Calidad de Servicio Percibida en Servicios de Voz y Video sobre IP

Pedro Casas Hernández
Diego Guerra Vidal
Ignacio Irigaray Bayarres

Tutor: Pablo Belzarena

Facultad de Ingeniería
Universidad de la República

Proyecto de fin de carrera, Ingeniería Eléctrica
Plan 97, Telecomunicaciones

30 de Agosto de 2005

Agradecimientos

Este proyecto no habría sido posible sin el apoyo de muchas personas.

Queremos dedicar y brindar nuestros más encendidos agradecimientos a nuestras familias. Por soportarnos todo este tiempo y estar siempre más allá de las diferencias.

A Jimena y Silvia agradecerles por ser la inspiración en las horas lentas.

A nuestro tutor Pablo Belzarena, agradecerle por su ayuda y su buena onda.

Queremos agradecer a todos los que participaron desinteresadamente en la realización de los tests subjetivos: federiquitin, nachopi, echi, edu, gonzuki, ale pini, nico, fabi, ale, merilyn, pipi, negro, ale fernandez, chino, coco, pablo aguirre, diego, leo, monzón, daniboy, el seba, daniel, maría del carmen, mayra, maledetto, olga, ceci, alba, hermann, ana, tito, julio, wilma, maggi, albert.

Queremos agradecer a Andrés Alcarraz, Federico Lecumberry, Pablo Cancela, Nacho Ramirez, y a Juan Cardelino por compartir su amplio conocimiento sobre las cosas con nosotros.

A Rafael Canetti, por liberar a Diegote

A nuestros amig@s, que siempre hacen más fácil el camino.

Queremos agradecer especialmente a Pablo Arias, Laura Aspirot, Paola Bermolen, Federico Larroca y Víctor González por el apoyo incondicional e incansable. 5 grandes valores.

Pedir disculpas por que nos estamos olvidando de muchos seguramente.

Muchas Gracias..... Pedro, Diego y Nacho

Contenido

Capítulo 1

Introducción

En este primer capítulo se presentan los distintos elementos que motivaron este proyecto (sección 1.1), los objetivos específicos planteados (sección 1.2), una breve descripción de la organización de este documento (sección 1.3) y un resumen de las tareas realizadas (sección 1.4).

1.1. Motivación

El abrumador crecimiento de Internet de los últimos años ha llevado a la proliferación de múltiples servicios que actualmente se ofrecen a través de la misma. Escuchar radio en vivo o hablar por Internet es algo cada día más normal y no resulta difícil pensar que a corto plazo se ofrecerán servicios del tipo TV o video bajo demanda (de hecho hoy se ofrecen en Uruguay servicios de este estilo [42, 43]). La tendencia al uso de Internet como la portadora de todos estos servicios es una realidad tangible. Sin embargo, este crecimiento desmedido no ha sido acompañado de un cambio estructural real que permita asegurar garantías de calidad a los usuarios finales: Internet es aún una red de mejor esfuerzo. Pero es claro que sin garantías de calidad no es posible la tarifación de servicios y la idea de *red convergente* quedaría en simplemente eso: una idea.

Este problema ha generado un creciente interés por parte de los proveedores de servicio en estimar la calidad ofrecida. El problema con la calidad de servicio (Quality of Service - QoS) es qué medir y como hacerlo. Tradicionalmente se ha estimado la QoS en base al estado de la red de transporte, midiendo valores promedio de probabilidad de pérdida de paquetes, retardos, ancho de banda disponible y demás. Algunos alegan sin embargo que las medidas tradicionales no son las adecuadas ya que no representan correctamente la QoS experimentada por el usuario final.

En los últimos años surge un nuevo enfoque para el problema de calidad de servicio en

Internet. Este se basa en que la calidad de un servicio es un tema completamente subjetivo y depende directamente de lo que el usuario perciba del mismo, independientemente del estado de la red que lo transporta. Esto es muy claro en los servicios de audio y video; en base a distintas técnicas de codificación de la señal y al uso de algoritmos de información redundante es posible que el usuario final experimente niveles de calidad aceptables aún frente a problemas serios en la red. Aparece así el concepto de **calidad de servicio percibida (PQoS)**.

Existen en la actualidad distintos algoritmos para estimar la calidad de servicio percibida en voz y video sobre IP. Muchos se basan en técnicas utilizadas en el diseño de redes de telefonía tradicional y en el desarrollo de nuevos algoritmos de codificación de audio y video; si bien en estas áreas han probado ser buenos estimadores, su pasaje a Internet no a sido del todo exitoso. Otras técnicas recientes han demostrado mejores resultados, pero al momento no es clara la generalidad de los mismos.

No obstante este desarrollo en el área, no existe hasta el momento (o al menos no hemos encontrado) una herramienta de uso libre que permita estimar, en base a estos algoritmos, la PQoS que un usuario experimentará al utilizar un servicio multimedia en Internet. La disponibilidad de una herramienta de este estilo permitiría lograr un mayor entendimiento de los problemas que han sido mencionados. Al mismo tiempo, una comparación imparcial de las bondades y desventajas de las técnicas existentes resultaría fundamental a la hora de encarar nuevos desarrollos en calidad de servicio.

Motivado en estas últimas observaciones, este proyecto intenta desarrollar un herramienta de software capaz de estimar la calidad de servicio percibida por un usuario de un servicio de voz y/o video sobre IP, utilizando para ello los algoritmos disponibles en la actualidad. La herramienta intentará ser integral, en el sentido de que englobará todos los problemas subyacentes al uso de estos algoritmos (transmisión y recepción de secuencias multimedia en tiempo real, mediciones en la red, etc..). Al mismo tiempo, se plantea un desarrollo modular de la herramienta que permita modificaciones y ampliaciones a futuro.

Para terminar, es importante destacar que este trabajo se enmarca dentro de un proyecto de estimación de QoS más general que el grupo de redes del IIE tiene en curso. Titulado METRONET, sus objetivos apuntan al desarrollo de metodologías que permitan realizar medidas de verificación de la QoS asegurada a usuarios de servicios de voz o video sobre Internet.

1.2. Objetivos del Proyecto

Presentaremos a continuación los objetivos específicos del proyecto:

Estudio e implementación de algoritmos de estimación de PQoS para los servicios de voz y video sobre IP.

Se plantea en primera instancia investigar el estado del arte en lo que respecta a algoritmos y técnicas de estimación de PQoS en voz y video sobre IP. En base a este estudio se seleccionarán los algoritmos más representativos para su eventual implementación (o adaptación en aquellos casos en los que sea posible la reutilización de código).

Desarrollo de una herramienta de software que permita estimar, desde los extremos involucrados en la transmisión y recepción de un servicio de voz o video sobre IP, la PQoS que experimentará el usuario de dicho servicio.

Se plantea el desarrollo de una herramienta de software que permita estimar la calidad de servicio percibida por un usuario de un servicio de voz y/o video sobre IP, utilizando para ello los algoritmos desarrollados. La herramienta deberá ser integral, en el sentido de que englobará todos los problemas subyacentes al uso de estos algoritmos (transmisión y recepción de secuencias multimedia en tiempo real, mediciones en la red, etc..). Al mismo tiempo se implementará un desarrollo modular de la herramienta para facilitar eventuales modificaciones y ampliaciones.

1.3. Estructura de la documentación

El contenido de esta documentación se distribuye como sigue:

En el capítulo 2 se introducen los conceptos generales de calidad de servicio (QoS - Quality of Service) en servicios de tiempo real sobre IP. Un resumen de los parámetros tradicionales de calidad de servicio se presenta en la sección 2.1. En las secciones 2.2 y 2.3 se describen las características principales de los servicios de voz y video sobre IP (VoIP y VideoIP respectivamente).

En el capítulo 3 se presenta el concepto de calidad de servicio percibida (PQoS - Perceived QoS). Las distintas técnicas de estimación de PQoS son presentadas en la sección 3.2. En las secciones 3.3 y 3.4 se describen las distintas herramientas y metodologías implementadas para el caso de técnicas de estimación intrusivas. Para el caso de técnicas no intrusivas se describen en la sección 3.5 las herramientas desarrolladas.

En el capítulo 4 se presentan los detalles de implementación de la herramienta de software desarrollada. El esquema de funcionamiento de dicha herramienta es presentado en la sección 4.1. En la sección 4.2 se describe la arquitectura del software, analizándose uno a uno los módulos que lo componen. La plataforma de pruebas utilizada en la calibración de los algoritmos y la realización de los test subjetivos se presentan en las secciones 4.5 y 4.6.

En el capítulo 5 se presentan los resultados obtenidos y la validación de la herramienta de medida.

En el capítulo 6 se presentan las conclusiones finales del proyecto y las posibles líneas de trabajo a futuro.

1.4. Tareas realizadas

El desarrollo de este proyecto se realizó en 4 etapas bien definidas:

1. Estudio del estado del arte en PQoS.
2. Implementación de los algoritmos de estimación de PQoS.
3. Realización de tests de calidad subjetivos, calibración de los algoritmos desarrollados y análisis de los resultados obtenidos.
4. Desarrollo de la herramienta de software.

En la figura 1.1 se presenta un diagrama cronológico del avance del proyecto; las etapas **(1)** y **(2)** se desarrollaron secuencialmente, mientras que las etapas **(3)** y **(4)** se desarrollaron en paralelo.

A continuación se describen las distintas etapas mencionadas.

(1) La primera etapa de este proyecto consistió en el estudio del estado del arte en lo que a estimación de PQoS en servicios de voz y video sobre IP respecta. Para ello fue necesario en primera instancia analizar las características principales de estos servicios. A partir de este estudio del área se concluyó que el mejor y más aceptado estimador de la calidad de servicio percibida por una persona es el **MOS (Mean Opinion Score)**. El MOS representa el valor promedio de calidad percibida que un grupo de personas asigna a cierto servicio (en una cierta escala de valores definida). La clasificación se realiza en base a una única secuencia del servicio a calificar (es decir, se determina como percibe la persona la secuencia de audio o video en cuestión independientemente de la secuencia original).

Otro de los estimadores de PQoS encontrados es el **DMOS (Degradation MOS)**. El mismo toma en cuenta la calificación que un grupo de personas asigna a la degradación percibida de cierto servicio. Para esto se utilizan dos secuencias en la estimación: la secuencia

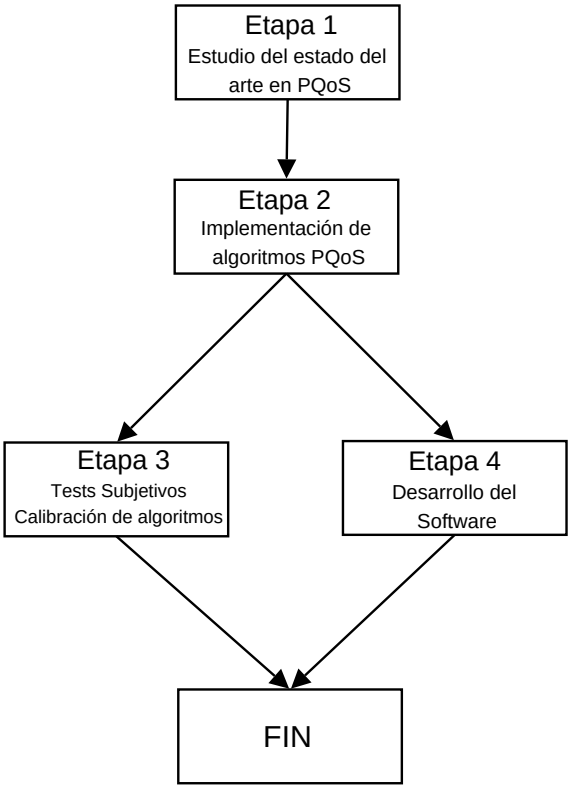


Figura 1.1: Orden cronológico de tareas desempeñadas

original y la secuencia distorsionada por efecto de su transmisión en tiempo real a través de la red.

En ambos casos es necesario realizar tests de calidad subjetivos para obtener los valores de MOS o DMOS correspondientes.

Este es básicamente el argumento de partida del proyecto: ¿cómo se puede estimar el valor del DMOS automáticamente, sin necesidad de realizar tests de calidad subjetivos?

Para responder a esta pregunta existen en la actualidad dos grandes grupos de técnicas o metodologías de estimación: las llamadas técnicas *intrusivas* y las técnicas *no intrusivas*. Las técnicas intrusivas se basan en inyectar en la red secuencias del servicio a clasificar para luego ejecutar, utilizando la secuencia distorsionada resultante y la secuencia original, un algoritmo de comparación de señales. Las técnicas no intrusivas permiten realizar la estimación en base al estado de la red de transporte y a las características del servicio en cuestión.

(2) El siguiente paso del proyecto consistió en implementar y/o reutilizar aquellos algoritmos de PQoS más utilizados o que a priori presentaban las mejores características. Esto resultó en la implementación de los siguientes algoritmos de audio y video:

- Structural Similarity Index Metric (SSIM) - comparación de secuencias de video.
- Métricas de distorsión espacio/temporal (ITS) - comparación de secuencias de video.
- Peak Signal to Noise Ratio (PSNR) - comparación de secuencias de video.
- E-Model - estimación no intrusiva en audio.
- Random Neuronal Networks - estimación no intrusiva en audio y video.

También se reutilizaron y adaptaron los siguientes algoritmos de audio:

- Perceptual Evaluation of Speech Quality (PESQ) - comparación de secuencias de audio.
- EMBSD (Enhanced Modified Bark Spectral Distortion) - comparación de secuencias de audio.
- MNB (Measuring Normalized Blocks) - comparación de secuencias de audio.

(3) Una vez implementados los algoritmos de estimación automática de PQoS (en base al DMOS) se procedió a realizar tests de calidad subjetivos para audio y video. Los resultados de estos tests son necesarios para llevar a cabo 3 tareas fundamentales:

- calibrar los algoritmos de comparación de secuencias.
- entrenar el sistema de aprendizaje supervisado implementado por las RNN.
- validar los resultados obtenidos.

(4) La última etapa del proyecto consistió en el desarrollo de una herramienta de software que automatiza el proceso de estimación de PQoS entre los extremos involucrados en la transmisión y recepción de un servicio de voz o video sobre IP.

Capítulo 2

Servicios de Tiempo Real sobre IP

En este capítulo se introducen los conceptos generales de calidad de servicio (QoS - Quality of Service) en servicos multimedia de tiempo real sobre IP. Un resumen de los parámetros tradicioneales de calidad de servicio se presenta en la seccion 2.1. Al mismo tiempo y motivados en lograr un mayor entendimiento de la gran variedad de elementos que componen estos servicios multimedia, se describen en las secciones 2.2 y 2.3 las características fundamentales de los servicios de voz y video sobre IP. En ambos casos se brinda un análisis de la influencia de los parámetros que afectan la calidad de estos servicios.

2.1. QoS - Calidad de Servicio

El concepto de QoS (Quality of Service) representa hoy en día un elemento fundamental a la hora de hablar de venta y consumo de servicios. Si bien no es un concepto nuevo, es muy fácil escuchar cada vez más la palabra QoS. La razón es simple: los consumidores de servicios son cada día más exigentes y la necesidad de poder cuantificar la calidad ofrecida se ha tornado un tema fundamental.

¿Pero qué significa exactamente QoS? Podría definirse QoS como el valor de un conjunto de parámetros de performance que aseguran al usuario de un servicio niveles aceptables de calidad. Como distintos tipos de servicio mantienen características particulares, cada uno tendrá su propia QoS. Por ejemplo, en el caso de telefonía tradicional se puede definir QoS como el tener un canal de $64Kbps$ durante el tiempo que dure la conversación y una disponibilidad de servicio de $99,999\,\%$. En el caso de Internet, las características heterogéneas de los distintos servicios que transporta hacen del tema calidad de servicio una problema mayor. Es difícil identificar en cada caso cuales son los parámetros de performance que aseguran niveles aceptables de calidad.

Los parámetros que normalmente se utilizan hoy en día en Internet son la pérdida de paquetes, los retardos, la variación del retardo (jitter) y el ancho de banda disponible entre otros. Las metodologías de medida que se utilizan trabajan principalmente con valores medios de estos parámetros. Sin embargo, resulta bastante claro que los valores medios de estos parámetros no son necesariamente los más adecuados. Basta pensar por ejemplo en la pérdida de paquetes: la degradación de un servicio frente a pérdidas individuales y periódicas será percibida de forma distinta que en el caso de pérdidas en ráfagas y a tiempos variables. A este respecto se han realizado recientemente algunos trabajos que parecen confirmar esta teoría (ver [29]).

Por último, es importante resaltar el trabajo que se ha venido desarrollando en la estandarización de estos conceptos. En un esfuerzo por definir claramente los parámetros de calidad de servicio para Internet el IETF (Internet Engineering Task Force), en particular uno de sus grupos de trabajo (el grupo IPPM - IP Performance Metrics) ha redactado varias recomendaciones al respecto (en el anexo H se presenta un resumen de dichas recomendaciones).

2.2. Especificaciones del servicio de VoIP

En esta sección se introducen los conceptos básicos para la comprensión del servicio de voz sobre IP[1]. Al conjunto de tecnologías que hacen posible la transmisión de la voz sobre redes que utilizan el protocolo IP se le llama VoIP.

Se pueden distinguir dos categorías dentro de estas tecnologías:

- Las que se encargan del tratamiento de la señal de voz. Digitalización, compresión, paquetizado, etc.
- Las encargadas de la configuración de la llamada (call setup).

En la primer categoría encontramos las que hacen posible la transmisión eficiente de la señal de voz por la red. El primer paso es pasar la señal del dominio analógico al dominio digital. Luego esta señal es codificada para reducir la tasa de información a transmitir. La señal codificada de manera eficiente es paquetizada para enviarla por la red. En el receptor el proceso es el inverso.

En la segunda categoría se ubican las tecnologías de intercambio de información sobre la configuración de la llamada. Existen dos estándares en la actualidad: H.323 y SIP. Esta información es utilizada para autorización, autenticación, resolución de direcciones, etc.

[1]Parte de esta sección se basa en una recopilación realizada por Gonzalo Mateos para su proyecto de fin de carrera.[36]

El proyecto se concentra en el tratamiento que recibe la voz y las degradaciones que sufre en las distintas etapas.

2.2.1. Codecs de audio

Una de los factores determinantes a la hora de transmitir una señal de voz por la red es el codec. El codec a utilizar determina la tasa de bits necesaria, y junto con esto la calidad de la voz. Podemos dividirlos en tres categorías según su principio de funcionamiento:

- Codecs de forma de onda (waveform)
- Codecs vocales
- Codecs híbridos

Codecs de forma de onda Estos codecs se basan en almacenar información sobre la forma en el tiempo de la señal. Por lo general son los que más ancho de banda consumen, dado que no utilizan ninguna característica especial de la señal. Pueden ser utilizados para transmitir cualquier tipo de señal, no solamente voz (musica,fax).

PCM (Pulse Code Modulation)

Codifica la forma de onda con una precisión de n bits por muestra. Las variantes (ley A y Mu) equivalen a un muestreo no uniforme que permite mejorar la relación señal a ruido con 8 bits por muestra. Se trata de la codificación básica de la telefonía pública a 64 kb/s.

- Se filtra la señal y muestrea la señal de voz a una frecuencia de 8 KHz. (4KHz es el ancho de banda mínimo para mantener la inteligibilidad de una señal de voz).
- Se cuantizan las muestras obtenidas en q niveles discretos (cuantización).
- Se codifican las muestras en palabras binarias de n bits (en gral. palabras de 8 bits).

DPCM (Differential PCM)

Se fundamenta en la predicción de muestras mediante la memorización en el tiempo. Se realiza la codificación de la diferencia entre la muestra y la predicción.

ADPCM (Adaptive Differential PCM)

Implementa la misma técnica que DPCM pero utilizando un algoritmo de predicción más eficiente. La predicción se realiza en base a un algoritmo autoadaptativo dependiente de la actividad de la señal vocal.

Codecs vocales Este tipo de codecs se basan en un modelo de como el sonido fue creado. Reconstruyen la señal solamente con la información del modelo, la forma de la señal es descartada por completo. El modelo es construido teniendo en cuenta el sistema vocal humano, básicamente un resonador y un tubo. Las desventajas que introducen estos codecs son: alta complejidad (cálculos necesarios) y dado que procesan la señal en tramas introducen un retardo significativo. Este tipo de codecs utilizan muy poco ancho de banda (2.4 Kb/s) pero se pierde completamente la naturalidad de la voz debido a la simplificación del modelo.

Codecs híbridos Dentro de esta categoría se encuentran los codecs que están a medio camino entre los anteriores. Incorporan elementos de los dos, resultando en tasas de transmision mas alta que los vocales pero menor que los de forma de onda. La complejidad es menor que los vocales y mayor que los de forma de onda. En términos de calidad se aproximan a los codificadores de forma de onda.

G.723 CELP (Code Excited Linear Prediction)

Los codecs CELP trabajan dividiendo la señal en segmentos del habla, denominados tramas. Usan un modelo del sistema vocal para remover la redundancia de la señal, permitiendo la transmisión a una tasa de datos más baja (típicamente entre 4 y 16 kbps para aplicaciones telefónicas). Los codecs CELP generalmente crean un retardo mayor que los codecs de forma de onda. Una trama de habla CELP no puede codificarse hasta que el codificador obtenga toda la información corrrespondiente a esa trama. Esto significa que hay un retardo de una trama entera antes de que el codec pueda empezar a procesar la señal.

GSM RPELTP (RegularPulse Excitation LongTerm Predictor)

Es el estándar europeo más utilizado en la telefonía celular. Una de sus ventajas sobre los demás codecs de esta categoría es su simplicidad. La tasa de bits más alta que permite es de 13 Kb/s.

Supresión de silencios y VAD (Voice Activity Detection)

Es un mecanismo complementario al empleo de codecs compresores para reducir el ancho da banda. Se pretende detectar períodos de silencio durante la conversación (mecanismos VAD, Voice Activity Detection) suprimiendo el envío de paquetes de voz mientras dure la situación. Como en la conversación telefónica cada interlocutor sólo habla la mitad del tiempo y realiza pausas entre frases, se pueden obtener reducciones de hasta el 60 % en el flujo de paquetes. La señal de silencio igualmente se codifica, pero el software de supresión evita que se envíen dichos bloques de datos.

Para evitar que el interlocutor piense que se ha cortado la comunicaciøn durante los intervalos de silencio la ITU-T especifica dos posibles soluciones:

- enviar periódicamente paquetes de silencio (SID, Silence Insertion Description) durante la pausa. Estos paquetes proporcionan una indicación del nivel de ruido que existe en el origen para que el receptor lo simule en el terminal remoto mediante un algoritmo de CNG (Comfort Noise Generation) o generador de ruido. (Recomendación ITU-T I.366.2)
- para evitar el envío de paquetes SID es posible marcar el bloque generado como NOTX (No Transmission). En el receptor se genera ruido ambiente a partir de una señal de ruido blanco o del muestreo del auricular.

Tamaño de los paquetes de voz

La elección del tamaño de los paquetes de voz es otro factor a tener en cuenta, dado que las cabeceras que se añaden a los paquetes generan un extra a la tasa normal del codec. Si se desea minimizar el impacto de las cabeceras en el tráfico, es preciso enviarlas el menor número de veces posible y para ello se envían paquetes de gran tamaño con varios bloques de datos en cada paquete.

Esta solución presenta un gran inconveniente porque cuanto mayor es el número de bloques de voz por paquete, mayor es el tiempo de empaquetado (tiempo que hay que esperar para llenar el paquete).

El retardo de punta a punta es uno de los aspectos más críticos de los sistemas de voz empaquetada y será lo que determine el máximo tamaño de paquete posible. Como la compresión de voz aumenta el tiempo de empaquetado, cuanto más complejo sea el algoritmo de compresión, menor deberá ser el tamaño de los paquetes para que el retardo no sea excesivo.

Ancho de banda para VoIP

Al estimar el ancho de banda necesario para VoIP es necesario tener en cuenta el tamaño de carga útil y la sobrecarga por cabeceras. El tamaño de carga útil viene determinado por el tamaño de los bloques de información que entrega el codificador y por el número de bloques que se desean transportar en un paquete. En el cálculo de la sobrecarga por cabeceras se tienen en cuenta las cabeceras que añaden los sucesivos protocolos (RTP, UDP, IP y capas inferiores). En el caso más sencillo, por ejemplo en una sesión de voz entre dos terminales VoIP, la cabecera RTP se compone de 12 octetos, a los que hay que sumar los 8 de la cabecera UDP y los 20 de la IP. Los octetos de los niveles inferiores dependen de la tecnología concreta utilizada (por ejemplo 6 octetos para PPP).

En la siguiente tabla se presenta el ancho de banda necesario para una llamada de voz en dos casos diferentes: codec G.711 de 64 Kbit/s (cláísico de la red telefónica) y el G.729 de 8 Kbit/s (codec de baja velocidad utilizado en el acceso a una red IP via módem)

De la tabla se observa claramente la influencia del tamaño total de la carga útil en la tasa total de envío y en el retardo de empaquetado. Cuanto mayor es la carga útil, menor es la tasa

Códec	Tasa nominal (Kbit/s)	Retardo empaquetado (ms)	Tamaño carga útil (octetos)	Tasa real (Kbits/s)		
				IP	IP/PPP	IP/AAL5
G.711	64	5	40	128	137.6	169.6
		10	80	96	100.8	127.2
		20	160	80	82.4	106
G.729	8	10	10	40	44.8	84.8
		20	20	24	26.4	42.4
		40	40	16	17.2	21.2

Tasa de envío nominal y real de dos códecs normalizados

de envío (se reduce la sobrecarga por cabeceras) pero mayor es el retardo (aumenta el tiempo de construcción del paquete). Para reducir la sobrecarga agregada en las distintas capas se utilizan algoritmos de compresión de cabeceras. Los mecanismos de compresión de cabeceras se aplican en el enlace, es decir que si un extremo utiliza este método el extremo remoto es responsable de la descompresión. Un ejemplo típico es el enlace de acceso vía módem desde un PC hasta el servidor de acceso remoto de un ISP.

Existen varios estándares de compresión de cabeceras (RFCs 2508, CRTP y 3095, ROCH) que permiten la compresión de cabeceras IP, UDP y RTP. Estos mecanismos suprimen la información redundante que se transmite repetidamente en una sesión RTP y consiguen reducir la sobrecarga de cabeceras a 2 o 4 octetos.

Protocolos de Tiempo Real sobre IP

Los protocolos de tiempo real para la transmisión de audio y vídeo por Internet se definen dentro de la RFC 1889 (actualizada en la RFC 3550). Esta recomendación incluye dos protocolos que constituyen el estándar de hecho: RTP (Real Time Protocol) y RTCP (Real Time Control Protocol). El RTP regula el intercambio de información en diferentes formatos (audio y video). El RTCP regula la comunicación de control que se establece entre los extremos, en paralelo con la transmisión de información.

La norma no establece qué protocolos deben utilizarse en las capas inferiores, por debajo de RTP/RTCP; sin embargo, en la mayor parte de los casos se emplea UDP.

En el anexo F se brinda un descripción detallada de estos protocolos.

2.2.2. Parámetros que afectan la calidad de la voz

Retardo punta a punta

Existen muchos factores que contribuyen al retardo de punta a punta: retardo del algoritmo de codificación, tiempo de empaquetado, tiempo de propagación (despreciable salvo en distancias muy grandes), tiempo de transmisión, tiempos de espera en las colas de la red (dependiente del tráfico en la red), tiempo de descompresión,etc.

El retardo total de punta a punta en una conversación telefónica debe mantenerse por debajo de un cierto nivel para minimizar la pérdida de interactividad entre los usuarios. La norma ITU-T G.114 especifica un retardo máximo de punta a punta de 150ms. Para valores mayores de retardo la comunicación se vuelve molesta por la pérdida de interactividad: la persona que habla al percibir que su interlocutor tarda en contestar, repite sus palabras, a la vez que recibe la respuesta procedente del otro extremo.

En la figura 2.1 se muestran los distintos componentes que contribuyen al retardo de punta a punta en VoIP:

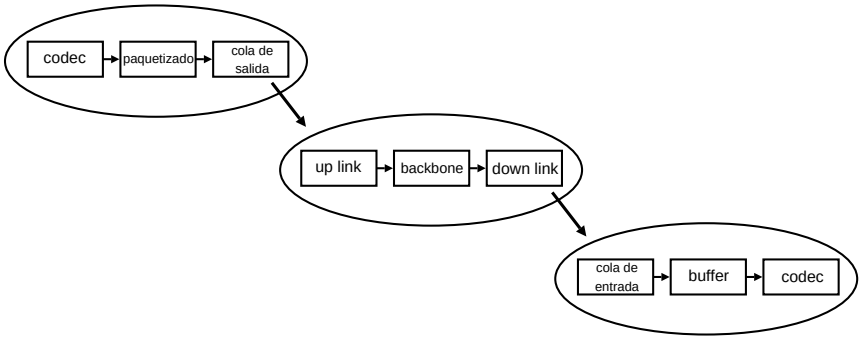


Figura 2.1: Componentes del retardo

Los retardos introducidos en cada instancia pueden a su vez ser clasificados en retardos fijos y variables. Por ejemplo, en el retardo introducido por la red puede considerarse un retardo de transmisión fijo debido a las distancias y un retardo variable debido a las condiciones cambiantes de la red.

Los componentes fundamentales en el retardo total de punta a punta son los siguientes:

1. Retardo de la Red
2. Retardo del Codec
3. Retardo del Buffer receptor

En la siguiente tabla se muestran valores típicos para el uso del codec G.729, tomando una carga útil de 10 bytes.

Componente	Retardo Fijo (ms)	Retardo Variable (ms)
Codec G.729	15	
Empaquetado	10	
Retardo de Cola		Depende del Up Link
Retardo de Red	50	Depende de la carga de la Red
Buffer receptor	50	
Total	**125**	

Variación del retardo (Jitter)

El transporte de voz empaquetada no es sensible sólo al retardo de punta a punta sino también a las fluctuaciones o variaciones de ese tiempo de retardo (jitter). Estas variaciones son debidas a la fluctuación en los tiempos de espera de los nodos de la red que dependen del tráfico concreto del momento. Dado que la generación de bloques de voz se realiza a tasa constante en el emisor, el algoritmo de decodificación espera (para un correcto funcionamiento) que la llegada también sea a tasa constante.

Para minimizar los efectos del jitter se implementa un almacenamiento temporal (en base a buffers) de paquetes en el receptor de manera tal de enviar al decodificador los bloques codificados en el emisor de manera sincrónica. El uso de buffers en el receptor también soluciona el problema de la llegada de tramas RTP fuera de orden, chequeando los números de secuencia de éstas. La desventaja que presenta el uso de buffers en la recepción es el aumento del retardo fijo total de punta a punta.

Pérdida de paquetes

La pérdida de paquetes es otro factor preponderante en lo que refiere a servicios en tiempo real. Dado que el tráfico de VoIP se implementa sobre UDP (RTP sobre UDP) el único control que se puede realizar sobre la pérdida de paquetes se da en las puntas de la transmisión. Los codecs implementan técnicas de corrección de errores para hacerlos transparentes al usuario, utilizando algoritmos de interpolación sobre los datos recibidos para generar la información perdida.

Sin embargo, cuando las pérdidas superan cierto umbral (del orden del 3 %) o cuando se dan en ráfagas ya dejan de ser útiles las técnicas mencionadas.

En la siguiente tabla se indican valores especificados por el grupo 12 de la ITU-T para los parámetros antes mencionados en distintas aplicaciones de voz :

Aplicación	Simetría	Retardo pta a pta	Jitter (ms)	Pérdida de paquetes
Conversación	Ida y vuelta	< 150 ms (Preferido) < 400 ms (Máximo)	< 1	< 3 %
Streaming de audio	Solo ida	< 10 s	< 1	< 1%
Mensajes de Voz	Solo ida	< 1 s (reproducir) < 2 s (grabar)	< 1	< 3%

2.3. Especificaciones del Servicio de VideoIP

En esta sección se introducen los conceptos básicos para la comprensión del servicio de video sobre IP[2]

En primer lugar se describien brevemente las técnicas de compresión utilizadas en video digital. A continuación se presentan los codecs más utilizados en la actualidad. Por último se analiza el impacto de los parámetros de la red sobre la calidad percibida.

Como toda imagen, un cuadro (frame) de una secuencia de video contiene información redundante. Suprimiendo dicha información la calidad no se ve muy deteriorada, y se reduce la información a transmitir. Para ello se utilizan transformadas invertibles de matrices en matrices que lo que hacen es tomar una matriz y obtener otra cuyos coeficientes de valor numérico significativo son pocos y se encuentran concentrados en las primeras posiciones. Estos coeficientes son los que en definitiva se transmitirán. Por lo general se utiliza la transformada del coseno discreta (DCT - Discrete Cosine Transform). Dado que el costo del algoritmo no es lineal con el tamaño de la matriz a transformar, se divide la imagen en matrices de 8×8 (a veces 16×16) llamadas Macro-Bloques (MB), y se aplica la transformación a cada uno de éstos por separado. A su vez, cada coeficiente es codificado utilizando VLC (Variable Length Coding - Codificación de Largo Variable), el cual aprovecha el hecho de que es muy probable una sucesión larga de ceros luego de un valor distinto del nulo. Lo que se hace es codificar el valor no nulo y cuántos ceros lo siguen. Esto tiene como contrapartida que no todos los macro-bloques tengan la misma cantidad de bits luego de codificados.

Una secuencia tiene información temporal redundante. Para aprovechar este hecho se utiliza la compresión temporal. Ésta consiste en analizar una secuencia de video para que en lugar de transmitir todos los cuadros consecutivos tan solo se codifique un cuadro y la diferencia entre éste y los cuadros cercanos. Por ejemplo, se codifica el cuadro 1 entero y en lugar de codificar el cuadro 2, tan solo se codifica aquella información que es distinta entre dichos cuadros. Esto permite que en las secuencias en las que la información es muy redundante (o sea existen muy pocas variaciones entre cuadros consecutivos) se consigan factores de compresión muy elevados, ya que la diferencia entre ellos es prácticamente nula. La mayoría de las técnicas de compresión temporal que se utilizan en la actualidad no se

[2]Las dos primeras partes de esta sección se basan en una recopilación realizada por Federico Larroca para el proyecto PDT "METRONET" [37].

basan tan sólo en la codificación de la difrencia entre cuadros consecutivos, sino que lo que codifican es la diferencia entre un cuadro y la predicción del siguiente. Esto eleva mucho el cómputo del procesado y permite obtener a cambio un flujo de datos mucho más reducido y una imagen de calidad óptima.
Existen tres posibilidades para los cuadros:

1. Que sea el cuadro "original". Es decir, que para decodificar el cuadro no es necesario ningún otro. Estos cuadros son llamados I-pictures (Intra pictures).
2. Que para decodificar el cuadro sea necesario el cuadro anterior porque se transmitió la diferencia entre la predicción a partir del ese cuadro y el de este instante. Estos cuadros son llamados P-picture (predicitive picture) y son necesarios menos bits para codificarlo que los cuadros I para la misma calidad.
3. Por último se encuentran los B-picture (Bidirectionally-predictive picture), que para decodificarlas son necesarios tanto el cuadro anterior como el siguientes, pues fue a partir de ambos que se predijo un valor para el cuadro de este cuadro. Utilizar este tipo de cuadros aumenta el delay pues es necesario esperar al siguiente cuadro para mostrar el de este instante, aunque tienen como ventaja que son los de mayor compresión de los tres.

Si la compresión espacial es con pérdida, se deben incluir periódicamente cuadros del tipo I-picture. Además, si quiero recibir una transmisión de video luego de que ésta ya haya comenzado, debo recibir algún cuadro del tipo I para empezar a decodificar los cuadros P o B. Por último, como cada tipo de cuadro tendrá distinto grado de compresión, esto aumentará el hecho de que cada MB tendrá distinto tamaño en bits luego de codificado.
Además para que el macro-bloque a partir del cual se predice sea el más adecuado, se pueden utilizar técnicas de compensación de movimiento para buscar el MB más similar al que se va a predecir.

Por lo tanto, y a modo de resumen, la secuencia típica de una cuadro al ingresar al codificador es (suponiendo que utilizaremos un cuadro tipo P): se predice su valor con el cuadro anterior, se le resta dicha predicción, a ésta se le aplica la DCT y se codifican los coeficientes con VLC, por último se transmite por el canal.

2.3.1. Codecs de video

A continuación se presentan los codecs más utilizados en video:

- **H.261** Es un codec pensado para utilizarse con teleconferencias sobre ISDN, por lo que su calidad no es lo más destacable. Consume un ancho de banda múltiplo de 64 kbps (P*64 kbps, con P natural), por lo que se lo conoce como P*64. Cuenta con un mecanismo para controlar la calidad en función del movimiento de la secuencia. Cuánto mayor el movimiento, menor la calidad de la imagen, de forma que la tasa sea constante.

- **H.263** Es una versión mejorada del H.261. Mejora la compensación de movimiento, tiene una mayor configurabilidad (menor tasa de bits o mayor recuperación frente a errores). También agrega soporte para pictures del tipo P y B. Además, soporta una mayor variedad de resoluciones que su antecesor, por lo que su uso se extiende más allá de la videoconferencia.

- **MJPEG** Aunque no es un CODEC de video propiamente dicho, se utiliza muy a menudo una sucesión de cuadros codificados con el formato JPEG. O sea, no se utiliza compresión temporal de ningún tipo.

- **MPEG-1** Genera datos a una tasa entre 1 y 1.5 Mbps. Tiene la calidad del VHS (352×288 y 30 fps). Está pensado para utilizarse en medios como el CD-i, por lo que utiliza ancho de banda sin mayor cuidado. Además, tiene una gran susceptibilidad a las pérdidas por su extenso uso de cuadros tipo P y B (el uso de éstas últimas hace que su utilización en aplicaciones con algún tipo de interactividad no sea lo más adecuado por la latencia extra). Por último, no implementa ningún tipo de escalabilidad (que será explicado más adelante).

- **MPEG-2** Es la extensión de MPEG-1, pero soporta mayores resoluciones aún y mejores prestaciones en audio. Esto trae como contrapartida un mayor ancho de banda consumido (entre 4 y 15 Mbps). Implementa algunas escalabilidades. Esto es enviar una transmisión base con lo necesario para que la calidad sea aceptable, y además enviar otras transmisiones (capas superiores) con la información extra necesaria para que la calidad sea la requerida. Esto es útil cuando se necesitan descartar algunos paquetes, pues se puede comenzar por los que pertenecen a las capas superiores. MPEG-2 implementa tres: escalabilidad SNR, espacial y temporal. La primera es cuantizar la base con menor cantidad niveles, y las capas superiores con mayor precisión. En la espacial se envía en la base la resolución mínima, y en las capas superiores se envía información para ir aumentando la resolución. Por último, la escalabilidad del tipo temporal es aquella donde se envía como base la cantidad mínima de cuadros por segundo, y se marcan como capas superiores los cuadros entre aquellos que forman la base. Nuevamente la idea para la que fue diseñado fue para la utilización en medios como el DVD o la transmisión satelital (donde se logra un mejor aprovechamiento del ancho de banda de los canales analógicos), y es por eso que no se usa casi en transmisiones por internet.

- **MPEG-4**. Este codec soporta tres rangos de generación de datos:

  1. Menor a 64 kbps
  2. Entre 64 y 384 kbps
  3. Entre 384 y 4000 kbps

  Fue diseñado para utilizarse en internet y para reproducir video de calidad variada. En el rango inferior, utiliza las mismas técnicas que MPEG tradicional (utilización de macro-bloques, bloques I, P y B, etc). En los rangos superiores utiliza un enfoque completamente distinto, pues separa la imagen en regiones (objetos). No se utilizan más los macro-bloques de tamaño fijo y se pasa a trabajar con macro-bloques de tamaño variable, donde cada macro-bloque puede llegar a representar un objeto (o partes del

mismo) en la secuencia. Además se diseñó mucho más tolerante a errores mediante el uso de marcadores de resincronización, cabeceras, etc. Sigue teniendo la escalabilidad de MPEG-2, pero ahora con los objetos. Por ejemplo, supongamos que tenemos una secuencia que consta de una persona hablando delante de un fondo relativamente estático. Se puede configurar que la resolución tanto temporal como espacial del objeto fondo sea la mínima, y darle gran calidad al objeto persona. Por último, tiene una estructura de capas:

- **VOP** Es la muestra temporal de un objeto.
- **GOV** Un conjunto de VOP se puede agrupar en un GOV para facilitar la resincronización o el acceso aleatorio a la secuencia.
- **VOL** Cada nivel o capa de la escalabilidad de cada objeto se encuentra en los VOL.
- **VO** Es un objeto de la secuencia.
- **VS** Es la secuencia completa.

Sin embargo, el concepto de objeto no se utiliza en las implementaciones de este codec por cuestiones claras de eficiencia. Cualquier algoritmo de segmentación (necesario para identificar objetos dentro de una imagen) implica tiempos de cálculo y recursos importantes que inhabilitan su uso. Por lo tanto, los macro-bloques en MPEG-4 difícilmente representen objetos.

2.3.2. Parámetros que afectan la calidad del video

La calidad del servicio de video en Internet percibida por un usuario es afectada básicamente por los parámetros de la red y por el tipo de codificación implementada. Es claro por ejemplo que el efecto de la pérdida de paquetes depende fuertemente del tipo de codificación utilizada; este factor resulta crítico en aplicaciones de video de alta calidad en donde, con la finalidad de disminuir el ancho de banda consumido, se implementan codificaciones que hacen uso de cuadros predictivos (MPEG-1, MPEG-2, MPEG-4) cuya pérdida resulta en falta de información para decodificar los cuadros siguientes.

Otros factores importantes que afectan la calidad percibida están directamente relacionados con la fuente de video, encontrando en este grupo la resolución del cuadro, la luminancia (niveles de gris) y la "profundidad"del color (nº de bits por pixel) y la tasa de generación de cuadros. La sincronización entre el audio y el video influye también sobre la opinión de la calidad, si bien no hay una relación directa entre ellos en lo que respecta al transporte (se transmiten por distintos canales y con distinta codificación).

Como se verá más adelante los efectos de la variación del retardo entre paquetes es el otro factor de crítico en la calidad percibida, aunque en este caso no es tan evidente a priori el porqué de esta fuerte influencia.

Por último, otro de los elementos a considerar es el aspecto temporal del video transmitido. Un video con poca variación espacial entre cuadros (informativo p.e.) será más robusto frente a pérdidas y jitter en el sentido de que al usuario le será más difícil notar la falta o retardo de la información. Por el contrario, aquellos videos con alta variación espacial entre cuadros (videos de acción o deportes) serán muy sensibles ante estos factores.

Pérdida de paquetes

La pérdida de paquetes es la principal causa de degradación de la calidad. Debido a la forma en que se codifica el video, la pérdida de un paquete en un cuadro puede afectar a los cuadros siguientes. Por ejemplo en el caso que sea un cuadro tipo I, la degradación de este provocará la degradación de los subsecuentes, hasta la recepción de un nuevo cuadro I.

En el trabajo de Liang, Apostolopoulos y Griod [1] se presenta un análisis del impacto de la pérdida de paquetes en la calidad de video comprimido. Para ello consideran que cada paquete representa un cuadro y si un cuadro se pierde se reemplaza por el anterior recibido. La codificación es tal que se manda un cuadro I cada N cuadros, los restantes son tipo P. Como medida de la distorsión total utilizan la suma de el MSE de cada cuadro (ver 3.4.1). La conclusión a la que llegan es que no sólo importa la media de paquetes perdidos, también importa la distribución de dichas pérdidas. Por ejemplo la distorsión total debido a dos pérdidas consecutivas es mayor que si las pérdidas fuesen independientes (esto es separadas por más de N cuadros).

Otro factor determinante en la influencia de las pérdidas sobre la calidad percibida es el codec utilizado. Se puede observar en la figura 2.2 la diferencia entre la distorsión producida por la pérdida de paquetes para un mismo video, codificado en MPEG-1 y en MPEG-4. En ambos casos la pérdida de paquetes se traduce en una pérdida de macro-bloques (MB). En el caso de MPEG-1 los MB son cuadrados de tamaño fijo y la tasa de actualización de los mismos (envío de MB de tipo I) es constante, por lo que la pérdida de éstos produce cuadrados erráticos en una posición determinada de la imagen hasta la llegada de un nuevo MB I. En el caso de MPEG-4 los MB son de tamaño variable y el uso de MB de tipo I es más "concentrado" en aquellas regiones de mayor movimiento. De esta forma la pérdida de un MB afectará principalmente las regiones de baja cantidad de movimiento. Este efecto se puede observar en la figura 2.2, donde la moto (es el elemento que se mueve) deja una "estela", correspondiente a los MB que van quedando como parte del fondo (bajo movimiento) sin ser actualizados.

Jitter

Veamos la influencia del jitter y las pérdidas en un simple ejemplo. La figura 2.3 muestra un flujo de paquetes de video en tres escenarios distintos: ausencia de pérdidas y jitter, canal

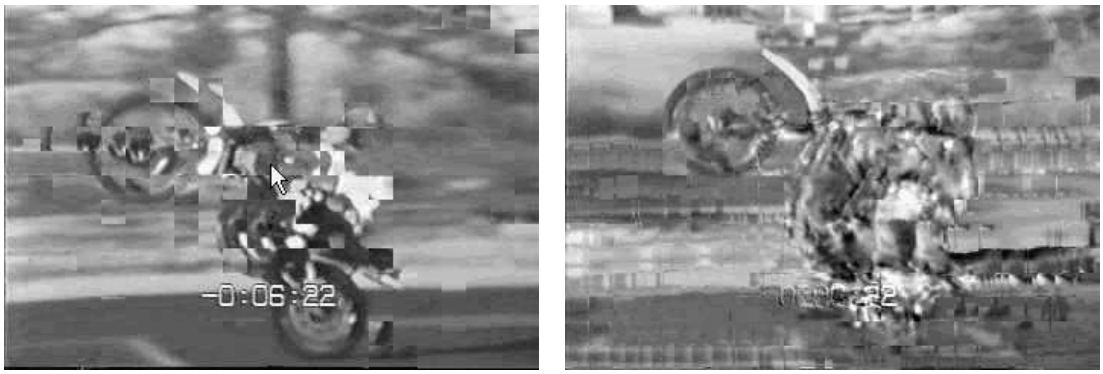


Figura 2.2: Efectos de la codificación (izquierda - MPEG-1, derecha - MPEG-4)

con jitter y canal con pérdidas.

En ausencia de pérdidas y con retardo fijo entre paquetes la decodificación se realiza en tiempo y forma, resultando en una reproducción suave y continua del video. En el caso del jitter, la variación del tiempo entre arribos hace que la decodificación no sea realizada en el momento correcto y, dependiendo de que tan grande sea dicha variación, pueda llegar a considerarse que el paquete se ha perdido. El usuario verá la imagen del último cuadro decodificado congelada (segundo cuadro en el ejemplo) hasta la llegada del próximo cuadro. El cuadro retrasado será reproducido en un tiempo menor al requerido, de forma de mantener el secuenciamiento temporal con el próximo cuadro. En el caso de las pérdidas, además de tener el efecto de congelamiento de la imagen se agrega el problema de la necesidad de los cuadros anteriores para la decodificación de los siguientes (aunque la inclusión de información redundante en ciertos cuadros disminuye este problema). Intuitivamente se puede ver entonces que los efectos de las pérdidas y del jitter sobre la percepción del usuario resultan similares.

Esta similitud, en principio intuitiva, motivó a varios investigadores de calidad perceptual en video a comparar empíricamente los efectos conjuntos de estos factores. En lo que sigue describiremos uno de los primeros trabajos en esta área realizado por *Claypool* y *Tanner* en el año 1999 ([2]).

El experimento implementado consistió en generar secuencias de video distorsionadas en base a distintos niveles de pérdida y jitter para luego someterlas a una clasificación subjetiva de calidad (ver Anexo D). Al mismo tiempo se consideraron secuencias de video con distinta variación temporal con el fin de determinar la influencia conjunta de todos los parámetros sobre la calidad percibida. De esta forma se lograron, para 3 categorías de variación temporal (baja, media y alta) secuencias del tipo jitter bajo, jitter alto, tasa baja de pérdidas y tasa alta de pérdidas. Se realizó un test subjetivo de MOS mediante ACR para 25 secuencias distintas, incluyendo además una indicación por parte de cada individuo del número de "eventos de distorsión" percibidos. Por último se compararon los resultados obtenidos en presencia de

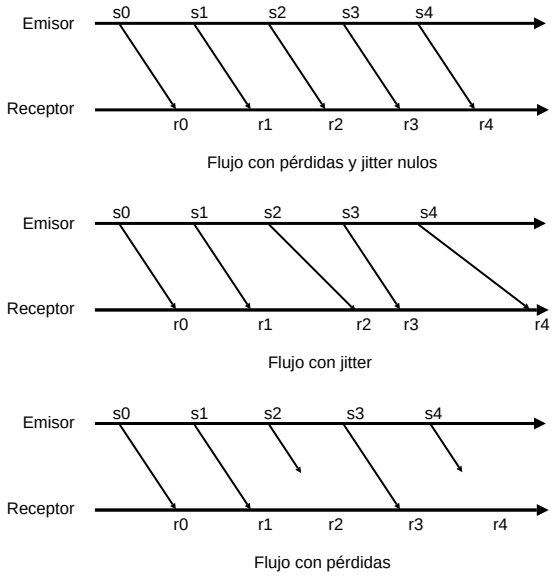


Figura 2.3: Efectos del jitter y las pérdidas

jitter vs. presencia de pérdidas.

Del estudio de los resultados experimentales se pueden presentar las siguientes conclusiones del efecto del jitter sobre la calidad percibida:

- Como se intuía previo al estudio, los efectos del jitter sobre la degradación de la calidad percibida son muy similares a los efectos de la las pérdida. Es más, la degradación es severa en presencia de bajos niveles de pérdida y/o jitter, aunque se observa que valores mayores de dichos parámetros no degradan proporcionalmente la calidad percibida.
- Como era de esperar, aquellas secuencias con menor variación temporal fueron más robustas frente a la presencia de jitter que las de mayor variación temporal.
- Se observa una alta correlación entre los valores de MOS obtenidos y el número promedio de eventos de distorsión detectados, lo que sugiere el uso del conteo de dichos eventos como estimación de la calidad percibida (por ejemplo, detectando cuantas veces se superan ciertos umbrales en los parámetros de cuestión).

Capítulo 3

Calidad de Servicio Percibida (PQoS)

3.1. Introducción

En el capítulo anterior se presentó el concepto de QoS como un conjunto de parámetros que garantizan un cierto nivel de performance. El problema con estas métricas es que el usuario final (persona no experta en QoS), consumidor de un servicio multimedia no está interesado en saber la probabilidad de pérdida de paquetes ni el jitter, quiere tener una estimación de la calidad con la que *percibirá* el servicio en cuestión. Surge así la idea de PQoS (perceived quality of service) como la calidad percibida por el usuario. Después de todo, calidad de servicio será para una persona lo que pueda percibir del mismo, independientemente del estado de la red que lo transporte.

Estimar la calidad de servicio percibida es un requisito fundamental en los sistemas de comunicación modernos por razones técnicas, legales y comerciales. Las medidas de calidad percibida pueden realizarse usando métodos objetivos o subjetivos como se muestra en la figura 3.1. Los métodos subjetivos definen la métrica más aceptada ya que representan una conexión directa con la calidad percibida por los usuarios. Estos métodos consisten en evaluar la opinión media de un grupo de personas, para ello se presentan distintas secuencias y cada individuo asigna un valor de calidad. El problema inherente a estos métodos es el tiempo necesario para realizarlos, el costo y que no pueden ser usados para monitorear la calidad en períodos largos de tiempo. Esto ha hecho a los métodos objetivos atractivos para estimar la calidad percibida en redes de comunicaciones.

Las medidas objetivas de calidad percibida pueden ser intrusivas o no intrusivas. Por intrusvio se entiende inyectar señales extra para estimar la calidad. Los métodos intrusivos son más precisos pero normalmente no son adecuados para monitorear la calidad en servicio, de-

bido a la inyección de señales extra y la necesidad de comparar estas con las señales originales.

Los métodos no intrusivos no requieren de señales extra y son adecuados para monitorear la calidad en servicio. Dependiendo del tipo de entrada al método se pueden clasificar como basados en señales, la entrada es la señal transmitida por la red, o basados en parámetros donde las entradas son parámetros de la red de comunicación y parámetros de la señal en cuestión (por ejemplo la tasa de bits en video).

Los métodos objetivos no dan un resultado directamente en opinión de las personas, sino que su resultado tiene una correlación con la calidad percibida. Esto hace necesaria su calibración en base a los resultados de los métodos subjetivos.

En lo que resta del capítulo se presenta una descripción general de los diferentes tipos de métodos, y luego una descripción detallada de los métodos utilizados en nuestro proyecto.

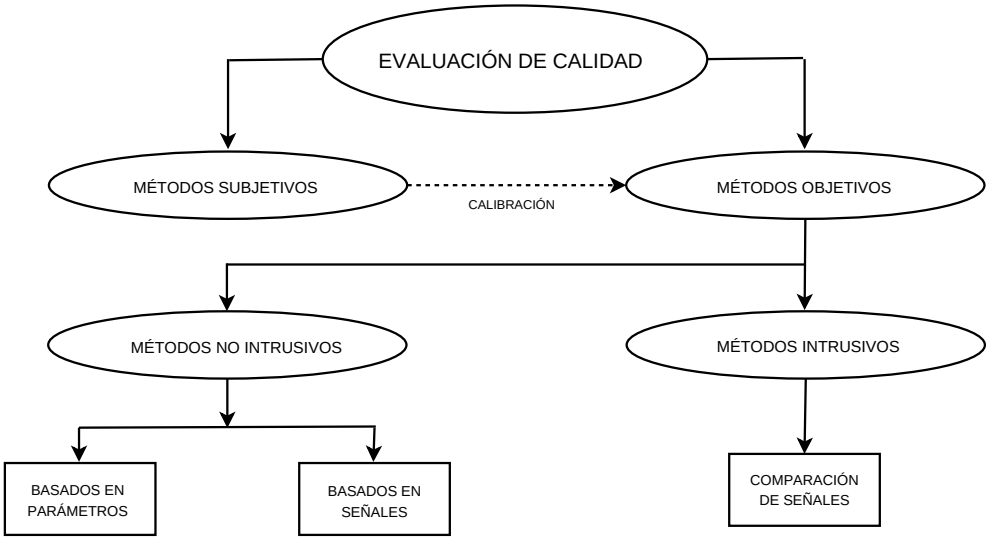


Figura 3.1: Clasificación de los métodos de asignación de calidad

3.2. Metodologías de Estimación de Calidad de Servicio Percibida

3.2.1. Métodos Subjetivos

Los distintos métodos subjetivos están normalizados por la ITU en las recomendaciones ITU BT.500 [15] para video e ITU P.800 [16] para audio.

Básicamente se pueden clasificar en Absolute Category Rating (ACR) que tiene como resultado el Mean Opinion Score (MOS), y Degradation Category Rating (DCR) que tiene como resultado el Degradation Mean Opinion Score (DMOS).

Los distintos tests son realizados normalmente en condiciones controladas en un laboratorio (por ejemplo cuartos aislados acústicamente). También se requiere mucho cuidado en el momento de definir las condiciones e interpretar los resultados.

En el Anexo D se presentan de forma más detallada los distintos métodos subjetivos explicados en las recomendaciones de la ITU, así como el tratamiento de los resultados.

Absolute Category Rating (ACR)

En este tipo de test, los participantes deben asignar un valor global de calidad a la señal[1] que se les presenta (por lo general ya transmitida), sin tener acceso a la señal original. De ahí el nombre de absoluto. Los valores de calidad se asignan de acuerdo a la siguiente tabla.

Valor de calidad	Calidad de la señal
5	Excelente
4	Buena
3	Regular
2	Pobre
1	Mala

El valor medio asignado por los participantes es el MOS. En el caso que se simulen conversaciones (ver Anexo D) el resultado es el MOS_c

[1] las señales pueden ser una imagen, un video o una secuencia de audio

Degradation Category Rating (DCR)

Cuando las señales son de buena calidad, los métodos ACR son insensibles a los pequeños cambios de calidad. En este tipo de caso se utilizan los métodos tipo DCR, en los cuales a los participantes se les presentan dos señales y deben asignar un valor a la degradación de la calidad de una respecto de la otra, de acuerdo a la siguiente escala:

Valor de calidad	Nivel de Degradación
5	Imperceptible
4	Perceptible pero no molesta
3	Un poco molesta
2	Molesta
1	Muy molesta

El valor medio asignado por los participantes es el DMOS.

Existen diversas variantes en este tipo de método dependiendo de las señales que se presenten juntas. Lo más común es presentar la señal original y luego la distorsionada (ya transmitida). Esto permite medir la fidelidad del sistema de comunicación utilizado.

Otros métodos Subjetivos

Se han propuesto recientemente nuevos métodos para una mejor asignación de calidad en servicios multimedia con gran variabilidad de calidad en el tiempo. Para ello se realiza una asignación continua de calidad por parte de los participantes mediante el uso de cursores electrónicos. Por más detalle ver el Anexo D.

3.2.2. Métodos Objetivos

Métodos Intrusivos de estimación de PQoS

Los métodos intrusivos normalmente utilizan dos señales de entrada, una de referencia (original) y una distorsionada (ya transmitida). Son considerados intrusivos debido a la introducción de señales auxiliares y la utilización de la red.

Existe una gran variedad de métodos tanto para audio como para video. Se pueden clasificar en dos grandes grupos. Están los que realizan comparaciones en el dominio del tiempo, como el Signal to Noise Ratio (SNR) o su versión para imagen y video PSNR. Estos métodos son muy simples de implementar, pero la correlación con las medidas subjetivas no es muy buena. El segundo grupo realiza medidas relevantes a la percepción, transformando las

señales al dominio de la percepción utilizando modelos de la percepción humana, ya sea de la audición o de la visión. Estos métodos son más complejos que los primeros pero presentan mejor correlación con los métodos subjetivos.

Métodos típicos de medida en el dominio de la percepción son el Perceptual Speech Quality Measure (PSQM), Measuring Normalizing Blocks (MNB), Enhanced Modified Bark Spectral Distorsion (EMBSD) y Perceptual Evaluation of Speech Quality (PESQ) para audio. Para video existen unos cuantos algoritmos como el basado en el índice de similitud estructural (SSIM), basado en distorsión espacio-temporal del Institue for Telecommunication Science (ITS), pero difieren en que es lo importante a la percepción. Cabe señalar que a diferencia del audio, donde el modelo de percepción esta bastante aceptado, en imágenes y video el conocimiento del mismo es limitado lo que hace que los distintos algoritmos tengan supuestos diferentes sobre la visión.

Algunos de los algoritmos mencionados fueron incluidos en nuestro proyecto. Estos últimos serán analizados con mayor profundidad en la sección 3.3 de audio y 3.4 de video.

El resultado que se obtiene con estos algoritmos es una medida de la distorsión relevante a la percepción. Es necesario llevar estos valores a una escala común, que permita compararlos entre si y que además sea de fácil comprensión para las personas. La escala que surge naturalmente es la utilizada en los métodos subjetivos. Para unificar las escalas se ajusta de forma paramétrica la relación entre valores subjetivos y objetivos de cada método. Por lo tanto es necesaria la realización de test subjetivos para todas las secuencias utilizadas.

En el caso del PESQ, este incluye el ajuste, por lo que la salida se da directamente en escala de DMOS.

Métodos No Intrusivos de estimación de PQoS

A diferencia de los métodos objetivos antes presentados donde el servicio debe ser interrumpido para inyectar las señales, los métodos no intrusivos pueden ser utilizados durante el servicio. Aquí cabe aclarar que no siempre es posible utilizar estos métodos en servicio, debido a que si bien no utilizan señales extra, sí pueden inyectar algún tipo de tráfico para estimar el estado de la red. Este es el caso de lo implementado en nuestro proyecto.

Estos métodos se pueden clasificar en basados en parámetros o basados en señales. Los últimos predicen la calidad utilizando la señal distorsionada sin necesidad de referencia. A este tipo de método se los denomina Null Reference. Los otros predicen la calidad a partir de el valor de parámetros de la red IP (por ejemplo probabilidad de pérdida, jitter, retardo) y de parámetros no específicos de la red (codec utilizado, eco, tasa de bits del video, etc). En nuestro proyecto nos concentramos en este último tipo de métodos. Ejemplo de estos métodos

son el E-Model y el uso de redes neuronales.

El E-Model es un modelo empírico matemático estandarizado por la ITU en la recomendación G.107 [18]. Es un conjunto de formulas que tienen como entrada parámetros de la red tradicional de circuitos conmutados y de la red de paquetes conmutados, y tiene como salida el factor de calidad el cual se puede mapear en MOS_c. Si bien es una herramienta para la planificación de redes, actualmeten es muy utilizada para predecir calidad percibida en VoIP.

Las redes neuronales se utilizan para aproximar la relación no lineal que existe entre calidad percibida (mejor dicho MOS) y el conjunto de parámetros considerado. Un conjunto de parámetros de entrada posible sería el formado por: la probabilidad de pérdida, retardo, jitter, codec utilizado, tasa de bits del video, lenguaje en audio, etc. Para lograr el mapeo deseado se debe generar una base de entrenamiento que consiste en un conjunto de valores de los parámetros y el correspondiente valor de calidad obtenido mediante tests subjetivos. El obtener una buena base de entrenamiento, es decir un rango considerable de variación de los parámetros, es la principal limitante debido al costo de los tests subjetivos.

Tanto el E-Model como las redes neuronales se analizan con mayor profundidad en la sección 3.5.

3.3. Métodos Intrusivos para estimación de PQoS en Audio

En esta sección se describen con mayor profundidad los métodos objetivos utilizados en el proyecto.

La sección comienza con una introducción a los conceptos fundamentales de la psicoacústica, describiéndose luego los métodos particulares.

3.3.1. Principios Psicoacústicos

La Psicoacústica estudia la relación entre los estímulos acústicos y las sensaciones percibidas por el sistema auditivo humano. Se describen a continuación los principios psicoacústicos utilizados. Estos son:

- Bandas Críticas
- Sonoridad

- Enmascaramiento

Bandas Críticas

El concepto de **bandas críticas** se basa en una propiedad singular de nuestro sistema auditivo. Este puede distinguir a bajas frecuencias tonos separados por unos pocos hertz, mientras que a altas frecuencias es necesario que estén separados cientos de hertz para poder diferenciarlos. El espectro audible es entonces "dividido"en bandas independientes que modelan esta diferencia.

La tarea de dividir el espectro es realizada por el oído interno; la energía de frecuencias pertenecientes a diferentes bandas excitan distintas zonas de la membrana basilar. Puede entenderse la banda crítica como la mínima banda de frecuencias alrededor de una frecuencia determinada que activan la misma zona de la membrana basilar. El sistema auditivo periférico puede modelarse como un conjunto de filtros pasabanda donde cada uno tiene una respuesta en frecuencia asimétrica.

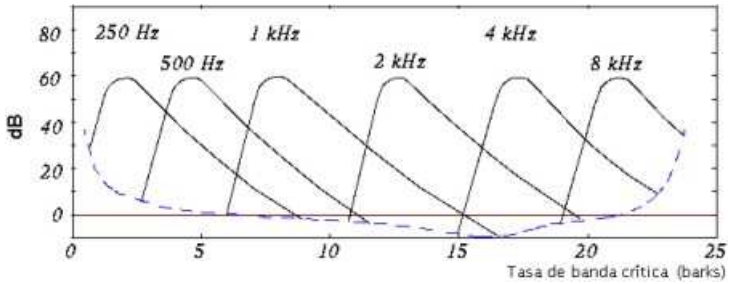


Figura 3.2: Enmascaamiento en frecuencia y bandas críticas.

En las figura 3.2 y 3.3 se pueden observar las bandas críticas, su frecuencia central, el ancho de banda y sus frecuencias de corte laterales. Dada su importancia se definió una unidad para la frecuencia perceptual, el Bark.

Sonoridad

La sonoridad hace referencia a que tan fuerte es un sonido. Para medir la sonoridad se debe responder la siguiente pregunta: ¿cuándo un sonido es igual de fuerte que otro de referencia? Se utiliza como referencia una sinusoide de 1 KHz, y se define que para 1 KHz el nivel de sonoridad (medido en fons) es igual a la nivel de presión sonora (SPL) medido en dB. Para observar la dependencia de la sonoridad con la frecuencia se trazan las curvas de igual sonoridad. Los sonidos sinusoidales contenidos en cada curva de la figura 3.4 tienen la misma sonoridad. Por ejemplo un sonido sinusoidal de 50 Hz y 40 dB de intensidad tiene la

número	frecuencia central (Hz)	banda crítica (Hz)	frecuencia de corte inferior (Hz)	frecuencia de corte superior (Hz)
1	50	-	-	100
2	150	100	100	200
3	250	100	200	300
4	350	100	300	400
5	450	110	400	510
6	570	120	510	630
7	700	140	630	770
8	840	150	770	920
9	1000	160	920	1080
10	1170	190	1080	1270
11	1370	210	1270	1480
12	1600	240	1480	1720
13	1850	280	1720	2000
14	2150	320	2000	2320
15	2500	380	2320	2700
16	2900	450	2700	3150
17	3400	550	3150	3700
18	4000	700	3700	4400
19	4800	900	4400	5300
20	5800	1100	5300	6400
21	7000	1300	6400	7700
22	8500	1800	7700	9500
23	10500	2500	9500	12000
24	13500	3500	12000	15500

Figura 3.3: Escala de Bark.

misma sonoridad que la sinusoide de 1 Khz y 0 dB, igual a 0 fon.

Empíricamente se observa que cada 10 fons la sonoridad se duplica. Se introduce también el concepto de **sone**: decimos que una señal tiene una sonoridad de un sone si es tan fuerte como una sinusoide de 1 KHz a 40 dB, duplicándose cada 10 dB, 50 dB 2 sone, etc.

Enmascaramiento

El enmascaramiento se refiere al efecto psicoacústico que sucede cuando la presencia de un sonido impide la percepción de otro. Un caso de enmascaramiento ocurre cuando dos personas no se escuchan al hablar debido al ruido del trafico en la ciudad; la presencia de este "ruido" produce un desplazamiento del umbral auditivo. Existen tres tipos de enmascaramiento: el pre-enmascaramiento, el enmascaramiento simultáneo y el post-enmascaramiento. El pre-enmascaramiento ocurre cuando el sonido enmascarador es posterior al sonido de prueba (enmascarado); este efecto resulta en la práctica poco relevante ya que tiene una duración del orden de 20 ms. El enmascaramiento simultáneo ocurre cuando los dos sonidos están presentes y el post-enmascaramiento ocurre cuando el sonido de prueba es posterior al enmascarador. Este último se debe al decaimiento de la percepción del sonido enmascarador. Este efecto

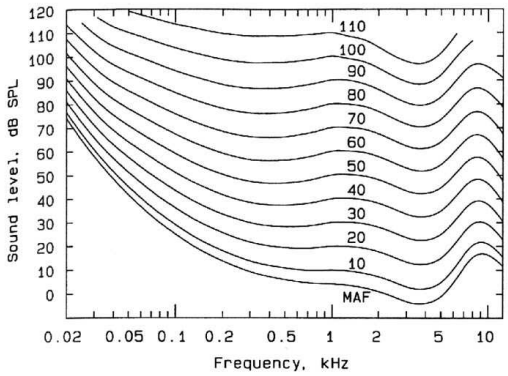


Figura 3.4: Curvas de igual sonoridad

dura entre 100 ms y 200 ms, y depende de si las frecuencias del sonido de prueba y del sonido enmascarador caen o no en la misma banda crítica.

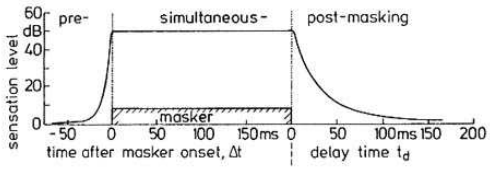


Figura 3.5: Tipos de enmascaramiento

3.3.2. Enhanced Modified Bark Spectral Distortion

El **EMBSD** es un medida objetiva de la degradación de la calidad de la voz que incorpora principios psicoacústicos. La principal hipótesis asumida es que la calidad de la voz está directamente relacionada con la sonoridad. La distorsión perceptualmente relevante por lo tanto va a ser proporcional a la diferencia de sonoridad entre la señal original y la degradada.

Descripción del Algoritmo

El algoritmo divide la señal en "cuadros" de 320 muestras. En caso de que sea un cuadro donde hay voz [2] se calcula la diferencia de sonoridad entre la señal original y la señal

[2]Supone que distorsión en cuadros sin voz no afectan la calidad.

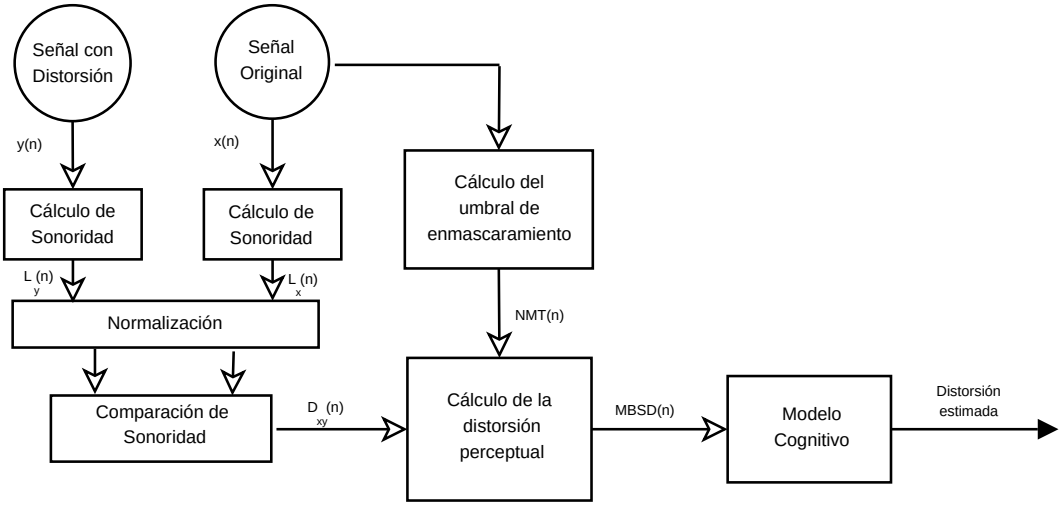


Figura 3.6: Diagrama de Bloques del EMBSD

distorsionada para cada banda crítica. La sonoridad de la diferencia (distorsión agregada) se compara con el umbral de enmascaramiento de ruido (se explicará más en detalle adelante). Si la sonoridad de la distorsión cae debajo del umbral de enmascaramiento, ésta no es tenida en cuenta ya que no es relevante a nivel perceptual. En caso contrario sí es tenida en cuenta. Después es evaluado por el modelo cognitivo que introduce el concepto de post-enmascaramiento. El resultado final del algoritmo es la suma de todas las distorsiones perceptualmente relevantes en cada uno de los cuadros y en cada una de las bandas críticas.

División en Bandas Críticas: el espectro de un cuadro es obtenido utilizando la FFT; éste es dividido en bandas críticas de acuerdo con la figura 3.3. Se tienen en cuenta las bandas de 4 a 18 barks, ya que son las que pertenecen al ancho de banda de interés.

Pre-énfasis: después de dividir el espectro en las correspondientes bandas críticas, el nivel de sonoridad es obtenido utilizando las curvas de igual sonoridad de la fig 3.6. Los datos situados entre dos curvas son interpolados.

Umbral de enmascaramiento de ruido: se mencionaba antes que la distorsión se tiene en cuenta sólo si su sonoridad es mayor al umbral de enmascaramiento de ruido. Existen dos tipos de enmascaramiento; cuando un tono enmascara a un ruido, o cuando un ruido enmascara a un tono, pensando en ruido como un sonido de espectro amplio. Cuando un tono enmascara a un ruido, el umbral está (14.5 + i) dB debajo de la potencia de la señal, siendo i la banda crítica. En el caso de un ruido enmascarando a un tono el umbral se encuentra 5.5 dB por debajo sin importar la frecuencia. Para medir si la señal es más parecida a un tono o a un ruido, se utiliza el SFM (Spectral Flatness Measure). El SFM se

define como el cociente entre la media geométrica del espectro Gm, y la media aritmética Am.

$$SFM_{dB} = 10 \log_{10} \frac{Gm}{Am}$$

Utilizando el SFM calculamos el índice de tonalidad como

$$\alpha = min(\frac{SFM_{dB}}{-60dB}, 1) \tag{3.1}$$

Si el SFM es menor a -60 db estamos frente a una señal muy parecida a un tono, y tomamos $\alpha = 1$ en caso de tener un SFM igual a 0, estamos frente a un señal de espectro plano. Utilizando el índice de tonalidad, calculamos el offset O(i) que es la diferencia entre la potencia de la señal y el umbral de enmascaramiento del ruido en cada banda crítica.

$$O(i) = \alpha(14{,}5 + i) + (1 - \alpha)5{,}5 \tag{3.2}$$

Con la potencia de la señal en la banda y con el offset (O(i)) podemos calcular el umbral de enmascaramiento de ruido . Existe un umbral absoluto, por lo que si para alguna banda el umbral de enmascaramiento es menor que el umbral absoluto, se utiliza el umbral absoluto en vez del calculado.

Modelo Cognitivo

Para crear el modelo cognitivo se tuvieron en cuenta dos principios psicoacústicos:

- El sistema auditivo integra el sonido por un período aproximado de 200 ms.
- El pre-enmascaramiento es relativamente corto, comparado con el efecto del post enmascaramiento.

Se define un segmento cognitivo como los cuadros sucesivos correspondientes a 200 ms. La distorsión perceptual P(j) se define como el máximo valor de la distorsión en un segmento cognitivo. La distorsión por post-enmascaramiento Q(j) se define como una fracción (80 %) de la distorsión del segmento cognitivo anterior . Considerando cuadros de 320 muestras que es aproximadamente 20 ms (8KHz), se tiene que un segmento cognitivo son 10 cuadros. Para el calculo del EMBSD: miramos el máximo de la distorsión en 10 cuadros consecutivos que supere al umbral de enmascaramiento. Lo comparamos con la distorsión del segmento cognitivo anterior, el máximo de los dos va a ser la distorsión en este segmento. El valor final se obtiene sumando estas distorsiones para todos los segmentos de la señal.

3.3.3. Perceptual Evaluation of Speech Quality (PESQ)

Introducción

El algoritmo PESQ compara la señal original $X(t)$ con la señal distorsionada $Y(t)$, que es el resultado de enviar $X(t)$ por un sistema de comunicación. El resultado del algoritmo es una predicción de la calidad subjetiva de $Y(t)$. La idea básica es mapear la señal original y la señal distorsionada en una representación interna utilizando un modelo perceptual. Las diferencias en esta representación de las señales son utilizadas por un modelo cognitivo para predecir la calidad subjetiva de la señal degradada. El aspecto más destacable de PESQ es que incorpora un mecanismo de alineación entre las señales, para la detección de retardos variables.

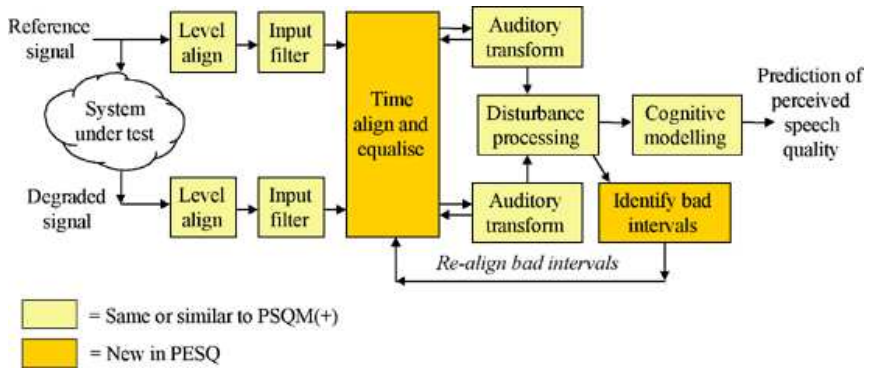


Figura 3.7: Funcionamiento de PESQ

Descripción del Algoritmo

La recomendación ITU-T P.862 define el algoritmo PESQ, que introduce mejoras significativas sobre su predecesor PSQM (ITU-T P.861). Las etapas involucradas se detallan a continuación.

Calibración: La primer etapa del algoritmo es compensar la ganancia del sistema bajo prueba. Ambas señales, la señal original $X(t)$ y la señal distorsionada $Y(t)$ son amplificadas para que tengan un nivel constante de energía. La banda de interés está entre los 300 Hz y los 3000 Hz, que corresponde con el ancho de banda de la telefonía. Además de alinear los niveles en el tiempo, es necesario compensar los niveles en el dominio de la frecuencia. Se "enventana" la señal y se normaliza en frecuencia.

Filtrado: En esta etapa se tienen en cuenta las características en las que se realizan los tests. Se filtra la señal simulando la respuesta del "tubo". Se implementa en el dominio de la frecuencia, transformando las señales por medio de la FFT, y aplicándole un filtro lineal a tramos. Se le aplica la FFT inversa para volver al dominio del tiempo.

Alineación Temporal: La rutina de alineación temporal provee una estimación del retardo entre las señales. El modelo perceptual compensa posibles variaciones en el retardo del sistema en evaluación utilizando esta información. El proceso de alineación temporal consta de las siguientes etapas.

- Estimación del retardo basado en la envolvente, utilizando toda la señal.
- División de la señal original en subsecciones básicas de voz, llamadas "utterances"[3]
- Estimación del retardo basado en la envolvente para cada "utterance"
- Ajuste fino del retardo basado en la correlación con precisión cercana a la muestra.
- División de la señal en "utterances", para realinearlos.
- Después del modelo perceptual se identifican secciones largas , con errores grandes, en busca de errores en la alineación

Transformación al dominio de la audición: En esta parte del proceso es donde se incluye la percepción humana. Las dos señales son transformadas teniendo en cuenta sus aspectos psicoacústicos. El resultado es una representación de la señal teniendo en cuenta su sonoridad en frecuencia, y el tiempo. A esta representación se le llama la superficie sensitiva.

Procesamiento de las Alteraciones: Las diferencias audibles entre la señal original y la distorsionada son divergencias entre las superficies sensitivas. Se calculan a partir de estas superficies alteraciones absolutas y alteraciones aditivas.

Mapeo del resultado crudo PESQ en un resultado MOS-LQO: El algoritmo antes descripto da como resultado un valor proporcional a la distorsión perceptual relevante entre ambas señales. Es deseable obtener un valor MOS-LQO (Mean Opinión Score-Listening Quality Objective) a partir de PESQ que nos permita una comparación lineal con el MOS. La recomendación ITU-T P.862.1 especifica una función 3.3 para realizar este mapeo.

$$y = 0{,}999 + \frac{4{,}999 - 0{,}999}{1 + e^{(} - 1{,}4945 * x + 4{,}66607)} \tag{3.3}$$

[3]Utterance: unidad natural del habla, separada por silencios o respiraciones.

Este mapeo fue realizado utilizando una base de datos con aplicaciones de todo tipo. Con otro tipo de funciones optimizadas para un tipo especifico de datos se podrían obtener mejores resultados para ese conjunto de datos particular.

3.3.4. Measuring Normalizing Blocks (MNB)

El algoritmo Measuring Normalizing Blocks fue desarrollado en el año 1998 por Stephen Voran. Al igual que PESQ y EMBSD se utiliza para cuantificar la calidad percibida de la voz al pasar por un sistema de comunicación. Las investigaciones recientes incluyen modelos de la percepción de la audición humana, tratan de obtener un estimador que "escuche"las señales de la misma forma que los seres humanos lo hacen. En [3] se hace incapié en que si bien en la teoría es un gran avance, en la practica cuando son evaluados estos estimadores muestran pocas ventajas. Voran afirma que la causa del pobre desempeño de estos sistemas se debe a dos posibles causas. La primer causa es que a pesar de que se conocen modelos detallados para los umbrales de la audición y sonoridad para distintas combinaciones de tonos puros y ruidos, las características de la audición, tales como la no linealidad y la variación temporal, hacen que conjugar esos resultados en un modelo para señales generales sea una tarea por lo menos difícil. Es necesario hacer simplificaciones, resultando en modelos moderadamente complejos, que no se adaptan adecuadamente a este tipo de señales. Como segundo punto señala que la percepción humana está compuesta por dos aspectos, la audición y la capacidad de juzgar . Muchos esfuerzos se hicieron para modelar la audición, seguidos de un modelo de la capacidad de juzgar relativamente trivial. El enfoque que se da en [3] es tener un modelo de la audición simple pero efectivo, seguido de un modelo más sofisticado para la capacidad de juzgar.

Descripción del Algoritmo

El primer paso es transformar las señales al dominio de la percepción, de modo de considerar las propiedades de nuestro sistema auditivo. Observando la correlación con los tests subjetivos el autor llega a la conclusión de que los modelos más complejos tienen muy pocas ventajas en comparación con los más simples; es por esto que sólo se tienen en cuenta para el modelo de la percepción la resolución no uniforme en frecuencia (escala de Bark) y la no linealidad en la percepción de la sonoridad. Nuestro sistema auditivo reacciona y se adapta distinto a distorsiones que abarcan distintas escalas en frecuencia y tiempo. Además una estimación perceptualmente consistente debe cubrir varias escalas de tiempo y frecuencia. Las distorsiones a una escala se deben remover para no ser tomadas en cuenta, a otras escalas. Trabajar partiendo de escalas "grandes"hacia escalas "pequeñas"emula patrones de adaptación y reacción a las distorsiones. Teniendo en cuenta lo anterior se definen dos estructuras jerárquicas formadas (MNB-I y MNB-II) por bloques de medida normalizadores en el dominio del tiempo y la frecuencia.

Cada uno de esos bloques tiene como entrada la señal original $X(f,t)$ y la señal distorsionada $Y(f,t)$ en el dominio de la percepción, y devuelve como salida un set de medidas, y una versión normalizada de $Y(f,t)$, $\tilde{Y}(f,t)$. Como se observa en la figura. 3.8 el bloque

TMNB calcula la diferencia en frecuencia, mide la distorsión perceptuales de esa diferencia en el tiempo y normaliza la señal distorsionada. La normalización es necesaria para que los bloques que siguen en la jerarquía, no tomen en cuenta la distorsión medida por este bloque.

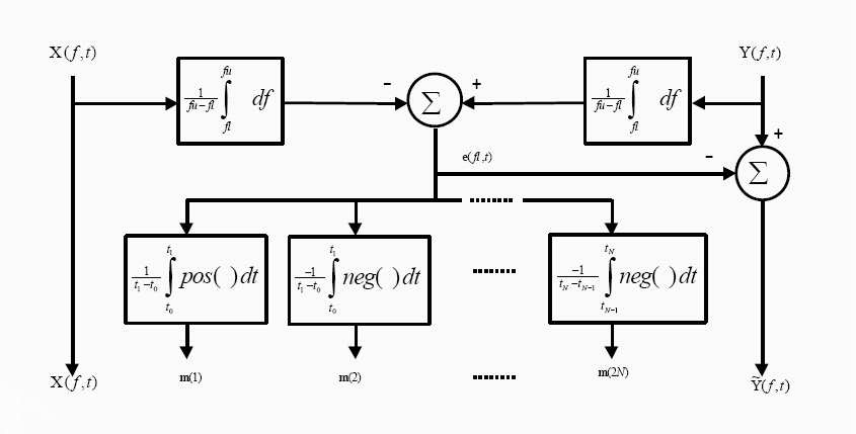


Figura 3.8: Estructura básica TMNB

FMNB Análogamente un bloque FMNB calcula las diferencias en el tiempo, mide la distorsión perceptual de la diferencia en frecuencia y normaliza la señal distorsionada. Al igual que en el TMNB es necesario normalizar la señal distorsionada para que los bloques que estén a continuación en la jerarquía no tengan en cuenta la distorsión ya medida.

Ambos bloques fueron diseñados para ser idempotentes lo que significa:

$$MNB(X,Y) = (X,\tilde{Y},m) \Longrightarrow MNB(X,\tilde{Y}) = (X,\tilde{Y},0) \tag{3.4}$$

Esto significa que una segunda pasada por un bloque MNB dado, no va a alterar la señal de salida, y el vector de medidas va a valer cero. La idempotencia de los bloques MNB es lo que permite ponerlos en cascada y asegurar que midan la distorsión en una determinada escala de tiempo o frecuencia una única vez.

El autor propone dos estructuras que representan un punto medio para el compromiso entre complejidad y performance. En la figura 3.10 se observa el detalle de la estructura MNB-I

Esta estructura da como resultado un vector m de 12 elementos. La combinación lineal de dichos elementos es un buen estimador para la distancia perceptual de dos señales de voz. Este valor es llamado distancia auditiva ("auditory distance"AD):

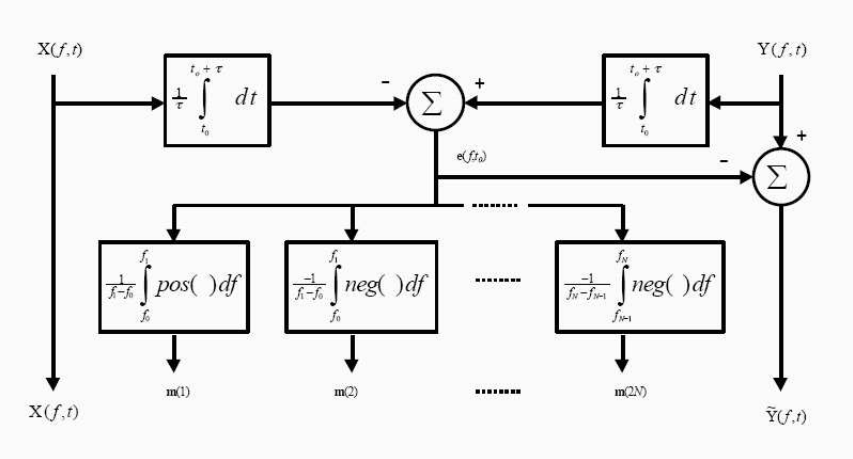


Figura 3.9: Estructura básica FMNB

$$AD = w^T.m \tag{3.5}$$

donde w es un vector que pondera los elementos del vector m.

Función Logística: la distancia auditiva AD es una estimación de la calidad percibida de la voz. Los tests subjetivos cubren un rango finito de valores, en el caso del MOS los valores posibles van de 1 a 5. Para aumentar la correlación con esos valores se mapea la distancia auditiva AD en un rango finito de valores. El autor utiliza la función logística con asíntotas 0 y 1.

$$L(z) = \frac{1}{1 + e^{az+b}} \tag{3.6}$$

3.4. Métodos Intrusivos para estimación de PQoS en Video

En esta sección se presentan los algoritmos implementados para estimar la degradación sufrida por un video al transmitirse por la red. Dichos algoritmos se basan en la comparación del video original con el distorsionado. De acuerdo a la clasificación dada en la introducción del capitulo son algoritmos objetivos intrusivos.

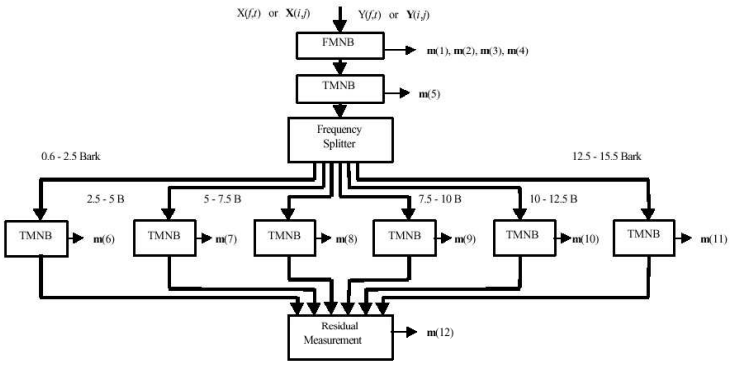


Figura 3.10: detalle de la Estructura MNB-I

3.4.1. Error cuadrático medio (MSE) y relación señal a ruido de pico (PSNR)

Se definen el MSE y PSNR como:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{3.7}$$

$$PSNR = 10\log_{10}\left(\frac{L^2}{MSE}\right) \tag{3.8}$$

donde N es el número de pixels de la imagen o video, x_i y y_i son el i-ésimo pixel de la imagen original y distorsionada respectivamente, y L es el rango de valores que toman los pixels (rango dinámico, por ejemplo en una imagen monocromática de 8 bits L=255).

Estos métodos de asignación de calidad han sido los más utilizados durante años debido a su simplicidad matemática, de cálculo y fácil interpretación física. Sin embargo han sido criticados debido a su baja correlación con los métodos subjetivos [7], esto es con la forma en que los humanos percibimos la calidad. En las siguientes figuras se muestran distintos tipos de errores, con efectos notoriamente diferente en la calidad percibida, pero con valores de MSE casi idénticos.

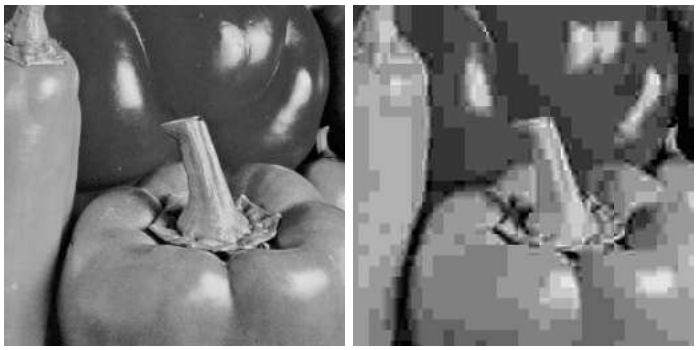


Figura 3.11: izquierda imagen original, derecha compresión JPEG; MSE=160

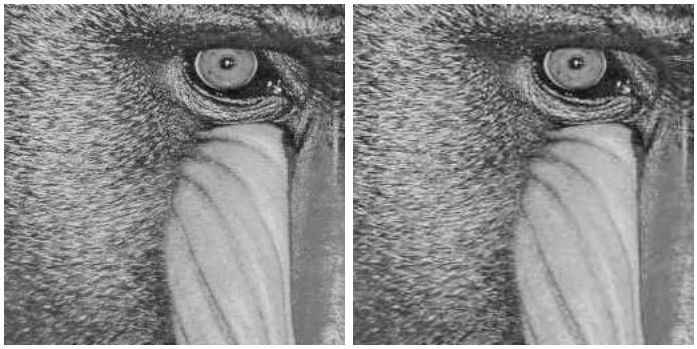


Figura 3.12: izquierda imagen original, derecha compresión JPEG; MSE=159

3.4.2. Métricas de distorsión espacio-temporales

Este algoritmo fue creado por el Institute for Telecommunication Sciences (ITS), presentado en [8]. El objetivo fue crear métricas que fuesen útiles para distintos rangos de calidad de video, y que a su vez fuesen eficientes en cuanto a la cantidad de información extra a ser enviada con el video, permitiendo su utilización en monitoreo de calidad punta a punta. En un principio no sería necesario enviar el video original completamente, sólo una cierta cantidad de valores. Para ello se toman ciertas zonas espacio-temporales (E-T) del video (las mismas en el original y el transmitido) y sobre ellas se calculan ciertos parámetros. Dichos parámetros son transmitidos por un canal auxiliar de forma de poder compararlos del lado receptor y obtener así una estimación de la degradación sufrida durante la trasmisión. Dependiendo del tamaño y cantidad de zonas seleccionadas es la reducción que se obtiene en el ancho de banda (en comparación a tener que enviar todo el video original).

El algoritmo se basa en medir cambios en la actividad espacial, tomando el gradiente como medida de ella. Se toma en cada frame un cierto número de regiones E-T, y se calcula el gradiente de la componente de luminancia en cada una de ellas. Utilizando el módulo del gradiente en cada región E-T y se calculan distintas estadísticas (como desviación estándar, valores medios, etc) que cuantifican la actividad espacial como función de la orientación angular. Luego se comparan los valores del video original y distorsionado para cada zona E-T obteniéndose una medida de la distorsión en cada E-T. Dichas medidas se combinan luego para obtener una medida de la distorsión total del video.

Implementación

La implementación se realizó acorde a lo sugerido en [8]. Para el filtro detector de bordes se utilizó una máscara horizontal (y una vertical) de 13x13 pixels que se comporta como un filtro pasabanda horizontal y pasabajos vertical, con pesos dados por:

$$w_x = k.\left(\frac{x}{c}\right).\exp\left[-\frac{1}{2}\left(\frac{x}{c}\right)^2\right] \tag{3.9}$$

donde x es el desplazamiento del pixel considerado desde el centro del filtro, c es una constante que determina el ancho del filtro pasabanda y k es una constante de normalización. En nuestro caso tomamos $c = 2$ y $k = 1$. Una vez filtrado el video original y distorsionado se divide cada uno en regiones E-T ("cubos"). Nuestra implementación del algoritmo toma regiones de tamaño 8 pixels horizontales x 8 pixels verticales x 6 frames, que según [8] es el tamaño óptimo para propósitos general (diferentes tasa de bits, etc). Una vez obtenidas las regiones E-T se procede a calcular los distintos parámetros. Para ello, primero se calcula en cada pixel (fila i, columna j, tiempo t) de las zonas E-T el módulo y fase del gradiente, definido como:

$$R(i,j,t) = \sqrt{H(i,j,t)^2 + V(i,j,t)^2} \tag{3.10}$$

$$\theta(i,j,t) = \arctan \frac{V(i,j,t)}{H(i,j,t)} \tag{3.11}$$

donde H y V son el resultado de aplicar al video el filtro definido por 3.9 y el transpuesto, respectivamente.

El primer parámetro f_1 se define como la desviación estándar de $R(i,j,t)$ calculada sobre cada región, obteniéndose una secuencia $f_1(s,t)$ donde s es la coordenada espacial y t la temporal. Los parámetros se calculan con un umbral de percepción (si son menores que un cierto valor P se les asigna el valor P), en el caso de f_1 se tomó $P = 12$. Este parámetro es sensible a la cantidad de actividad espacial dentro de cada región E-T, la presencia de blurring tiende a disminuir su valor, mientras que el ruido tiende a aumentarlo.

El otro parámetro, f_2 que definiremos más adelante, es sensible a la variación en la orientación de la actividad espacial. La imagen con gradientes horizontales y verticales HV, contiene los pixels $R(i,j,t)$ que son bordes horizontales y verticales (los bordes diagonales son igualados a cero). La imagen con gradientes diagonales, $\overline{HV}$, contiene los pixels $R(i,j,t)$ que son bordes diagonales (los bordes horizontales y verticales son igualados a cero)(ver figura 3.13). Los valores del módulo del gradiente menores a un cierto r_{min} son igualados a cero con el fin de asegurar valores de θ correctos.

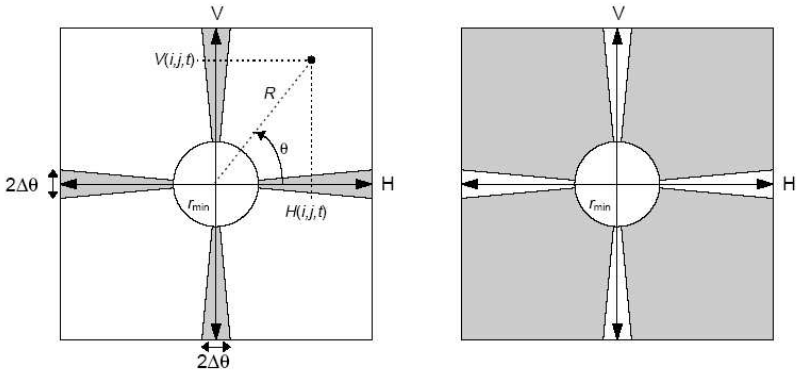


Figura 3.13: izquierda HV, derecha $\overline{HV}$. Las zonas grises indican los pixels con valor diferente de cero.

El valor de los pixels de HV y $\overline{HV}$ se pueden expresar como:

$$HV(i,j,t) = \begin{cases} R(i,j,t) & \text{si } R(i,j,t) \geq r_{min} \text{ y } m\frac{\pi}{2} - \Delta\theta < \theta(i,j,t) < m\frac{\pi}{2} + \Delta\theta \text{ (con } m = 0,1,2,3) \\ 0 & \text{de otra forma.} \end{cases} \tag{3.12}$$

$$\overline{HV}(i,j,t) = \begin{cases} R(i,j,t) & \text{si } R(i,j,t) \geq r_{min} \text{ y } m\frac{\pi}{2} + \Delta\theta \leq \theta(i,j,t) \leq (m+1)\frac{\pi}{2} - \Delta\theta \text{ (con } m = 0,1,2,3) \\ 0 & \text{de otra forma.} \end{cases} \tag{3.13}$$

Siguiendo la recomendación de [9] y al igual que lo implementado en [8], tomamos $r_{min} = 20$ y $\Delta\theta = 0{,}05236$ radianes. Finalmente se calcula f_2 como

$$f_2 = \frac{mean\left[HV(i,j,t)\right]}{mean\left[\overline{HV}(i,j,t)\right]} \tag{3.14}$$

El valor medio se calcula en cada región E-T obteniéndose una secuencia $f_2(s,t)$ donde s es la coordenada espacial y t la temporal. Los valores medios se calculan con un umbral de percepción que en nuestro caso es igual a 3. Como mencionamos antes f_2 es sensible a cambios en la orientación del gradiente. Por ejemplo si los bordes verticales u horizontales sufren más blurring que los diagonales el valor de f_2 de la E-T distorsionada es menor que el de la original, en cambio si aparecen bordes verticales u horizontales (blocking en la codificación jpeg, pérdida de paquetes) el valor de f_2 para el video distorsionado es mayor que para el video original.

Una vez obtenidos los parámetros se procede a compararlos para determinar la distorsión sufrida por el video. Para esto se definen la ganancia y pérdida (gain y loss) de actividad espacial, como:

$$gain(s,t) = \left[\log_{10}\frac{f_{out}(s,t)}{f_{in}(s,t)}\right]^{+} \tag{3.15}$$

y

$$loss(s,t) = \left[\frac{f_{out}(s,t) - f_{in}(s,t)}{f_{out}(s,t)}\right]^{-} \tag{3.16}$$

donde

$$[x]^{+} = max[x,0] \quad y \quad [x]^{-} = min[x,0] \tag{3.17}$$

Una vez obtenidos los valores de gain y loss se toma un promedio con los peores casos para cada frame resultando en una secuencia temporal de gain y loss. En nuestro caso tomamos el promedio con el 5 % de los peores casos. Tomando el promedio de la secuencia temporal se obtienen valores de gain y loss para todo el video.

Por último se toma una combinación lineal de gain y loss de f_1 y f_2 como estimación de la degradación percibida. La combinación es tal que el resultado es siempre menor o igual a cero, donde cero significa no degradación.

3.4.3. Calidad de Video basada en la distorsión estructural

La mayoría de los algoritmos para asignación de calidad consideran la imagen distorsionada como la imagen de referencia (perfecta) superpuesta con una imagen de ruido. La idea de estos algoritmos es evaluar cuán perceptible es la señal de error para el sistema visual humano (HVS según la sigla en inglés). El problema es que el HVS es un sistema sumamente complicado y el conocimiento existente sobre el mismo es limitado. En [10, 11, 12] se introdujo una "nueva filosofía" en el diseño de métricas de calidad:
" La función principal del sistema visual humano es extraer información estructural del campo visual, y el sistema visual humano está altamente adaptado para este propósito. Por lo tanto, una medida de la distorsión estructural debería ser una buena aproximación a la distorsión percibida en una imagen".

Cuando se habla de estructura lo que se quiere decir es que existe una dependencia entre las muestras de una imagen, que aumenta a medida que estas muestras están más cercanas. A diferencia de los métodos basados en el HVS que considera diferencias pixel a pixel, en este método se considera el entorno que rodea al pixel. Otra ventaja del presente método es que se independiza (casi) del HVS y sus problemas actuales.

Índice de Similitud Estructural

Se pueden admitir diferentes implementaciones de esta "nueva filosofía" dependiendo de como se interpreten los conceptos de información y distorsión estructural. Siguiendo lo propuesto por [10], se considera información estructural aquellos atributos que reflejan la estructura de los objetos independiente del nivel de luminancia y contraste de la imagen. Por lo tanto obtenemos una asignación de calidad que separa la medida de distorsión de luminancia, contraste y estructural.

En [11] se propuso un algoritmo que calcula el "índice de similitud estructural". Consideremos dos señales $\mathbf{x}$ e $\mathbf{y}$, positivas, alineadas entre si (por ejemplo dos imágenes a ser comparadas); y sean μ_x, μ_y, σ_x^2, σ_y^2 y σ_{xy} la media de $\mathbf{x}$, la media de $\mathbf{y}$, la varianza de $\mathbf{x}$, la varianza de $\mathbf{y}$ y la covarianza de $\mathbf{x}$ e $\mathbf{y}$, respectivamente. El valor medio y la desviación estándar se consideran como estimativos de la luminancia y el contraste de la imagen respectivamente. La covarianza se puede pensar como una medida de cuánto cambia una imagen de forma no lineal respecto a la otra. Para comparar la luminancia, contraste y estructura de dos imágenes se definen las siguientes magnitudes:

$$l(\mathbf{x},\mathbf{y}) = \frac{2\mu_x\mu_y}{\mu_x^2+\mu_y^2}, \quad c(\mathbf{x},\mathbf{y}) = \frac{2\sigma_x\sigma_y}{\sigma_x^2+\sigma_y^2}, \quad e(\mathbf{x},\mathbf{y}) = \frac{\sigma_{xy}}{\sigma_x\sigma_y} \tag{3.18}$$

Los dos primeros términos dependen exclusivamente de la luminancia y contraste de las imágenes comparadas, mientras que un cambio en la luminancia y/o el contraste no tiene ningún impacto en el tercero. Geométricamente $e(\mathbf{x},\mathbf{y})$ corresponde al coseno del ángu-

lo entre los vectores $\mathbf{x} - \mu_x$ y $\mathbf{y} - \mu_y$, y por lo tanto independiente del módulo. Cuando $(\mu_x^2 + \mu_y^2)(\sigma_x^2 + \sigma_y^2) \neq 0$, el índice de similitud se define como:

$$S(\mathbf{x}, \mathbf{y}) = l(\mathbf{x}, \mathbf{y}) \cdot c(\mathbf{x}, \mathbf{y}) \cdot e(\mathbf{x}, \mathbf{y}) = \frac{4\mu_x \mu_y \sigma_{xy}}{(\mu_x^2 + \mu_y^2)(\sigma_x^2 + \sigma_y^2)} \tag{3.19}$$

Un problema que presenta (3.19) es que cuando $(\mu_x^2 + \mu_y^2)$ o $(\sigma_x^2 + \sigma_y^2)$ son cercanos a cero, la medida resulta inestable. Para evitar este problema se define el índice de similitud estructural (SSIM según la sigla en inglés) entre $\mathbf{x}$ e $\mathbf{y}$ como:

$$SSIM(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x \mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{3.20}$$

donde C_1 y C_2 son constantes definidas como:

$$C_1 = (K_1 L)^2 \quad y \quad C_2 = (K_2 L)^2, \tag{3.21}$$

con L rango dinámico de los valores de los pixels, K_1 y K_2 constantes pequeñas de forma tal que C_1 y C_2 tengan efecto sólo cuando $(\mu_x^2 + \mu_y^2)$ o $(\sigma_x^2 + \sigma_y^2)$ son chicos. De acuerdo con [10] el resultado de (3.20) no depende significativamente del valor de las constantes.

El SSIM así definido tiene las siguientes propiedades:

1. $SSIM(\mathbf{x}, \mathbf{y}) = SSIM(\mathbf{y}, \mathbf{x})$
2. $SSIM(\mathbf{x}, \mathbf{y}) \leq 1$
3. $SSIM(\mathbf{x}, \mathbf{y}) = 1$ si y solo si $\mathbf{x} = \mathbf{y}$

Basados en la "nueva filosofía", si consideramos que una de las imágenes tiene calidad perfecta, el SSIM da una medida cuantitativa de la distorsión presente en la otra imagen. La distorsión es nula si ambas imágenes son iguales ($SSIM = 1$), y cuanto menor sea el valor de SSIM mayor la distorsión.

El índice SSIM se calcula para cada pixel utilizando una ventana cuadrada de un número fijo de pixels (8 en este caso). El valor total de calidad para la imagen se obtiene promediando los valores del SSIM en cada pixel. También es posible utilizar funciones más complejas para obtener valores totales de calidad, por ejemplo tomar el promedio ponderando los pixels dependiendo de la posición en la imagen.

Calidad de Video

Básicamente lo que se hace es calcular un SSIM para cada frame del video, combinando luego dichos valores para obtener una medida de calidad para todo el video. Para ello se

toman un número (R_s) de ventanas de 8×8 pixels por frame de manera aleatoria. De esta forma se reduce significativamente la cantidad de cálculo a realizar, por ejemplo tomando R_s= 0.2 % del número total de ventanas posibles (esto es moviendo pixel por pixel) se obtiene una variación en el resultado del orden del 1 %. Se aplica el algoritmo de forma independiente a las componentes Y, Cb y Cr y luego se combinan en un valor local usando una suma ponderada. Llamemos $SSIM_{ij}^{Y}$, $SSIM^{C}b_{ij}$ y $SSIM^{C}r_{ij}$ al valor de SSIM de la componente Y, Cb y Cr de la j-ésima ventana del i-ésimo frame del video. El índice local está dado entonces como

$$SSIM_{ij} = W_Y SSIM_{ij}^{Y} + W_{Cb} SSIM^{C}b_{ij} + W_{Cr} SSIM^{C}r_{ij} \tag{3.22}$$

donde los valores considerados fueron $W_Y = 0{,}8$, $W_{Cb} = 0{,}1$ y $W_{Cr} = 0{,}1$, los mismos que [10]. Una vez obtenidos los valores locales, se combinan para obtener un índice de calidad del frame en cuestión, usando

$$Q_i = \frac{\sum_{j=1}^{R_s} w_{ij} SSIM_{ij}}{\sum_{j=1}^{R_s} w_{ij}} \tag{3.23}$$

donde Q_i representa el índice de calidad del i-ésimo frame, y w_{ij} es el peso dado a la j-ésima ventana del i-ésimo frame.

Finalmente, se obtiene el valor de calidad para todo el video:

$$Q = \frac{\sum_{i=1}^{F} W_i Q_i}{\sum_{i=1}^{F} W_i} \tag{3.24}$$

donde F es el largo en frames del video y W_i es el peso asignado al i-ésimo frame. Para elegir los pesos los factores psicológicos son importantes, y aquí entra de nuevo el HVS. De acuerdo con [10] se consideraron dos métodos. El primero se basa en que zonas oscuras llaman menos la atención que las claras. Utilizando el valor medio μ_x de la componente Y como estimativo de la luminancia, se toma el peso local como:

$$w_{ij} = \begin{cases} 0 & \text{si } \mu_x \leq 40 \\ (\mu_x - 40)/10 & 40 < \mu_x \leq 50 \\ 1 & \mu_x > 50 \end{cases} \tag{3.25}$$

El segundo ajuste tiene en cuenta si existe mucho movimiento en el video. Esto tiene en cuenta que la distorsión percibida depende del movimiento presente en el video. Para esto se calcula el vector de movimiento de cada ventana, buscando la ventana en el frame siguiente que maximice la correlación con la primera. Si m_{ij} representa el módulo del vector de movimiento de la j-ésima ventana del i-ésimo frame, el nivel de movimiento del frame i se estima como

$$M_i = \frac{\sum_{j=1}^{R_s} m_{ij}}{R_s K_M} \tag{3.26}$$

donde la constante K_M sirve como normalización del nivel de movimiento ($K_M = 16$). El peso de cada frame es

$$W_i = \begin{cases} \sum_{j=1}^{R_s} m_{ij} & \text{si } M_i \leq 0{,}8 \\ (1{,}2 - M_i)/0{,}4 & 0{,}8 < M_i \leq 1{,}2 \\ 0 & M_i > 1{,}2 \end{cases} \tag{3.27}$$

3.4.4. Verificación de los algoritmos de video

Para verificar la consistencia de los algoritmos implementados se utilizó una serie de videos, disponibles en [13], que consisten en varios videos de referencia con distinta calidad (desde 50Mb/s a 700 Kb/s), y por cada uno de estos 16 videos con diferentes tipos de distorsión. A los videos distorsionados se le realizaron test subjetivos de acuerdo con [15]. Esta secuencia de videos surge en el marco de un proyecto llevado acabo por el Video Quality Expert Group (VQEG) en el cual se pretendía estandarizar (generar una ITU Recommendation) la estimación objetiva de calidad de video([14]). Básicamente el proyecto consistió en evaluar una cierta cantidad de métodos propuestos por distintos grupos, y dependiendo del resultado recomendar uno o varios métodos para la estimación objetiva. Dentro de estos métodos esta el Peak Signal to Noise Ratio y el algoritmo basado en métricas de distorsión espacio-temporales propuesto por el ITS. La correlación de los resultados subjetivos con los objetivos se estudió mediante cuatro métricas diferentes (ver [14]), luego de ajustar el mapeo entre valores objetivos y subjetivos utilizando una función logística de cuatro parámetros libres. La conclusión a la que llegaron es que salvo un par de métodos (cuya desempeño fue peor), todos son estadísticamente equivalentes. Por lo tanto no se llegó a ninguna recomendación.
Utilizando los algoritmos con estos videos (solamente 40) se observa que:

- El algoritmo que presenta una mejor correlación entre valores subjetivos y objetivos es el basado en el índice de similitud estructural (SSIM), como se puede ver claramente a partir de las gráficas de valores subjetivos contra objetivos, ver figura 3.14.

- El algoritmo presentado por el ITS requiere más tiempo de cómputo que los otros dos. Para ser más precisos demora 67 minutos (implementación en Java) en asignar la calidad a un video, contra 5 minutos que demoran los otros. Esto es una característica poco deseable teniendo en cuenta la forma en que se utilizan los mismos.

A medida que se realizaron las distintas pruebas se hicieron dos modificaciones al algoritmo basado en el SSIM. Primero, observamos que asignar el peso a cada frame utilizando la cantidad de movimiento (3.26), hace que el resultado varíe en el orden del 10 % de una corrida a otra. Eliminando esto se observa que: el resultado no cambia de manera significativa, se eliminan las variaciones, y además se reduce el tiempo de cálculo. La segunda modificación consistió en que en vez de calcular el SSIM para el video como el promedio de los SSIM de cada frame, se toma el promedio entre el x % de los frames con valores más bajos de SSIM. Esto disminuyó el error cuadrático, no muy significativamente, en el ajuste entre resultados

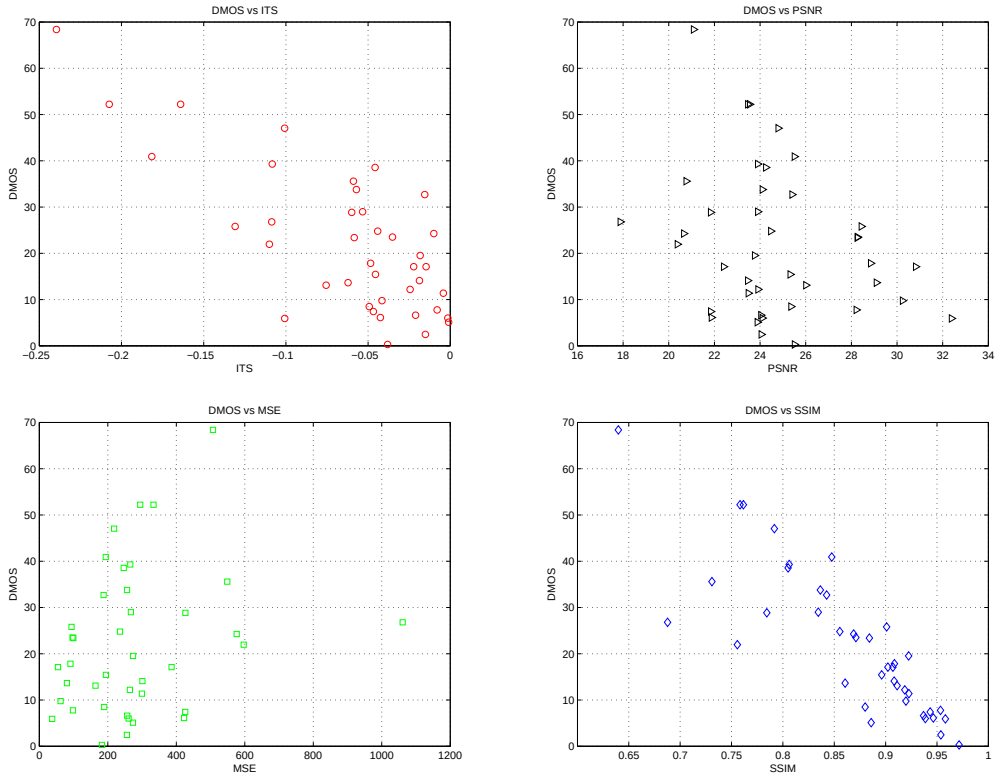


Figura 3.14: Arriba izquierda DMOS vs ITS, arriba derecha DMOS vs PSNR, abajo izquierda DMOS vs MSE y abajo derecha DMOS vs SSIM

subjetivos y objetivos. En la figura 3.15 se muestra el resultado del SSIM considerando el promedio de todos los frames y del 5 % de los peores casos.

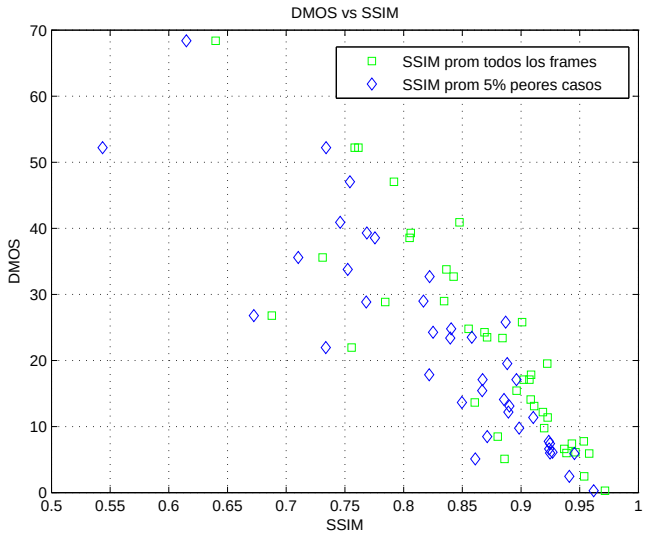


Figura 3.15: DMOS vs SSIM como el promedio de todos los frames y el 5 % de los peores casos

3.5. Métodos no Intrusivos para estimación de PQoS en audio y video

3.5.1. Random Neuronal Networks

Otra de las técnicas de estimación de PQoS utilizadas en el proyecto consistió en la implementación de un sistema de aprendizaje supervisado constituido por una red neuronal. La idea de este sistema es mantener por un lado las características de automatización de las herramientas antes descritas, utilizando a su vez un algoritmo inteligente de clasificación que reproduzca el comportamiento humano y que haga posible el uso de técnicas *no intrusivas* para dicha estimación (el concepto de técnicas no intrusivas hace alusión a la descripción brindada en la sección 3.2.2, aunque dependerá de la cantidad de tráfico de test que se utilice para estimar el estado de la red). Una de las principales virtudes de este tipo de sistemas es su habilidad para adaptarse a las condiciones cambiantes del medio como es el caso de la dinámica de las redes IP; esta capacidad de adaptación se logra mediante una fase de entrenamiento de la red de neuronas.

Las principales aplicaciones de las redes neuronales pueden clasificarse en tres grandes categorías:

- Reconocimiento de patrones.
- Sistemas predictivos de clasificación.
- Aplicaciones de control y optimización.

En este proyecto se decidió utilizar un modelo particular de red neuronal conocido como *red neuronal aleatoria (Random Neuronal Network, RNN)*, denominada también *G-Network* en honor a su creador Erol Gelenbe. Motivó esta decisión el gran éxito obtenido en la aplicación del mismo en varios campos de la ingeniería, incluyendo la resolución de problemas de optimización NP completos (Non-deterministic Polynomial time) [24, 25], problemas de imágenes de generación de texturas [26, 30], algoritmos de compresión de imagen y video [27, 28, 34] y en particular problemas de clasificación de calidad percibida en voz y video sobre IP [32, 33]. Precisamente en estos dos últimos trabajos se basa la implementación del mencionado sistema de clasificación.

Metodología implementada

¿Cómo se utiliza la red neuronal en la clasificación? La idea es simple: la calidad de servicio percibida para una aplicación de tiempo real en IP depende básicamente del estado de la red de transporte (pérdidas, retardo, jitter), de las características del contenido a transmitir (codec, robustez del algoritmo de codificación), de las prestaciones de los equipos utilizados en la transmisión y recepción (velocidad de procesador, memoria, tarjetas dedicadas) y del software de capa de aplicación utilizado. Si se asume que los equipos utilizados y el software

involucrado son adecuados para la aplicación en cuestión, es posible relacionar la calidad percibida con un conjunto de parámetros estimables (aunque no fácilmente en todos los casos) de la red y del contenido. En la figura 3.16 se muestra la estructura básica de este sistema. Por lo tanto, si se encontrara alguna función que mapeara este conjunto de parámetros en alguno de los parámetros subjetivos de calidad descritos en 3.2.1 sería posible clasificar la aplicación en cuestión incluso antes de utilizarla. En este punto se plantea el uso de la red neuronal.

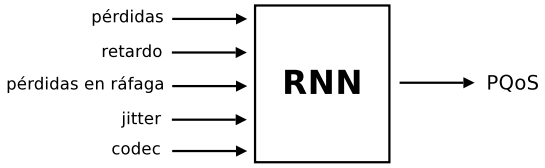


Figura 3.16: Estructura del sistema de clasificación

El éxito empírico obtenido en [32, 33] por este tipo de redes neuronales y con los objetivos mencionados puede ser justificado en la teoría en base a los resultados obtenidos en [31], donde se demuestra que para cualquier función continua f de varias variables definida en un conjunto compacto C, es posible construir una red neuronal aleatoria de estructura predefinida que aproxime el mapeo generado por f hasta cierto grado de precisión deseado. Si bien el conjunto de parámetros disponible no representa en teoría un conjunto compacto (se puede pensar por ejemplo que cada parámetro toma cualquier valor en R^+), en la práctica claramente existe un conjunto acotado de valores posibles. Al mismo tiempo, no resulta difícil pensar que la función f que se intenta estimar es una función continua (al menos respecto de los parámetros de red, ya que el resto no son continuos). Esto último justifica el uso de este sistema de aprendizaje supervisado.

Estructura de la red de neuronas

A continuación se describe brevemente la estructura del modelo de red de neuronas utilizado. En el anexo C se brinda una descripción más detallada de este modelo.

En la figura 3.17 se presenta un diagrama de la red de neuronas utilizada. Esta posee una estructura de tipo "feed-forward"de tres etapas: una primera etapa de neuronas de entrada a las cuales ingresan los distintos parámetros de la estimación (*neuronas de entrada*), una etapa de salida que devuelve los resultados (*neuronas de salida*) y una etapa intermedia de neuronas que interconecta las etapas de entrada y salida(*neuronas escondidas*). La característica "feed-forward"de la red de neuronas determina la no existencia de loops cerrados en la estructura de la red.

El número de neuronas de entrada y salida coincide en general con el número de parámetros de entrada y salida del sistema respectivamente. El número de neuronas intermedias

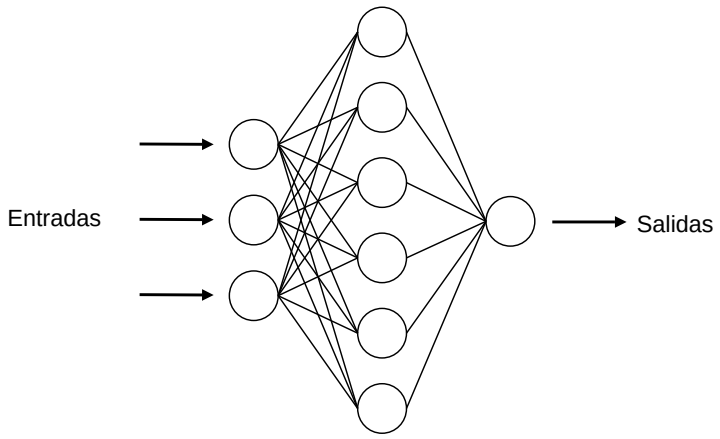


Figura 3.17: Estructura de la red de neuronas

permite agregar grados de libertad al sistema: un número menor de neuronas intermedias resultará en principio en un ajuste de menor precisión; sin embargo, un número elevado de neuronas intermedias puede llevar a problemas de inestabilidad de la red (la estimación se ajusta muy bien a la muestra de datos utilizada en la fase de entrenamiento pero se pierde capacidad de estimación ante la llegada de entradas no contempladas en dicha base de entrenamiento).

Una vez descrita la estructura básica de la red de neuronas se pasará a describir brevemente el proceso de implementación de la misma; este proceso se compone de dos etapas consecutivas: una primera etapa donde se *entrena* o calibra la red para que clasifique como lo haría un ser humano, y una segunda etapa en donde se utiliza la red calibrada como un sistema de clasificación automático.

Aprendizaje de la red

- El primer paso a dar consiste en definir cuales serán los parámetros sobre los que se trabajará, considerando para cada uno de ellos un conjunto de valores más representativos (p.e. tomar los valores más usuales). Se define así el dominio de la función mapeo.
- A continuación se deberían seleccionar de dicho dominio cierto número de combinaciones (las más probables), de manera tal de cubrir razonablemente el conjunto a mapear. Estas combinaciones se definen como *configuraciones*. Ahora bien, para evitar cantidades abrumadoras de configuraciones se toman los valores variando unicamente de a dos parámetros por vez, dejando el resto de los parámetros fijos (p.e. si tuviésemos 6 parámetros con 3 valores posibles cada uno, el número de configuraciones sería de $3^6 = 729$!!!).

- En base a cada configuración se genera (p.e. mediante una red de pruebas controlada) una señal distorsionada del servicio a clasificar (secuencia de audio o video). Esto resulta en un conjunto de elementos de la forma $[S_{original}, S_{distorsionada}, N_{configuración}]$.
- Luego se realiza un test subjetivo de calidad de servicio (ref: sección MOS) para cada uno de los elementos del conjunto, resultando en una base de datos final que contiene, para cada configuración de parámetros, un valor del parámetro subjetivo seleccionado en dicho test.
- Finalmente se divide esta base de datos en dos subconjuntos: uno que se utilizará para la calibración de la red (muestra de entrenamiento) y otro que se aplicará en la etapa de validación (muestra de validación). Dado que la etapa fundamental de todo el sistema es la de aprendizaje, se toma la muestra de entrenamiento de tamaño mayor al de la muestra de validación (p.e. 75 % de los elementos).

La calibración de la red se realiza en base a la minimización de una función de costo que penaliza la diferencia entre la predicción realizada por la red neuronal y el valor real del parámetro subjetivo obtenido del test. En el Anexo A (Anexo: Descripción del modelo de la RNN) se encuentran los detalles del algoritmo de aprendizaje.

Validación y uso de la red calibrada

Antes de utilizar la red calibrada se utilizan las muestras restantes de la base de datos (muestra de validación) para validar el proceso de aprendizaje. Se comparan los resultados obtenidos en los tests subjetivos con los estimados por la red de neuronas y si las diferencias son razonables (dependerá de la aplicación en particular) se da por finalizada la implementación del sistema. A partir de este momento será razonable obtener buenos resultados para valores de entrada *cercanos* a los utilizados en la implementación de la red. En caso de que las configuraciones de parámetros de entrada se tornen muy distintas a las *observadas* por la red en las etapas anteriores será necesario recalibrar el sistema, incluyendo en los nuevos tests subjetivos estas nuevas configuraciones.

Ventajas y desventajas de la metodología

A diferencia de las metodologías antes descritas, en las cuales (salvo en el caso particular del E-Model) la clasificación se basa en comparación de señales (con los respectivos tiempos insumidos), el uso de la red neuronal permite la implementación de una herramienta de clasificación en tiempo real. Si bien la fase de aprendizaje es lenta e involucra necesariamente la realización de tests subjetivos, el tiempo que insume la clasificación es mínimo (menor a 1 segundo). Otra ventaja que en principio ofrece esta metodología es el hecho de que no es necesaria la transmisión por la red de una secuencia real de audio o video como en las metodologías de comparación, resultando en una técnica de estimación menos invasiva. Ahora bien, la determinación de parámetros de la red se realizará en general en base a técnicas de medición activas (inyección de tráfico en la red), por lo cual esta última ventaja será relativa

a la cantidad de tráfico involucrado en la estimación de los parámetros de la red.

La principal desventaja que presenta esta metodología es la fuerte dependencia que posee con la fase de entrenamiento. Como se explicó antes, en caso de que los valores de los parámetros de entrada se aparten del conjunto de entradas utilizado en la calibración, los resultados obtenidos no serán buenos. Es posible reducir los efectos de este problema mediante el uso de un conjunto de configuraciones para la etapa de entrenamiento que cubra la mayor cantidad de casos posibles, introduciéndose así un compromiso entre robustez del sistema y costos de calibración (tiempos involucrados en los tests subjetivos y costos de implementación de los mismos).

3.5.2. E-Model

El **E-Model**, abreviado de European Telecommunications Standards Institute (ETSI) Computational Model, definido en el ETSI Technical Report ETR 250 [17] y luego estandarizado en la ITU-T Recommendation G.107 [18], surge como una herramienta para la planificación de redes telefónicas híbridas (mezcla de redes de circuitos conmutados (SCN) y de paquetes conmutados (PSN)). Esta herramienta permite a los proyectistas ver como los distintos parámetros de transmisión afectan la calidad que percibirán los usuarios finales. Si bien no existe un acuerdo para esto [18], el E-Model esta siendo ampliamente utilizado como un método no intrusivo para estimar la calidad de servicio (QoS) en aplicaciones de voz sobre IP (VoIP).

El principio fundamental del E-Model se basa en un concepto introducido por J. Allnatt: "Factores psicológicos en la escala psicológica son aditivos"; esto simplifica bastante lo que al principio podría ser una relación muy compleja entre los distintos factores que determinan la calidad. La salida del E-Model es el factor de clasificación de la transmisión (transmission rating factor) **R**, el cual se calcula como:

$$R = R_0 - I_s - I_d - I_{e-eff} + A \tag{3.28}$$

Donde:

- R_0 (relación señal a ruido básica, en el punto de 0 dB_r): representa los efectos del ruido. Depende de los distintos ruidos presentes, desde el ruido ambiente hasta el ruido introducido por los circuitos que forman la red.
- I_s es la suma del deterioro que se produce de forma simultánea con la transmisión de la voz (ruido de cuantización, sidetone, nivel de recepción de la voz). Tanto este factor como el anterior están determinados por parámetros de la telefonía convencional.
- I_d es la suma del deterioro que se produce de forma retardada. Está determinado por el eco del lado receptor y transmisor y el retardo absoluto boca-oído. Este factor además

del eco da cuenta de la pérdida de inteeractividad que sufren los usuarios por retardos muy grandes. Los parámetros que influyen son el retardo medio de ida, ida y vuelta, y el retardo desde el lado receptor hasta el punto en la conexión donde se produce el acoplamiento de señales como fuente de eco. Por interesarnos en una conexión sobre IP exclusivamente, tomamos este ultimo igual que el retardo medio de ida.

- I_{e-eff} tiene en cuenta la distorsión de la voz producida por el codec que se utilice y por la pérdida de paquetes. Se calcula como:

$$I_{e-eff} = I_e + (95 - I_e).\frac{Ppl}{\frac{Ppl}{BurstR} + Bpl} \tag{3.29}$$

  donde

  - I_e representa la distorsión introducida por el codec, y está cuantificada de forma empírica para distintos codecs, en [19] se encuentran valores para los diferentes codecs.
  - Ppl es la probabilidad de pérdida de paquetes, que incluye tanto los paquetes perdidos en la red (poe ejemplo descartados en las colas de los routers), como los paquetes perdidos en el buffer del codec debido al jitter.
  - Bpl (Packet-loss Robustness) es una medida de la robustez del codec frente a la pérdida de paquetes, depende de si el codec implementa algún algoritmo de reconstrucción. En [19] se encuentran valores para distintos codecs.
  - BurstR (Burst Ratio) permite incluir el efecto de pérdidas en ráfagas. Se define como el cociente entre el largo medio de las ráfagas observadas en la secuencia de llegada sobre el largo medio de ráfagas esperado suponiendo pérdidas independientes. Un valor de uno implica pérdidas independientes mientras que un valor mayor implica pérdidas en ráfagas.

- A (factor de expectativa o ventaja) se debe a los diferentes niveles de exigencia de acuerdo al sistema de comunicación utilizado (por ejemplo teléfono cableado A=0, celular A=10)

El E-Model será utilizado en una red de conmutación de paquetes, simplificándose así el modelo implementado. De esta forma el resultado sólo dependerá de los parámetros de transporte de la red IP.

Para lograr esta simplificación se tomaron los valores por defecto dados en [18] para los parámetros clásicos de una red SCN, quedando como variables del modelo la probabilidad de pérdida de paquetes, el Burst Ratio, el retardo absoluto de ida y vuelta, y los parámetros dependientes del codec (I_e y Bpl).

Una vez obtenido el valor R, es posible mapearlo a parámetros utilizados en las medidas subjetivas de calidad como el MOS_c (Conversational MOS).

Si bien el E-Model es atractivo para la predicción de calidad presenta varias limitaciones. Está basado en fórmulas empíricas y por lo tanto es aplicable solamente a un cierto conjunto de codecs y condiciones de la red. Para cada configuración es necesario la validación del modelo de forma experimental, comparando los resultados con medidas subjetivas de calidad (costosas en tiempo y recursos). Actualmente ha sido verificado para cierto rango de valores de los parámetros de entrada [18].

Capítulo 4

Implementación y Aplicaciones

Una vez presentados los conceptos sobre los cuales se basa el proyecto, pasaremos a describir la herramienta de software desarrollada para la estimación de la PQoS desde las puntas de una conexión. En la secciones 4.1 y 4.2 se describe la estructura del software. En particular, la sección 4.2 presenta los detalles de implementación de los distintos módulos que componen el software. En la sección 4.5 se presentan los detalles relativos a la plataforma de pruebas implementada para la obtención de datos y validación de la herramienta. Por último, en la sección 4.6 se describe la realización de los test subjetivos.

4.1. Descripción de la Herramienta de Medida

De acuerdo a los objetivos marcados la herramienta de software a implementar debía tener la capacidad de estimar la calidad de servicio percibida por un usuario de servicios de voz y/o video sobre IP, teniendo acceso únicamente a los equipos de las puntas del servicio. Esto imponía el desarrollo de un sistema que fuese capaz de establecer una conexión con el equipo de la otra punta para poder evaluar las condiciones del enlace sobre el cual se transportaría dicho servicio. Este esquema de trabajo presentaba a su vez dos posibles arquitecturas de implementación:

Cliente-Servidor

En esta arquitectura, uno de los equipos de las puntas cumple el papel de proveedor o servidor de aplicaciones mientras que el otro adopta el papel de consumidor o cliente. El servidor es quien concentra la complejidad del problema, encargándose de todos los elementos necesarios para estimar la calidad que recibirá el cliente (figura 4.1).

Cuando el cliente desea evaluar la calidad que recibirá se conecta con el servidor y solicita, para cierto servicio que éste provee, una estimación de la PQoS que obtendrá en ese

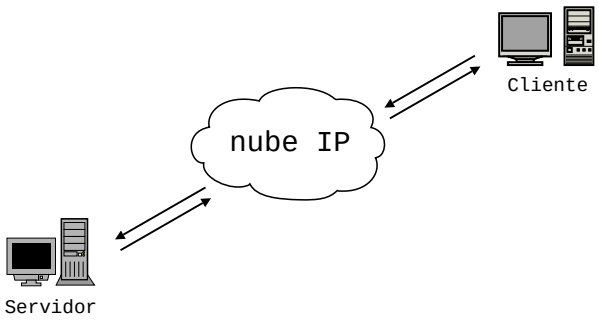


Figura 4.1: Arquitectura Cliente-Servidor

momento. Dependiendo del tipo de metodología a utilizar para realizar esta estimación (ver 3), el servidor deberá o bien enviar una muestra del servicio que ofrece (supongamos por ejemplo una secuencia de video corta) que el cliente deberá almacenar en tiempo real, o bien estimar los parámetros de la conexión. En el primer caso se agrega una complejidad extra al cliente quien no solo deberá poseer la capacidad de almacenar la muestra enviada sino que a su vez deberá retornar esta muestra al servidor (por una conexión sin pérdida) para su eventual comparación. En caso de tener que estimar el estado de la conexión y suponiendo una inyección de paquetes de test por parte del servidor, el cliente deberá ser capaz de retornar los paquetes recibidos para su eventual procesamiento. En la figura 4.2 se presentan brevemente ambas situaciones.

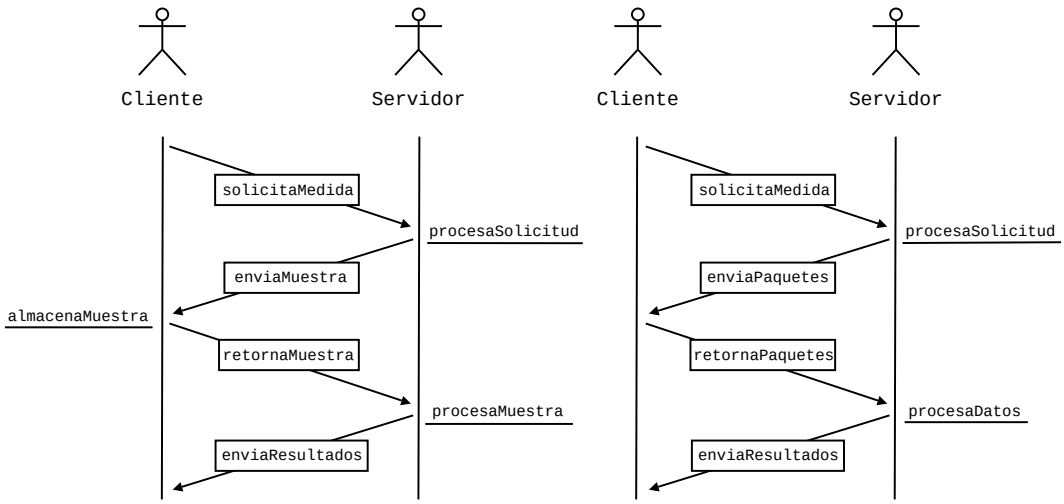


Figura 4.2: Esquema de funcionamiento - Arquitectura Cliente-Servidor

Ahora bien, no todos los servicios involucran directamente al cliente y al servidor como los

extremos de la conexión. Consideremos por ejemplo el caso de una conversación de voz entre dos clientes de un servicio de los que hoy en día se ofrecen gratuitamente en Internet (Skype [38], MSN [39], etc...). En este caso el servidor sólo interviene en el establecimiento de sesión entre los dos usuarios; el envío de la voz codificada se da luego entre los dos clientes a través de una conexión *peer-to-peer*, convirtiéndose ambos en las puntas entre las cuales se desea estimar la PQoS (figura 4.3). En este caso el esquema anterior no puede ser implementado como tal y alguno de los clientes debe adquirir responsabilidades de "servidor"para realizar la estimación. Por esta razón se decidió adoptar un esquema de funcionamiento distinto que se describe a continuación.

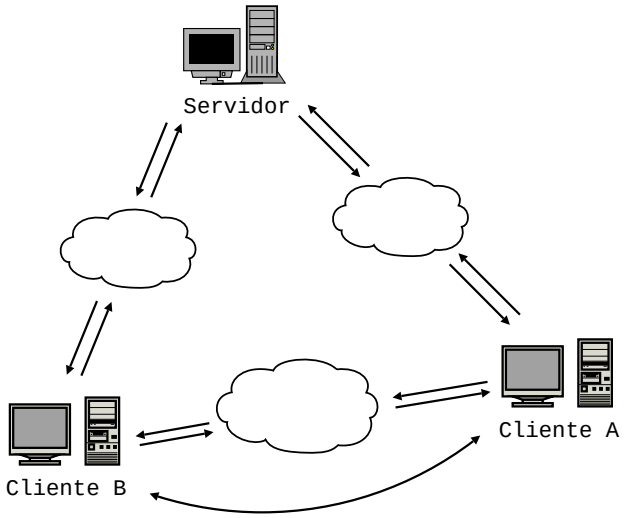


Figura 4.3: Problema del esquema Cliente-Servidor

Cliente/Servidor-Servidor/Cliente

En esta arquitectura, ambos extremos poseen la capacidad de funcionar tanto en modo servidor como en modo cliente, dependiendo de cual de las dos puntas sea la que solicite la estimación de calidad (figura 4.4). Si bien la complejidad del problema pasa ahora a concentrarse en ambas puntas, perdiéndose la deseable característica de mantener "clientes tontos", resulta posible evitar los problemas antes mencionados.

El funcionamiento de esta nueva arquitectura es levemente distinto al del esquema anterior. En principio ambas puntas adquieren el papel de servidor al ejecutarse la herramienta en cada terminal, entendiéndose por servidor en este caso aquel extremo que espera por una petición de medida. Esto se logra en base a una estructura de multithreading que mantiene en todo momento un escucha a la espera de una petición en ambas puntas (en la sección 4.2.1 se analiza esta estructura). Cuando una de las dos puntas realiza un pedido de medición pierde automáticamente la calidad de servidor y adopta el rol de cliente, pasándose a una estructura

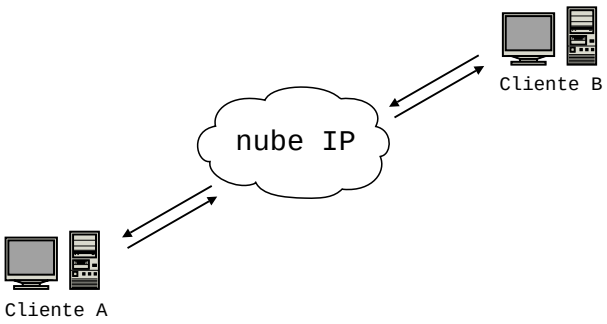


Figura 4.4: Arquitectura Cliente-Servidor Servidor-Cliente

clara de cliente-servidor similar a la analizada anteriormente. En la figura 4.5 se presentan estos conceptos.

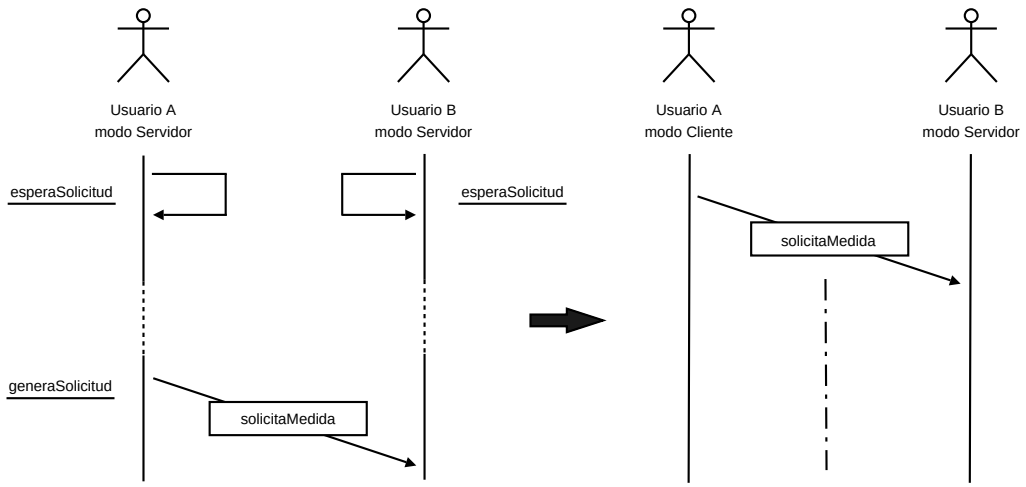


Figura 4.5: Esquema de simetría

La principal ventaja que se obtiene con este esquema de simetría es la capacidad que adquieren ambas puntas de procesar y generar información indistintamente. En un principio ya no es necesario que el cliente retorne información alguna al servidor, ya que él mismo puede ahora encargarse de procesar los datos recibidos. Como desventaja principal figura el hecho de que una herramienta más compleja debe estar instalada en ambos extremos de la conexión y activa al mismo tiempo en ambas puntas al momento de realizar la estimación.

4.2. Arquitectura del Software

Una vez presentada la arquitectura de funcionamiento adoptada pasaremos a describir someramente la arquitectura del software. La herramienta desarrollada está compuesta por 5 módulos independientes entre sí. A su vez, cada módulo está compuesto por varios submódulos o bloques. Cada uno de ellos será analizado en profundidad en las siguientes secciones. En la figura A.1 se presenta un diagrama de sistema que presenta cada uno de estos módulos.

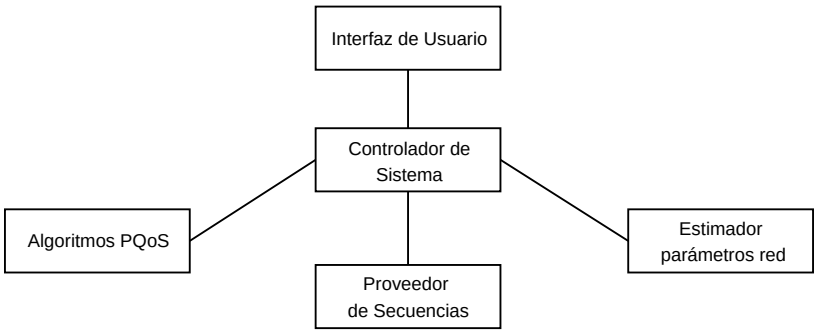


Figura 4.6: Módulos de sistema

Controlador de Sistema: el controlador de sistema es el cerebro del software. Administra el establecimiento de conexión y el intercambio de información entre los equipos de las puntas, controlando al mismo tiempo el resto de los módulos de sistema. Está compuesto por 3 bloques que permiten desacoplar los distintos modos de funcionamiento del sistema:

Figura 4.7: Controlador de Sistema

- **Bloque Controlador:** es el bloque central que implementa todas las funcionalidades de control del sistema independientemente del modo de operación.
- **Bloque Cliente:** incorpora las funcionalidades del bloque controlador correspondientes al modo de operación cliente.
- **Bloque Servidor:** incorpora las funcionalidades del bloque controlador correspondientes al modo de operación servidor.

Algoritmos de PQoS: el módulo de algoritmos de PQoS es el módulo más importante de todo el sistema. Implementa los distintos algoritmos expuestos en el capítulo 3. Se compone

de dos bloques distintos, uno que concentra las implementaciones de audio y otro que abarca las de video:

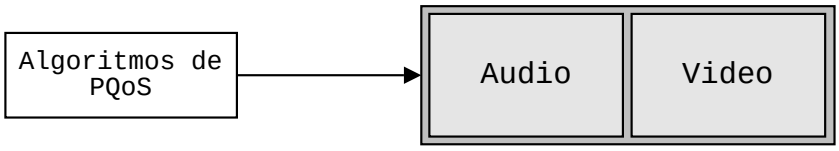


Figura 4.8: Algoritmos PQoS

- **Bloque Audio:** contiene las distintas herramientas de estimación de PQoS desarrolladas para audio. Abarca los algoritmos de PESQ, EMBSD, MNB, E-Model y la RNN entrenada para audio.
- **Bloque Video:** contiene las distintas herramientas de estimación de PQoS desarrolladas para video. Abarca los algoritmos de SSIM, ITS, PSNR y la RNN entrenada para video.

Proveedor de Secuencias: este módulo es el encargado de suministrar al módulo de PQoS las parejas de secuencias de audio y/o video necesarias para las metodologías de comparación de señales. Tres son los bloques que abarca este módulo:

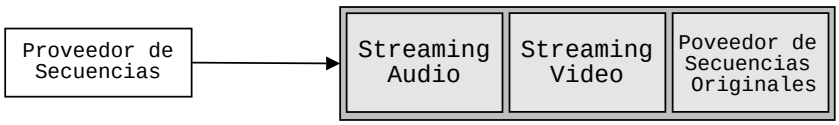


Figura 4.9: Proveedor de secuencias

- **Bloque Proveedor de Secuencias Originales:** las secuencias de audio y video de prueba originales, representativas de las distintas categorías dentro de cada servicio son administradas por este bloque. Para cada una de estas secuencias registra información relativa a su contenido.
- **Bloque Streaming Audio:** este bloque es el encargado de realizar la transmisión y recepción en tiempo real de las secuencias de audio suministradas por el bloque anterior, generando así las secuencias de audio distorsionadas. Su implementación se basa en la Java Media Framework de Sun (ver apéndice G).
- **Bloque Streaming Video:** cumple las mismas tareas que el bloque anterior pero para el caso de secuencias de video. Su implementación se basa en la plataforma de streaming que brinda el Video Lan Client (ver apéndice G).

Estimador de Parámetros de Red: este módulo es el encargado de estimar los parámetros de calidad objetivos de la red que transporta el servicio a calificar. Estos datos son luego utilizados por los algoritmos no intrusivos del módulo de PQoS. Dentro de este módulo se identifican dos bloques con tareas claramente desacopladas.

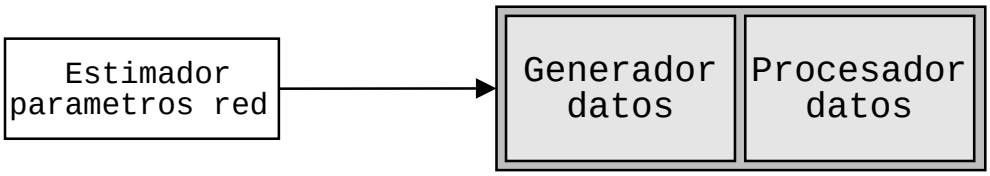


Figura 4.10: Estimador de parámetros de red

- **Bloque Generador de Datos:** es el encargado de inyectar tráfico de prueba en la red y recopilar todos los datos obtenidos.
- **Bloque Procesador de Datos:** se ocupa de procesar la información suministrada por el bloque anterior para estimar los distintos parámetros de la conexión de red.

Interfaz de Usuario: el módulo de interfaz de usuario define una capa gráfica intermedia que interactúa entre el usuario final y el módulo controlador de sistema, facilitando el uso de la herramienta desarrollada. Su implementación se basa entre otras en un conjunto de librerías Java de código abierto para diseño de interfaces gráficas de ususario [40].

Lenguaje de programación

Una vez presentada la arquitectura pasaremos a describir el lenguaje de programación escogido. Se puede observar que la descripción brindada hace fuerte hincapié en la estructura modular del software. Esto motiva fuertemente el uso de un paradigma de programación orientado a objetos cuyas características facilitan la implementación de dicha estructura (en particular, la característica de bajo acoplamiento que este ofrece). De los lenguajes conocidos por el grupo de proyecto solo Java posee estas características. Al mismo tiempo, las características de portabilidad que éste ofrece hacen más atractiva su elección.

Ahora bien, es sabido que la performance que Java ofrece en aplicaciones con restricciones temporales altas no es muy buena. Esto motivaría el uso de algún otro lenguaje con mejores prestaciones de tiempo (por ejemplo C++), pero lamentablemente nuestro conocimiento en este área es escaso (*"cuando solo se tiene un martillo como herramienta, todos los problemas se parecen a un clavo"*).

Investigando un poco más sobre este problema de tiempos encontramos una herramienta que a la postre nos permitiría solucionar varios problemas: la JNI (Java Native Interface). La

JNI es una interfaz que permite la interacción entre código interpretado (máquina virtual de Java) y código *nativo* (C o C++). Esto nos permitió implementar los algoritmos del módulo de PQoS en C/C++ y utilizarlos desde el lado de Java (en el anexo A se brindan detalles del uso de la JNI).

En resumen, el desarrollo del software se realizó principalmente en Java, incluyendo implementaciones en C y C++ para algunos de los módulos descritos y utilizando para la interacción de lenguajes la interfaz JNI.

4.2.1. Módulo Controlador de Sistema

El módulo controlador de sistema es el encargado de gestionar la interacción entre los distintos módulos que componen el software. Permite dos modos distintos de operación: un modo de tipo servidor y otro de tipo cliente. Como se mencionó en la sección 4.2, se compone de 3 bloques distintos.

Bloque Controlador

Basado en una estructura de multithreading, el bloque controlador implementa todas las funcionalidades de manejo del sistema:

- **Demonio de conexión:** se activa al ejecutarse la aplicación. En base a threads se implementa una rutina de espera (un *escucha*) que aguarda por una solicitud de conexión. Al recibirse un pedido de conexión la rutina de espera se detiene, iniciándose la fase de establecimiento de conexión de control. Como se explicó en la sección 4.1, esta solicitud de conexión condiciona el modo de funcionamiento de cada una de las puntas. La rutina de espera se pone nuevamente en funcionamiento al finalizar la estimación de PQoS.
- **Establecimiento de conexión de control:** permite el establecimiento de conexión entre ambos extremos en base al uso de sockets TCP. Implementa un protocolo de comunicación básico que permite coordinar la interacción entre el servidor y el cliente.
- **Estimación del estado de la conexión:** permite realizar la estimación de los parámetros de red mediante llamadas al módulo estimador de parámetros. Utiliza el canal de control para el intercambio de información relativa a las características de la medición. Las llamadas al módulo estimador se hacen a través de threads.
- **Envío y captura de secuencias multimedia:** el envío y captura de secuencias multimedia se realiza mediante llamadas al módulo proveedor de secuencias. Al igual que en el caso anterior, todas las llamadas se realizan a través de threads independientes.
- **Estimación PQoS:** permite seleccionar el tipo de algoritmo a utilizar para la estimación de PQoS.

Bloque servidor y bloque cliente

Los bloques servidor y cliente concentran las tareas relativas a los dos modos de funcionamiento posible del sistema. Ambos utilizan las funcionalidades implementadas por el bloque controlador.

4.2.2. Módulo de Estimación de PQoS

Este módulo implementa todas las funcionalidades necesarias para el cálculo del DMOS, para audio y video. Los algoritmos que lo componen permiten realizar tanto medidas intrusivas como no intrusivas.

Se compone de dos bloques distintos, uno que concentra los algoritmos de audio y el otro abarca los de video.

Algoritmos de Audio

Los algoritmos de audio que comparan señales utilizados en nuestro software son tres, en dos casos (EMBSD, MNB) sus autores publican el código fuente permitiendo su libre utilización. Por último la norma P.862 trae una implementacion de referencia que se utilizó con fines educativos. Como el desarrollo del software se estaba realizando en Java, se planteaba una disyuntiva, portar el código escrito en C a Java, o utilizar una característica de Java (JNI-Java Native Interface), que permite realizar llamados a funciones escritas en otros lenguajes. Decidirse por portar todos los códigos a Java, tiene la ventaja de la portabilidad, se compila una vez y corre en muchas plataformas a coste de una peor performance debido a su arquitectura de Maquina Virtual, además del tiempo que insume portar un código de un lenguaje a otro, que aunque no presente desafíos conceptuales, puede resultar bastante tedioso. Utilizar el código escrito en C mediante la JNI brinda una velocidad de ejecución mayor, e insume menos tiempo su adaptación al software. Como nuestro interés es que corra en los sistemas operativos Windows y Linux, perder la portabilidad que ofrece Java no es grave. La elección fue la de utilizar los códigos nativos escritos en C, llamándolos desde Java mediante la JNI.

Formatos de audio Los formatos de audio posibles son muchos, varían en tasa de muestreo, cantidad de bits por muestra, número de canales, codificación, y además existen varios formato de archivo que contienen los datos de audio más una cabecera que contiene la información necesaria para decodificar el audio. Los códigos del EMBSD y del MNB reciben como entrada dos archivos de audio en formato PCM 16 bits con signo , un sólo canal, a 8 KHz sin formato de archivo (raw data), además de dos parámetros que indican el orden de los bytes (MSB-LSB LSB-MSB) y el offset entre las señales respectivamente. El código de PESQ recibe dos archivos con audio en formato PCM, un solo canal,16 bits con signo, 8 KHz

con formato de archivo WAV.

Figura 4.11: Archivo de audio tipo

Conversión Para que el software se pueda utilizar de manera automática, sin preocuparse por formatos, tasa de muestreo, orden de los bits, etc. se desarrollo una herramienta que convierte un archivo que contiene audio en cualquier formato, y formato de archivo AU o WAV, en dos archivos, uno con formato de archivo WAV, y otro sin formato (RAW) donde los datos de audio tienen el formato necesario para correr los algoritmos ya mencionados (PCM con signo, 16 bits, 8KHz, MSB-LSB). Para realizar este parte del software utilizamos una parte de la API de Tritonus [44].
Esta API es una implementación de la API de sonido de Java para Linux, ofrece muchas mejoras para el programador, además de mejoras técnicas en comparación con la original. Para el software utilizamos una parte de la API, en forma de plug-in que funciona en todas las plataformas, este plug-in brinda facilidades para transformar el audio entre distintas codificaciones, tasas de muestreo, número de canales, etc. .

Implementación del E-Model

El E-Model se implemento a partir de la norma [18]. El primer paso fue implementarlo en MatLab, debido a la sencillez del mismo. Esta implementación fue verificada con la implementación de referencia de la norma, escrita en Basic.
Por último se realizó en Java, para poder adaptarse fácilmente a el resto del módulo de PQoS.

Algoritmos de Video

Al igual que con el E-Model, el primer paso fue la implementación en MatLab debido a el fácil manejo de matrices y la existencia de funciones que facilitaron la implementación de

los distintos algoritmos.

En este caso no se contaba con ningún tipo de implementación de referencia, por lo que las pruebas fueron muy básicas. Para el SSIM y el PSNR las primeras pruebas consistieron en correr estos algoritmos con matrices tridimensionales, que simulan un video, idénticas y verificar que el resultado fuera acorde. Esto permitió detectar algunos errores en la implementación.

El segundo paso fue correr los algoritmos con los videos disponibles del VQEG (ver 3.4.4). Estos videos se encuentran sin ningún tipo de compresión (un video de 10 segundos ocupa 180 MB aproximadamente) y están codificados en YUV 4:2:2, estos es, por cada pixel hay un valor de luminancia (Y), pero las cromas se encuentran submuestreadas a la mitad. Adaptamos los algoritmos, de forma tal de ir procesando de a un cuadro del video. Esto permite cargar en memoria el video de a un cuadro, y evita cargar el video entero, lo cual es inviable debido a el tamaño del mismo.

En este punto se encontró otra dificulta, que era el tiempo que demoraba la ejecución de los algoritmos (sobre todo el ITS). Se pasó a la implementación en Java de los mismos. Las implementaciones de Java bajaron los tiempos apreciablemente lo que permitió hacer una verificación primaria de los algoritmos. Los resultados de ésta se presentaron en 3.4.4.

Decodificación de los videos

El siguiente paso fue la implementación de un módulo de software que permitiera el uso de los algoritmos con videos codificados. Esto se puede realizar de dos formas, o bien decodificar todo el video, o ir decodificando de a un cuadro, a medida que es procesado por el algoritmo. La primer opción no es viable debido al tamaño que ocupa un video sin codificar (del orden de 1 MB por cuadro), aunque resultaba más simple. Para llevar a cabo la segunda opción se comenzó utilizando la Java Media Framework (JMF) (por más información ver Apéndice G).

Se comenzó modificando una clase que se da como ejemplo [45], la cual en teoría permite avanzar el número de cuadros deseado. El problema que apareció es que el método que permite avanzar por cuadros en el video no funciona correctamente para algunas codificaciones, por ejemplo Mpeg-1. En vez de avanzar cuadro por cuadro, repetía un mismo cuadro N veces y luego avanzaba al cuadro correcto. Por ejemplo, supongamos que obtenemos el cuadro 1 del video correctamente, cuando avanzamos al siguiente cuadro (supuestamente el 2) en realidad lo que obtenemos es el cuadro 1 nuevamente. Esto sucede así hasta que avanzamos N veces, momento en el cual el cuadro que pedimos (N+1) coincide con el cuadro N+1 del video. Por lo tanto se perdían los restantes cuadros. Dado que el número N era constante en cada video, y los cuadros a los cuales se podía accederse eran siempre los mismos para cada video, parece razonable suponer que el problema era con la codificación temporal y que solo se accedía a los cuadros tipo I. Esto es coherente con el hecho de que en los video que no tienen compresión

temporal el método funciona correctamente.

A su vez se encontró otro problema, la estructura de la JMF no permite acceder al control de posición en dos videos del mismo tipo a la vez (esto es reconocido como una limitación de la JMF por la gente de Sun). Este problema es no menor teniendo en cuenta la forma en que se utilizan los algoritmos. La posible solución a esto era tener los videos originales en dos formatos distintos, uno para la transmisión (formato original) y otro distinto para realizar la comparación posterior. El problema es la modificación que se introduce en el video original al recodificarlo.

A pesar de este último problema se siguió intentando obtener un decodificación por cuadro exitosa, en base a otras implementaciones de prueba. Este problema perduró durante casi un mes, hasta que integrantes del grupo Multimedia del IIE recomendaron el uso de algunas herramientas para el tratamiento de video. Estas son la FFmpeg y FOBS. La FFmpeg [46] es una herramienta que permite grabar y convertir señales de audio y video. Por su parte la FOBS [47] es un conjunto de clases de *C++* que facilita el manejo de la FFmpeg.

Estas herramientas permitieron sortear los problemas antes mencionados. Para poder integrar estas herramientas con los algoritmos, se tuvieron que implementar estos últimos en *C++*. Como resultado del cambio de lenguaje se obtuvo una reducción en el tiempo de ejecución de los algoritmos.

Al igual que en los casos anteriores la interacción entre Java y los algoritmos, implementados en código nativo, se realiza a través de la JNI antes mencionada.

4.2.3. Módulo Estimador de Parámetros de Red

Este módulo es el responsable de estimar el estado de la conexión entre los equipos involucrados. Realizando las medidas desde las puntas de la conexión se estiman, para ambos sentidos, los siguientes parámetros:

- Porcentaje de pérdida.
- Largo promedio de ráfaga de pérdida.
- Retardo promedio de ida y vuelta (RTT promedio).
- Jitter promedio.

Si bien no todos son luego considerados por el módulo de PQoS, se optó por abarcar un número mayor de parámetros para hacer del módulo estimador un elemento genérico y útil para alguna otra aplicación que requiera su uso.

Como se mostró en la sección anterior, las tareas del módulo estimador se concentran en dos bloques independientes; el bloque generador de datos y el bloque procesador de datos. El bloque generador es el encargado de realizar las medidas sobre el enlace mismo, recopilando los datos obtenidos para su eventual procesamiento. Las medidas realizadas por este bloque son de tipo activas, inyectándose tráfico de test entre las puntas de la conexión.

Para llevar a cabo esta tarea, el bloque generador hace uso de los 4 elementos funcionales que lo componen:

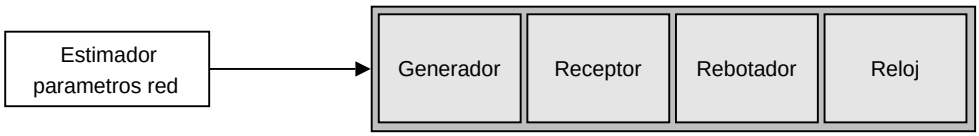


Figura 4.12: Estimador de parámetros de red

- **Generador de Paquetes:** genera y envía tráfico de test de tipo UDP. Permite configurar tiempo entre paquetes y tamaño de los mismos. La generación y el envío de paquetes se implementa en base al paquete **java.net** de la API estándar de Java, que permite entre otras el manejo de sockets y datagramas UDP.

- **Rebotador de Paquetes:** permite recibir todos los paquetes que arriban a cierta dirección de capa 4, reenviándolos hacia otra dirección especificada (en lo que sigue diremos "rebotar"el paquete).

- **Receptor de Paquetes:** recibe todos los paquetes que arriban a cierta dirección de capa 4 especificada, guardando información relativa a cada uno.

- **Reloj de alta precisión:** reloj implementado en código nativo (código C) con mejores prestaciones que el disponible en Java.

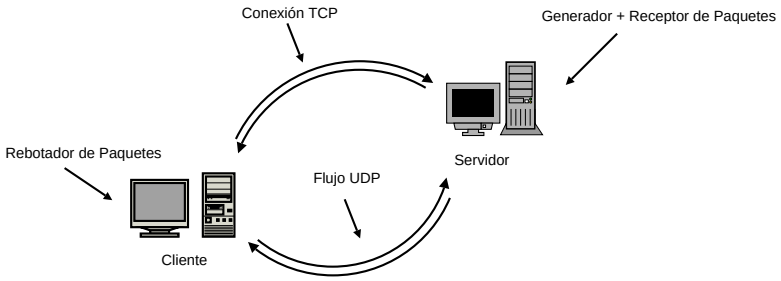


Figura 4.13: Estimación de parámetros de la conexión

En las figuras 4.13 y 4.14 se esboza la metodología de medida implementada por el bloque generador. Tanto el generador como el receptor de paquetes corren en la punta servidora, mientras que el cliente se limita simplemente a rebotar los paquetes que arriben. Para realizar una medida es necesario establecer primero una conexión segura de mensajes de control (a través del módulo controlador de sistema, utilizando sockets TCP) de forma tal de poder coordinar correctamente los pasos a seguir.

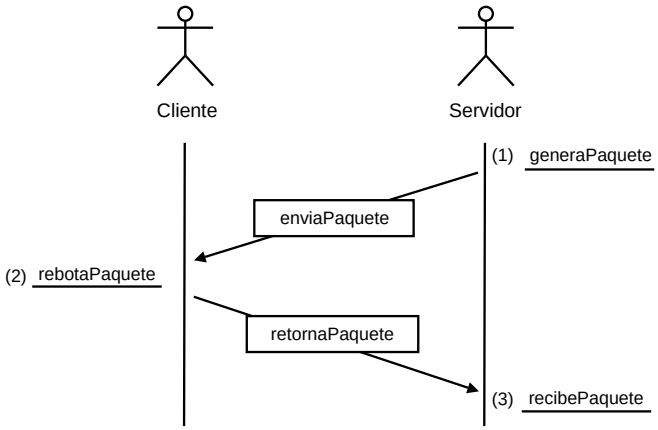


Figura 4.14: Metodología de medida

El servidor informa al cliente datos relativos a la medida (p.e. nº máximo de paquetes a transmitir) y espera su confirmación para comenzar. Del lado del cliente se activa el rebotador de paquetes y se avisa al servidor. El generador construye y envía paquetes UDP hacia el cliente (de acuerdo a la política especificada), introduciendo en la carga útil un número de secuencia (entero creciente) y una estampa del tiempo actual (punto (1)). Para cada paquete que arriba al rebotador se registra su número de secuencia (esta información es luego utilizada por el procesador de datos para discriminar sentidos de pérdida), se anexa en su carga útil una nueva estampa de tiempo actual y se envía de vuelta hacia el servidor (punto (2)). Finalmente, al retornar al servidor se registra el tiempo de arribo y los datos incluidos en la carga útil (punto (3)). Una vez que todos los paquetes han sido enviados y antes de terminar la medida, el cliente envía al servidor los números de secuencia de todos los paquetes que fueron rebotados. Como resultado de la medida se obtiene una secuencia de datos de la forma:

$$<N^o \text{ de secuencia, Bandera de pérdida, Sentido de pérdida, } T_{enviado}, T_{rebotado}, T_{recibido}>$$

La bandera de pérdida indica si el paquete fue perdido o no; en caso afirmativo, el sentido de pérdida indica si la misma fue a la ida o la vuelta, quedando en 0 los tiempos de envío, rebote y recibo. Aquí terminan las tareas del bloque generador, dando paso al bloque procesador para el cálculo de parámetros.

De la discusión presentada en 2.1 queda claro que los parámetros a determinar no son necesariamente los más indicados. Sin embargo, como se mencionó en el capítulo 1 el desarrollo de un módulo robusto de procesamiento de datos (de alto contenido estadístico) es uno de los puntos que quedó por fuera del alcance de este proyecto. De todas maneras, la modularidad de la implementación hace posible que en una etapa posterior puedan anexarse otros módulos de procesamiento de datos con mejores prestaciones que el implementado.

Los cálculos realizados por el módulo de procesamiento desarrollado son muy simples; las pérdidas promedio se calculan en base al recuento del número de paquetes perdidos en cada sentido. Para el cálculo de los largos promedio de las ráfagas de pérdida se utiliza la información adicional de los números de secuencia. El RTT promedio se calcula en base a los tiempos de envío y recibo de paquetes. El cálculo del jitter promedio en ambos sentidos es probablemente el punto más complicado; en consecuencia se presentan a continuación los detalles del mismo. En la figura 4.15 se muestran los tiempos involucrados entre la partida y el arribo de dos paquetes de test consecutivos:

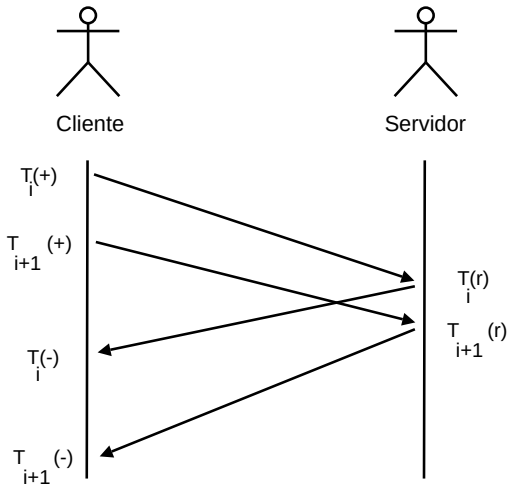


Figura 4.15: Cálculo del jitter

Sean $\Delta T(+)$ y $\Delta T(r)$ los tiempos entre paquetes consecutivos en el envío y el rebote:

$$\Delta T(+) = T_{i+1}(+) - T_i(+) \text{ y } \Delta T(r) = T_{i+1}(r) - T_i(r)$$

Recordando que el jitter representa la variación del retardo de paquete, se tiene:

$$Jitter_{ida} = \frac{\sum_i |(T_{i+1}(+) - T_i(+)) - (T_{i+1}(r) - T_i(r))|}{\sum_i (T_{i+1}(+) - T_i(+))}$$

$$Jitter_{vuelta} = \frac{\sum_i |(T_{i+1}(r) - T_i(r)) - (T_{i+1}(-) - T_i(-))|}{\sum_i (T_{i+1}(r) - T_i(r))}$$

Este cálculo del jitter presenta dos particularidades:

- En la diferencia entre tiempos se considera el valor absoluto, para evitar la cancelación de valores. En algunos casos puede resultar de utilidad diferenciar los valores de jitter positivos (*clumping*) de los negativos (*dispersión*).
- El valor calculado es relativo al tiempo medio entre envío y rebote de paquetes en cada caso. Si bien la definición formal brindada por el IPPM (ver anexo H) expresa el jitter como la diferencia entre los tiempos considerados, creemos que es más representativo asignar valores relativos.

Problemas de tiempo en Java

A la hora de trabajar con tiempos en Java nos encontramos con un gran problema inherente a este lenguaje de programación: la portabilidad e independencia de plataforma logradas traen como contrapartida un bajo desempeño en aplicaciones de alta demanda temporal. En efecto, estas características de independencia que Java brinda dependen fuertemente de una capa de software adicional (la famosa *Java Virtual Machine*) que introduce latencias mayores en la ejecución [35].

En nuestro caso particular comprobamos que las consultas temporales que Java ofrece son de baja resolución (del orden de algunos ms). Esto representa un problema importante a la hora de realizar medidas de tiempo, ya que retardos o jitter menores a dicha resolución no pueden ser estimados. El problema se ve acentuado en plataformas Windows (versiones 95, 98 y 2000), donde la resolución de tiempos es del orden de los 10 ms (esto se puede verificar fácilmente, ejecutando por ejemplo la herramienta *ping* desde un terminal de comandos). Al mismo tiempo, la precisión del reloj que Java utiliza es bastante pobre, afectando las rutinas de espera que el generador de paquetes utiliza para los envíos de paquetes consecutivos. Frente a esta problemática se propusieron dos posibles soluciones:

1. **Solución 1:** implementar todo el bloque generador de datos (generador, rebotador y receptor) en código nativo (C/C++), donde es posible realizar consultas de tiempo de mayor precisión y resolución más alta vía rutinas del sistema operativo.
2. **Solución 2:** implementar solamente las rutinas de espera y un reloj de alta resolución en código nativo.

En ambos casos se logra la interacción con Java vía la interfaz JNI antes descrita. La solución 1 brinda la ventaja de tener que acceder al lado nativo una sola vez por cada corrida de estimación de parámetros. Si tomamos en cuenta que esta interacción insume aproximadamente 250 μs, se observa en este sentido una gran ventaja. Sin embargo, la complejidad de implementación que representa la misma no justifica su elección.

Se decidió entonces el desarrollo de un reloj de mejores prestaciones que el disponible que permitiera generar estampas de tiempo con mayor resolución y rutinas de espera de mayor precisión. La desventaja que esta solución presenta se aprecia en la resolución alcanzada, del orden de los 250 μs mencionados. Igualmente y para las tiempos manejados decidimos que este era el camino a seguir (por detalles de implementación ver anexo A). En las siguientes figuras se muestran resultados comparativos del uso de nuestra implementación frente a la ofrecida por la plataforma de desarrollo:

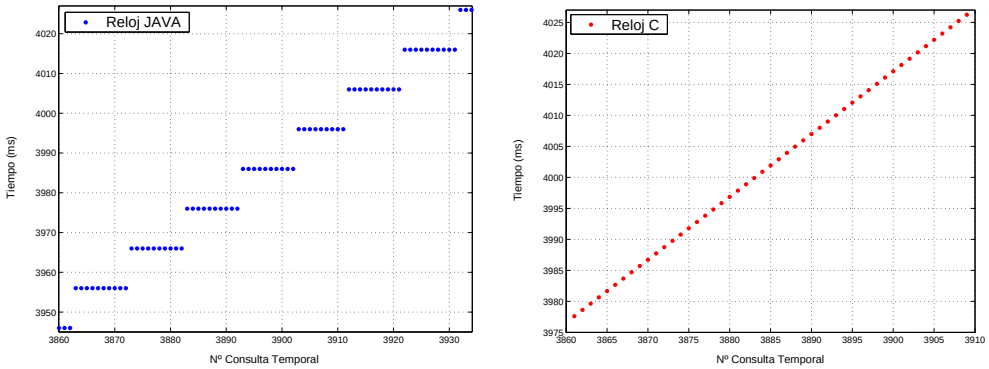


Figura 4.16: Resolución de las consultas de tiempo

En la figura 4.16 se comparan las resoluciones logradas. En ambos casos se realizan consultas del tiempo actual a intervalos iguales de 1 ms. Como se explicó antes, se puede apreciar que en la implementación Java (izquierda) la resolución alcanzada es de 10 ms y consultas a intervalos de tiempo menores dan resultados erróneos. Por el contrario, las consultas realizadas mediante nuestra implementación (derecha) muestran resultados correctos. Otra de las ventajas logradas se aprecia en la precisión de las rutinas de espera. En este caso la experiencia de prueba consistió en enviar paquetes entre dos máquinas (conectadas con un cable de red cruzado) a intervalos de tiempo de 1ms, registrando en la máquina destino los tiempos de arribo de paquetes mediante el uso de la herramienta de captura *tcpdump* [41]. Respecto de esta experiencia, cabe realizar dos aclaraciones:

- El estado de sobrecarga de la máquina destino era óptimo, en el entendido de que solamente se ejecutaba la herramienta de captura.
- Si bien la herramienta de captura introduce alteraciones en las medidas, estas son

despreciables a la escala de tiempos de la experiencia (la resolución de tiempos del tcpdump es del orden de μs).

En la figura 4.17 se muestran los resultados obtenidos. Se puede apreciar claramente que la dispersión obtenida con nuestra implementación es menor.

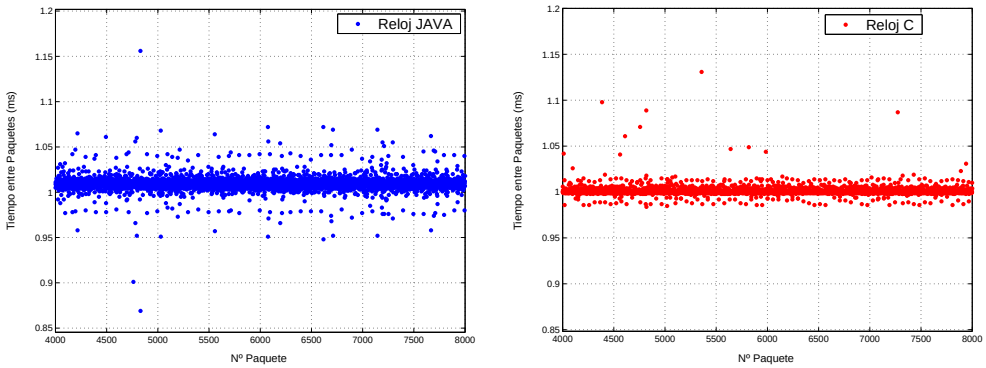


Figura 4.17: Precisión de las rutinas de espera

4.2.4. Módulo Proveedor de Secuencias

Este módulo es el encargado de enviar y recibir en tiempo real las secuencias multimedia de prueba que se utilizan en los algoritmos de comparación de señales. Si bien su implementación se basa principalmente en herramientas ya desarrolladas (JMF y VLC), podemos decir que la misma fue una de las tareas que más tiempo insumió. En particular, se encontraron serios problemas con las características de streaming y captura (cuando hablamos de captura hacemos referencia a escritura a disco) de secuencias en tiempo real de la JMF y el VLC que complicaron el problema.

Bloques de streaming multimedia: problemas de la JMF y el VLC

Cuando comenzamos con la fase de desarrollo de la herramienta de medida nos encontramos con un problema que en un principio habíamos pasado por alto: la robustez de la plataforma de streaming multimedia a utilizar. Con los algoritmos de estimación de PQoS ya implementados necesitábamos una herramienta que nos permitiera enviar secuencias de tests a través de la red y capturarlas en destino para luego poder probar los algoritmos implementados y realizar tests de calidad subjetivos. Varios meses antes (a principios de año) ya habíamos encontrado una herramienta que permitía hacer todo esto, y con la virtud principal de estar completamente desarrollada en Java: la Java Media Framework (ver anexo G). En su momento realizamos unas cuantas pruebas (basándonos en ejemplos incluidos en la página

web [45]), enviando distintas secuencias de audio y video y al parecer la plataforma funcionaba correctamente. Sin embargo, las pruebas realizadas probaron no ser lo suficientemente exhaustivas porque al momento de utilizar la JMF para chequear la validez de los algoritmos nos encontramos con muchas y no gratas sorpresas.

El primer problema detectado fue en la captura de las secuencias en destino. Aún bajo condiciones óptimas de la red y utilizando equipos de grandes prestaciones para el envío y la captura se obtienen en la mayoría de los casos secuencias de duración menor, perdiéndose la información al principio de las mismas. Atribuimos este problema a la estructura de funcionamiento de la JMF: la misma está compuesta por una especie de "máquina de estados"y el pasaje al estado de captura parece darse al momento de arribo de los datos; esto lleva a que, si al momento de este cambio de estado se estuviese ejecutando algún proceso con mayores prioridades los datos se perderían. Para solucionar este problema decidimos concatenar las secuencias de tests con secuencias muy cortas de silencio o cuadros negros (audio y video respectivamente) para evitar la pérdida de información relevante. Esto nos llevó a tener que modificar nuestros algoritmos de comparación de señales, agregando un etapa de pre-procesamiento de los datos para remover la información agregada.

El segundo problema encontrado fue que en la recepción de secuencias de alta tasa de bits (p.e. en el caso de video) no es posible para la JMF (aún bajo condiciones óptimas de la red) procesar todos los paquetes que arriban, resultando en secuencias con menor cantidad de frames. Si bien resulta un tanto difícil notar la diferencia visualmente, al correr los algoritmos de comparación los resultados no son los esperados (es decir, si la captura fuese correcta, la secuencia capturada debería ser exactamente igual a la distorsionada y por ende la comparación frame por frame debería arrojar resultado nulo). En el caso de audio y con la salvedad de la pérdida de información al principio, la herramienta funciona correctamente. Este problema resulta realmente crítico para nuestros algoritmos de comparación de secuencias de video, ya que ninguno de ellos compensa desfasajes temporales (como por ejemplo sí lo hace el PESQ en audio). En este punto se nos plantearon tres posibles caminos a seguir:

1. Agregar a los algoritmos de comparación de secuencias de video una fase de sincronización temporal.
2. Conseguir otra herramienta de streaming con mejores características que las ofrecidas por la JMF para el caso de video.
3. Intentar estimar los números de frames perdidos en la transmisión (ya sea por problemas en la red o por el problema antes descrito) para realizar la comparación de forma coherente.

Sin duda alguna la mejor solución es la primera; sin embargo, una buena implementación de la etapa de sincronización temporal no resulta trivial (una simple correlación daría claramente resultados pobres) y probablemente insumiría demasiado tiempo (escapando al alcance de este proyecto). La solución número 3 fue en una primera instancia la adoptada; dado que

los códigos fuente de gran parte de la JMF están disponibles para su uso libre, nos concentramos en identificar que parte podría ser modificada para obtener la información deseada. Después de mucho buscar encontramos una clase de software encargada del depaquetizado de la información que ofrecía, para cada paquete arribado, la estampa de tiempo asignada en el proceso de empaquetado para su eventual reconstrucción en destino. Estas estampas nos permitían entonces saber que paquetes no habían llegado al destino (por haberse perdido en la red o por no ser levantados por la JMF), evitando así correr la comparación para los frames identificados. Si bien esta solución parecía muy buena a priori, no era lo bastante general como para ser la definitiva. En efecto, el problema de captura de tiempo real de la JMF complicaba la implementación del otro grupo de algoritmos (los que utilizan las RNN como sistema de aprendizaje) ya que la relación entre el estado de la red y la clasificación subjetiva no sería correcta. Finalmente decidimos buscar otra herramienta de streaming que contemplara todos los casos presentados.

En este punto recordamos que otro de los grupos de proyecto del área de redes de datos estaba trabajando con un servidor de streaming de video que al parecer ofrecía muy buenas prestaciones: el Video Lan Client(ver [48]). Si bien el desarrollo de este software no es en código Java (esto complica mucho su integración a nuestra herramienta de medida), decidimos darle una oportunidad y nos embarcamos en su prueba. Los resultados obtenidos fueron bastante buenos; en particular, el problema de captura en tiempo real de la JMF fue superado y la correlación de la calidad percibida con el estado de la red comenzó a ser más coherente. Sin embargo, encontramos nuevos problemas con la captura a disco. Para algunos formatos de video (MPEG4 en particular) la escritura a disco que implementa el VLC corrompe de alguna forma las cabeceras de información, siendo imposible acceder luego al archivo capturado por otros medios distintos al VLC. Esto limitaba fuertemente el uso de nuestros algoritmos de comparación de señales a aquellos formatos de video que podían ser capturados correctamente (MPEG1 y MPEG2). Igualmente y por cuestiones de tiempo decidimos trabajar con esta herramienta para el caso de video, agregando al problema la restricción del tipo de codificación para el uso de algoritmos de comparación.

En síntesis, la solución final para la plataforma de streaming se compone del uso de la JMF para la transmisión de secuencias de audio (con la salvedad del agregado de silencios cortos al principio de cada secuencia de test) y del VLC para la transmisión de secuencias de video (agregando la restricción del tipo de codificación aceptado). Los bloques de streaming de audio y streaming de video descritos en la sección 4.2 implementan las funcionalidades de manejo de estas dos herramientas respectivamente.

Bloque proveedor de secuencias originales

Este bloque es el encargado del manejo de las secuencias multimedia de test que se inyectan en la red para el caso de los algoritmos de comparación de señales. Estas secuencias de test fueron elegidas de acuerdo a distintas categorías de servicio que el usuario puede seleccionar al momento de solicitar la estimación de PQoS. En el caso de video se dispone de secuencias representativas de las siguientes categorías:

1. Acción o Deportes (secuencias de video de alta cantidad de movimiento).
2. Videoconferencia o Noticias (secuencias de video de baja cantidad de movimiento).
3. Otras (secuencias de video de cantidad de movimiento medio).

Para el caso de audio se dispone de secuencias con distintas codificaciones, incluyendo PCM, G.723 y GSM. En este caso la selección de la secuencia de test no es realizada por el usuario sino que el servidor se encarga de ello, de acuerdo al servicio a prestar.

Otra de las funcionalidades que ofrece este bloque es la de consulta de las características del video a transmitir (en particular, tasa de bits y tasa de cuadros); esta información es utilizada por los algoritmos no intrusivos (por la RNN) para realizar la estimación de PQoS.

4.3. Metodologías de medida de PQoS - Casos de Uso

Una vez descrita la implementación del software y a modo de resumen se presentan los casos de uso típicos de la herramienta desarrollada. La descripción de los mismos es completamente informal y no intenta en absoluto seguir las reglas del UML; su finalidad es resumir la secuencia de eventos involucrados en la estimación de la PQoS.

En ambos casos de uso, el cliente (usuario) está interesado en estimar cuál será la PQoS que experimentará al utilizar un servicio multimedia de tiempo real (streaming de audio y/o video) contra cierto servidor (en nuestro caso, otro cliente). La distinción que se presenta entre el uso de metodologías intrusivas o no intrusivas radica en la diferencia de los eventos que se desencadenan en cada caso. Para el usuario final, la elección del uso de una u otra técnica es en principio transparente (ver A.7). Para terminar, cabe aclarar que al hablar de cliente se hace referencia al software que corre en una de las puntas y no al usuario de la herramienta.

Estimación de la PQoS mediante técnicas intrusivas

En este caso se utiliza una técnica de estimación de tipo intrusiva (ver capítulo 3). Recordemos que el esquema de funcionamiento de nuestra herramienta es del tipo cliente-servidor, donde los roles de ambas puntas se deciden por cuál es el extremo que solicita la medida. El caso de uso es desencadenado por el usuario de la herramienta al momento de solicitar la estimación:

1. **Usuario** solicita estimación de PQoS, seleccionado tipo y categoría de servicio.
2. **Cliente** envía pedido de conexión al **Servidor**.
3. **Servidor** recibe pedido de conexión y se establece la conexión de control.
4. **Servidor** pregunta al cliente la metodología de estimación a utilizar.

5. **Cliente** informa que la metodología a utilizar es de tipo intrusiva.
6. **Cliente** informa tipo y categoría de servicio seleccionados por el **Usuario**.
7. De acuerdo a estos datos el **Servidor** selecciona la secuencia multimedia a enviar, avisando al **Cliente** del comienzo de la transmisión.
8. **Cliente** se prepara para capturar dicha secuencia, indicando al **Servidor** que inicie la transmisión.
9. **Servidor** envía secuencia seleccionada.
10. **Cliente** captura la secuencia enviada.
11. **Cliente** ejecuta algoritmo de comparación de secuencias, utilizando la secuencia capturada y la secuencia original.
12. **Cliente** informa al **Usuario** los resultados de la estimación.

Estimación de la PQoS mediante técnicas no intrusivas

En este caso se utiliza una técnica de estimación de tipo no intrusiva (ver capítulo 3). Nuevamente el caso de uso es desencadenado por el usuario de la herramienta al momento de solicitar la estimación:

1. **Usuario** solicita estimación de PQoS, seleccionado tipo de servicio.
2. **Cliente** envía pedido de conexión al **Servidor**.
3. **Servidor** recibe pedido de conexión y se establece la conexión de control.
4. **Servidor** pregunta al **Cliente** la metodología de estimación a utilizar.
5. **Cliente** informa que la metodología a utilizar es de tipo no intrusiva.

En este punto se asume que el servidor conoce las características del servicio que brindará al cliente. Más precisamente, en el caso de streaming de video el servidor sabe cuál es el video a transmitir (y por ende puede conocer sus características); en el caso de streaming de audio conoce la codificación que utilizará. Como se verá en el capítulo 5, el servidor debe conocer estos datos para realizar la estimación. En la herramienta de software desarrollada se permite al usuario del servicio establecer estos parámetros (ver anexo B).

6. **Servidor** estima las características del servicio solicitado por el **Cliente**.
7. **Servidor** avisa al cliente del inicio de la estimación de los parámetros de red.
8. **Cliente** pone en marcha el bloque rebotador de paquetes, indicando al **Servidor** que inicie la transmisión.

9. **Servidor** envía y recibe paquetes de test con características similares a las del servicio solicitado por un lapso de aproximadamente 20 segundos.
10. **Servidor** procesa la información de los paquetes de test, estimando los parámetros del enlace con el cliente.
11. En base a las características del servicio y los parámetros de red estimados, el **Servidor** ejecuta algoritmo de estimación de PQoS e informa al **Cliente** los resultados de la misma.
12. **Cliente** informa al **Usuario** los resultados de la estimación.

4.4. Posibles aplicaciones de la herramienta desarrollada

Para finalizar con este capítulo presentaremos a continuación algunos posibles usos de la herramienta desarrollada:

- **Monitoreo continuo de calidad**

  El monitoreo continuo de la PQoS permite a un usuario final o ISP conocer las características del servicio recibido/ofrecido a lo largo del tiempo. Para el usuario puede ser interesante identificar los horarios donde recibirá mayor calidad para hacer uso del servicio en los mismos; para el ISP puede ser de utilidad conocer los períodos con mayores restricciones de calidad para incentivar el consumo diferenciado.

- **Verificación de contratos de servicio**

  Es posible para el usuario poder verificar si se cumplen o no los acuerdos de servicio acordados con el proveedor. La herramienta permite al usuario tener una evidencia tangible de la calidad experimentada.

- **Venta diferenciada de servicios multimedia**

  El conocimiento de la PQoS para distintas demandas de recursos permite al proveedor de servicios ofrecer tarifas de servicio diferenciadas de acuerdo a la calidad deseada por el usuario.

- **Servicios de streaming multimedia de calidad adaptativa**

  La velocidad de procesamiento obtenida con los algoritmos no intrusivos permite al proveedor de servicios adaptar las características del servicio ofrecido de acuerdo a la PQoS experimentada por el usuario.

- **Diseño de redes**

  Es posible diseñar la estructura de una red de servicios multimedia en base al conocimiento de la PQoS obtenida en los puntos terminales.

4.5. Plataforma de Pruebas

Para poder llevar a cabo los tests subjetivos, calibrar los algoritmos y comprobar los resultados es necesario disponer de una plataforma de pruebas que permita configurar los parámetros de la red. La plataforma de pruebas se compone de un "enrutador"de pruebas que simula pérdidas, retardos y jitter.

Enrutador de pruebas

Como ya fue explicado en el capítulo anterior, es necesario realizar tests subjetivos. Necesitamos para ello poder simular las condiciones de la red de una manera controlada. Con este fin se implementó un "enrutador"que recibe los paquetes en un puerto, los manipula de manera de obtener los parámetros deseados y los envía a la maquina receptora. Los parámetros a manipular son básicamente las pérdidas, el jitter y el retardo.

Modelo de las Pérdidas

Como primera aproximación al problema se utilizó un modelo muy simple para las pérdidas. Cada n paquetes el "enrutador" descarta uno. Si bien tiene la ventaja de ser muy simple, este modelo es muy poco realista.

Un resultado más apropiado se obtiene al considerar un modelo de Gilbert de dos estados (ver anexo E). Este modelo cuenta con un estado donde el paquete es enviado, y otro estado donde el paquete es descartado

- Estado 0: *El paquete es enviado*
- Estado 1: *El paquete es descartado*

El modelo introduce el concepto de pérdidas en ráfagas; la probabilidad de pérdida de un paquete depende de lo sucedido al paquete anterior. Hay probabilidades de pérdida distintas si el paquete anterior fue descartado o no. Se obtiene así un modelo con dos parámetros, el porcentaje de pérdida y el largo medio de las ráfagas. Tomando por ejemplo una red con una probabilidad de pérdida p dada, este modelo permite por ejemplo distribuir dichas pérdidas en pocas ráfagas largas (lo que ocurriría con un buffer de recepción lleno) o muchas ráfagas cortas (lo que es más común en una red inalámbrica).

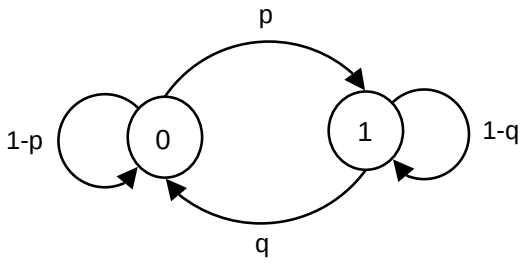


Figura 4.18: Modelo de Gilbert

Implementación del jitter

Para introducir jitter a un flujo de paquetes, el *enrutador* de pruebas implementa un buffer en donde los paquetes son enviados con una cadencia distinta que en la recepción. Enviando los paquetes con un tiempo menor entre ellos que a el tiempo entre paquetes en la recepción se obtienen paquetes con jitter negativo. Para enviar paquetes con jitter positivo, se retrasa el envió del paquete mediante una rutina de espera de alta precisión (con una resolución del orden de 250 μs)

Implementación del retardo

El tamaño del buffer es directamente proporcional al retardo introducido. Modificando el tamaño de los buffers se modifica el retardo. Es en los flujos bidireccionales donde se torna relevante el retardo, un retardo mayor a los $200ms$ provoca una pérdida de interactividad importante, haciéndose casi imposible mantener una conversación con un retardo mayor de $400ms$.

4.6. Realización de los Tests Subjetivos

Los tests subjetivos que se realizaron pretenden medir la degradación sufrida por la señal al transmitirse por la red. De acuerdo con lo mencionado en el capitulo 3 son del tipo DCR. Esta elección se debe a que los métodos objetivos utilizados toman la señal original como la referencia "perfecta", y en base a ésta asignan la calidad de la distorsionada.

Se generó una colección de señales imponiendo los parámetros de la red en el simulador de pruebas (pérdidas, ráfagas, etc.) que en principio podrían tener cualquier granularidad. La cantidad de tests a realizar es directamente proporcional a la cantidad de muestras y al número de parámetros. Dado que insumen bastante tiempo y que es necesario la colaboración de mucha gente, se eligió la cantidad mínima de parámetros y los valores de estos donde sus

Calificación	Concepto
5	No se nota la degradación
4	Se nota pero no molesta
3	Se nota y es un poco molesta
2	Es bastante molesta
1	Insoportable

Cuadro 4.1: Valores posibles para los Tests Subjetivos

efectos se hacen más relevantes.

Se realizaron los tests de la manera más dinámica posible, minimizando su duración para reducir así las molestias causadas a las personas que colaboraron en esta etapa del proyecto. Para esto se generó una lista de reproducción con las señales a observar para utilizarlas de manera casi automática con cualquier reproductor multimedia que hubiese instalado. Se confeccionó un disco con las señales y las listas de reproducción, que permitió hacer los tests en las computadoras de los voluntarios. Utilizamos planillas que contenían un espacio para puntuar las secuencias, los valores a utilizar (tabla 4.1), y unas breves instrucciones.

4.6.1. Tests Subjetivos de Video

La escala utilizada fue la escala clásica de cinco valores para test DCR. Los tests se realizaron, en la medida de lo posible, de acuerdo a las recomendaciones de la ITU como se explica en el Apéndice D.

Para realizar los tests de video se generaron 75 parejas (original y distorsionado) a partir de 40 videos distintos de corta duración (entre 10 y 20 segundos). Estos videos fueron elegidos de forma de poder estudiar la influencia de los parámetros de la red sobre la calidad percibida dependiendo del video original. Se consideraron como parámetros relevantes del video original el tipo de codificación (MPEG-1 o MPEG-4), la tasa de bits del video (como medida de la calidad del mismo) y la cantidad de movimiento presente, separándolos en tres categorías: alta, media y baja.

Otro factor que influyo en la elección de los videos fue el contenido, y el nivel iluminación de los mismos. Se intentó que fueran interesantes, pero que a su vez no despertaran reacciones fuertes en los participantes, de manera de no desviar la atención de los mismos. Se eligieron videos que no fuesen excesivamente oscuros, ya que esto impide visualizar correctamente las perturbaciones.

Las 75 secuencias distorsionadas fueron generadas utilizando el VLC [48] como software para el streaming, y el enrutador antes mencionado. Los parámetros de la red considerados

fueron la probabilidad de pérdida y la duración media de las ráfagas, dado que el jitter tiene similares efectos que las pérdidas como se menciona en 2, y el retardo no influye en la calidad de servicios sin interactividad.

Se varió la probabilidad de pérdida desde 0.2 % hasta 10 % y la ráfaga media de 2 a 20 paquetes. Los valores de pérdida y ráfaga media se eligieron de forma de tener resultados realistas (estos valores definen las probabilidades de transición entre los estados del modelo de Gilbert, ver apéndice E).Por ejemplo ráfagas muy largas y probabilidades de pérdida muy chicas, hacen que sea difícil que ocurra un cambio de estado en videos de 20 segundos.

Las parejas se dividieron en tres grupos de 25 (formando los tests tipo A, B y C), de forma de que en cada uno de ellos no se repitiera ningún video, de que las personas puedan utilizar todos los valores de la escala, y de que la media del test esté cerca del valor medio de la escala (3). Cada participante recibe una planilla correspondiente al tipo de test que va a realizar y posteriormente se muestran los videos. Primero se pasa el video original y después el distorsionado, seguido de 10 segundos de pantalla negra, momento en el que los participantes asignan el valor de calidad.

4.6.2. Tests Subjetivos de Audio

Para los tests de audio se grabaron 24 señales con frases cortas (aprox. 6 segundos), donde cada frase tiene sentido, pero no tiene conexión el contenido entre señales distintas. Para evitar el acostumbramiento a la voz, las señales fueron grabadas por dos locutores distintos. Para transmitir las señales se utilizaron las herramientas descritas en la sección 4, variando la codificación. De las codificaciones disponibles se consideraron aquellas utilizadas con mayor frecuencia en aplicaciones de VoIP. Los codecs utilizados fueron GSM, G.723, y PCM $\mu - law$. Cada una de las 24 señales fue codificada en los tres codecs mencionados y se utilizó el "enrutador" de pruebas para simular los efectos de la red. En este caso se consideraron pérdidas desde el 1 % hasta un 40 % y ráfaga media de 2 a 19 paquetes.

Al transmitir las señales de audio y medir los parámetros impuestos a la red, observamos que el modelo de pérdidas que funcionaba bien con los videos, con el audio daba resultados impredecibles. Investigando las causas, se conluyó que el número de paquetes involucrados en un stream de audio de pocos segundos de duración no eran suficientes para lograr una estadística decente. Rara vez el número de paquetes enviados superaba los 150, por lo que en algunos casos (p.e. pérdidas muy bajas) no se perdían paquetes. La solución encontrada a este problema fue modificar un poco el "enrutador", de forma tal que registrara lo sucedido con cada paquete de la señal de audio enrutado. En base a esta información se calcula la probabilidad de pérdida y la ráfaga media.

Al igual que en los videos, se generaron tres tipos de test (A,B,C) cada uno formado por 24 señales. A cada participante se le otorga una planilla correspondiente al tipo de test que va a realizar y posteriormente se le hace escuchar las señales. Primero se pasa la señal original

y después la distorsionada, seguido de un silencio de unos 8 segundos, tiempo necesario para pensar y asignarle un valor a la degradación.

4.6.3. Conclusiones sobre la realización de los Tests Subjetivos

La realización de los tests subjetivos permitió comprobar lo costoso (en términos de tiempo en este caso) que resultan. Tomando en cuenta que los tests de video tenían una duración de 30 minutos más unos 10 minutos para prepararlo y explicar la metodología, realizar el test a unas 50 personas insume aproximadamente unas 35 horas. Además de este tiempo hay que considerar el tiempo que lleva procesar los datos. El tiempo insumido en realizar los tests subjetivos fue aproximadamente una semana, sin contar el tiempo involucrado en la generación de las secuencias y en el procesamiento de datos.

En las tablas 4.2, 4.3 se muestra el valor medio y la desviación estándar media de los tests. Se puede observar que la media en todos los tests está muy cercana al valor medio de la escala, que era una de las metas al diseñarlos.

Tipo de Test	**Valor medio**	**Media de las desviación estándar**
Test A	3,1089	0,5138
Test B	3,0218	0,5110
Test C	2,9533	0,4918

Cuadro 4.2: Promedios estadísticos de los tests de Video

Tipo de Test	**Valor medio**	**Media de las desviación estándar**
Test A	3,0379	0,7573
Test B	3,1111	0,6724
Test C	2,9757	0,7501

Cuadro 4.3: Promedios estadísticos de los tests de Audio

Se observa que la media de la desviación estándar es más alta en los tests de audio que en los tests de video (casi un 50 % más). Los propios participantes de los tests manifestaban que era más sencillo puntuar los videos que el audio. Además algunas señales no presentaban la inteligibilidad deseable, ya sea por las condiciones en que fueron grabadas como por la propia dicción del locutor.

Capítulo 5

Resultados y Validación del Software

5.1. Resultados de los algoritmos intrusivos de Audio

En esta sección se presentan y analizan los resultados obtenidos por los algoritmos objetivos de audio. Para poder comparar el desempeño de los algoritmos se utiliza el coeficiente de correlación y el error absoluto medio. El coeficiente de correlación indica si los datos están alineados. Su valor está entre 0 y 1 (en módulo); valores cercanos a 1 indican una relación lineal entre los datos, mientras que valores cercanos a 0 indican lo contrario. El error absoluto medio nos da una idea de cuanto nos alejamos de la medida en promedio. Cuanto mayor es el error absoluto medio menos confiable es la medida.

Resultados del algoritmo PESQ

Este algoritmo fue el elegido por la ITU-T entre varios (entre ellos MNB y el EMBSD) para convertirse en un estándar (P.862). Es de esperar que su desempeño sea superior al de los otros algoritmos.

En la figura 5.1 se muestra una gráfica que contiene los resultados de los tests subjetivos (DMOS) contra la salida de PESQ. Se puede observar una relación lineal entre DMOS y PESQ. El coeficiente obtenido muestra una alta correlación entre PESQ y DMOS .

- Coeficiente de Correlación entre PESQ y DMOS $\mathbf{R = 0.88}$

Se grafica además la recta que mejor se ajusta a los datos. En la literatura [4] se presentan resultados para la correlación cercanos a 0,95. La diferencia era esperable, considerando

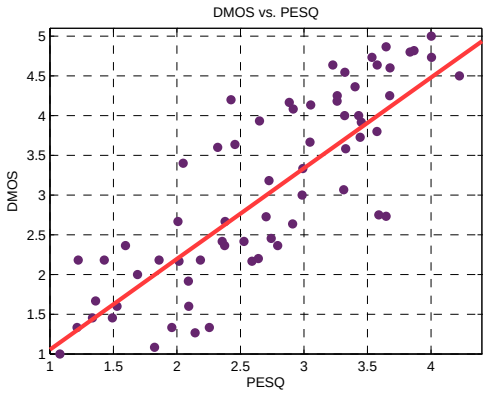


Figura 5.1: Resultados del algoritmos PESQ

la forma en que fueron realizados los tests subjetivos. Si bien se intentó seguir la norma, fue imposible ajustarse a todos los requisitos. De todas maneras se obtuvo una correlación importante.

Análisis de los resultados

La norma indica que el algoritmo esta calibrado de manera general para dar un valor en la escala clásica de DMOS (1 a 5). Observando las diferencias entre los valores de DMOS y los valores de PESQ, se observa que el error absoluto medio es 0.56. Calibrando el algoritmo para nuestro set de datos particular se espera obtener mejores resultados. Con este fin se dividió la muestra en dos, con el 70 % se realizó una regresión lineal, validando los resultados con el 30 % restante. El error absoluto medio se reduce en este caso a 0.43.

Para realizar un análisis cualitativo de los datos se divide la escala de DMOS en tres niveles de calidad de servicio según la siguiente tabla.

DMOS	**Calidad**
$DMOS > 3{,}7$	**buena**
$3{,}7 > DMOS > 2{,}3$	**regular**
$DMOS < 2{,}3$	**mala**

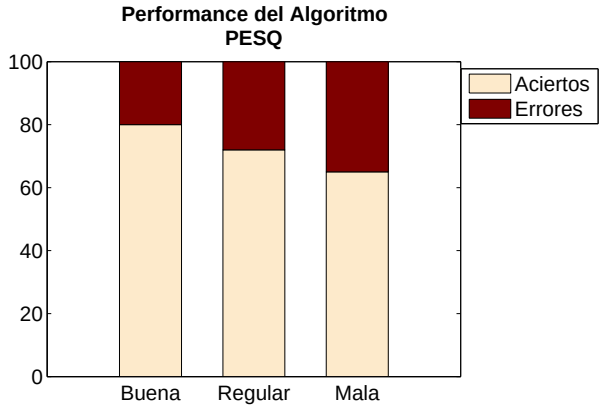
Cuadro 5.1: Clasificación del Servicio VoIP

Se observa que en promedio PESQ califica satisfactoriamente un 73 % de los casos. Si se desglosa por nivel de servicio, se observa una mejor predicción para servicios de alta calidad (señales poco degradadas). En esta franja PESQ tiene un 80 % de aciertos. Sin embargo, en el nivel de servicio malo se obtienen los peores resultados, obteniéndose un 65 % de aciertos.

Una observación importante es que la diferencia entre PESQ y DMOS nunca es tan grande como para que una señal clasificada como servicio bueno sea clasificada por PESQ como servicio malo y viceversa. Los resultados para cada nivel se muestran a continuación.

Clasificación DMOS-PESQ	**Bueno**	**Regular**	**Malo**
Bueno	80 %	20 %	0 %
Regular	19 %	72 %	9 %
Malo	0 %	35 %	65 %

Cuadro 5.2: Resultados por categoría de servicio.



Las señales que clasifican en el último nivel por lo general tienen una probabilidad de pérdida alta. Las pérdidas influyen de dos maneras: directamente por la falta de información, e indirectamente aumentando la desalineación como se verá más adelante. PESQ incluye un etapa de alineación temporal diseñada para ajustar retardos variables. Si bien esta etapa amortigua el efecto no lo elimina completamente. Se justifica el peor desempeño en las señales de calidad baja por este último motivo.

Resultados del algoritmo EMBSD

El algoritmo EMBSD cuenta con el modelo de la percepción más complejo; es el único de los utilizados en el proyecto que incluye el efecto del post-enmascaramiento. Sin embargo no posee ningún tipo de consideración en lo que alineación se refiere.

A diferencia de PESQ, EMBSD devuelve un valor que es una medida de la distorsión perceptualmente relevante introducida a la señal.

- Coeficiente de Correlación entre EMBSD y DMOS $\mathbf{R = -\ 0.77}$

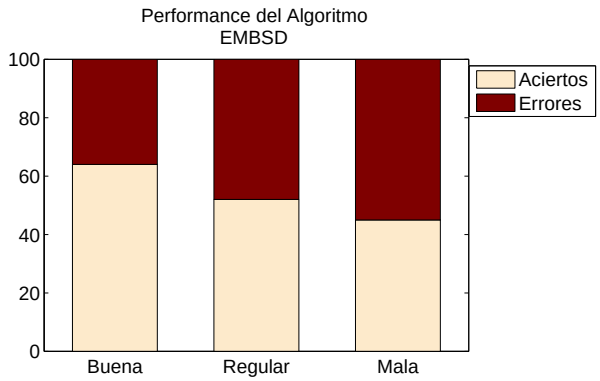
Se obtuvo una correlación negativa entre el resultado del algoritmo y el DMOS como era de esperar.

Se realizó una regresión lineal con el 70 % de los datos, y se comprobó el resultado con el 30 % restante. El resultado fue un error medio de 0,77. El efecto del error se hace más notorio si se divide el servicio en tres calidades como se hizo con PESQ. En este caso el algoritmo clasifica satisfactoriamente en el 55 % de los casos.

Si desglosamos por categoría observamos un caso en que la señal era de categoría buena, y el algoritmo clasifica como de categoría mala.

Clasificación DMOS-EMBSD	**Bueno**	**Regular**	**Malo**
Bueno	64 %	32 %	4 %
Regular	19 %	52 %	29 %
Malo	0 %	55 %	45 %

Cuadro 5.3: Resultados por categoría de servicio.



Además se observa en el cuadro 5.3 que de las señales que entran en la categoría malas solo un 45 % fueron correctamente categorizadas. Como veremos mas adelante esto se debe fundamentalmente a el desfasaje introducido en las señales por las perdidas.

Resultados del algoritmo MNB

La idea original de MNB es la de medir distorsiones perceptualmente significativas a distintas escalas de tiempo y frecuencia. El autor sostiene que la manera natural de los humanos de comparar es yendo de escalas grandes a escalas pequeñas. Al igual que EMBSD, el algoritmo MNB no cuenta con ningún mecanismo de sincronización, por lo que es de esperar que con pérdidas grandes (señales muy degradadas) se comporte peor. El autor propone dos estructuras (MNB-I y MNB-II).

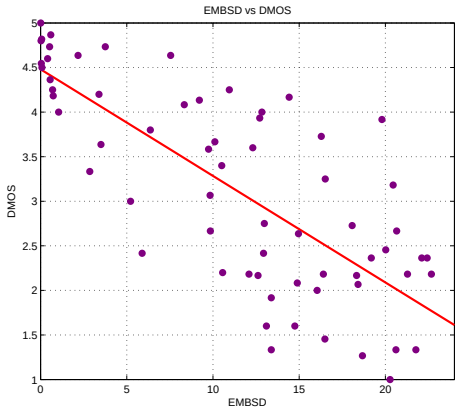


Figura 5.2: Resultado del algoritmos EMBSD

- Coeficiente de Correlación entre MNB-I y DMOS **R = - 0.68**
- Coeficiente de Correlación entre MNB-II y DMOS **R = - 0.71**

De las figuras 5.3 y 5.4 se puede apreciar que los resultados son muy similares, siendo la estructura MNB-II la que da mayor correlación con el DMOS. Por lo tanto en adelante estudiaremos esta estructura. El error absoluto medio es de 0.66, valor intermedio entre PESQ y EMBSD.

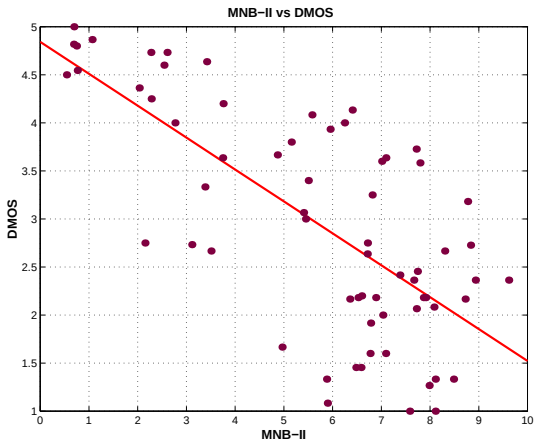


Figura 5.3: Resultado del algoritmos MNB-II

Estudiando por niveles como en los casos anteriores se observa que MNB-II predice satisfactoriamente un 52 % de los casos. Observando el detalle por categoría el peor desempeño se

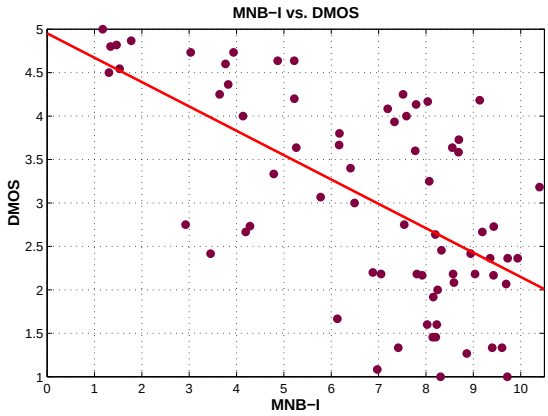
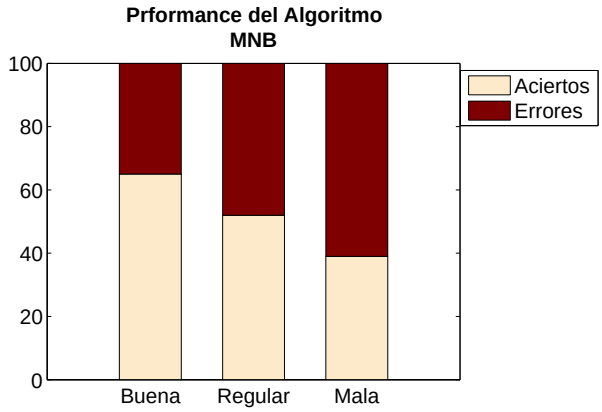


Figura 5.4: Resultado del algoritmos MNB-I

da en el nivel bajo. Esto se puede ver en la gráfica 5.3, la dispersión de los puntos aumenta a medida que aumenta el valor de MNB-II . Si miramos en la categoría de calidad buena, una secuencia es clasificada como mala.

Clasificación DMOS-MNB-II	**Bueno**	**Regular**	**Malo**
Bueno	65 %	30 %	5 %
Regular	13 %	52 %	35 %
Malo	0 %	61 %	39 %

Cuadro 5.4: Resultados por categoría de servicio.



Algunas Conclusiones

En la figura 5.5 se aprecia que cuando ocurren pérdidas, las señales se desfasan a partir de ese punto. Los algoritmos que no tienen en cuenta este efecto comparan partes distintas de la señal, dando un resultado poco confiable. Esto implica que el resultado dependerá fuertemente de la posición de las pérdidas.

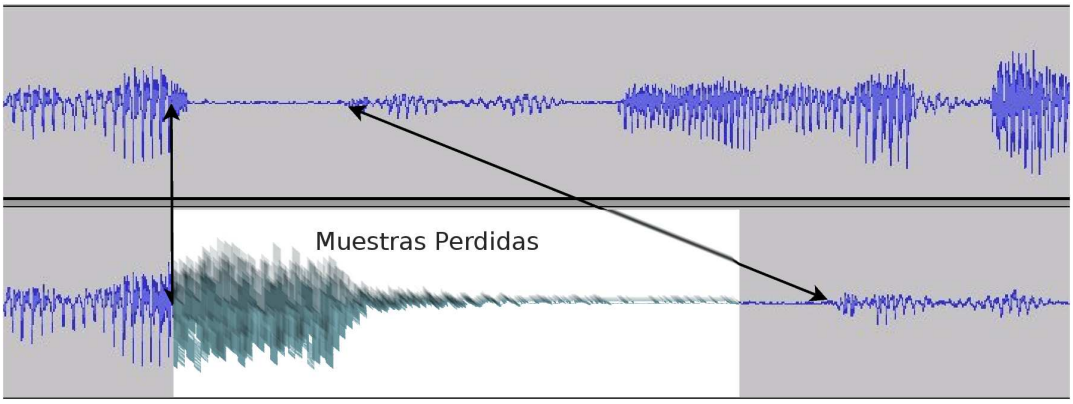


Figura 5.5: Desfasaje temporal entre la señal original y la señal distorsionada

Para cuantificar el efecto de la desalineación se realizó el siguiente experimento. Dos señales de voz con exactamente la misma distorsión; ésta ocurre sobre la misma palabra y dura el mismo tiempo. En un caso la distorsión se da al comienzo y en el otro al final. Podemos asegurar que si hiciéramos tests subjetivos se obtendría el mismo valor de DMOS. A continuación se comparan los resultados de los tres algoritmos.

Algoritmo	**Pérdidas al comienzo**	**Pérdidas en el medio**	**Pérdidas al final**
PESQ	3.9	3.7	3.7
EMBSD	2.4	3.1	4.6
MNB-II	2.42	3.0	4.4

Como se puede apreciar el resultado de PESQ varía muy poco al cambiar la pérdida de lugar. Sin embargo las predicciones de DMOS de EMBSD y MNB-II varían entre 2.4 y 4.5 a medida que la pérdida ocurre más sobre el final de la señal. La causa detrás de estas variaciones es la alineación temporal. Cuando la pérdida ocurre al principio de la señal, esta queda desalineada con la original a partir de ese punto. Los algoritmos que no tienen en cuenta la alineación (EMBSD,MNB) consideran como distorsión segmentos de la señal que simplemente están desfasados. PESQ, que cuenta con un potente algoritmo de sincronización resulta menos perjudicado por este efecto.

Observando este último resultado, se puede concluir que la alineación juega un papel fundamental en el desempeño de los algoritmos. PESQ fue el que obtuvo los mejores resultados.

Los otros dos algoritmos fallan a medida que disminuye el nivel de calidad, llegando a obtener más errores que aciertos en la clasificación.

5.2. Resultados de los algoritmos intrusivos de Video

En esta sección se presentan los resultados obtenidos para los algoritmos intrusivos de video utilizados.

A diferencia de los resultados presentados en el capitulo 3 (3.4.4) con los videos del VQEG, aquí se consideran los videos generados mediante la herramienta de streaming y el enrutador antes mencionado. Estos dos tipos de video presentan diferencias sustanciales. Los primeros tiene diferentes tipos de distorsiones (TV analógica y digital), la luminancia media de los mismos es reajustada luego de la distorsión, y lo que es más importante la duración del video original y el distorsionado es la misma. Esto es, a cada frame del video original le corresponde un frame en el distorsionado. En cambio, los videos generados no tienen ningún post-procesamiento (se quiere estimar la calidad "en servicio"), y la duración del video original y el transmitido no es la misma; o sea que hay frames en el video original que no aparecen en el transmitido. Esto se debe a la pérdida de paquetes en la red, y a que no se implementa ningún tipo de recuperación frente a pérdidas (por ejemplo repetir el último frame recibido correctamente). Esta diferencia de tamaño torna inválida la visión clásica del video distorsionado como el original más una señal de error.

En la figura 5.6 se muestra el resultado del DMOS obtenido de los test subjetivos, contra el valor del SSIM; en la figura 5.7 la salida del MSE contra el DMOS, figura 5.8 el resultado del PSNR contra el DMOS, y en la figure 5.9 el resultado del ITS contra el DMOS.

Como se puede observar en las figuras la correlación entre los resultados de los algoritmos y el valor subjetivo es pobre. A diferencia de lo que ocurría con los videos del VQEG, en este caso el que presenta una mayor correlación es el PSNR. Para cuantificar el grado de correlación de los ajustes calculamos para cada uno el error absoluto medio. Si bien el menor error se presenta en el PSNR, el valor del mismo es comparable con el del resto de los algoritmos. Los valores obtenidos para el error absoluto medio son: $SSIM = 0{,}60$, $MSE = 0{,}49$, $PSNR = 0{,}48$ e $ITS = 0{,}53$.

Cabe aclarar que en las gráficas presentadas se excluyó un punto correspondiente a un video de muy baja calidad (en su versión original), para el cual las personas asignaron valores de degradación muy grandes que no correspondían con la realidad.

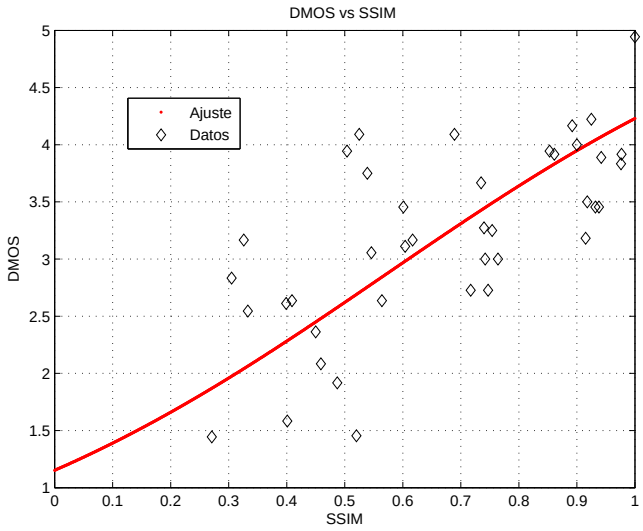


Figura 5.6: Resultados SSIM vs DMOS, Error Absoluto Medio= 0.60

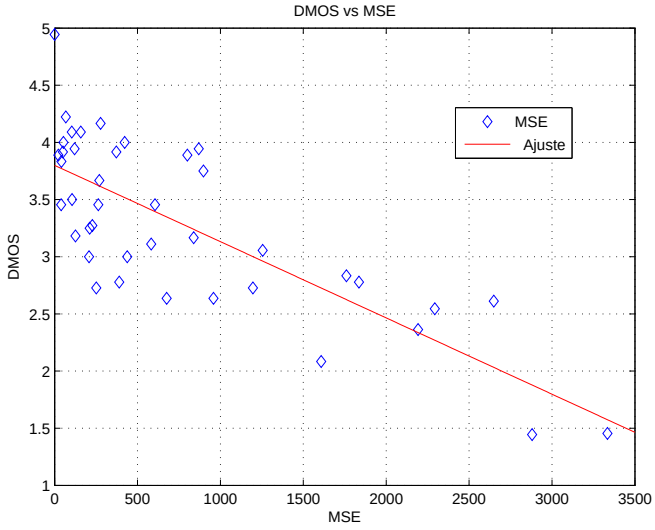


Figura 5.7: Resultados MSE vs DMOS, Error Absoluto Medio= 0.49

Causas de error identificadas

Las causas de esta baja correlación pueden ser varias. Primero hay que recordar que no existe un algoritmo infalible para la asignación de calidad percibida en video. A esto se le

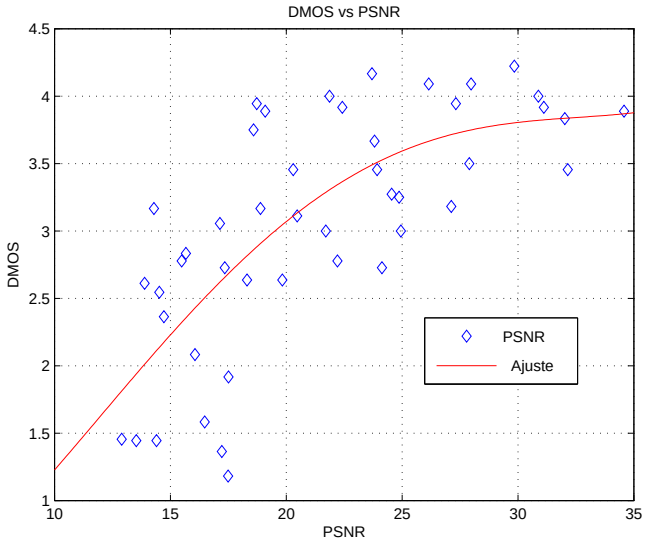


Figura 5.8: Resultados PSNR vs DMOS, Error Absoluto Medio= 0.48

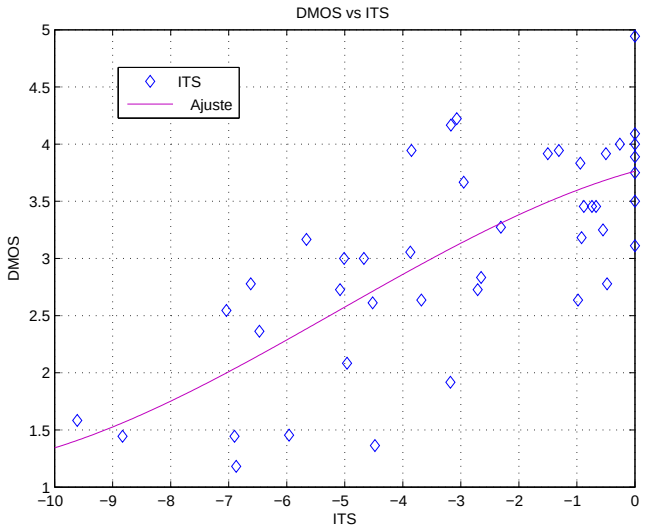


Figura 5.9: Resultados ITS vs DMOS, Error Absoluto Medio= 0.53

suman problemas propios de implementación.

La pérdida de frames antes mencionada produce un desfasaje en la comparación, por lo que el algoritmo termina comparando frames distintos. Ésta ocurre cuando hay pérdidas en ráfaga durante la transmisión.

El resultado de este efecto sobre la correlación con los resultados subjetivos depende de la ocurrencia de las ráfagas. Por ejemplo, en los videos que sólo ocurren pocas ráfagas aisladas, las personas parecen no molestarse demasiado, sin embargo esto produce el desfasaje en el algoritmo. A su vez las consecuencias de dicho desfasaje dependen del video en cuestión, sobre todo de la cantidad de movimiento presente en el mismo, y en el momento en que ocurra.

Para ver la dependencia con la cantidad de movimiento se realizó la siguiente prueba, se le quitaron los dos primeros frames a dos videos y se compararon por medio del SSIM con el video original respectivo. Se utilizó un video titulado *orange*, en el cual el fondo es bastante estático y los niveles de luminancia son muy similares (como dice el nombre es todo naranja); por lo que la diferencia entre frames consecutivos es chica. El otro video titulado *donut* presenta una cantidad de movimiento mucho mayor que orange. El resultado del SSIM fue 0,85 para el primero y 0,60 para el segundo.

En cuanto al momento en que ocurre el desfasaje, es claro que si se pierden frames al principio del video, el resultado del algoritmo será peor que si se pierden la misma cantidad de frames al final del mismo. Esto se ve claramente en dos versiones del video llamado *bungge*, los dos tienen un $DMOS = 4$, pero por el motivo antes expuesto el SSIM tiene como resultado 0,6 en uno y 0,9 en el otro.

El problema del desfasaje se podría evitar agregándole a los algoritmos implementados un módulo que permita sincronizar la comparación. Dicho módulo no parece ser fácilmente implementable, teniendo en cuenta que hay frames muy deteriorados, y agregaría mucho tiempo de cómputo. Otra posible solución, más factible en el marco del presente proyecto, seria reconocer los frames que se pierden en el momento de la recepción. Esto se lograba con la JMF ya que se tenía acceso a la clase depaquetizadora, y observando las estampas de tiempo se podía determinar los frames perdidos. Por motivos antes expuestos se decidió no utilizar la JMF y por lo tanto este método no fue implementado.

Otro problema radica en la diferencia que existe entre lo decodificado por la herramienta de streaming (VLC) y lo que efectivamente recibe el algoritmo para la comparación. Esto hace que el video distorsionado que ven las personas sea diferente al que utiliza el algoritmo.

En la figura 5.10 se muestra el frame utilizado por el algoritmo (esto es lo que se obtiene de la FFmpeg). Como se puede observar al comparar con la figura 5.11, la cual es el mismo frame pero desplegado por el VLC, la diferencia es apreciable. Esto se produce cuando los frames están muy dañados y no se puede decodificar algún macro bloque. Este efecto se puede

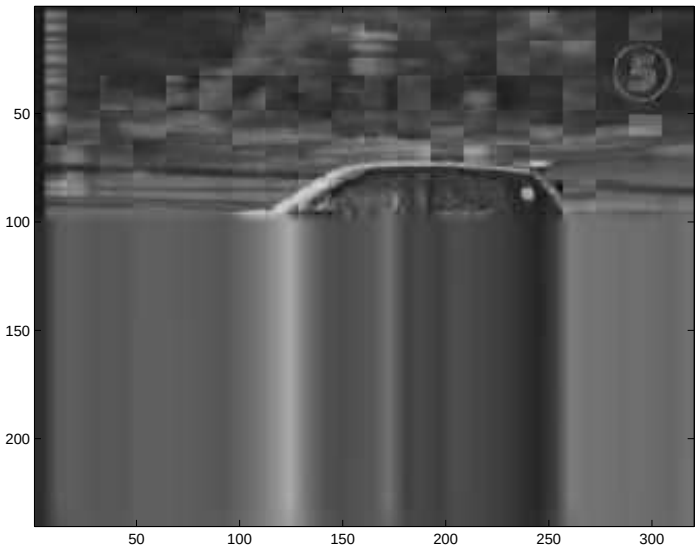


Figura 5.10: Frame decodificado por FFmpeg

Figura 5.11: Frame decodificado por el VLC

propagar a otros frames en el caso que el macro bloque sea tipo I.

El problema se debe a no utilizar una misma herramienta para desplegar a pantalla y

decodificar el video, frame por frame, para usar el algoritmo. Esta unidad se pretendía lograr en un principio mediante el uso de la JMF, pero por todos los problemas expuestos anteriormente, se tuvieron que integrar diferentes herramientas.

Conclusiones

La implementación actual de los algoritmos enfrenta dos problemas graves. Estos son el desfasaje que se produce en la comparación debido a la pérdida de frames, y la diferencia entre lo observado por las personas y lo que efectivamente compara el algoritmo. Estos problemas afectan de forma diferente a los distintos algoritmos. A su vez el impacto depende del video en cuestión, como ya se mencionó anteriormente. A pesar de estos problemas se obtiene una predicción razonable, con un error absoluto medio de 0,5. Teniendo en cuenta que los valores subjetivos de DMOS tienen un intervalo de confianza del 95 % de 0,4 en media, el error que tienen los algoritmos es aceptable. Para disminuir el valor del error absoluto medio es necesario eliminar los problemas antes expuestos.

De los algoritmos considerados el que tiene el ajuste con menor error es el PSNR, con el beneficio adicional de ser el que menos demora en ejecutarse. Por lo tanto en la implementación actual es el algoritmo recomendado.

5.3. Resultados de los algoritmos no intrusivos de Video

En esta sección se presentan los resultados obtenidos con las Random Neural Networks (RNN) en la estimación de la calidad percibida en video.

La implementación de las redes consta de dos partes, la primera es generar la red con una estructura dada y entrenarla con un conjunto de configuraciones (valores de entrada y salida correspondiente). La segunda etapa es utilizar dicha red como un sistema de clasificación automático.

Se pretende estimar la calidad percibida a partir del estado de la red y del tipo de servicio que se pretenda transmitir. Los parámetros posibles de entrada son la probabilidad de pérdida y la duración media de las ráfagas de pérdida en la red, y distintos parámetros propios del video a transmitir. Estos son la tasa de bits, tasa de cuadros, cantidad de movimiento presente y la codificación (MPEG-1 o MPEG-4). La salida es el valor de DMOS.

Los datos para entrenar la red surgen de los tests subjetivos realizados. En el momento de generar los videos distorsionados se estimaba el estado de la red (probabilidad de pérdida y ráfaga media). Esto permitió generar 75 configuraciones distintas de parámetros de la red,

parámetros del video y el valor de DMOS correspondiente.

El primer paso es crear una red con una estructura dada. La estructura está dada por la cantidad de neuronas de entrada (cantidad de parámetros de entrada), el número de neuronas escondidas y el número de neuronas de salida (parámetros de salida, uno en nuestro caso). Las neuronas escondidas se pueden pensar como parámetros libres para realizar el ajuste de la función (que se supone existe) entre entradas y salidas. Cuantas más neuronas escondidas se tomen mejor será el ajuste a los valores de entrenamiento, pero si se toman muchas la red se ajusta demasiado a dichas muestras y pierde la capacidad de predicción. Se dice que la red está sobre entrenada. Se presenta un compromiso entre el error del ajuste a las muestras de entrenamiento y la capacidad de predicción de la red. De acuerdo a los valores presentados en la literatura existente y a experiencia propia se tomaron diez neuronas escondidas.

La primer estructura considerada fue con tres entradas: probabilidad de pérdida, ráfaga media y codificación. Se observó que independientemente del número de neuronas escondidas que se tomaran el ajuste era muy malo. Esto se debe a que la función de la que se hablaba antes no queda bien determinada con estas entradas. Se fueron agregando entradas y se observó que el error del ajuste disminuía por lo que al final se consideraron 6 entradas, que son todos los parámetros disponibles.

Entrenamiento de la RNN

En la figura 5.12 se muestra el resultado del ajuste de la red utilizando todos los datos para su entrenamiento. Para ello se compara el valor de DMOS estimado por la RNN contra el valor utilizado en el entrenamiento de la red. El error absoluto medio de la estimación es de 0,50.

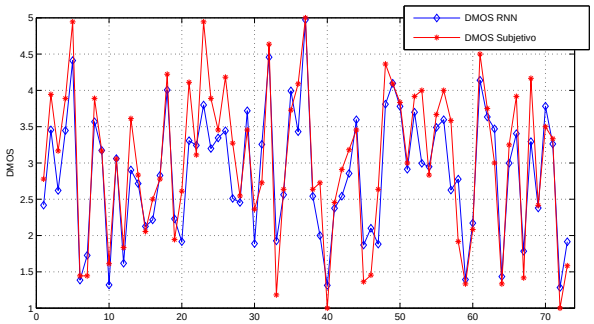


Figura 5.12: DMOS RNN y DMOS subjetivo para las muestras de entrenamiento

Se puede observar que el ajuste es en general bueno. Las diferencias más grandes se dan en los valores extremos. Esto se debe a la existencia de pocas muestras de entrenamiento con valores muy altos de DMOS o muy bajos. El resultado es que el ajuste sea mejor para valores cercanos al 3. En la eventual realización de nuevas pruebas subjetivas este es un punto importante a tener en cuenta.

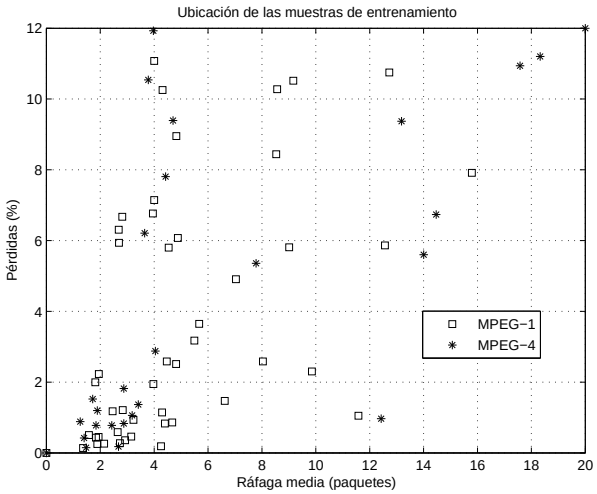


Figura 5.13: Ubicación de las muestras de entrenamiento utilizadas.

En la figura 5.13 se pueden observar las muestras de entrenamiento que fueron utilizadas. Estas están ubicadas según el valor de pérdida y ráfaga media estimado para cada una y separadas por el tipo de codificación. Se puede apreciar una gran concentración de puntos en los valores bajos de pérdida y ráfaga media. Esto se eligió así porque es el lugar donde el valor de DMOS varía más rápido. Se observa que la concentración en esta región es mayor para MPEG-4 que para MPEG-1. La consecuencia de esto es que la RNN subestime los valores altos de DMOS principalmente para MPEG-1. Como la cantidad de muestras MPEG-1 es casi el doble que las MPEG-4 se observa que en general la RNN subestima el DMOS para valores altos del mismo. De todas formas existen zonas que no fueron debidamente cubiertas para ninguno de los dos codecs.

Otra forma de ver la calidad del ajuste es mediante la figura 5.14. En esta figura se grafican los valores reales de DMOS contra la estimación de la RNN y la recta $y = x$. Si el ajuste fuese perfecto todos los puntos deberían caer sobre esta recta. También se puede apreciar el efecto de la falta de muestras con valores altos y bajos de DMOS. Para valores altos de DMOS la mayoría de los puntos se encuentran por encima de la recta $y = x$ lo que indica que los valores de DMOS son subestimados por la RNN. Lo contrario sucede para valores muy bajos de DMOS y estaría equilibrada para valores un por debajo de 3.

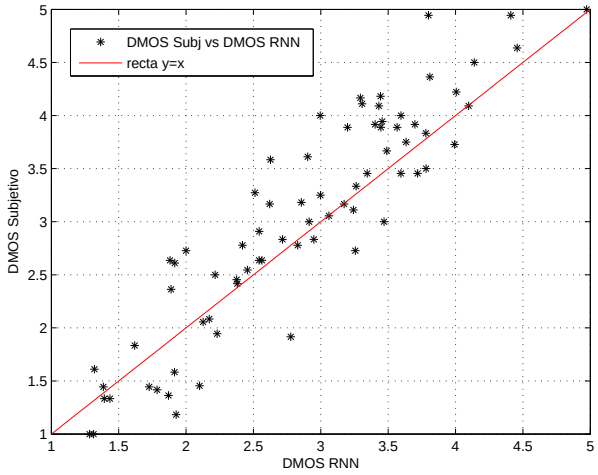


Figura 5.14: DMOS Subjetivo vs DMOS RNN para las muestras de entrenamiento

Validación de la RNN

Una vez verificado el ajuste a las muestras de prueba, se procede a estudiar la predicción de la RNN. Para ello se ordenan las muestras de forma aleatoria y se separan en dos grupos. Un grupo de 60 muestras se utiliza para entrenar la red y las 14 restantes para validar la predicción. Como se mencionó en la sección 5.2, uno de los videos no se consideró en ninguno de los algoritmos.

En la figura 5.15 se muestra el valor de DMOS estimado por la RNN y el valor subjetivo correspondiente. Se puede observar que el ajuste es aceptable. El error absoluto medio para las 14 muestras es de 0,40. Este valor es inferior a lo obtenido con los métodos objetivos donde el error absoluto medio más bajo es de 0,48.

Resultados obtenidos

Una vez que se tiene la red entrenada, esta permite estudiar la influencia de los distintos parámetros en la calidad percibida. La validez de este estudio radica en la validez de los valores utilizados para entrenar la red y la existencia de muestras suficientes para tener estimaciones consistentes en todo el rango de variación de las entradas. En el presente estudio se obtuvo un error en la predicción considerable, y además no se cubre todo el rango de manera

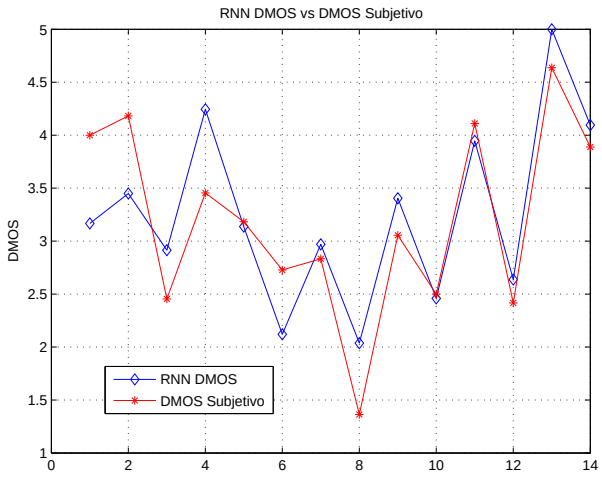


Figura 5.15: DMOS Subjetivo y DMOS RNN para las muestras de validación

uniforme, como ya se mencionó antes. Esto hace que no se pueda medir cuantitativamente la influencia de los diferentes parámetros en la calidad, pero si tener una idea cualitativa.

Para ver como afectan estos parámetros, lo que se hizo es formar un grilla con la probabilidad de pérdida y el valor de ráfaga media e ir cambiando los otros parámetros. En particular se estudió el efecto del tipo de codificación y la cantidad de movimiento presente en el video.

En la figura 5.16 se presenta una curva donde se muestra la variación del DMOS en función de la probabilidad de pérdida y la ráfaga media para video MPEG-1, con una tasa de bits de 500 kb/s, tasa de cuadros $24cdr/s$ y cantidad de movimiento alto.

En la figura se aprecia nuevamente el efecto de subestimación de la calidad para valores bajos de pérdida. Cabe aclarar que no todos los puntos que aparecen en la gráfica tienen sentido. La ráfaga media esta medida en paquetes por lo tanto se deben perder por lo menos esa cantidad de paquetes para que el punto en cuestión sea válido. Teniendo en cuenta que en promedio se envían 1000 paquetes por video, una pérdida del 1 % corresponde a 10 paquetes. Hecha esta salvedad analicemos la forma de la curva.

Como era de esperarse el valor de DMOS cae rápidamente con las pérdidas. A su vez se observa un incremento en el DMOS al aumentar la ráfaga media para un mismo valor de pérdida. Si pensamos en la ocurrencia de eventos de pérdida, independientemente de su tamaño, tenemos que para un valor dado de probabilidad de pérdida ocurrirán más eventos cuanto menor sea la ráfaga media. Lo que esta curva nos dice es que de cierta forma la gente

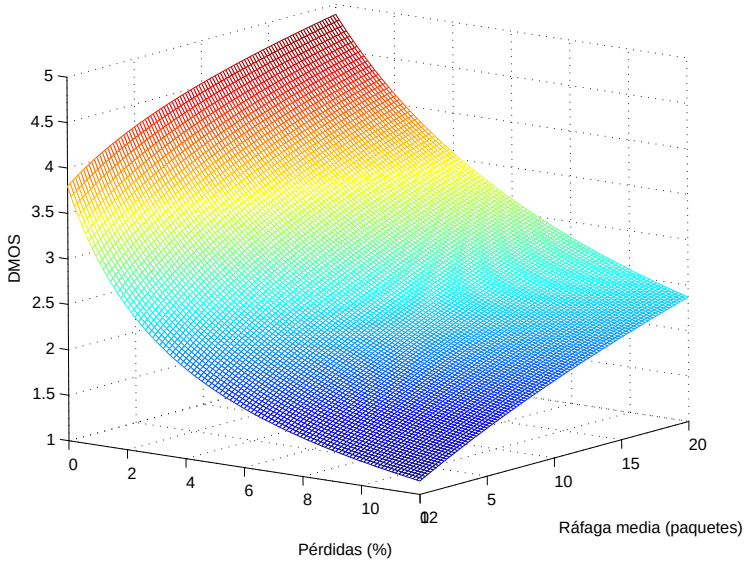


Figura 5.16: DMOS vs pérdida y ráfaga media. Cantidad de movimiento alta.

responde a la cantidad de eventos de pérdida. Este efecto es mencionado en [2]. Sin embargo es de esperarse que para valores de ráfaga muy grande el valor de DMOS caiga, ya que en ese caso faltarían partes considerables del video.

Las curvas para MPEG-1 con diferente cantidad de movimiento son muy similares a la anterior. La diferencia es la velocidad con la que el DMOS cae en función de la probabilidad de pérdida. En la figura 5.17 se muestra la variación del DMOS en función de la probabilidad de pérdida para las tres categorías de cantidad de movimiento. Las tres curvas tienen la misma tasa de bits, tasa de cuadros y ráfaga media. Se observa que la curva de cantidad de movimiento bajo queda por encima de las otras dos para valores de pérdida altos. Si pensamos en la pérdida de un frame, es claro que esta será más notoria cuanto mayor sea la cantidad de movimiento presente en el video (manteniendo la tasa de cuadros constante).

Otro estudio interesante es comparar la diferencia entre las dos codificaciones. Primero veamos la curva que se obtiene para MPEG-4 (figura 5.18), cantidad de movimiento alto. Se puede observar que si bien presenta valores altos de DMOS para valores de pérdida y ráfaga media bajos, los mismos caen rápidamente en función de las pérdidas.

Para poder comparar mejor la curva para MPEG-1 con la de MPEG-4 se presenta la siguiente gráfica (figura 5.19). En dicha figura se muestra el valor de DMOS contra las pérdidas para un valor de ráfaga media fija (5 paquetes). El resto de los parámetros: tasa de bits y tasa de cuadros son iguales en ambos casos. Si bien las curvas no comienzan en el mismo punto, se

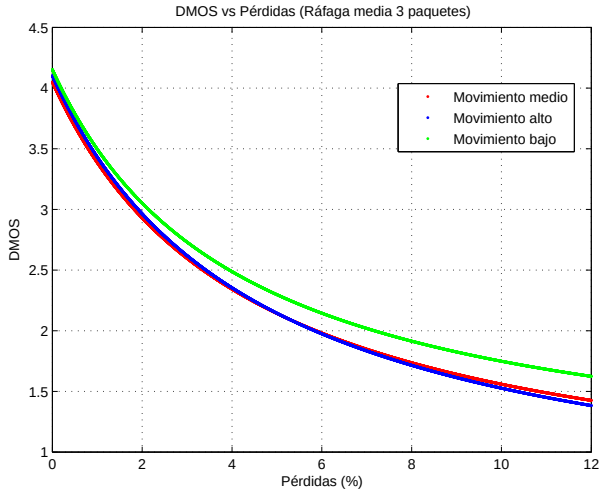


Figura 5.17: DMOS vs pérdida para las tres categorías de cantidad de movimiento. Ráfaga media 3 paquetes.

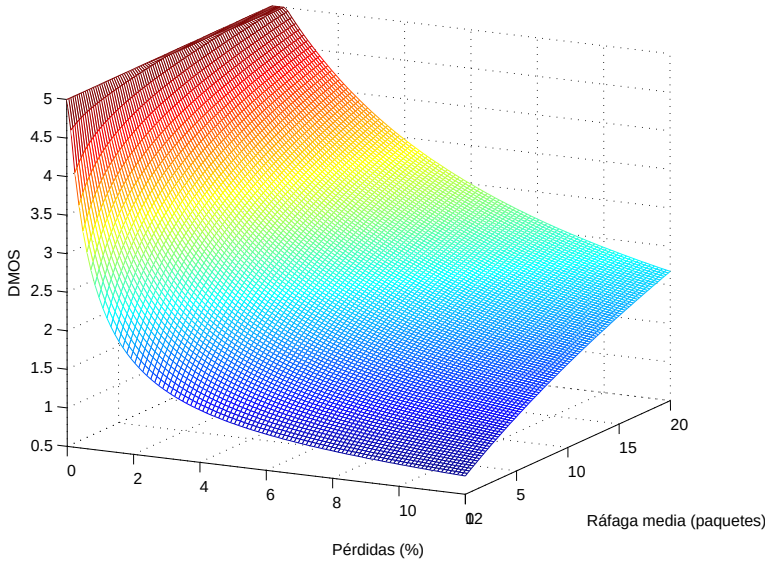


Figura 5.18: DMOS vs pérdida y ráfaga media. Cantidad de movimiento alta.

aprecia claramente que la velocidad con la que cae el valor de DMOS en MPEG-4 es mayor que en MPEG-1. Esto se debe a la diferencia en la influencia de las pérdidas en uno y otro caso. Como se mencionó en el Capitulo 2, la presencia de macro bloques corruptos produce cuadrados erróneos en la imagen, de ubicación fija, en el caso de MPEG-1. Para MPEG-4 los macro bloques son variables y dependen de la cantidad de movimiento presente en el video, por lo que un error en los mismos produce cuadrados erróneos que se mueven en la imagen.

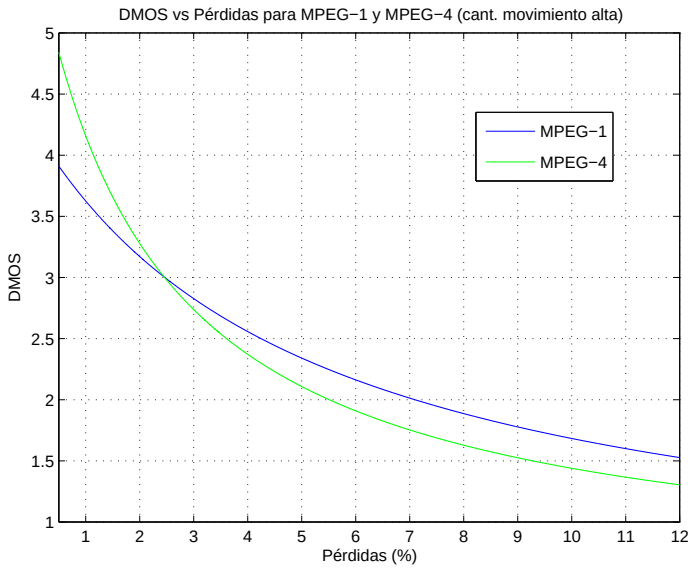


Figura 5.19: DMOS vs pérdida, para MPEG-1 y MPEG-4. Cantidad de movimiento alta.

Es de esperar que la cantidad de movimiento juegue un papel más importante en el valor de DMOS para el caso de MPEG-4. Para ver este efecto, se muestra en la figura 5.20 el valor de DMOS contra las pérdidas (ráfaga media fija en 5 paquetes), para las tres categorías de cantidad de movimiento.

El resultado es el esperado, cuanto mayor la cantidad de movimiento más rápido disminuye el DMOS con la probabilidad de pérdida. Se puede observar que la diferencia es más notoria entre los videos clasificados como bajos y los restantes. Esto es entendible dado que la clasificación fue realizada de manera subjetiva, lo que hace que la separación de algunos videos entre cantidad media o alta sea cuestionable.

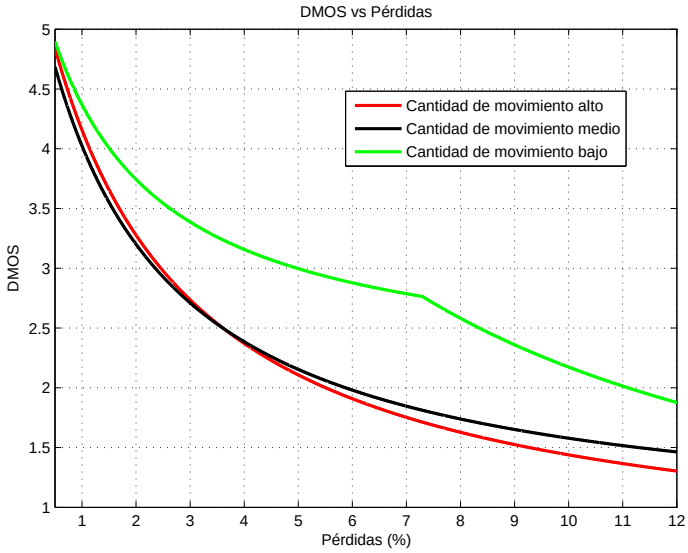


Figura 5.20: DMOS vs pérdida para las tres categorías de cantidad de movimiento. Codec MPEG-4, ráfaga media de 5 paquetes.

Conclusiones

Se obtuvo un herramienta de estimación de PQoS con una performance mayor a la de los métodos intrusivos. Además presenta ventajas adicionales sobre estos. No es necesario transmitir un video para poder predecir la calidad. Esto evita tener que disponer de un número importante de videos que sean representativos de las distintas categorías. La otra ventaja es que el tiempo que demora la RNN en obtener el resultado es del orden del segundo, mientras que los métodos intrusivos requieren un tiempo del orden del minuto.

5.4. Resultados de los algoritmos no intrusivos de Audio

En esta sección se presentan los resultados de los algoritmos no intrusivos para audio. Primero se presentan los resultados obtenidos con las Random Neural Networks (RNN), y luego los resultados obtenidos con el E-Model.

5.4.1. Random Neural Networks para Audio

La forma en que se utilizan las RNN ya se explicó en la sección anterior para el caso de video.

En este caso la estructura de la RNN consta de tres neuronas de entrada, diez neuronas escondidas y una neurona de salida. Las entradas son la probabilidad de pérdida, la ráfaga media y el tipo de codec utilizado en el streaming. Los posibles codecs son G.711 (PCM), GSM y G.723. Estos presentan diferencias en la forma de codificar la señal y como resultado, diferencias en la degradación que producen en la misma. En orden creciente de degradación se tiene PCM, G.723 y por último GSM. Al igual que en el caso de video la salida es el valor de DMOS.

El conjunto de configuraciones (valores de entrada y salida correspondiente) necesario para entrenar y validar la RNN surge de los tests subjetivos. En el momento de generar las señales distorsionadas para realizar dichos tests, se estimaba el estado de la conexión y se registraba el tipo de codec. Se generaron de esta manera 74 configuraciones distintas.

Entrenamiento de la RNN

Se puede apreciar en la figura 5.21 el ajuste de la RNN a las muestras de entrenamiento. Se puede observar claramente que el ajuste es peor que en el caso de video. El error absoluto medio en video fue 0,50, mientras que en este caso es 0,70. Se probaron diferentes cantidades de neuronas escondidas pero este valor no se logró modificar apreciablemente. Cabe aclarar que de las 74 muestras disponibles dos no fueron tomadas en cuenta por causas que se expondrán más adelante.

Se observa que en general la RNN sobrestima la calidad. En la figura 5.22 se muestra el valor de DMOS obtenido en los tests subjetivos en función del valor de DMOS estimado por la RNN y se grafica la recta $y = x$. Se puede observar claramente la sobrestimación del DMOS por parte de la RNN, casi todos los valores se encuentran a la derecha de la recta $y = x$.

Esta sobrestimación del DMOS se debe a la presencia de muchas configuraciones con DMOS alto en relación con las de DMOS bajo. En la figura 5.23 se muestra la ubicación

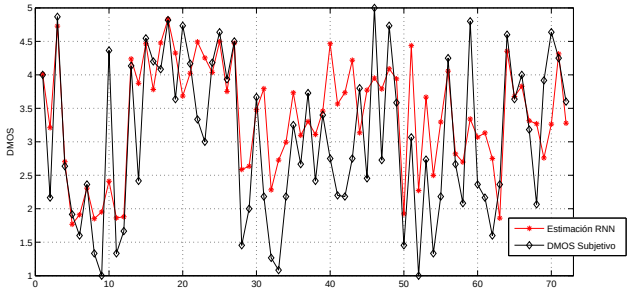


Figura 5.21: DMOS RNN y DMOS subjetivo para las muestras de entrenamiento

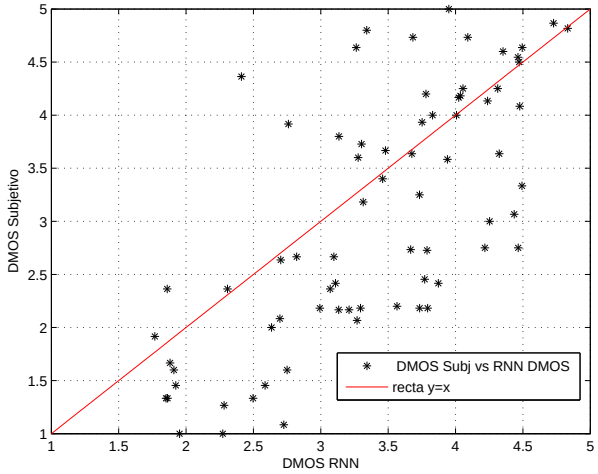


Figura 5.22: DMOS subjetivo contra DMOS RNN, para las muestras de entrenamiento.

de las configuraciones utilizadas en el plano pérdida - ráfaga media, separadas por el codec utilizado. Si bien la concentración para valores bajos es buena, existen zonas muy grandes en las que no hay muestras.

Además de los problemas que causa la mala distribución de las muestras de entrenamiento, existe otro inconveniente que tiene que ver con el valor estimado de los parámetros de la red de transporte. Como se mencionó anteriormente (ver 4.2.4) la herramienta de streaming utilizada para audio (JMF) pierde los primeros paquetes enviados independientemente del estado de la red de transporte. Para evitar que las personas tuvieran en cuenta este efecto en el momento de realizar los tests subjetivos, se le agregó un segundo de silencio al comienzo de

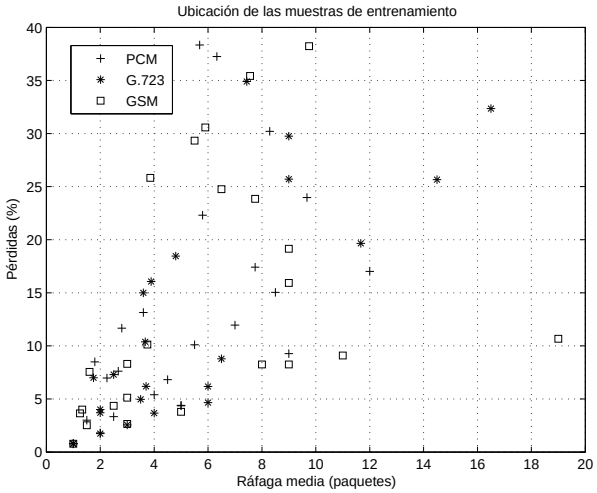


Figura 5.23: Ubicación de las muestras de entrenamiento utilizadas.

cada señal de audio. El problema es que la mayoría de la señales duran entre 3 y 4 segundos, por lo que el segundo de silencio representa una parte importante de la señal a transmitir. El inconveniente es que se puede dar la pérdida de paquetes correspondientes al silencio, por lo cual no son tomados en cuenta por los tests subjetivos. Como resultado algunas muestras tienen valores altos de pérdida, pero la señal se ve poco distorsionada. Esta fue la causa por la que se eliminaron dos muestras de las 74 existentes. Estas presentaban valores de pérdida muy grandes pero valores de DMOS también muy grandes, debido a que las pérdidas se habían dado casi exclusivamente en el silencio. Esta es una causa importante del mal ajuste de la RNN a las muestras. Cabe aclarar que en el momento de la inclusión del silencio no se esperaba que este afectara tanto los resultados de la estimación de los parámetros. Para evitar este problema sería necesario estimar la cantidad de paquetes correspondientes al silencio, y contar las pérdidas luego de transmitido éste.

Validación de la RNN

Para validar las estimaciones realizadas por la RNN, se procedió igual que en el caso de video. Se ordenaron las muestras de forma aleatoria y se separaron en dos grupos, uno de 60 para entrenar la RNN y el otro de 12 para validar la predicción. Los resultados se muestran en la figura 5.24. Se observa que la predicción de la RNN es pobre, como era de esperarse a partir de los resultados observados para las muestras de entrenamiento. El error absoluto medio que se obtuvo para las muestras de validación es 0,70. Este valor es superior a lo obtenido en el caso del PESQ, donde el error absoluto medio fue 0,42.

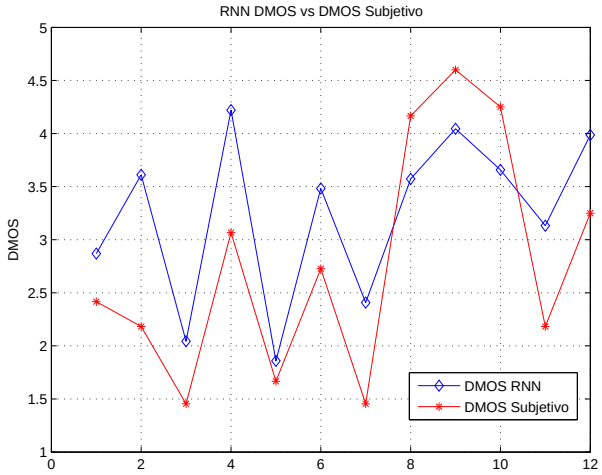


Figura 5.24: DMOS estimado RNN y DMOS subjetivo para las muestras de validación.

Resultados

Una vez obtenida la red entrenada se procede a estudiar la influencia de los parámetros de entrada en el valor de DMOS. Como se mencionó en la sección anterior, los resultados se toman de forma cualitativa.

En la figura 5.25 se muestra el valor de DMOS en función de la probabilidad de pérdida y la ráfaga media de pérdidas, utilizando PCM. Cabe aclarar nuevamente que no todos los puntos de la superficie tienen sentido ya que la ráfaga media esta expresada en paquetes. Considerando que en una señal de audio se transmiten en promedio 100 paquetes, el 1 % corresponde a un paquete. Para que un valor de ráfaga media dado tenga sentido, se tienen que perder por lo menos esa cantidad de paquetes.

Se puede observar en la figura como el valor de DMOS decrece con la probabilidad de pérdida. A su vez presenta un aumento con el valor de la ráfaga media, similar al observado en el caso de video.

Las curva para los restantes codecs es muy similar a la anterior. El impacto del codec en el valor de DMOS se observa en la figura 5.26. En dicha figura se muestra el valor de DMOS en función de la probabilidad de pérdida para los diferentes tipos de codificación. La ráfaga media es de 5 paquetes en los tres casos. El resultado coincide con el esperado; el codec que presenta valores más altos de DMOS para un valor de pérdida dado es PCM, luego G.723 y por último GSM. Valores de referencia de la distorsión introducida por cada uno de los

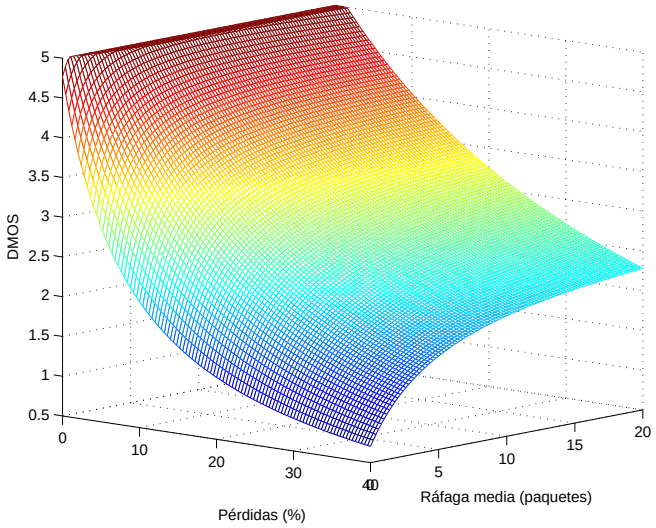


Figura 5.25: DMOS vs pérdida y ráfaga media. Codificación PCM

codecs se pueden encontrar en [19].

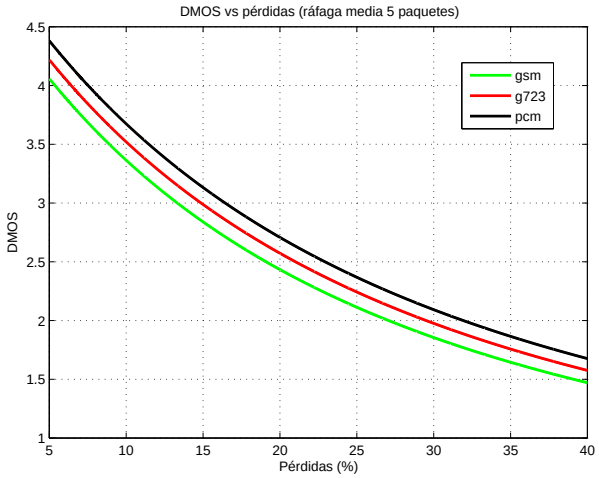


Figura 5.26: DMOS vs pérdida, para los 3 codecs. Ráfaga media constante (5 paquetes).

Conclusiones

Los resultados obtenidos con la presente implementación de la RNN para audio no son satisfactorios. Su desempeño es peor que el de los métodos intrusivos, en particular Pesq. La causa de este pobre desempeño radica en la muestra de entrenamiento utilizada, como se explicó anteriormente en esta sección. Para obtener mejores resultados sería necesario volver a realizar las secuencias de audio y los tests subjetivos correspondientes. Debido al tiempo que esta tarea insume no fue posible llevarla a cabo.

5.4.2. Resultados E-Model

El E-Model fue originalmente desarrollado como herramienta de planificación para redes telefónicas. Tiempo después fue actualizado para considerar redes de paquetes conmutados. Es una función que mapea los parámetros de la comunicación en un valor proporcional a la calidad percibida.

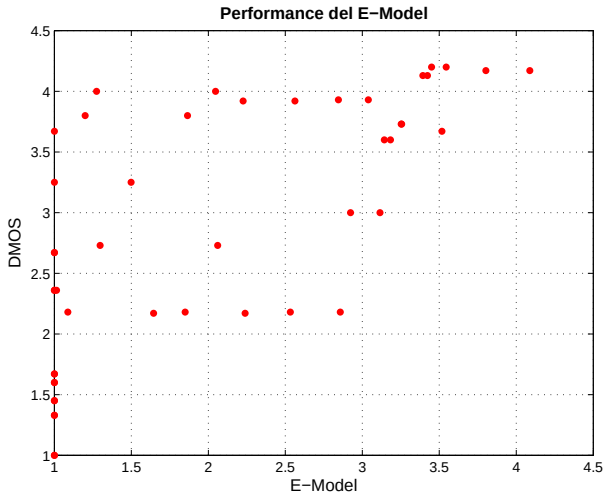


Figura 5.27: Performance del E-Model

En la figura 5.27 se observa la pobre performance del E-Model. El coeficiente de correlación es de 0.62, lo que indica poca linealidad entre los datos. Se observa que para una gran cantidad de valores el E-Model da una predicción de DMOS igual a 1.

Existen varios factores que influyen en el desempeño de este algoritmo. Como se mencionó en la sección anterior las señales de audio tenían al principio una sección de silencio, cuya duración era comparable en algunos casos con la duración de la sección de voz. Las

pérdidas ocurridas en la sección de silencio no son consideradas en los tests, por lo que nos encontramos con señales con 30 % de pérdidas y un DMOS de 4.3. Es muy probable que gran porcentaje de estas pérdidas se dieran en la sección con silencio por lo que en la realidad el porcentaje de pérdidas "escuchado" al momento de hacer los tests subjetivos sea bastante menor.

Otro aspecto a tener en cuenta es que la norma no asegura los resultados del modelo cuando el valor de la ráfaga media (ver 3.5.2) es mayor que 2, cosa que ocurre en muchos de los casos.

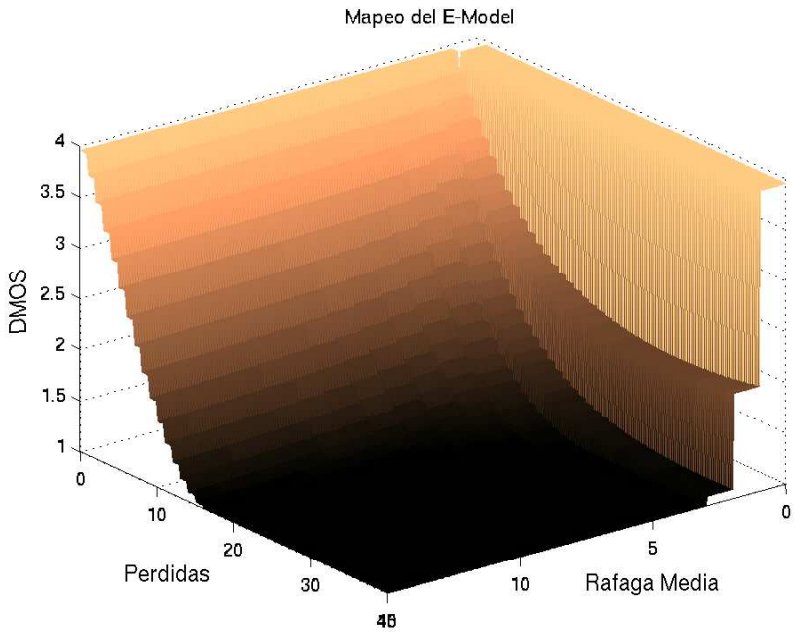


Figura 5.28: Performance del E-Model

En la gráfica 5.28 se observa el mapeo del E-Model para el codec GSM, variando las pérdidas de 0 % a 40 % y la ráfaga media de 0 a 15. Como se esperaba la función es monótona decreciente para valores crecientes de pérdida y ráfaga. A medida que aumenta la ráfaga media la influencia del porcentaje de pérdida se hace más pronunciado.

Al ser el E-Model una herramienta desarrollada inicialmente para planificación de redes telefónicas, incluye en el modelo el efecto del retardo. La pérdida de interactividad en una conversación es un factor muy importante, pasados los 400 ms se torna imposible llevar adelante una conversación fluida. Recientemente se propuso la utilización del E-Model junto

con PESQ. Este último es utilizado para estimar la degradación de la señal, mientras que el E-Model cuantifica los efectos del retardo en una conversación.

Capítulo 6

Conclusiones y Trabajo a Futuro

En este capítulo se presentan las conclusiones finales del proyecto. En la sección 6.1 se analiza el trabajo realizado, evaluando el cumplimiento de los objetivos marcados. El capítulo finaliza en la sección 6.2 con algunas propuestas de trabajo a futuro.

6.1. Conclusiones

En base al trabajo realizado se consideran satisfactoriamente cumplidas las metas establecidas. En lo que respecta al primer objetivo, se realizó un estudio del estado del arte en la estimación de la PQoS en voz y video sobre IP que dejó de manifiesto el creciente interés en el área y su continuo desarrollo.

Este estudio abarcó numerosos temas desconocidos por el grupo de proyecto (codificación de audio y video, programación en distintos lenguajes y realización de tests subjetivos entre otros) y contempló una gran variedad de áreas temáticas: redes de datos, tratamiento de señales, desarrollo de software y hasta algunos aspectos de la percepción humana.

Dada la amplitud del tema en cuestión se optó por un enfoque global del problema, atacándolo desde los distintos puntos de vista existentes en la actualidad. Esto permitió evaluar el desempeño de un variado conjunto de herramientas, sentando así las bases para futuros desarrollos en el área.

Para alcanzar el primer objetivo se llevaron a cabo las siguientes tareas:

- Estudio y selección de los algoritmos de estimación de PQoS para audio y video.
- Implementación de los algoritmos de video SSIM, ITS y PSNR.

- Implementación del algoritmo E-Model para audio.
- Adaptación de los algoritmos de audio PESQ, EMBSD y MNB.
- Implementación de una herramienta de software que permite manipular redes neuronales aleatorias (RNN) de tipo feed-forward (creación, entrenamiento y utilización).
- Realización de tests subjetivos de calidad para audio y video.

Este último punto merece un comentario aparte; la falta de experiencia en la realización de tests subjetivos influyó significativamente en los resultados obtenidos. Si bien desde un principio se consideró un punto importante, no se le atribuyó la importancia debida. La realidad de los hechos demostró que este punto era realmente crítico; los tiempos involucrados en las distintas etapas (preparación, realización de los tests y procesamiento de los datos) hicieron difícil repetir la experiencia. De todas maneras y a pesar de haber realizado una única sesión de tests, los resultados obtenidos son aceptables.

En lo que respecta al segundo objetivo del proyecto, se desarrolló una herramienta de software que engloba todos los problemas subyacentes al uso de los algoritmos y técnicas de estimación de PQoS implementadas (transmisión y recepción de secuencias multimedia en tiempo real, mediciones en la red, etc.). Este aspecto resulta de gran importancia teniendo en cuenta que las herramientas disponibles son en su mayoría implementaciones de los distintos algoritmos.

Durante el desarrollo se puso especial énfasis en la modularidad del programa, facilitando el agregado de nuevos componentes (nuevos algoritmos y metodologías de estimación de PQoS, nuevos módulos de generación de tráfico, etc.).

Las tareas involucradas en el segundo objetivo fueron las siguientes:

- Desarrollo de una arquitectura de funcionamiento simétrico de tipo cliente/servidor-servidor/cliente
- Desarrollo de una herramienta de estimación de parámetros de red.
- Desarrollo de un reloj de alta resolución.
- Desarrollo de una herramienta de streaming de audio en Java.
- Adaptación de una herramienta de streaming de video.
- Desarrollo de la interfaz gráfica de usuario del programa.
- Desarrollo de un enrutador simple que emula el comportamiento de una red IP real.

6.2. Trabajo a Futuro

Dentro de los perspectivas a futuro se plantean en esta sección distintas mejoras a la herramienta de software desarrollada, así como también distintas propuestas que continúan la línea de trabajo del proyecto.

6.2.1. Mejoras de la herramienta desarrollada

- **Realizar una calibración exhaustiva de los algoritmos desarrollados.** Como ya fue mencionado, uno de los grandes problemas enfrentados fue la calibración de los algoritmos desarrollados. La experiencia adquirida en la realización de los tests subjetivos permite asegurar que una nueva etapa de tests, contemplando los distintos problemas encontrados, brindaría mejoras notorias en el desempeño de los algoritmos de estimación de PQoS, principalmente en los desarrollados en base a las RNN.
- **Mejora de los algoritmos intrusivos.** El desfasaje introducido por la pérdida de información en la transmisión de secuencias mostró ser determinante en el desempeño de los algoritmos de estimación intrusivos. Agregar una fase previa de alineación temporal mejoraría los resultados obtenidos (el algoritmo PESQ es una clara muestra de esto). Otra solución posible y que en algún momento se evaluó implementar sería lograr que la herramienta de envío y captura de secuencias informe a los algoritmos la ubicación de las pérdidas.
- **Integrar las herramientas de manipulación y streaming de video.** Para realizar una buena estimación de la calidad percibida es necesario que lo que "ve" el algoritmo de comparación de secuencias sea igual a lo que el usuario visualiza (problema mostrado en 5.2).

6.2.2. Propuestas que continúan la línea de trabajo del proyecto

- **Validar la herramienta en ambientes no controlados.** Los tiempos del proyecto no permitieron validar el software desarrollado en ambientes no controlados (Internet). Es necesario verificar su funcionamiento en una red real, generalizando así su aplicación.
- **Generar nuevos módulos de software.** Uno de los puntos que sería deseable encarar es el desarrollo de nuevas funcionalidades para el programa. Por ejemplo, nuevos módulos de estimación de PQoS, de monitoreo continuo de calidad, etc. La modularidad del diseño permite acoplar fácilmente nuevos módulos.
- **Mejorar el módulo de medición de parámetros de red .** Si bien el módulo desarrollado cumple las funciones básicas de medida, queda claro que es posible mejorarlo ampliamente.
- **Investigar a fondo los efectos de los factores que influyen en la PQoS en voz y video sobre IP.**

Bibliografía

[1] Y. J. Liang, J. G. Apostolopoulos, y B. Girod, *Analysis of packet loss for compressed video: Does burst-length matter?,* ICASSP, 2003.

[2] M. Claypool y J. Tanner, *The Effects of Jitter on the Perceptual Quality of Video,* ACM Multimedia Conference, Volume 2, Orlando, FL, October 30 - November 5, 1999.

[3] S. Voran, *Objective Estimation of Perceived Speech Quality – Part I:Development of the Measuring Normalizing Block Technique,* IEEETrans. on Speech and Audio Processing, Vol. 7, No. 4, July 1999.

[4] W. Yang, *Enhanced Modified Bark Spectral Distortion (EMBSD): An Objective Speech Quality Measure Based on Audible Distortion and Cognition Model*, Dissertation, Temple University, Philadelphia, USA, May 1999.

[5] ITU-T Recommendation P.862, *Perceptual evaluation of speech quality (PESQ), an objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs*, Feb. 2001.

[6] J. G. Beerends , A. P. Hekstra , A. W. Rix , and M. P. Hollier *Perceptual Evaluation of Speech Quality (PESQ), the new ITU standard for end-to-end speech quality assessment. Part II − Psychoacoustic model*, October 1998

[7] B. Girod, *"What's wrong with mean-squared error"*, in Digital Images and Human Vision, A. B. Watson, ed., pp. 207-220, MIT Press, 1993.

[8] *"Spatial-Temporal Distortion Metrics for In-Service Quality Monitoring of Any Digital Video System"*, SPIE International Symposium on Voice, Video, and Data Communications, Boston, MA, September 11-22, 1999.

[9] ANSI T1.801.03, *"American National Standar for Telecommunication - Digital Transport of One-Way Video Telephony Signals - Parameters for Objective Performance Assessment"*, American National Standars Institute, 1996.

[10] Z. Wang, L. Lu y A. C. Bovik, *"Video Quality Assessment Bades on Structural Distorsion Measurement"*, Signal Processing: Image Communication, vol. 19, N°2, pp. 121-132, february 2004

[11] Z. Wang, *Rate scalabel foveated image and video communications.* PhD thesis, Dept. of ECE, The University of Texas at Austin, Dec. 2001.

[12] Z. Wang, L. Lu y A. C. Bovik,*"Why is image quality assessmetn so difficult?,"* in Proc. IEEE Int. Conf. Acosut., Speech, and Signal Processing, Orlando, May 2002.

[13] VQEG: The Video Quality Expert Group, **http://www.vqeg.org/.**

[14] VQEG, *"Final report from the video quality experts group on the validation of objective models of video quality assessment,"* Mar. 2000, **http://www.vqeg.org/.**

[15] ITU-R, Recommendation BT.500-11, *Methodology for the subjective assessment of the quality of television pictures,* 2002.

[16] ITU-P, Recommendation P.800, *Methods for subjective determination of transmission quality,* 1996.

[17] ETSI , *"Speech Communication Quality from Mouth to Ear of 3.1'kHz Handset Telephony across Networks,"* Technical Report. ETR 250, 1996.

[18] ITU-T, Recommendation G.107, *"The E-model, A Computational Model for Use in Transmission Planning,"*, Marzo 2005.

[19] ITU-T, Recommendation G.113, *"Transmission impairments due to speech processing,"* Mayo 2002.

[20] J.C. Bolot, *"End-to-end frame delay and loss behavior in the Internet,"* in In Proc. ACM SIGCOMM, Sept. 1993, pp. 289-298.

[21] E. Gelenbe, *Random neural networks with negative and positive signals and product form solution,* Neural Computation, vol. 1, no. 4, pp. 502-511, 1989.

[22] E. Gelenbe, *Stability of the random neural network model,* in Proceedings of Neural Computation Workshop, pp. 56-68, Berlin,West Germany, Febrero 1990.

[23] E. Gelenbe, *Learning in the recurrent random neural network,* Neural Computation, vol. 5, no. 1, pp. 154-511, 1993.

[24] E. Gelenbe, V. Koubi, and F. Pekergin, *Dynamical random neural network approach to the travelling sales man problem,* Elektrik, vol. 2, no. 1, pp. 1-9, Abril 1994.

[25] Anoop Ghanawani, *A Qualitative comparison of neural network models applied to the vertex covering problem,* Elektrik, vol. 2, no. 1, pp. 11-18, 1994.

[26] V. Atalay and E. Gelenbe, and N. Yalabik, *The random neural network model for texture generation,* International Journal of Pattern Recognition and Artificial Intelligence, vol. 6, no. 1, pp. 131-141, 1992.

[27] C. Cramer, E. Gelenbe and P. Gelenbe, *Image and video compression,* IEEE Potentials, vol. 17, no. 1, pp. 29-33, 1998.

[28] C. Cramer, E. Gelenbe, and H. Bakircioglu, *Low bit rate video compression with neural networks and temporal sampling,* Proceedings of the IEEE, vol. 84, no. 10, pp. 1529-1543, Octubre 1996.

[29] L. Aspirot, P. Belzarena, G. Perera, B. Bazzano, *End to end quality of service prediction based on functional regression,* Conferencia HET-NET 2005. **http://iie.fing.edu.uy/investigacion/grupos/artes/publicaciones/hetnet05.pdf**

[30] V. Atalay and E. Gelenbe, *Parallel algorithm for colour texture generation using the random neural network model,* International Journal of Pattern Recognition and Artificial Intelligence, vol. 6, no. 2, pp. 437-446, 1992.

[31] E. Gelenbe, Z. H. Mao, and Y. D. Li, *Function approximation with spiked random networks,* IEEE Transactions on Neural Networks, vol. 10, no. 1, pp. 1-9, 1999.

[32] S. Mohamed and G. Rubino, *A study of real-time packet video quality using random neuronal networks,* IEEE Transactions On Circuits and Systems for Video Technology, vol. 12, no. 12, pp. 1071-1083, Diciembre 2002.

[33] S. Mohamed , G. Rubino and M. Varela, *Performance evaluation of real-time speech through a packet network: a random neuronal networks-based approach,* Performance Evaluation, vol. 57, no. 2, pp. 141-162, 2004.

[34] E. Gelenbe, H. Bakircioglu, T. Kocak *Image processing with the random neuronal network,* in Proceedings of the IEEE Digital Signal Processing Conference, vol. 1, pp. 243-248, Junio 1997.

[35] M. Claypool y J. Tanner, *Java Jitter- The Effects of Java on Jitter in a Continuous Media Server,* IEEE Multimedia Technology and Applications (MTAC), Anaheim, CA, September 1998.

[36] M. Caetano, G. Mateos y F. Morales, *Sip y Asterix - Implementaciones de VoIP basadas en SIP: un estudio enfocado a la QoS utilizando Asterix,* Proyecto de fin de carrera - Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, UDELAR.

[37] L. Aspirot, P. Belzarena, P. Casas, V. González y F. Larroca, *Proyecto PDT S/CO/P/17/02: Medición de performance de punta a punta en servicios de voz y video sobre IP,* Informe de avance 1, Noviembre 2004.

[38] Skype, **http://www.skype.com**

[39] MSN Messenger, **http://www.msn.com**

[40] JGoodies, **http://www.jgoodies.com**

[41] TCPDump, **http://www.tcpdump.org**

[42] TV Ciudad en vivo, **http://www.teveciudad.org.uy**

[43] Ciudad Digital on line, **http://usa.ciudaddigital.com.uy/online**

[44] Tritonus: Open Source Java Sound, **http://tritonus.org**

[45] JMF Solutions **http://java.sun.com/products/java-media/jmf/2.1.1/solutions**

[46] FFmpeg **http://ffmpeg.sourceforge.net**

[47] FOBS **http://fobs.sourceforge.net**

[48] Video Lan Client **http://www.videolan.org**

[49] Java Native Interface **http://www.programacion.com/tutorial/jni**

[50] The Java Tutorial **http://java.sun.com/docs/books/tutorial/**

[51] Java Media Framework **http://java.sun.com/products/java-media/jmf**

Apéndice A

Descripción detallada del software

En este anexo se describe en detalle la arquitectura del software desarrollado. En primera instancia y a modo de clasificación general se presenta en la sección A.1 la estructura de paquetes del programa. A continuación se describen en la sección A.2 y para cada uno de estos paquetes, las distintas clases de software que lo componen. En cada caso se brinda un diagrama de clases que facilita la comprensión del diseño. Por último, en la sección A.3 se brinda una introducción a la JNI (Java Native Interface), ampliamente utilizada en el desarrollo del software.

A.1. Paquetes de software

En la figura A.1 se recuerda la composición del software por módulos brindada en el capítulo 4.

La estructura de paquetes de software representa fielmente esta clasificación. El módulo *controlador de sistema* es representado por las clases del paquete **pqos.control**. El paquete **pqos.metrologia** abarca las distintas clases que permiten estimar los parámetros de la conexión (módulo *estimador parámetros red*). Las funcionalidades del módulo *proveedor de secuencias* son implementadas por las clases del paquete **pqos.multimedia**. El módulo *algoritmos PQoS* se corresponde con el paquete **pqos.algoritmos**. Por último, el paquete **pqos.gui** incluye las distintas clases encargadas de la interfaz gráfica de usuario (módulo *interfaz de usuario*). En la figura A.2 se presenta el diagrama de paquetes correspondiente:

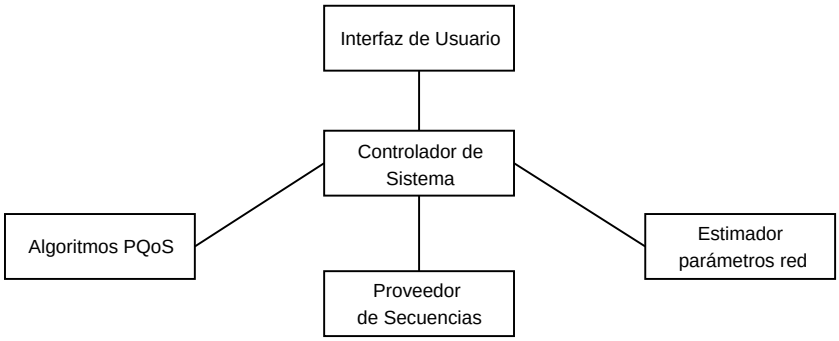


Figura A.1: Módulos que componen el software

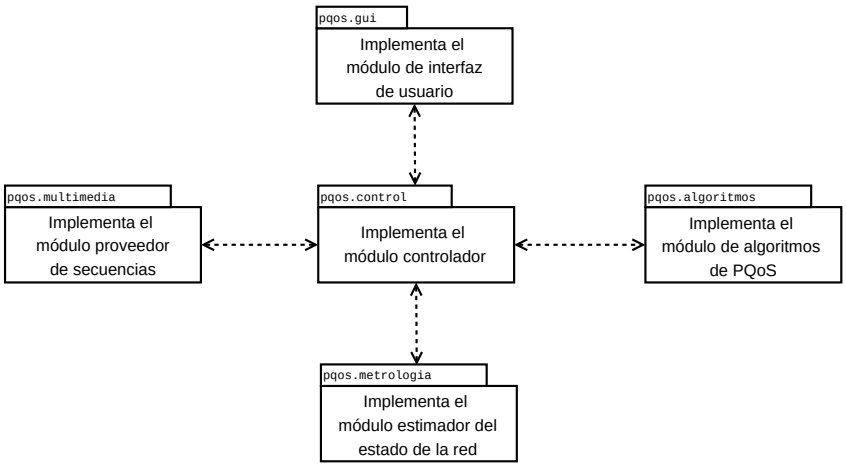


Figura A.2: Paquetes del software

A.2. Clases de software

Paquete pqos.controlador

Este paquete está compuesto por la siguientes clases de software (figura A.3):

- ***Controlador***: implementa todas las funcionalidades del software mediante métodos estáticos: establecimiento de conexión, envío, recibo y captura a disco de secuencias multimedia, estimación de parámetros de red y cálculo de PQoS. Contempla para cada una de dichas funcionalidades tanto el aspecto cliente como el aspecto servidor del programa (por ejemplo, para el establecimiento de conexión implementa los métodos *iniciarConexionServidor(...)* e *iniciarConexionCliente(...)*).

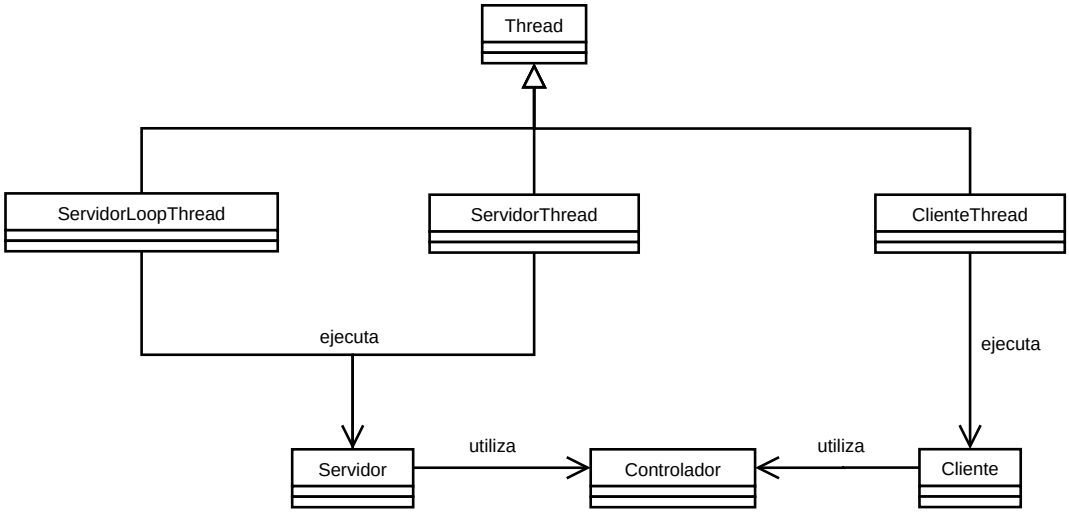


Figura A.3: Diagrama de clases del paquete pqos.controlador

- ***Cliente***: utiliza los métodos de la clase *Controlador* relacionados con el lado cliente para llevar a cabo las distintas tareas mencionadas.
- ***Servidor***: utiliza los métodos de la clase *Controlador* relacionados con el lado servidor para llevar a cabo las distintas tareas mencionadas.
- ***ClienteThread***: esta clase se encarga de ejecutar la clase *Cliente* en un thread paralelo al thread principal (el thread de la interfaz gráfica).
- ***ServidorThread***: esta clase se encarga de ejecutar la clase *Servidor* en un thread paralelo al thread principal.
- ***ServidorLoopThread***: implementa el demonio de conexión del programa; mediante la ejecución continua de la clase *Servidor* se logra una rutina de espera que aguarda por una solicitud de conexión del cliente.

Paquete pqos.metrologia

Este paquete está compuesto por la siguientes clases de software (en figura A.4, la clase *Controlador* se resalta del resto por no pertenecer a este paquete; su inclusión permite un mejor entendimiento del funcionamiento del programa. Al mismo tiempo, las líneas de asociación que involucran esta clase se trazan en punteado para discriminarlas del resto)

- ***EnviaUDP***: genera y envía paquetes de tipo UDP de tamaño variable y a intervalos de tiempo determinados.

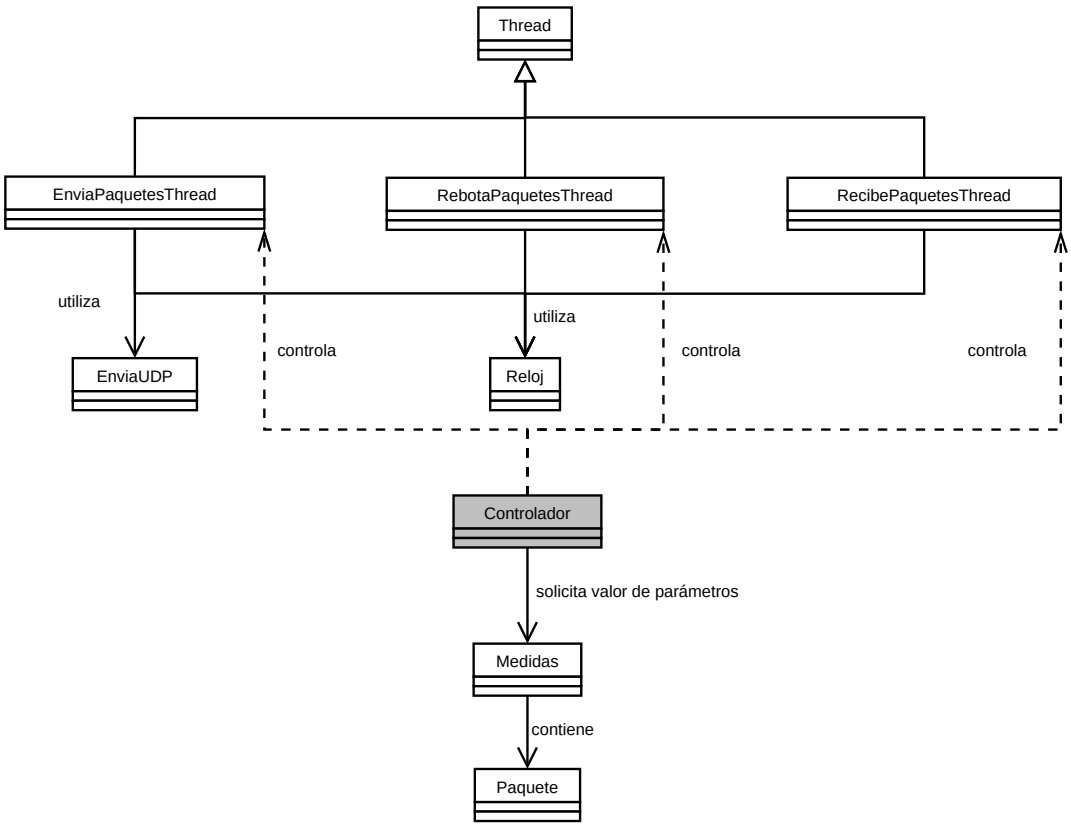


Figura A.4: Diagrama de clases del paquete pqos.metrologia

- ***EnviaPaquetesThread***: ejecuta la clase *EnviaUDP* en un thread paralelo al principal. Incluye
- ***RebotaPaquetesThread***: esta clase se encarga de "rebotar"los paquetes de test utilizados en la estimación de los parámetros de red. Recibe los paquetes UDP que arriban a cierta dirección de capa 4 preestablecida, reenviándolos hacia otra dirección especificada.
- ***RecibePaquetesThread***: recibe los paquetes de test rebotados por la clase *RebotaPaquetesThread*, guardando la información correspondiente en un contenedor de paquetes.
- ***Reloj***: esta clase permite la interacción con el reloj de alta resolución desarrollado en código nativo (C) a través de la interfaz JNI. En lo que respecta a la implementación del reloj en C, se utilizaron las funciones QueryPerformanceFrequency y QueryPerformanceCounter (implementadas en la librería de sistema operativo *kernell32.dll*) que permiten consultar la frecuencia y valor actual del reloj del SO respectivamente.

- ***Medidas***: a partir de los datos recopilados por la clase *RecibePaquetesThread* calcula el valor de los distintos parámetros de la conexión. Se calculan en particular (tanto para la dirección de ida como para la vuelta): el porcentaje de paquetes perdidos, el largo promedio de la ráfaga de pérdida, el retardo de ida y vuelta promedio y el jitter.
- ***Paquete***: guarda información de cada datagrama UDP recibido.

Paquete pqos.multimedia

Este paquete está compuesto por la siguientes clases de software (figura A.5):

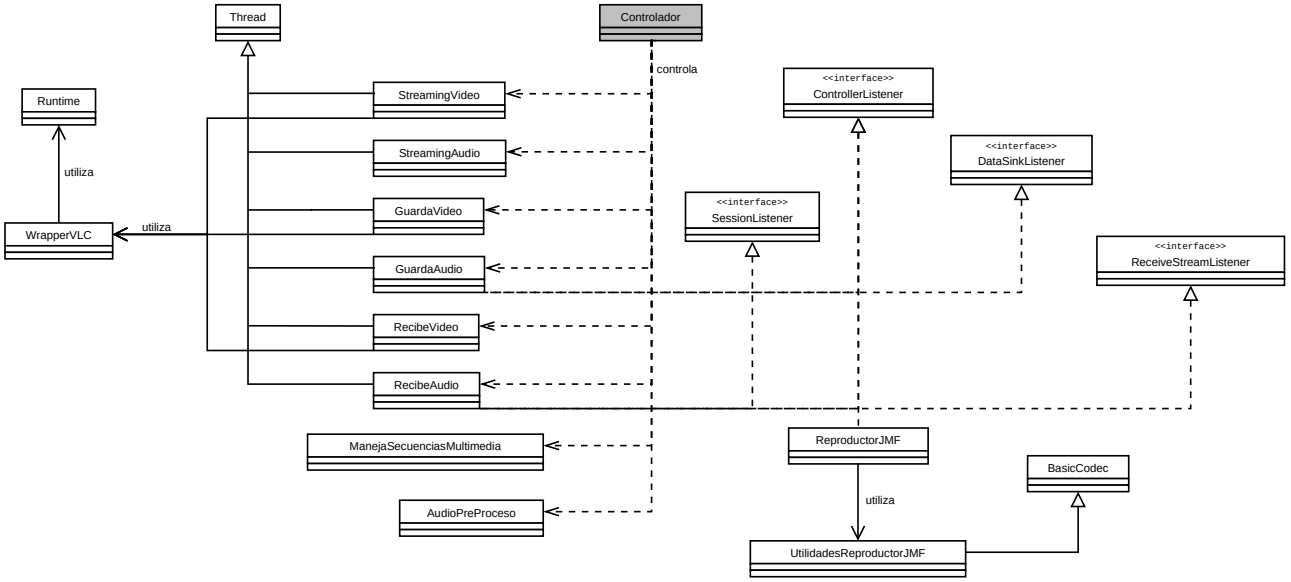


Figura A.5: Diagrama de clases del paquete pqos.multimedia

- ***WrapperVLC***: esta clase permite utilizar desde Java la herramienta de streaming de video VLC([48]). Implementa métodos estáticos para realizar las siguientes tareas: reproducir un video, enviar un streaming de video a una dirección establecida, recibir a pantalla y/o guardar a disco un streaming de video. Estos métodos utilizan la clase *Runtime* para acceder al entorno de ejecución en el cual está corriendo la aplicación. De esta forma es posible controlar el programa de video a través del intérprete de comandos del SO. Para el manejo eficiente del VLC se utilizan dos herramientas auxiliares que se incluyen en el software: cmdow.exe (menejo de ventanas) y ps.exe (manejo de procesos).
- ***StreamingVideo***: permite enviar un streaming de video en base al uso de la clase *WrapperVLC*. Esta clase se implementa como un thread que corre en paralelo al thread principal.

- ***StreamingAudio***: permite enviar un streaming de audio en base al uso de las librerías de la JMF. Esta clase se implementa como un thread que corre en paralelo al thread principal.
- ***GuardaVideo***: permite guardar un streaming de video a disco en base al uso de la clase *WrapperVLC*. Esta clase se implementa como un thread que corre en paralelo al thread principal.
- ***GuardaAudio***: permite guardar un streaming de audio en base al uso de las librerías de la JMF. Esta clase se implementa como un thread que corre en paralelo al thread principal.
- ***RecibeVideo***: permite recibir a pantalla un streaming de video a disco en base al uso de la clase *WrapperVLC*. Esta clase se implementa como un thread que corre en paralelo al thread principal.
- ***RecibeAudio***: permite reproducir un streaming de audio en base al uso de las librerías de la JMF. Esta clase se implementa como un thread que corre en paralelo al thread principal.
- ***ManejaSecuenciasMultimedia***: esta clase guarda información de las secuencias de audio y video de test (definidas como "secuencias originales"en el capítulo 4) que se utilizan en la estimación de PQoS mediante técnicas intrusivas.
- ***AudioPreProceso***: acondiciona las secuencias de audio previo al uso de los algoritmos de estimación de PQoS (adaptación de formato y sustracción del silencio inicial introducido a consecuencia del problema mencionado en 4.2.4).
- ***ReproductorJMF***: permite reproducir localmente una secuencia de audio en base al uso de las librerías de la JMF.
- ***UtilidadesReproductorJMF***: implementa utilidades que utiliza la clase *ReproductorJMF*.

Paquete pqos.algoritmos

Este paquete está compuesto por la siguientes clases de software (figura A.6):

- ***DMOS***: esta clase ejecuta en forma de thread cualquiera de las clases que implementan los algoritmos de estimación de PQoS.
- ***RNN***: implementa el algoritmo de las Random Neuronal Networks. Permite crear la red de neuronas, entrenarla y utilizarla para la estimación de PQoS.
- ***EModel***: implementa el algoritmo del E-Model.
- ***EMBSD***: implementa, en base a la JNI, las llamadas al algoritmo EMBSD, implementado en código nativo (C).

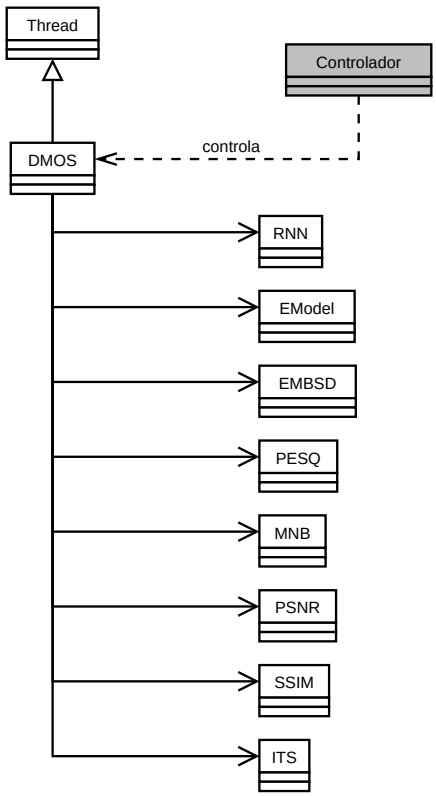


Figura A.6: Diagrama de clases del paquete pqos.algoritmos

- ***PESQ***: implementa, en base a la JNI, las llamadas al algoritmo PESQ, implementado en código nativo (C).
- ***MNB***: implementa, en base a la JNI, las llamadas al algoritmo MNB, implementado en código nativo (C).
- ***PSNR***: implementa, en base a la JNI, las llamadas al algoritmo PSNR, implementado en código nativo (C).
- ***SSIM***: implementa, en base a la JNI, las llamadas al algoritmo SSIM, implementado en código nativo (C).
- ***ITS***: implementa, en base a la JNI, las llamadas al algoritmo ITS, implementado en código nativo (C).

Paquete pqos.gui

Este paquete está compuesto por la siguientes clases de software (figura A.7):

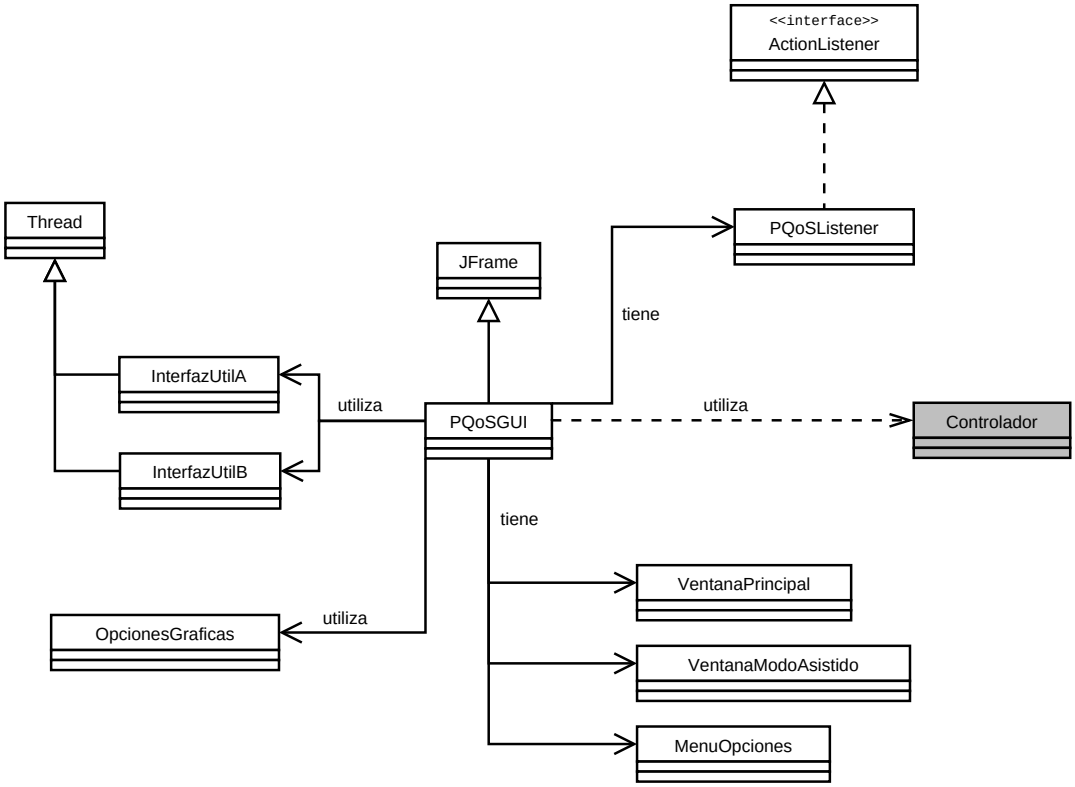


Figura A.7: Diagrama de clases del paquete pqos.gui

- ***PQoSGUI***: es la clase principal de la interfaz gráfica de usuario. Engloba las distintas ventanas de uso del programa.
- ***VentanaPrincipal***: representa la ventana principal del programa, correspondiente al modo de funcionamiento "estándar"de la interfaz (ver B).
- ***VentanaModoAsistido***: representa la ventana "asistida"del programa, correspondiente al modo de funcionamiento "asistido"de la interfaz (ver B).
- ***MenuOpciones***: implementa el menú de opciones de la interfaz.
- ***OpcionesGraficas***: permite definir las características gráficas de la interfaz.
- ***PQoSListener***: implementa el "escucha"principal de la interfaz. Espera a que el botón de inicio (ver B) sea presionado para desencadenar, de acuerdo a las selecciones realizadas por el usuario, la tarea solicitada.
- ***InterfazUtilA***: thread de ayuda para el manejo de la interfaz gráfica.
- ***InterfazUtilB***: thread de ayuda para el manejo de la interfaz gráfica.

A.3. Interfaz Nativo-Java

En esta sección se presenta una breve descripción de la Interfaz Nativo-Java (Java Native Interface), ampliamente utilizada en el desarrollo de la herramienta de software. Esta descripción se basa en una traducción del tutorial de SUN [49]. Para acceder a mayor información acerca del uso de la JNI consultar el tutorial oficial de SUN [50].

Introducción

A paritr de la versión 1.1 de Java se incluye en el kit de desarrollo JDK la Interfaz Nativo Java (JNI). Por un lado, la JNI define un estándar de nombres y convenciones de llamadas para que la Máquina Virtual Java (VM) pueda encontrar las implementaciones de software en código nativo. Al mismo tiempo es posible acceder a las funciones JNI desde el código nativo para manipular objetos Java, liberar objetos Java, crear nuevos objetos, llamar a métodos Java, etc. A continuación se presentarán las utilidades básicas que brinda esta interfaz.

Declarar Métodos Nativos

En el lado del lenguaje Java, se declara un método nativo con la palabra clave native y un cuerpo de método vacío. En el lado del lenguaje nativo, se proporciona la implementación del método nativo. Se debe tener cuidado en la declaración de los métodos y clases en código nativo; sus nombres deben corresponder a la implementación de la función nativa con la signatura del método en el fichero de cabecera del lenguaje Java. La herramienta javah, permite generar prototipos de funciones nativas que corresponden con la declaración del método nativo en lenguaje Java.

Mapear entre Tipos Java y Tipos Nativos

La JNI define un mapeo entre los tipos del lenguaje Java y los tipos de lenguaje nativo (C/C++). En la siguiente tabla se presentan los tipos nativos correspondientes a los tipos primitivos Java.

Java	Nativo
boolean	jboolean
byte	jbyte
char	jchar
short	jshort
int	jint
long	jlong
float	jfloat
double	jdouble
void	void

Acceder a Strings Java

Los Strings son un tipo de objeto Java particularmente útil. La JNI proporciona un conjunto de funciones para manipulación de Strings que hace sencilla la tarea de manejar Strings de Java en el código nativo. El programador puede traducir entre Strings Java y Strings nativos en los formatos Unicode y UTF-8.

Acceder a Arrays Java

Los Arrays son otro tipo de objetos Java muy utilizados. Se pueden utilizar las funciones de manipulación de arrays de la JNI para crear y acceder a los elementos de un array.

Llamar a Métodos Java

La JNI soporta un completo conjunto de operaciones de llamada que permiten invocar métodos Java desde código nativo. Se localiza el método utilizando su nombre y su firma. Se pueden llamar tanto a métodos estáticos como a métodos no estáticos. Se puede utilizar la herramienta javap para generar signaturas de métodos al estilo JNI para los ficheros de clases.

Acceder a Campos Java

La JNI permite localizar un campo Java utilizando el nombre y la signatura del campo. Se pueden localizar y acceder tanto a campos estáticos como campos no estáticos. Se puede utilizar la herramienta javap para generar signaturas de campos al estilo JNI para los ficheros de clases.

Capturar y Lanzar Excepciones

Es posible también generar, capturar y lanzar excepciones desde el lado nativo.

Referencias Locales y Globales

El código nativo se puede referir a objetos Java utilizando referencias locales o globales. Las referencias locales son sólo válidas dentro de una llamada a método nativo. Son liberadas automáticamente después de que retorne el método nativo. Las referencias globales deben asignarse y liberarse explícitamente.

Threads y Métodos Nativos

A través de la JNI es posible ejecutar métodos nativos en un entorno multi-thread. La JNI ofrece constructores de sincronización básicos para métodos nativos.

Apéndice B

Manual de Usuario

En este anexo se presenta el manual de usuario de la herramienta de software desarrollada. En la sección B.1 se describen los pasos a seguir para realizar la instalación del programa. En las secciones B.2 y B.3 se describe el funcionamiento del programa en los dos modos de funcionamiento que ofrece: el *modo estándar* y el *modo asistido*. Por último, es importante resaltar que este manual de usuario describe el funcionamiento del software considerando solamente el aspecto *cliente* del mismo (ver documentación, capítulo 4).

B.1. Instalación

La instalación del programa se realiza mediante la ejecución del programa **PQoSwin32.exe**. Este permite al usuario seleccionar el directorio destino donde se copiarán los distintos archivos que componen el software de estimación de PQoS.

Para llevar a cabo la intalación del programa se deben seguir los siguientes pasos:

1. Ejecutar el instalador PQoSWin32.exe.
2. A continuación se desplegará en pantalla una ventana donde se podrá ingresar el directorio destino de la instalación (figura B.1). La ventana informa el espacio requerido por la instalación y el espacio disponible en el destino seleccionado.
3. Seleccionar la opción **Instalar**. Se desplegará en pantalla una ventana que indicará el avance de la instalación (figura B.2).
4. Al finalizar la misma se creará en el **Menú de Programas** un enlace a la aplicación instalada.
5. La instalación modifica algunas variables de entorno, por lo que será necesario reiniciar la sesión del usuario antes de utilizar el programa.

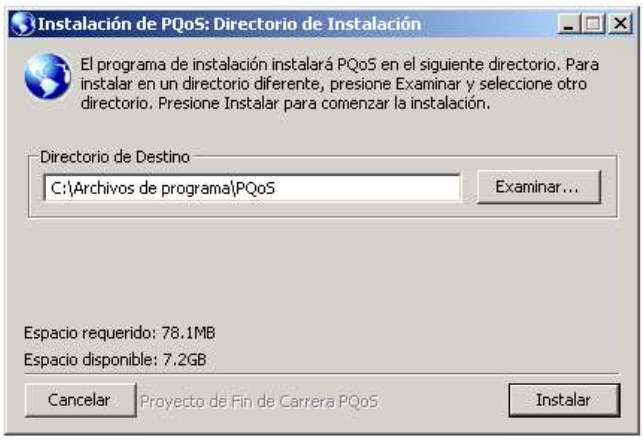


Figura B.1: Intalación del programa - selección del directorio destino

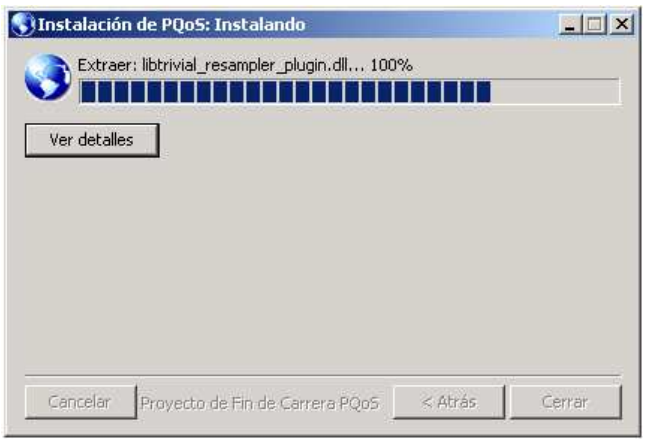


Figura B.2: Intalación del programa - avance de la instalación

Windows 2000 y Windows XP son los dos sistemas operativos donde el programa ha sido probado.

B.2. Modo Estándar

El programa permite dos modos distintos de funcionamiento: el modo estándar y el modo asistido. El modo estándar permite acceder directamente a todas las opciones de funcionamiento de la herramienta. El modo asistido permite al usuario ser guiado a través de las

distintas opciones sin más necesidad que leer en pantalla las instrucciones que debe seguir.

En la figura B.3 se muestra la pantalla inicial del programa, desplegada al ejecutarse la aplicación. El modo de funcionamiento del programa es por defecto el modo estándar. En esta pantalla se identifican los siguientes elementos:

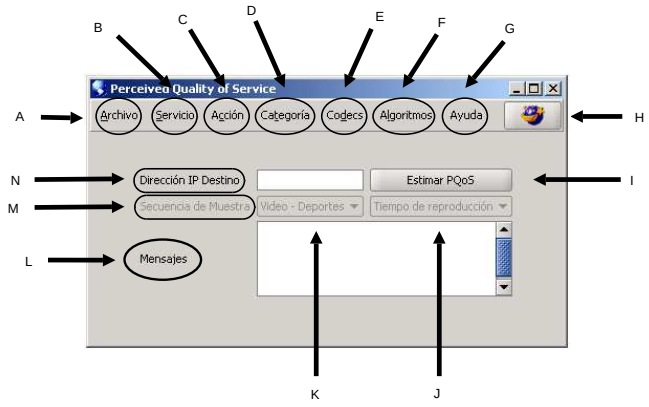


Figura B.3: Modo de funcionamiento estándar

A. Menú Archvio

B. Menú Servicio

C. Menú Acción

D. Menú Categoría

E. Menú Codecs

F. Menú Algoritmos

G. Menú Ayuda

H. Botón de selección de modo de funcionamiento (estándar/asistido)

I. Botón de inicio (estimación de pqos/streaming multimedia de prueba)

J. Selección del tiempo de reproducción del streaming de prueba

K. Selección de la secuencia multimedia del streaming de prueba

L. Ventana de mensajes

M. Sección de secuencias de muestra de streaming multimedia

N. Campo de ingreso de la dirección IP destino de la petición del usuario

Menú Archivo

El menú de archivo permite seleccionar la reproducción de archivos multimedia (figura B.4). La reproducción de video se realiza en base al programa VLC (Video Lan Client), mientras que la reproducción de audio se realiza con la JMF (Java Media Framework).

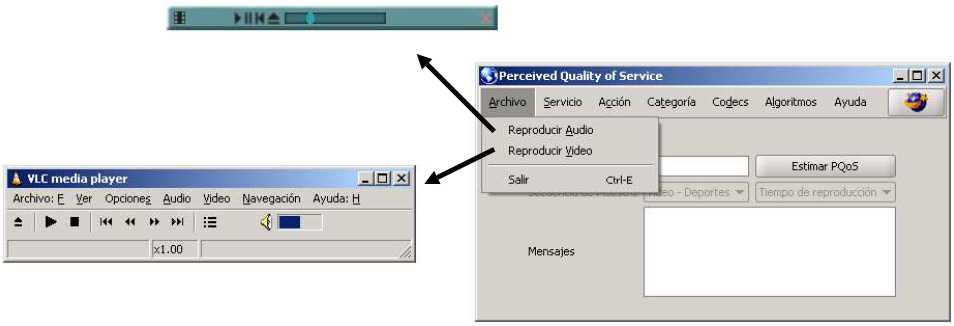


Figura B.4: Menú Archivo - reproducción multimedia

Menú Servicio

El menú de servicios permite seleccionar el tipo de servicio para el cual se realizará la estimación de PQoS. Las opciones disponibles abarcan los servicios de *streaming de audio* y *streaming de video*.

Menú Acción

El menú de acciones permite seleccionar la acción a realizar. Dos son las acciones posibles: la acción *predecir calidad* es la fundamental y permite realizar la estimación de PQoS para el servicio solicitado en el menú de servicios.; la acción *streaming de prueba* permite realizar un streaming multimedia de prueba desde el otro extremo de la conexión (recordar que el software del otro extremo corre como servidor, ver documentación, capítulo 4).

Menú Categoría

En caso de seleccionar el servicio de *streaming de video* y la acción *predecir calidad*, el menú de categorías permite seleccionar la cantidad de movimiento del video que se pretende "descargar". Esta clasificación asume por parte del usuario un cierto conocimiento del contenido del servicio para el cual se solicita la estimación de PQoS. Las opciones que se ofrecen incluyen: *movimiento alto*, *movimiento medio* y *movimiento bajo*. En caso de desconocer esta información se deberá seleccionar la categoría *movimiento alto*, contemplándose así el peor caso.

Menú Codecs

El menú de codecs permite seleccionar, dependiendo del servicio escogido, el codec que se utilizará. Para el caso de video se ofrece codificación en *MPEG-1* y *MPEG-4*; en audio es posible escoger *G.711*, *G.723* y *GSM*.

Menú Algoritmos

En caso de seleccionar la acción *predecir calidad*, el menú de algoritmos permite seleccionar los algoritmos que se utilizarán en la estimación de PQoS. Dependiendo del tipo de algoritmos seleccionados (intrusivos y/o no intrusivos) se realizarán la o las mediciones correspondientes (ver documentación, capítulo 4). Los algoritmos disponibles para el caso de video son: *PSNR*, *SSIM*, *ITS* y *RNN video*. Para el caso de audio es posible seleccionar: *Pesq*, *EMBSD*, *MNB*, *E-Model* y *RNN audio*.

Menú Ayuda

El menú de ayuda permite activar la ayuda en línea del programa, desplegando en pantalla una ventana con información de utilidad.

Sección de secuencias de muestra de streaming multimedia

En caso de seleccionar la acción *streaming de prueba* se activa en la pantalla principal la sección de secuencias de muestra (figura B.5). Esta permite al usuario realizar un streaming de prueba de un video o secuencia de audio que seleccionará (a través del menú de selección K) durante un tiempo a elección (menú de selección L). Dentro de las opciones se ofrecen secuencias de video del tipo *deportes*, *noticias*, *acción*, así como también secuencias de audio. El menú de codecs permite seleccionar el codec que se utilizará en la prueba.

Figura B.5: Streaming multimedia de prueba

Ventana de mensajes

En esta ventana se despliegan todos los mensajes de información generados por el programa. Estos incluyen tanto mensajes de error como mensajes de avance de la estimación.

Campo de dirección IP destino

Permite ingresar la dirección IP del equipo destino contra el cual se realizará la estimación de PQoS. En caso de no ingresar una dirección IP destino se despliega en la ventana de mensajes un aviso correspondiente.

Botón de inicio

Al presionar este botón se procesan las distintas opciones seleccionadas, llevándose a cabo una de las siguientes tareas:

- Estimación intrusiva de PQoS para audio.
- Estimación intrusiva de PQoS para video.
- Estimación no intrusiva de PQoS para audio.
- Estimación no intrusiva de PQoS para video.
- Streaming de prueba de audio.
- Streaming de prueba de video.

En caso de que las selecciones realizadas no definan claramente una de estas tareas se despliega en la ventana de mensajes un aviso de la información faltante (figura B.6) . El programa se bloquea durante 10 segundos antes de permitir un nuevo intento.

Figura B.6: Mensajes desplegados

Botón de selección de modo de funcionamiento

Presionar este botón implica un cambio del modo de funcionamiento del programa.

Resultados de la estimación

Los resultados de la estimación de PQoS se despliegan en pantalla una vez finalizada la tarea seleccionada (figura B.7).

Figura B.7: Resultados de la estimación de PQoS

B.3. Modo Asistido

El modo de funcionamiento asistido presenta la principal ventaja de guiar al usuario en la determinación de la tarea a realizar. Con el fin de facilitar el uso del programa se presentan opciones más restringidas; en particular, no es posible seleccionar el tipo de algoritmo a utilizar (se utilizan algoritmos por defecto). En la figura B.8 se presenta un resumen gráfico del modo de funcionamiento asistido.

1. La primer ventana del modo asistido permite seleccionar el tipo de servicio para el cual se desea realizar la estimación de PQoS (streaming de audio o video).

2a. En caso de seleccionar el servico de streaming de audio se escoge a continación la codificación que se utilizará en el mismo.

2b. En caso de seleccionar el servico de streaming de video se debe elegir primero la categoría del video (idem modo estándar), escogiendo a continuación la codificación a utilizar.

3. La siguiente ventana permite seleccionar la metodología de estimación de PQoS que se utilizará (*intrusiva* o *no intrusiva*).

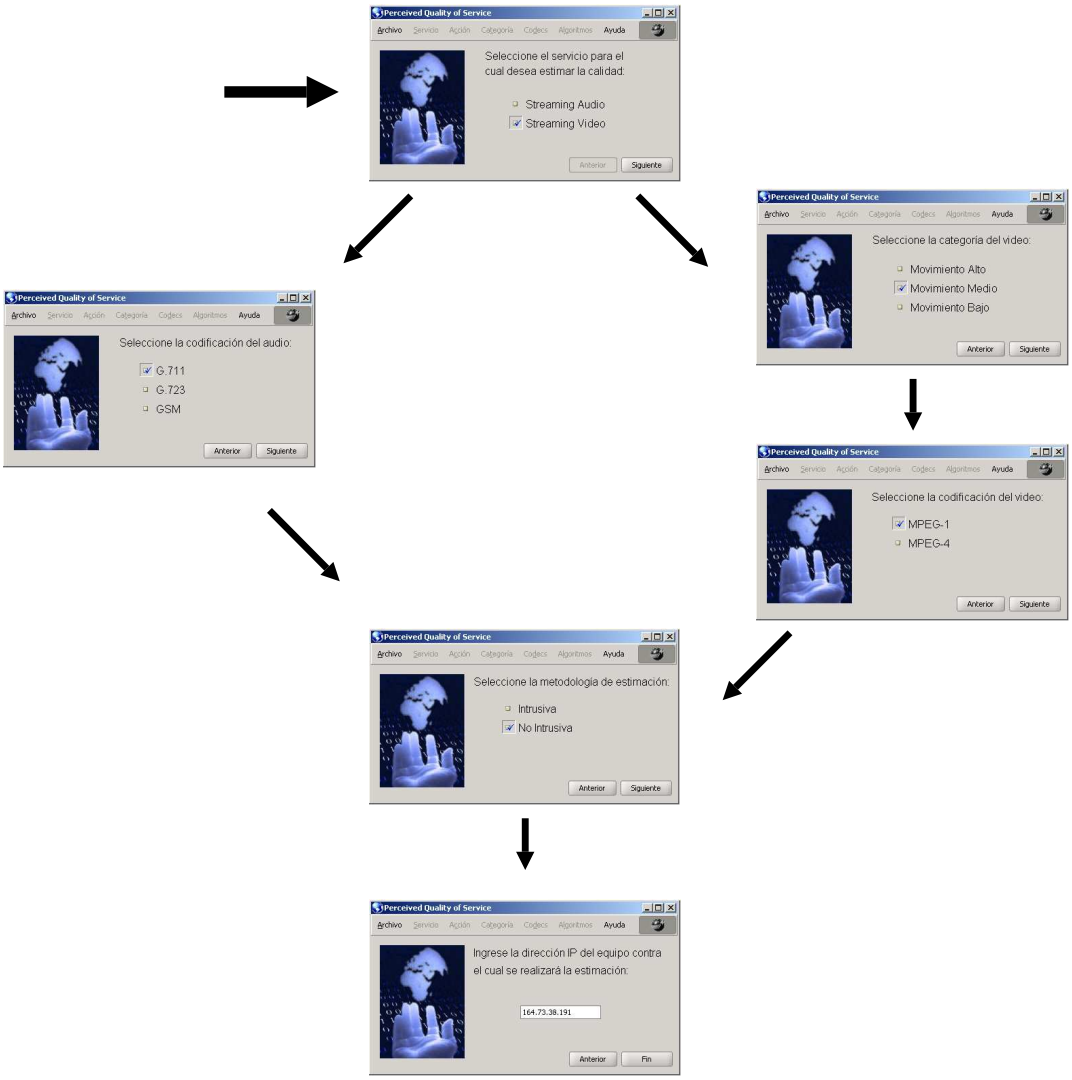


Figura B.8: Modo asistido

4. En la última ventana se debe ingresar la dirección IP del equipo contra el cual se realizará la estimación.

Apéndice C

Random Neuronal Networks

En este anexo se describe el modelo de la red neuronal aleatoria utilizado en el proyecto. Se presentan dos enfoques distintos de análisis. Un primer enfoque que presenta a la red neuronal aleatoria como un conjunto de neuronas interconectadas entre sí que intercambian impulsos excitadores e inhibidores a tasas de generación exponencialmente distribuidas, y un segundo enfoque que analiza el modelo como una red de Jackson a la cual se agregan clientes *negativos*.

C.1. Descripción del modelo de la RNN

Una red RNN no es más que una generalización de las redes de Jackson a la cual se agregan clientes negativos (al arribar a una cola disminuyen el número de clientes en una unidad en lugar de aumentarlo). Conocidas también como G-Networks (en honor a su creador, E.Gelenbe), las RNN presentan la virtud de ser muy estables es comparación con el resto de las redes neuronales. En el modelo implementado cada i-ésima neurona de la red se comporta como un enlace ./M/1, donde definiremos N_t^i como el número de clientes en el enlace i en tiempo t. La tasa de servicio del enlace i se denota por r_i, y luego de dejar el i-ésimo enlace el cliente deja la red con probabilidad d_i, se dirige al enlace j como un cliente positivo con probabilidad p_{ij}^+ o como un cliente negativo con probabilidad p_{ij}^-. Del exterior de la red sólo arriban también clientes positivos y negativos de acuerdo a un proceso de Poisson de tasa λ_i^+ y λ_i^- respectivamente. Las transferencias entre enlaces se realizan de forma instantánea.

Considerando el segundo enfoque, se define una neurona de la red como un elemento que tiene asociado un cierto valor natural característico N_t^i llamado *potencial de la neurona*. Todas las neuronas interconectadas entre sí generan impulsos excitadores e inhibidores que modifican su propio potencial y el de las neuronas vecinas. Los impulsos excitadores incrementan el potencial de la neurona en una unidad, mientras que los inhibidores lo reducen en una unidad. Al mismo tiempo, este potencial disminuye en una unidad al generarse un

impulso de cualquier tipo. Cuando N_t^i es mayor a cero se envían impulsos hacia el resto de las neuronas a intervalos de tiempo iid (independientes e idénticamente distribuidos) con distribución exponencial de parámetro r_i. Cada impulso generado es enviado hacia la j-ésima neurona de la red como excitador (con probabilidad p_{ij}^+), como inhibidor (con probabilidad p_{ij}^-) o hacia afuera de la red (con probabilidad d_i).

En lo que sigue se continuará con el enfoque de redes de Jackson. Del análisis de redes se puede ver que cuando al sistema se asocia un proceso de Markov $N_t = (N_t^1, N_t^2, ..., N_t^M)$, donde M representa el número de neuronas de la red y el mismo resulta estable, su distribución estacionaria cumple:

$$P(N_t = (k_1, k_2, ..., k_M)) = \prod_{i=1}^{M} (1 - \rho_i) \cdot \rho_i^{k_i} \tag{C.1}$$

donde ρ_i representa la probabilidad de que el i-ésimo enlace no esté vacío (normalmente se lo define como carga o utilidad).

La utilidad de cada enlace se obtiene mediante la resolución del siguiente sistema no lineal:

$$\rho_i = \frac{\lambda_i^+ + \sum_{j=1}^{M} \rho_j \cdot \omega_{ji}^+}{r_i + \sum_{j=1}^{M} \rho_j \cdot \omega_{ji}^-} \tag{C.2}$$

donde ω_{ji}^+ y ω_{ji}^- corresponden a las tasas de generación de impulsos excitadores e inhibidores de la neurona j a la neurona i respectivamente ($\omega_{ji}^+ = r_j * p_{ji}^+$ y $\omega_{ji}^- = r_j * p_{ji}^-$)

Antes de avanzar en el análisis de la red es importante tener en cuenta que en este caso particular de implementación se utiliza una red feed-forward de tres etapas: una primera etapa de enlaces de entrada a las cuales sólo arriban clientes positivos del exterior, una etapa de salida que sólo envía clientes al exterior y una etapa intermedia de enlaces (pensando en neuronas, se las define como *neuronas escondidas*) que interconecta entrada y salida. Al mismo tiempo se consideran solamente los clientes positivos que arriban desde el exterior de la red.

Consideremos ahora la información obtenida de los test subjetivos. Recordando que para cada secuencia sometida al test subjetivo se tiene la configuración de parámetros (llamemosle $\gamma_s = (\gamma_{1s}, ..., \gamma_{Ps})$, donde P es el número de parámetros considerados y γ_{is} su valor correspondiente) y el valor de MOS correspondiente (llamemosle μ_s), tenemos un set de pares (γ_s, μ_s) que nos permitirá entrenar la red de neuronas. Para ello se tendrán P neuronas de entrada y una neurona de salida y se buscará que cuando la tasa de arribo a las neuronas de entrada i ($i = 1..P$) sea $\lambda_i^+ = \gamma_{is}$, la salida sea $\rho_o = \mu_s$.

Esto resulta en un problema de minimización en el parámetro ω de la salida:

$$\omega_o = argmin_\omega \frac{1}{2} \sum_{s=1}^{S} (\rho_o(\omega, \gamma_s) - \mu_s)^2 \tag{C.3}$$

Este problema puede ser resuelto mediante el uso de técnicas estándar de optimización. El algoritmo de resolución puede ser consultado en [23].

Apéndice D

Medidas Subjetivas de calidad

En este anexo se presentan los métodos subjetivos utilizados para medir la calidad de los distintos medios de transmisión[1].

Existen dos clases de asignación subjetiva de calidad. Primero, están los que miden la performance del sistema de transmisión bajo condiciones óptimas. Estos son los llamados de asignación de calidad. La otra clase mide la habilidad del sistema de transmisión en mantener la calidad bajo condiciones no óptimas. Estos son llamados de asignación de defectos.
Los distintos métodos se presentan en las recomendaciones, ITU BT.500 [15] para imagen y video, y ITU P.800 [16] para voz y sonido.

D.1. Métodos Subjetivos para Audio

En la recomendación P.800 ([16]) se describen los distintos métodos separados en dos grandes categorías, test de conversación y test en los que solo se escucha (Listening test).

Test de conversación

En este tipo de test la idea es reproducir en un laboratorio una conversación telefónica con las condiciones reales de servicio. Para eso se colocan dos personas en cabinas acústicamente aisladas. La recomendación especifica la dimensión de las cabinas, la decoración, la atenuación al sonido externo, la densidad espectral de ruido ambiente, etc. Al final de cada conversación los participantes le asignan un valor a la calidad de acuerdo a la siguiente escala

[1] Por medio de transmisión se entiende cualquier parte que lo componga, por ejemplo un codec, el medio físico, etc.

Excelente	5
Buena	4
Razonable	3
Pobre	2
Mala	1

A la media aritmética de la colección de resultados se le llama valor medio de opinión de la conversación (mean conversation-opinion score) y se representa por el símbolo MOS_c.
Si bien este test es lo más adecuado para asignar valores subjetivos de calidad a conversaciones telefónicas los requisitos técnicos son inalcanzables para nosotros, así como la implementación de las conversaciones complicada. Por esto optamos en utilizar el segundo tipo de test, los Listening test.

Listening test-Absolute Category Rating (ACR)

Estos test consiste en que distintas personas escuchen muestras de audio (frases cortas) y les asignen un valor a la calidad utilizando la escala presentada anteriormente. El nombre de absolute category rating viene de que las personas juzgan la calidad solo escuchando la señal "distorsionada" (ya transmitida) y no tienen acceso a la señal original.
En este método no se espera el mismo realismo que en los test de conversación (no se está simulando una conversación) y por lo tanto las restricciones son menos severas. Al igual que en el otro método se especifica el lugar en que se deben realizar los test y la calidad del sistema de reproducción y grabación de las muestras de audio.

Las muestras de audio consisten en cierto número de frases simples, con sentido y de corta duración. Las frases se agrupan de manera aleatoria de forma que no tengan sentido entre si. La duración debe ser de entre 2 y 3 segundos.

La persona escucha las distintas muestras de audio, separadas entre si por unos diez segundos. En este tiempo el sujeto debe dar un valor de calidad a la muestra que acaba de escuchar, siguiendo la escala antes mencionada. A la media aritmética de los resultados de las distintas personas se denomina resultado de opinión media (mean opinion score) MOS.

Listening test-Degradation Category Rating (DCR)

Este método es una modificación del anterior, las muestra se presentan en parejas, separadas entre si por 1 segundo, donde la primera es la señal de referencia (sin deterioro) y la segunda es la señal ya transmitida. Una vez que escucharon ambas los individuos tienen que asignarle un valor a la degradación sufrida de acuerdo a la siguiente escala

La degradación es inaudible	5
La degradación es audible pero no molesta	4
La degradación es un poco molesta	3
La degradación es molesta	2
La degradación es muy molesta	1

El valor medio de los resultados obtenidos es el $DMOS$ (degradation mean opinion score).

Listening test-Comparison Category Rating (CCR)

Este método es muy similar al anterior. Las muestras se presentan en parejas, señal de referencia y señal distorsionada. Pero a diferencia del DCR donde la primer señal es siempre la de referencia, en el CCR el orden es aleatorio. Las personas asignan un valor a la calidad de la segunda señal comparada con la primera de acuerdo a la siguiente escala

Mucho mejor	3
Mejor	2
Apenas mejor	1
Igual	0
Apenas peor	-1
Peor	-2
Mucho peor	-3

El valor medio de los resultados es el $CMOS$ (comparison mean opinion score).

La ventaja de los dos últimos métodos (DCR y CCR) es que permiten evaluar la influencia que tiene el sistema de transmisión en la degradación de la calidad.

D.2. Métodos Subjetivos para Imagen y Video

En la recomendación [15] se presentan varios métodos de asignación de calidad. Describiremos primero los métodos que mejor se adecuan a nuestro interés, el DSIS (double-stimulus impairment scale) y el DSCQS (double-stimulus continuous quality scale). Ambos métodos son de comparación de señales[2] (señal de referencia y señal distorsionada), por eso el nombre de doble estímulo. Luego describiremos una serie de métodos alternativos de asignación que pueden llegar a ser útiles bajo ciertas circunstancias.

La recomendación especifica condiciones generales para realizar los diferentes test. Se indica las condiciones de visualización, esto es, niveles de iluminación, contraste y resolución

[2]las señales pueden ser una imagen, una secuencia de imágenes o un video

de los monitores utilizados. Se dan pautas para la elección de las señales a utilizar, así como requisitos que deben cumplir los participantes.

DSIS (double-stimulus impairment scale)

Este método sirve para la evaluación de sistemas de transmisión nuevos así como para determinar la influencia de ciertos parámetros en la fidelidad de la transmisión. Lo primero es escoger una cierta cantidad de muestras de video (asociadas con distintos valores del parámetro a evaluar) que cubran el rango de degradación, en pequeños intervalos.

El método consiste en comparar la degradación de la señal de prueba en relación a la señal de referencia. Existen dos variantes en la forma de presentar las señales:

1. La señal de referencia y distorsionada se presentan una sola vez (ver figura D.1 a).
2. La señal de referencia y distorsionada se muestran dos veces (ver figura D.1 b).

Luego de ver las señales las personas tienen un tiempo para votar, utilizando la siguiente escala

La degradación es imperceptible	5
La degradación es perceptible pero no molesta	4
La degradación es un poco molesta	3
La degradación es molesta	2
La degradación es muy molesta	1

Al comienzo de cada test se les explica a los participantes sobre la asignación, la escala a utilizar, la forma y tiempo en que se presentan las señales (video de referencia, video gris, video de prueba, tiempo para votar). Estas ideas se ejemplifican presentado señales para que las personas se acostumbren al mecanismo del test, sin tener en cuanta el resultado asignado.

El grado de degradación de las señales presentadas en cada test debe ser tal que las personas utilicen todo la escala de asignación y que la media sea cercana a 3.
Una vez obtenidos los valores se hace un estudio estadístico de los mismos. Dicho estudio se explica en las sección D.2.

DSCQS (double-stimulus continuous quality scale)

Este método es útil cuando las secuencias utilizadas no cubren todo el rango de calidad. El individuo ve las dos señales (referencia y distorsionada) y debe asignar la calidad de ambas. A diferencia del DSIS en el cual primero se muestra la referencia y luego la distorsionada, en

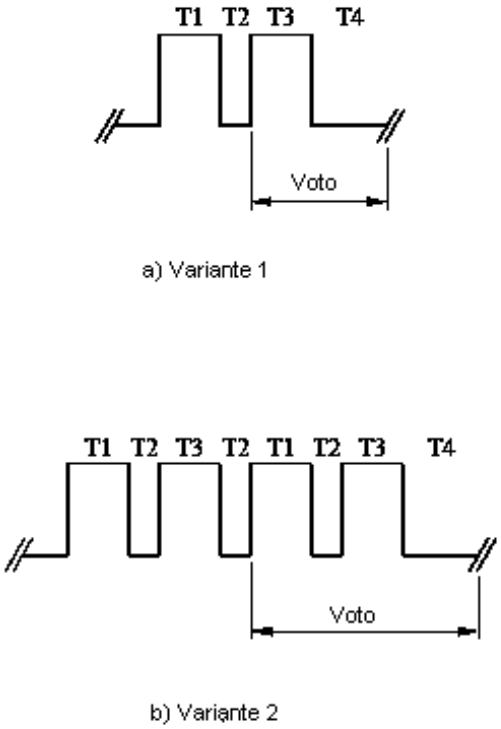


Figura D.1: Formato de presentación método DSIS

este caso se presenta de forma aleatoria.

Se presentan ambas señales y durante la siguiente repetición (ver figura D.2) los participantes deben asignar la calidad de cada señal haciendo una marca en una escala continua (ver figura D.3). Las escalas se imprimen de a pares, igual que se presentan las señales. La escala es continua para evitar errores de cuantización, pero esta dividida en cinco intervalos de igual longitud que coinciden con la escala clásica de cinco puntos (ver sección D.1). Estas divisiones se incluyen para guiar a los participantes.

La escala continua se pasa de medida de longitud a valores entre 0 y 100. Luego se calcula

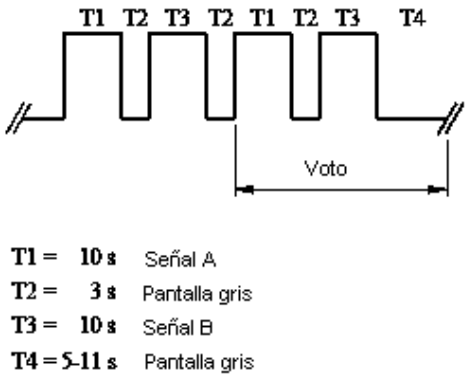


Figura D.2: Formato de presentación método DSCQS

Figura D.3: Escala utilizada en el método DSCQS

la diferencia entre ambas marcas. El estudio posterior se explica en la sección D.2.

Métodos alternativos

SS (single stimulus)

En este método, una sola señal se presenta y la persona otorga una valor de calidad de acuerdo a la escala clásica de cinco puntos (ver sección D.1).

Cuando se quiere utilizar este método para estudiar la influencia de ciertos parámetros, las señales se pueden preparar de dos formas. En la primera, cada señal representa un cierto nivel de un único parámetro. En la otra, cada señal representa un nivel de cada uno de los parámetros a estudiar, pero entre señales, cada nivel de cada parámetro aparece con todas las conbinaciones posibles de niveles de los otros parámetros. Ambos métodos permiten atribuir los resultados a los distintos parámetros. Con el ultimo método se puede detectar interacción entre parámetros.
Las sesiones de test consiste de una serie de asignaciones a distintas señales. Las señales se presentan de forma aleatoria, debiendo ser las secuencias distintas para distintas personas. Hay dos variantes de presentación:

1. Las señales de prueba se presentan una sola vez; al comienzo de introducen señales para que la gente se acostumbre. El proceso para asignar la calidad es el siguiente. Se muestra una pantalla gris, luego la señal de prueba y luego nuevamente la pantalla gris. La duración media es de 3, 10 y 10 segundos respectivamente. La persona asigna la calidad durante la señal de prueba o en la pantalla gris que la prosigue.

2. Las señales se presentan tres veces organizando la sesión en tres presentaciones, cada una incluyendo todas las señales a presentar. La primer presentación se usa para estabilizar la opinión de los participantes, por lo que no se tiene en cuenta su resultado. El resultado asignado a cada señal es la media de la segunda y tercer presentación. Se imponen las siguientes restricciones al orden de las señales en cada presentación:

   - una señal no puede estar en la misma posición que en otra presentación.
   - una señal no se encuentra inmediatamente después que la misma señal de las otras presentaciones.

   La presentación típica es: señal de prueba y pantalla gris. La duración es de 10 y 5 segundos. Los sujetos solo pueden votar durante la pantalla gris.

Método de comparación de señales

En este método se muestran dos señales y el sujeto asigna un valor a la relación de calidad entre estas dos utilizando la escala presentada en la sección D.1. A diferencia de los métodos vistos antes en este caso lo que se muestra no es la señal de referencia y la distorsionada, si no dos señales cualesquiera.
El procedimiento para realizar el test es similar al del método SS.

SSCQE (single stimulus continuous quality evaluation)

La introducción de la compresión digital produce un deterioro en la calidad de los videos que son dependientes de la escena y varían con el tiempo. Incluso en videos de corta duración la calidad puede fluctuar dependiendo del contenido de la escena. De acuerdo a lo expuesto en [15], los métodos antes presentados no son suficientes para asignar la calidad de este tipo de señal. Además, el método de doble estímulo utilizado en el laboratorio no es una buena réplica del servicio real que es de estímulo único (SS). Consideraron útil que la calidad de video digital se mida de forma continua.

Para llevar esto acabo se utiliza un sistema electrónico conectado a una computadora que registra de forma "continua" (2 veces por segundo) el valor de calidad asignado por el sujeto. El sistema en cuestión es un cursor que se mueve en una recta, y la posición del mismo indica la calidad. La escala utilizada es la misma que en la sección D.2, variando entre 100 y 0.
A los sujetos se le presentan sesiones con el siguiente formato:

- *Segmento de programa (SP):* un SP corresponde a un tipo de programa (deportes, noticias, acción, etc.) procesado de acuerdo a alguno de los parámetros de calidad (PC) que se evalúen (bitrate, probabilidad de perdidas); cada PS debe durar por lo menos 5 minutos.
- *Sesión de test (ST):* una ST es una serie de una o más combinaciones de SP/PC sin separaciones y ordenadas pseudo-aleatoriamente. Cada ST contiene al menos una vez todos los SP y PC pero no necesariamente todas las combinaciones SP/PC; cada ST tiene una duración de entre 30 y 60 minutos.

Los resultados se juntan de todas las sesiones de test. Se puede construir un gráfico de calidad media en función del tiempo con el que se muestra en la figura D.4.

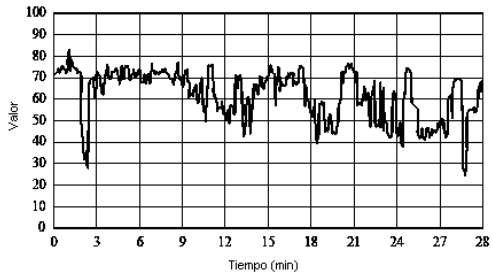


Figura D.4: Calidad media en función del tiempo

SDSCQE (simultaneous double stimulus for continuous quality evaluation)

Cuando se quiere evaluar la fidelidad de un sistema de transmisión, señales de referencia deben ser introducidas. Este método es muy similar al anterior, cambiando la forma en que se presentan las señales a los participantes. Éstos observan las dos señales (referencia y distorsionada) simultaneamente. Los sujetos deben evaluar la fidelidad de la señal distorsionada moviendo el cursor. Cuando la fidelidad es perfecta el cursor debe estar en el máximo (valor 100) y cuando la fidelidad es nula en el fondo (valor 0).

Análisis y presentación de los datos

Después de realizar un método se recoge una cantidad importante de datos. Estos datos, en forma de hojas de valores de cada sujeto, o su equivalente electrónico, se debe reducir por técnicas estadísticas para obtener resultados que resuman la performance del sistema evaluado.
Este análisis es aplicable a todos los métodos antes presentados. En algunos los valores asignados por los sujetos se encuentran en escalas de cinco puntos. En otros es continuo, en este caso los resultados se normalizan a valores enteros entre 0 y 100.

El primer paso es calcular el valor medio para cada condición del test. Por condición del test se entiende, las señales distorsionadas de acuerdo a un conjunto de valores de los parámetros que afectan las calidad. El valor medio esta dado por:

$$\overline{u}_j = \frac{1}{N}\sum_{i=1}^{N} u_{ij} \tag{D.1}$$

donde u_{ij} es el valor asignado por el sujeto i a la condición de test j, y N es el número de participantes. Para condición de test se calcula el intervalo de confianza. Se recomienda ([15]) calcular el intervalo de confianza del 95 % dado por:

$$[\overline{u}_j - \delta_j, \overline{u}_j + \delta_j] \tag{D.2}$$

donde

$$\delta_j = 1{,}96\frac{S_j}{\sqrt{N}} \tag{D.3}$$

donde la desviación estándar estimada, S_j, esta dada por

$$S_j = \sqrt{\sum_{i=1}^{N} \frac{(u_{ij} - \overline{u}_j)^2}{N-1}} \tag{D.4}$$

Con probabilidad 95 % el verdadero valor medio se encuentra dentro de este intervalo.

Eliminación de sujetos

Presentamos ahora el método recomendado en [15] para eliminar los sujetos que podrían haber asignado resultados incoherentes.

Lo primero es ver si la distribución de valores de la presentación es normal o no usando el test β_2 (calculando el cociente entre el momento de cuarto orden y el cuadrado del momento de segundo orden). Si β_2 está entre 2 y 4 , la distribución se puede considerar normal. Para cada condición de test, se compara el valor asignado por cada participante u_{ij} con la media $\overline{u}_j$ más, dos veces la desviación estándar S_j si la distribución es normal, o más $\sqrt{20}$ veces S_j si la distribución es no normal. A esta suma se le denomina P_j. También se compra con la media menos dos ($\sqrt{20}$) veces la desviación estándar si la distribución es normal (no normal). Esta cantidad se denomina Q_j. Cada vez que el valor asignado por el sujeto i se encuentre por encima de P_j un contador asociado a cada sujeto, p_i, se incrementa. De la misma forma, cada vez que el valor asignado por el sujeto i se encuentra por debajo de Q_j un contador, q_i, se incrementa. Al final los siguientes cocientes se calculan, $p_i + q_i$ dividido por la cantidad de resultados de cada individuo en la presentación y el valor absoluto de $p_i - q_i$ dividido por $p_i + q_i$. Si el primer cociente es mayor a 0.05 y el segundo menor a 0.3, el sujeto i debe ser eliminado.

Se puede expresar matemáticamente de la siguiente manera. Para cada condición de test calcular el valor medio, la desviación estándar y el coeficiente β_{2j} dado por:

$$\beta_{2j} = \frac{m_{4j}}{m_{2j}^2}, \qquad con \qquad m_{xi} = \frac{\sum_{i=1}^{N}(u_{ij} - \overline{u}_j)^x}{N} \tag{D.5}$$

Para cada sujeto i calcular p_i y q_i (inicializados a 0) de la siguiente forma:

si $2 \leq \beta_{2j} \leq 4$

si $u_{ij} \geq \overline{u}_j + 2S_j \Longrightarrow p_i = p_i + 1$;

si $u_{ij} \leq \overline{u}_j - 2S_j \Longrightarrow q_i = q_i + 1$;

de otra forma

si $u_{ij} \geq \overline{u}_j + \sqrt{20}S_j \Longrightarrow p_i = p_i + 1$;

si $u_{ij} \leq \overline{u}_j - \sqrt{20}S_j \Longrightarrow q_i = q_i + 1$;

Si $\frac{p_i+q_i}{J} > 0{,}05$ y $\left|\frac{p_i-q_i}{p_i+q_i}\right| < 0{,}3$ el sujeto i debe ser descartado, donde J es el número de condiciones de test de la presentación.
Una vez descartados los valores de los sujetos que dieron respuestas incoherentes, de acuerdo al criterio anterior, se debe recalcular el valor medio. Este sera el resultado del test en cuestión. Cabe aclarar que el criterio de eliminación de sujetos descrito anteriormente no es válido para los métodos de evaluación continua de calidad.

Apéndice E

Modelo de Pérdidas

E.1. Modelo de Gilbert

En este anexo se presenta el modelo utilizado para simular la pérdida de paquetes en una red IP.
La pérdida de paquetes en una red IP no es independiente de paquete a paquete, sino que dichas pérdidas ocurren en ráfagas. Bolot, en [20] estudió la distribución de la pérdida de paquetes en internet, y concluyó que se podía aproximar por un modelo Markoviano de pérdidas cono el de Gilbert. Teniendo esto en cuenta, se utilizó un modelo de Markov de dos estados (Modelo de Gilbert) para simular las pérdidas en una red IP.
En la figura E.1 se muestra el Modelo de Gilbert con probabilidades de transición p y q. Como mencionamos antes, consta de dos estados. Una transición al estado 0 corresponde a

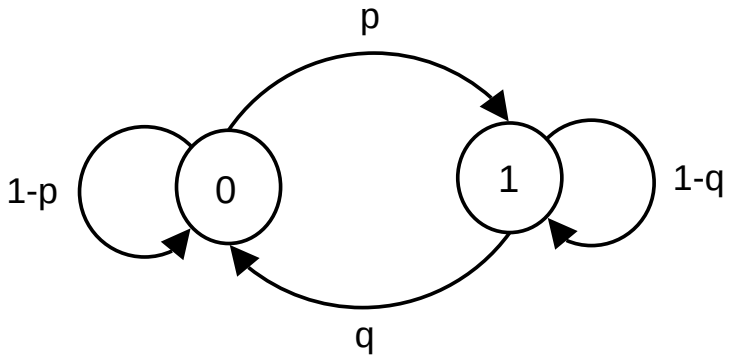


Figura E.1: Modelo de Gilbert.

que el paquete ha sido recibido, y una transición al estado 1 corresponde a que el paquete se

perdió. El hecho de que sean dos estados implica que la transición para cada paquete que se manda sólo depende del estado actual, o sea de lo que pasó con el paquete anterior.
Una forma alternativa de describir el modelo es en función de la probabilidad condicional e incondicional de pérdida, que denotaremos p_c y p_u respectivamente. Es fácil ver que $p_c = 1-q$.
El calculo de la probabilidad incondicional es un poco más complicado.
Llamemos π_0 y π_1 a la probabilidad de estar en el estado 0 y 1 respectivamente. En equilibrio debemos tener:

$$\pi_0 = (1-p)\cdot\pi_0 + q\cdot\pi_1 \quad y \quad \pi_1 = 1-\pi_0 \implies \pi_0 = \frac{q}{p+q},\ \pi_0 = \frac{p}{p+q} \tag{E.1}$$

Teniendo en cuenta que π_1 representa la probabilidad media de que el sistema se encuentre en el estado 1, es decir de perder paquetes. Por lo tanto es igual a p_u.
La probabilidad de que se pierdan k paquetes consecutivos y luego se reciba uno es:

$$p_k = (1-q)^{k-1}\cdot q \tag{E.2}$$

A partir de ésta se puede calcular el largo medio de las ráfagas de pérdida como:

$$\overline{k} = \sum_{k=1}^{\infty} k\cdot(1-q)^{k-1}\cdot q \ = -q\cdot\frac{\partial}{\partial q}\left(\sum_{k=1}^{\infty}(1-q)^k\right) \ = \frac{1}{q} \tag{E.3}$$

Se podrían implementar modelos más complejos como Markovianos de orden n, donde la pérdida de un paquete depende de los sucedido con los n paquetes anteriores. El número de estados crece exponencialmente, un modelo de orden n tiene 2^n estados. Este tipo de modelo permite tener en cuenta dependencias largas. Si bien permiten modelar mejor algunas situaciones, comparar el estado de la red de dos caminos diferentes se torna complicado. Se deben comparar 2^n parámetros y además no es claro como compararlos, ya que estos parámetros no tienen un significado tan simple como la probabilidad de pérdida media.
Esto ha hecho del modelo de Gilbert uno de los más utilizados en la simulación de pérdidas.

Apéndice F

RTP y RTCP

F.1. Real Time Protocol

Una aplicación que envía contenido multimedia sobre la red en tiempo real, le debe agregar números de secuencia y marcas de tiempo a los segmentos antes de pasarlo a la capa de transporte. Estos son necesarios para reconstruir el contenido en el receptor sin tener que esperar a bajar el contenido completo (o una parte suficientemente grande). Es conveniente tener una estructura estandarizada de paquetes que tenga campos que indiquen el tipo de codificación, marcas de tiempo, número de secuencia y otros campos potencialmente útiles. En el RFC 1889 se define el estándar RTP que cumple con las características mencionadas. Puede ser utilizado para transportar formatos comunes como WAV o GSM para audio y MPEG1 o MPEG2 para video. También permite utilizar formatos propietarios de sonido y video.

F.1.1. Conceptos Básicos

El protocolo RTP es utilizado generalmente sobre UDP, segmentos de datos de video o audio son generados por el transmisor de la aplicación multimedia, estos segmentos son encapsulados en paquetes RTP, y cada paquete RTP es luego encapsulado en segmentos UDP. Como RTP provee servicios a la capa de aplicación, puede ser visto como una subcapa de la capa de transporte, como se muestra en la figura F.1.

Desde la perspectiva del desarrollador, RTP no forma parte de la capa de transporte. Para el desarrollador RTP forma parte de la capa de aplicación, se debe escribir el código que encapsula los segmentos de contenido multimedia, en paquetes RTP, la aplicación envía luego estos paquetes por medio de un socket UDP. En nuestro caso utilizamos una API de SUN llamado Java Media Framework (JMF) que nos brinda los servicios del protocolo RTP.

Si utilizamos RTP para enviar una señal de voz codificada en PCM a 64 Kbps, y supon-

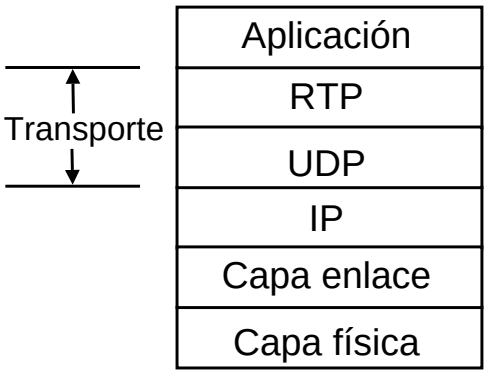


Figura F.1: RTP como una subcapa de la capa de transporte

gamos que la aplicación genera segmentos de 160 bytes cada 20 milisegundos, la aplicación le adjunta al segmento de audio la cabecera RTP. La cabecera incluye el tipo de codificación, un número de secuencia y una marca de tiempo. Los paquetes RTP formados por los datos más la cabecera son enviados por un socket UDP. Del lado del receptor, la aplicación extrae el segmento de audio y utiliza la información de la cabecera RTP para poder decodificar y reproducir correctamente el segmento de audio.

RTP no provee ningún mecanismo para garantizar la entrega, ni para garantizar el orden de los paquetes. Es más, el encapsulamiento RTP sólo es visto en las "puntas"del sistema, los enrutadores no distinguen entre datagramas que llevan carga RTP y los que no. Cada fuente de contenido multimedia tiene su propio stream de paquetes RTP, en una videoconferencia entre dos participantes se pueden abrir cuatro streams RTP, dos para el audio y dos para el video (uno para cada lado).

F.1.2. Cabecera de los paquetes RTP

Los cuatro campos principales de la cabecera RTP son: el tipo de carga útil, el número de secuencia, la marca de tiempo y el identificador de la fuente.

Figura F.2: Campos de la cabecera RTP

Tipo de carga útil El campo tipo de carga útil tiene 7 bits de largo, esto nos permite 2^7, o sea 128 tipos de carga útil diferentes. Para un stream de audio, el tipo de carga útil indica el tipo de codificación (por ejemplo PCM, G.723, etc). Si el transmisor decide cambiar el tipo de codificación en medio de una sesión, le informa al transmisor del cambio por medio de este campo. El transmisor podría cambiar la codificación para aumentar la calidad, o para reducir el flujo de bits de la sesion.

Tipo de carga útil	Codificación	Frecuencia de Muestreo	Throughput
0	PCM ley mu	8 KHz	64 Kbps
1	1016	8 KHz	4.8 Kbps
3	GSM (G.723)	8 KHz	13 Kbps
7	LPC	8 KHz	2.4 Kbps

Ocurre lo mismo para los streams de video, el tipo de carga útil indica el tipo de codificación (MPEG1, MPEG2, H.231, etc).

Tipo de carga útil	Codificación del Video
26	Motion JPEG
31	H.261
32	MPEG1

Número de Secuencia El campo reservado para el número de secuencia tiene 16 bits de largo. El número de secuencia se incrementa con cada paquete RTP enviado, y es utilizado por el receptor para detectar paquetes perdidos. Si el receptor recibe los paquetes con número de secuencia 86 y 89, sabe que los paquetes 88 y 89 se perdieron. Se puede implementar en el receptor algún algoritmo que regenere los datos perdidos para que su incidencia sea mínima (por ejemplo, agregar ruido blanco, o "interpolar"los datos faltantes.

Marca de Tiempo La marca de tiempo (32 bit) contiene el instante de creación de el primer byte del paquete RTP. El receptor utiliza esta marca para remover el jitter de los paquetes y para mantener el sincronismo de los distintos streams RTP.

Identificador de la fuente de sincronismo (SSRC) El campo SSRC tiene 32 bits de largo e identifica la fuente del stream RTP. Cada stream en una sesión RTP tiene un SSRC distinto. El SSRC no es la IP del transmisor, sino un numero aleatorio que el transmisor asigna a cada stream.

RTP Control Protocol (RTCP)

El protocolo RTP consta de en realidad de dos protocolos, el RTP y el RTCP. El RTCP se basa en la transmisión periódica de paquetes de control a todos los participantes de la sesión, utilizando el mismo mecanismo de transporte que los paquetes RTP. Utiliza un puerto distinto que RTP , por lo general se utilizan puertos consecutivos donde el par es asignado a

el flujo RTP y el impar al flujo RTCP.

Principales Funciones

Feedback RTCP es el encargado de la distribución de la información de realimentación de la calidad de los datos. Uno podría tomar acciones para mejorar el rendimiento de la aplicación con esta información, por ejemplo cambiar la codificación, para disminuir la tasa de bits.

Identificación Es necesario tener un identificador persistente a nivel de capa de transporte para las fuentes RTP. Este identificador se llama Nombre Canónico. Dado que el SSRC puede cambiar debido a un conflicto o debido a el reinicio del programa, los receptores utilizan el nombre canónico para mantener una lista de los participantes de la sesión. También se utiliza para agrupar distintos streams de un participante, por ejemplo para sincronizar audio con video.

Control de Participantes Las funciones anteriores obligan a los participantes de la sesión a enviar periódicamente paquetes RTCP, la tasa de envío de esos paquetes tiene que ser controlada para permitir acceso a mas participantes. Dado que los participantes envían los paquetes de control a todos los demás participantes, cada uno observa localmente el número de participantes y calcula a partir de este número la tasa de envió de paquetes de control.

Apéndice G

Java Media Framework (JMF)

G.1. Java Media Framework

Resumen Este anexo tiene como fin presentar la JMF (Java Media Framework). El mismo se basa en la documentación disponible, elaborada por los programadores de SUN [51]. Los temas que se presentan son los siguientes:

- Una presentación de los "Medios Basados en el Tiempo"(Time-Based Media)
- El funcionamiento de JMF para trabajar con estos tipos de medios
- Presentando medios basados en el tiempo con JMF
- Procesando medios basados en el tiempo con JMF
- Capturando datos de medios con JMF

Introducción Java Media Framework (JMF) es un API (Application Programming Interface), para la incorporación de medios basados en el tiempo (time-based medias) en aplicaciones Java. Los medios basados en el tiempo se caracterizan por cambiar su contenido con el transcurso del tiempo. Dentro de esta categoría encontramos: audio, video, secuencias MIDI y animaciones. Inicialmente, la JMF 1.0 habilitaba a los programadores para desarrollar aplicaciones que simplemente presentaban estos tipos de medios. Actualmente la JMF 2.0, extiende el área de trabajo para proveer soporte para captura y almacenaje de medios. Se puede controlar el tipo de procesamiento que es efectuado durante la reproducción mediante la utilización de plug-ins. Los objetivos principales del diseño de JMF 2.0 API son:

- Programación más amigable
- Soporte para la captura de medios.

- Habilita el desarrollo de aplicaciones para flujos multimedia y conferencias en Java
- Mantener la compatibilidad con JMF 1.0 API

Datos Basados En El Tiempo (Time-Based Media)

Toda información que tenga cambios significativos con respecto al tiempo puede ser catalogada como un medio basado en el tiempo, como lo son los clips de audio, secuencias MIDI, clips de video, etc. Estos medios pueden ser obtenidos de diversas fuentes, como archivos locales o remotos, cámaras, micrófonos y difusiones en vivo. A continuación se presenta un modelo que describe las características de estos medios y la manipulación que a estos se les debe aplicar.

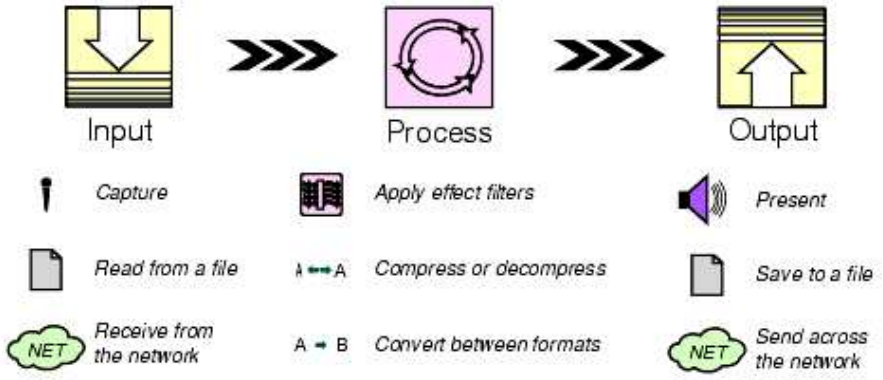


Figura G.1: Modelo de Procesamiento de Medios

Las características claves de los medios basados en el tiempo son las siguientes:

Flujo de Medios (Streaming Media) La característica principal de este medio es que requiere de un tiempo de entrega y de procesamiento, y por esto se debe controlar, ya que una vez iniciado el flujo de datos se deben satisfacer ciertos límites de tiempo.

Presentación de Medios (Output) La mayoría de estos medios pueden ser presentados a través de dispositivos de salida tales como parlantes y monitores, u otros destinos (ej.: a la red). Comúnmente a estos destinos de medios se le llaman DataSinks.

Procesamiento de Medios (Process) En muchas instancias la información contenida en un medio es manipulada antes de ser presentado al usuario, ya sea multiplexándola, filtrándola, comprimiéndola, o convirtiéndola en otro tipo de medio.

Captura de Medios (Input) Estos pueden ser capturados desde una fuente en vivo para procesarla y reproducirla o puede ser adquirida de un archivo de forma remota.

Entendiendo a JMF

JMF provee una arquitectura y un protocolo de mensajes unificado, para administrar la adquisición, procesamiento y entrega de medios basados en el tiempo. Para llevar a cabo su cometido, JMF cuenta con las siguientes características:

Arquitectura de alto nivel

Una parte integral de la JMF son las fuentes de datos y los reproductores para la administración de la captura, presentación y procesamiento de los medios, presentando la estructura de la figura 2. Algunos de los principales elementos (clases e interfaces) de JMF se presentan a continuación:

- Time model
- Managers
- Event Model
- Controls
- Data Model

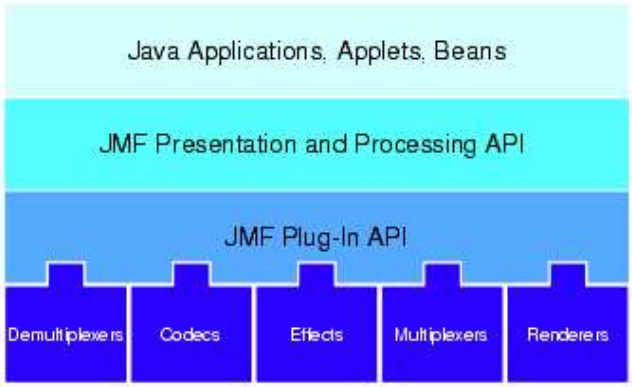


Figura G.2: Arquitectura de Alto Nivel

Presentación

El proceso de presentación es modelado por la interfaz controladora, que define el estado básico y el mecanismo de control para el objeto presentado o capturado. En este caso se definen 2 tipos de controladores: Reproductores y Procesadores.

Procesamiento

El procesador es un reproductor que toma una fuente de datos, realiza algunos procesos definidos por el usuario, y luego entrega la información de medio procesada. Las etapas por las que el medio atraviesa son la de demultiplexación, trascodificación, multiplexación y presentación.

Captura

Un dispositivo de captura multimedia puede actuar como una fuente de entrega de medios basados en el tiempo. Algunos dispositivos entregan múltiples flujos de datos que pueden ser separados mediante el procesamiento.

Almacenamiento y transmisión de medios

Un DataSink es usado para leer medios desde una fuente y presentarlo en algún destino. Particularmente el DataSink puede escribir datos a un archivo, escribir datos a través de la red o funcionar como un difusor RTP.

Presentando medios basados en el tiempo con JMF

Para presentar medios con JMF se debe usar un reproductor, la reproducción puede ser programada o se puede controlar desde el panel. Un Processor es un tipo especial de reproductor que provee control de cómo se procesan los datos antes de ser presentados. Un Processor puede ser creado usando un ProcessorModel, llamando al método Manager.createRealizedProcessor. El ProcessorModel define los requerimientos de entrada y salida para el Processor. Se puede crear indirectamente un reproductor usando el media Manager. Para mostrar el reproductor se deben agregar los componentes de objetos en su espacio de presentación de la applet o en la ventana de aplicación. Cuando se necesita crear un nuevo reproductor, éste se pide al Manager llamando a los métodos createplayer o createprocessor. Un reproductor generalmente tiene dos tipos de componentes en la interfaz, un componente visual y un panel de control, en JMF esto se puede ajustar a elección, cada componente debe ser agregado a la applet para que pueda ser usado, ya sea el control de volumen, de progreso, etc. También se puede ajustar la posición inicial, el contador de tiempo y otros componentes típicos de los reproductores. Un reproductor puede proveer información sobre sus parámetros actuales, incluyendo velocidad de reproducción, tiempo transcurrido y duración. Para

obtener la velocidad se llama a getRate quien retorna la velocidad como un número flotante, para obtener el tiempo se llama a getMediaTime la cual retorna un objeto Time, para obtener la duración del clip se llama a getDuration. Para responder a eventos durante la reproducción se debe implementar una interface llamada ControllerListener, la cual debe ser implementada en una clase y luego registrar esta clase llamando addControllerListener sobre el controlador del cual se quiere recibir los eventos. Para sincronizar la reproducción de múltiples flujos estos se asocian con una misma base de tiempo, para esto se usa $getTimeBase$ y $setTimeBase$, por ejemplo se puede ajustar player1 usando la base de tiempo de player2: $player1.setTimeBase(player2.getTimeBase())$;

Procesando medios basados en el tiempo con JMF

Un Processor puede ser usado como un reproductor programable que sirva para decodificar procesos, pero también puede ser usado para codificar y multiplexar los datos de medios capturados. Se puede controlar lo capturado por el Processor de diferentes maneras:

- Usar un ProcesorModel para construir un Processor que tenga ciertas características de entrada y salida
- Usar el método TrackControl.setFormat para especificar que formato de conversiones se realizan para cada pista
- Usar el método Processor.setOutputContentDescriptor para especificar el formato de multiplexación de los datos de salida
- Usar el método TrackControl.setCodecChain para seleccionar el efecto o codec plug-ins que es usado por el Processor
- Usar el método TrackControl.setRenderer para seleccionar el plug-in de render usado por el Processor

Configurando el Processor

Seleccionando las opciones de procesamiento de las pistas:

Para seleccionar que plug-ins son usados para cada pista en el flujo de medios se debe hacer lo siguiente:

- Llamar el método PlugInManager.getPlugInList para determinar que plug-ins están disponibles. El PlugInManager retorna una lista de plug-ins que tienen la especificación de los formatos de entrada y salida y el tipo de plug-ins.
- Llamar el método getTrackControls sobre el Processor para crear una instancia de la clase TrackControl para cada pista del flujo de medios

- Llamar los metodos TrackControl.setCodecChain o TrackControl.setRenderer para especificar el plug-in que se desea usar para cada pista. Cuando se usa el método setCodecChain para especificar el codec y plug-in de efecto para un Processor, el orden en que el plug-in aparece realmente en la cadena de procesamiento está determinado por los formatos de entrada y salida de cada plug-in cargado.

Convirtiendo datos de medios de un formato a otro

Se puede seleccionar el formato para una pista en particular a través del TrackControl:

- Llamar getTrackControls sobre el Processor para hacer un TrackControl para cada pista en el flujo de medios. El Processor debe estar en el estado Configurado antes de llamar getTrackControls.
- Usar el método TrackControl.setFormat para especificar el formato a que se quiere convertir la pista seleccionada.

Especificando el formato de salida

Se puede usar el método Processor.setContentDescriptor para especificar el formato de salida de los datos para el Processor. Se puede tener una lista de los formatos soportados llamando getSupportedContentDescriptors. Se puede también seleccionar el formato de salida usando un ProcessorModel para crear el Processor.

Especificando el destino

Se puede especificar un destino para el flujo de medios seleccionando una presentación particular para una pista a través de la clase TrackControl o usando la salida del Processor como entrada a un DataSink particular. También se puede usar la salida del Processor como entrada a otro reproductor o a un Processor con distinto destino.

Seleccionando una presentación

Para seleccionar la presentación que se desea usar se debe hacer lo siguiente: Llamar getTrackControls sobre el Processor para hacer un TrackControl para cada pista del flujo de medios. El Processor debe estar en estado Configurado antes de llamar getTrackControls. Llamar el método TrackControl setRenderer para especificar el plugin de renderizado.

Escribiendo datos de medios en un archivo

Se puede usar un DataSink para leer datos desde el objeto de salida del Processor DataSource y enviar los datos a un archivo. Se deben seguir los siguientes pasos para realizarlo:

- Recibir la salida DataSource desde el Processor llamando getDataOutput
- Construir un DataSink llamando Manager.createDataSink. Ubicarlo en la salida del DataSource y un MediaLocator que especifica la ubicación del archivo al que se quiere escribir
- Llamar open sobre el DataSink para abrir el archivo
- Llamar start sobre el DataSink para comenzar a escribir los datos. El formato del archivo escrito es controlado a través del Processor

Capturando datos de medios con JMF

Se puede usar JMF para capturar datos de medios desde un dispositivo de captura como un micrófono o una cámara de video. Los archivos capturados pueden ser procesados y almacenados para un uso futuro. Para capturar datos de medios se debe hacer lo siguiente:

- Localizar el dispositivo de captura que se quiera usar consultando el CaptureDeviceManager
- Hacer un objeto CaptureDeviceInfo para el dispositivo
- Hacer un MediaLocator desde el objeto CaptureDeviceInfo y usar éste para crear un DataSource
- Crear un reproductor o Processor usando el DataSource
- Iniciar el reproductor o Processor para empezar el proceso de captura

Accediendo a dispositivos de captura

Para acceder a dispositivos de captura se debe hacer a través del CaptureDeviceManager. El CaptureDeviceManager es un registro central para todos los dispositivos de captura disponibles para la JMF. Es posible hacer una lista con los dispositivos disponibles llamando al método CaptureDeviceManager.getDeviceList.

Almacenando datos de medios

Para almacenar los datos capturados en un archivo, se necesita usar un Processor en lugar de un reproductor. Se usa un DataSink para leer los datos desde la salida del Processor y enviarlos al archivo.

Apéndice H

IPPM Group

H.1. Parámetros objetivos de QoS

H.1.1. Introducción

El IPPM (IP Performance Metrics) es uno de los grupos de trabajo del IETF (Internet Engineering Task Force) cuya tarea se concentra en el estudio de la performance de Internet. El objetivo de este grupo es alcanzar un escenario en el cual tanto proveedores como usuarios de servicios de transporte en Internet tengan un común entendimiento de la performance de la red (redes) que proveen/utilizan, teniendo a su vez nociones claras de la credibilidad de dicha performance. Para alcanzar dicho objetivo es necesario definir métricas (parámetros de red) que cumplan con los siguientes principios:

- Toda métrica debe ser concreta y bien definida.
- Toda métrica debe evitar situaciones de *performance inducida* (engaño)
- Toda métrica debe permitir un claro entendimiento de la performance experimentada o brindada por los distintos actores de Internet.

En este sentido se han definido más de 30 métricas y metodologías de medida, todas ellas clasificadas dentro de las siguientes categorías (se especifica en cada caso la recomendación correspondiente):

- Conectividad (RFC 2678)
- Retardo en un sentido (RFC 2679)
- Pérdida de paquetes en un sentido (RFC 2680)
- Retardo de ida y vuelta (RFC 2681)

- Patrón de pérdidas en un sentido (RFC 3357)
- Variación del retardo (RFC 3393)
- Mediciones con flujos periódicos (RFC 3432)

En esta sección se resumen las distintas recomendaciones (RFC) brindadas por el IPPM hasta el momento. Se comenzará con un repaso de la recomendación 2330 donde se especifican claramente los términos que se utilizarán en la definición y análisis de las métricas estudiadas. Por último es importante aclarar que en esta sección se utilizarán indistintamente los conceptos de métrica y parámetro de red; cuando hablamos de métrica hacemos referencia a toda cantidad cuidadosamente especificada cuyo valor es relevante en el estudio del desempeño de una red. En este entendido es posible determinar la performance de un red en base a un conjunto de dichos parámetros o métricas.

H.1.2. Marco de Trabajo del IPPM (RFC 2330)

El RFC 2330 tiene como objetivos la definición de un marco de trabajo general para el desarrollo de las distintas métricas de red. Este marco especifica detalles de terminología, unidades de medida, metodologías de medida, etc.. que se utilizarán en el conjunto de recomendaciones mencionadas.

Terminología

La siguiente lista define en forma precisa las nociones de 'host', 'router', 'enlace', 'camino', 'nube IP' y 'conmutador' necesarias para un correcto desarrollo de las métricas para caminos:

- *Host*: computadora capaz de comunicarse utilizando los protocolos de Internet.
- *Router*: host que permite la comunicación a nivel de capa de red entre otros hosts mediante el envió de paquetes IP.
- *Enlace*: conexión entre dos o mas hosts a nivel de capa de enlace.
- *Camino*: secuencia de la forma $(H_0, L_1, H_1, L_2, ..., L_n, H_n)$, donde H_i indica el iésimo host del camino, L_i es el enlace entre H_{i-1} y H_i y $H_1..H_{n-1}$ son routers. Cada par (L_i, H_i) es definido como un salto (hop). Observar que camino es un concepto unidireccional.
- *Nube*: grafo unidireccional (cíclico o no) cuyos vértices son routers y sus arcos son enlaces que conectan pares de routers. Formalmente, ethernets y otros tipos de enlaces que conectan mas de dos routers son modelados como vértices del grafo. El término

"conectarse a una nube" se debe interpretar como conectarse a algún router de la misma a través de un enlace ajeno a la nube.

- *Conmutador*: enlace que conecta un host con una nube o nubes entre si.

Unidades de medida

El RFC 2330 define las unidades de medida que el grupo utilizará en las distintas métricas. Especifica que el sistema de unidades a utilizar será el sistema Universal de medida, teniendo en cuenta las siguientes observaciones:

- Solo son aceptadas unidades basadas en miles de las unidades aceptadas (Km, mm, μs son unidades aceptadas, mientras que cm no lo es).
- Cuando una unidad es expresada como combinación de unidades aceptadas, las unidades basadas en miles van al principio (es válido Km/s pero no m/ms).
- La unidad de información es el bit.
- Los prefijos asociados a bits tienen significado decimal y no el convencional asociado a recursos de memoria.
- Cuando una métrica se define en términos de otras se la denomina 'métrica derivada'.

Metodologías de medida

Para un conjunto de métricas definidas pueden existir distintas metodologías de medida. Una lista parcial incluye:

- Medida directa mediante el uso de tráfico de test inyectado. Ejemplo: uso de un paquete de eco request para medir el RTT.
- Proyección de una métrica a partir de medidas de nivel inferior. Ejemplo: dadas las medidas de retardo y ancho de banda para cada salto de un camino, es posible proyectar el retardo total para el camino para un paquete de tamaño conocido.
- Estimación de una métrica a partir de un conjunto de medidas relacionadas. Ejemplo: dada una serie de medidas del retardo de un camino de un solo salto para distintos tamaños de paquete, es posible estimar el retardo de propagación para el enlace de dicho salto.

- Estimación de una métrica en cierto tiempo a partir de un conjunto de métricas determinadas en otros tiempos. Ejemplo: dada una serie de medidas de capacidad de un enlace en un tiempo pasado, junto con una serie de medidas de retardo para ese tiempo pasado y para el tiempo actual, y dado un modelo dinámico para dicha capacidad, es posible estimar la capacidad observada en el tiempo actual.

Toda metodología de medida debe presentar la propiedad de continuidad: para pequeños cambios en las condiciones las medidas tienen pequeñas alteraciones. Una métrica que tiene al menos una metodología asociada que presenta continuidad es denominada como continua.

Nótese que algunas métricas toman valores discretos por definición (como por ejemplo el numero de saltos en un camino), por lo que no pueden ser continuas en el sentido antes descrito.

Por último, es importante mencionar que algunas metodologías de medida pueden ser 'no invasivas' en el sentido de que el algoritmo implementado no modifica, o a lo sumo mínimamente, el valor de la métrica de performance a medir. Ejemplo: en una red WAN de alta velocidad y en condiciones de baja carga, el uso de paquetes 'ping' para determinar el retardo de ida y vuelta no afecta la medida (siempre y cuando se utilice una pequeña cantidad de estos).

Errores e incertidumbres

Aún la mejor metodología de medida aplicada sobre la métrica más estable y perfectamente determinada puede presentar errores o incertidumbres. Por lo tanto, a la hora de definir una nueva metodología de medida es fundamental conocer, cuantificar y tomar en cuenta todas aquellas posibles fuentes de error e incertidumbre involucradas. Por ejemplo, al desarrollar un método de medida de retardos que utiliza una computadora es imprescindible distinguir el retardo que esta introduce del retardo que efectivamente se quiere medir. Observar que una buena forma de evitar este retardo extra es trabajar con herramientas de bajo nivel ('Kernel-level') en contraposición a aquellas de nivel de aplicación.

Métricas empíricas

En algunos casos podemos encontrarnos con métricas cuya especificación no brinde suficiente información como para clasificarla como una métrica analítica. En dicho caso la métrica es definido como una "métrica empírica", siendo estrictamente necesario contar con un método de medida claramente especificado para su determinación.

Relojes: características y temas relacionados

El RFC 1305 (Network Time Protocol) especifica las características del instrumento utilizado en muchas de las métricas antes definidas: el reloj.

- Offset o corrimiento: se define el offset de un reloj en un determinado tiempo como la diferencia entre el tiempo entregado por dicho reloj y el tiempo "real".
- Exactitud: define que tan cercano a cero es el valor absoluto del offset en determinado momento.
- Inclinación (skew): es la diferencia en frecuencia (derivada primera del offset respecto del tiempo real) entre el tiempo del reloj y el tiempo real.
- Variación en el skew (drift): es la derivada segunda del offset respecto del tiempo real.
- Resolución: es la unidad de tiempo más pequeña por la cual es modificado el tiempo del reloj. Brinda una cota inferior a la incertidumbre de una medida.

Dado que en muchas de la medidas relacionadas con el tiempo se utiliza más de un reloj, se incluye terminología para la comparación de tiempos entregados por relojes distintos. En general, dados dos relojes C_1 y C_2 basta con sustituir la noción de tiempo real por el tiempo entregado por uno de los dos relojes, supongamos C_1, para que las definiciones previas sean relativas al reloj C_1. Por ejemplo, el offset del reloj C_2 relativo al reloj C_1 en un determinado instante es $T_{C2} - T_{C1}$ (se lo denomina offset relativo).

La resolución de cualquier medida realizada con más de un reloj pasa a ser la suma de las resoluciones de los relojes utilizados. Este término pasa a ser importante, por ejemplo en aquellos casos en los que se utilizan diferencias de estampas de tiempo para realizar cierta medida.

Dos relojes se dicen sincronizados si los offsets relativos de uno con respecto al otro valen cero. Nótese que dos relojes pueden ser muy inexactos pero al mismo tiempo estar sincronizados, lo cual es importante ya que en varias medidas de performance la sincronización es más importante que la exactitud (por ejemplo, en la determinación del retardo en un sentido).

Toda metodología que involucre dos relojes usualmente trata de asegurar previo a la medida la sincronzación de los mismos. Una forma efectiva de lograrlo es obtener la noción de tiempo de una fuente externa a los hosts involucrados. Para ello es posible utilizar el NTP (Network Time Protocol) que permite la sincronización de dos relojes mediante el intercambio de paquetes, aunque claramente esto depende de algunas características de la red (particularmente retardo) las cuales pueden ser exactamente las que se quieran medir. Una forma más exacta para lograr la sincronización de relojes es el uso de un sistema de posicionamiento global o GPS en cada host.

Tiempo de Cable (Wire Time)

Las mediciones de performance de punta a punta en Internet son modificadas por los hosts involucrados, introduciendo retardos o cuellos de botella adicionales que nada tienen que ver con la red en si (si pensamos en la red como los enlaces y routers intermedios). Habrá que distinguir entonces aquellos casos en los que se deba tener en cuenta o no estas modificaciones; por ejemplo, si la métrica a determinar involucra solo aspectos de la red se deberá trabajar lo más cercano posible a la capa de red para eliminar los efectos introducidos por las capas superiores. Por el contrario, si lo relevante incluye aspectos de capa de aplicación (por ejemplo, retardo en una conversación de VIO) todos los efectos se tomarán en cuenta.

De esta discusión surge la necesidad de desacoplar las características de la red de las de los hosts. Para ello se introduce la noción de "tiempo de cable", que solo tomará en cuenta al host H observando el enlace E en cierto punto :

- Para un paquete P dado se define el "tiempo de cable de llegada" de P a H en E como el primer tiempo T en el cual algún bit de P fue observado por H en cierto punto de E.

- Para un paquete P dado se define el "tiempo de cable de salida" de P desde H en E como el primer tiempo T en el cual todos los bits de P fueron observados por H en cierto punto de E.

Cuando corresponda, las métricas deberán ser definidas en términos de tiempos de cable y no en tiempos de host, de manera de remarcar la separación entre retardos debidos al host y retardo de la red.

Tres tipos de métrica: Medida , Muestra y Estadística

- **Medida:** métrica que en cierto sentido es atómica. Por ejemplo, una única instancia del TCP throughput de un host a otro puede ser definida como una medida (aunque la misma conste de la medida de varios paquetes).

- **Muestra:** métrica derivada de un conjunto de medidas individuales. Por ejemplo, una muestra del retardo de ida entre dos hosts puede ser definida como el conjunto de medidas individuales de retardo de ida tomadas a lo largo de una hora y a intervalos separados exponencialmente con una media de 1 segundo.

- **Estadística:** métrica derivada de una muestra mediante el cálculo de cierto valor estadístico sobre las medidas individuales que conforman la misma. Por ejemplo, el valor promedio (la *media*) del retardo de ida del ejemplo anterior.

Métodos de recolección de Muestras

Al momento de elegir un método para tomar muestras es importante tener en cuenta que en algunos casos el mismo puede alterar las características del parámetro a medir haciendo que no refleje lo que realmente sucede. Diremos que una muestra es *no desviada* si el método de recolección no altera sus características. A continuación se presentan distintos métodos de recolección de muestras:

Una forma muy común de tomar muestras es la del muestreo periódico, realizando cada medida a intervalos de tiempo fijos y ctes. El muestreo periódico presenta algunos problemas importantes:

- Si la métrica a medir tiene un comportamiento periódico de período igual a algún múltiplo del período de muestreo, es altamente probable que la muestra tomada solo refleje parte del comportamiento real.
- El muestreo periódico es fácil de predecir, siendo muy susceptible a mecanismos que modifiquen la métrica de manera tal que revele un comportamiento inducido.
- El método en si puede perturbar la métrica a determinar. Por ejemplo, si el método de medida es un método activo (inyecta tráfico) puede llegar a inducir en la red un estado de sincronzación que magnifique efectos a priori despreciables.

Otro forma de recolectar muestras es mediante un muestreo aleatorio, donde las medidas se realizan a intervalos de tiempo independientes y aleatorios con una cierta distribución de probabilidad asociada $G(t)$. Las ventajas respecto del muestreo periódico son significativas; en general, evita los efectos de sincronización mencionados y conduce a muestras no desviadas. Como contrapartida, el muestreo aleatorio presenta ciertas debilidades:

- Dificulta el análisias en el dominio de frecuencia.
- Como se discutirá mas adelante, a menos que la distribución $G(t)$ sea exponencial el muestreo continua siendo un tanto predecible.

Como ejemplo de muestreo aleatorio veremos el muestreo de Poisson. Es posible demostrar que si $G(t)$ es una distribución exponencial de media λ :

$$G(t) = 1 - e^{-\lambda t}$$

entonces el arribo de nuevas muestras individuales no puede ser predecido y la muestra no se desvía. Es más, la muestra no se desvía asintóticamente aún si el efecto del muestreo afecta

el estado de la red. Dicho proceso de muestreo es conocido como un muestreo Poisson.

Dado que la distribución exponencial no está acotada puede llegar a generar intervalos de muestreo grandes indeseables en algunos casos. Para acelerar la convergencia de la estimación derivada del muestreo puede ser conveniente acotar el máximo intervalo de muestreo a cierto tiempo dT, utilizando por ejemplo una distribución uniforme:

$$G(t) = Unif[0, dT]$$

Claramente este tipo de muestreo se torna altamente predecible si un intervalo de tiempo cercano a dT a pasado sin haberse realizado el muestreo.

El muestreo Poisson se realiza generando intervalos de tiempo independientes y exponencialmente distribuídos, tomando una muestra individual al final de cada intervalo. Se puede probar fácilmente que si en un tiempo T se realiza un muestreo Poisson durante un tiempo dT en el cual se realizan N mediciones, entonces las muestras individuales obtenidas estarán uniformemente distribuídas en el intervalo $[T, T+dT]$. Por lo tanto, otra forma de implementar un muestreo Poisson es tomar N y dT y generar N tiempos de muestreo uniformemente distribuídos en $[T, T + dT]$.

Distribuciones Estadísticas

Una forma de describir una muestra es mediante una distribución de probabilidad (informalmente, la noción de percentiles). La función empírica de distribución (EDF) para un conjunto de medidas es una función $F(x)$ que asigna para cualquier x la proporción del total de medidas que fueron $\leq$ x. El uso de histogramas es otra forma de resumir la información de una muestra; en ambos casos se pierde la información del orden de ocurrencia de cada medida.

El término *percentil* refiere al menor valor de x para el cual $F(x) \geq$ cierto porcentaje.

Para terminar, se definirá el término mediana de una distribución. La misma se define como el pto. X para el cual la probabilidad de observar un valor $\leq$ X es igual a la probabilidad de observar un valor $>$ X. Para una muestra (ordenada en orden ascendente), la mediana será el percentil 50, si el número de medidas es impar, o el promedio de las medidas centrales si es par.

Paquetes de tipo P

Una propiedad fundamental de varias métricas en Internet es que el valor de la misma depende del tipo de paquete IP utilizado. Por ejemplo, consideremos la métrica de conectividad IP: los resultados dependerán de si el interés está en conectividad para paquetes destinados

a puertos TCP bien conocidos o puertos UDP no reservados, o aquellos con IP checksum inválido, o aquellos con TTL igual a 16, etc. En el caso de firewalls o RSVP esta distinción es fundamental.

Esta distinción lleva a la definición de un paquete genérico definido como un *paquete de tipo P*, donde P será definido explícitamente (exactamente que tipo de paquete es), parcialmente (por ejemplo, paquete con un payload de 8 bytes) o simplemente como un paquete genérico. Siempre que el valor de una métrica dependa del tipo de paquete involucrado se especificará en la misma el tipo de paquete o se incluirá la frase "tipo-P"; por ejemplo, "conectividad-IP-tipo-P".

H.1.3. Conectividad (RFC 2678)

Introducción

En esta sección se describen las distintas métricas relacionadas con el concepto de *conectividad*. Si bien es un elemento que normalmente se asume, es claro que el primer paso para determinar el desempeño de una red es determinar claramente si los distintos puntos de la misma son alcanzables.

Conectividad instantánea en un sentido

Nombre de la métrica

Conectividad-unidireccional-instantánea-tipo-P

Parámetros de la métrica

- 1. Src: dirección IP de un host.
- 2. Dst: dirección IP de un host.
- 3. T: cierto tiempo.

Unidades de la métrica

Boolean.

Definición

Src tiene *conectividad-unidireccional-instantánea-tipo-P* con Dst en T si un paquete de tipo P, transmitido desde Src a Dst en T, arriva a Dst.

Uno puede pensar que la conectividad unidireccional es difícil de medir en ausencia de conectividad en la dirección de retorno. Sin embargo, existe la posibilidad de que cierto proceso en el host Dst note la llegada de paquetes desde Src y lo reporte a través de un canal externo o más tarde cuando Dst tenga conectividad con Src.

Conectividad instantánea en dos sentidos

Nombre de la métrica

Conectividad-bidireccional-instantánea-tipo-P

Parámetros de la métrica

- 1. A1: dirección IP de un host.
- 2. A2: dirección IP de un host.
- 3. T: cierto tiempo.

Unidades de la métrica

Boolean.

Definición

Las direcciones A_1 y A_2 tienen conectividad-bidireccional-instantánea-tipo-P en el tiempo T si A_1 tiene conectividad-unidireccional-instantánea-tipo-P con A_2 y viceversa.

Una definición alternativa sería que A_1 y A_2 están conectados bidireccionalmente en T si A_1 tiene conectividad unidireccional instantánea con A_2 en T y A_2 tiene conectividad unidireccional instantánea con A_1 en T+dT, donde T+dT es el tiempo en el cual el paquete enviado desde A_1 llega a A_2. Esta definición es mas útil a la hora de realizar medidas porque utiliza una respuesta de A_2 para determinar la conexión bidireccional, aunque la misma rompe la simetría entre A_1 y A_2 y requiere noción del tiempo de un paquete entre ambos hosts.

Conectividad en un sentido

Nombre de la métrica

Conectividad-unidireccional-tipo-P

Parámetros de la métrica

- 1. Src: dirección IP de un host.
- 2. Dst: dirección IP de un host.
- 3. T: cierto tiempo.
- 4. dT: duración de tiempo.
- 5. [T,T+dT]: intervalo de tiempo.

Unidades de la métrica

Boolean.

Definición

Src tiene conectividad-unidireccional-tipo-P con Dst durante el intervalo [T,T+dT] si para algún T' perteneciente al intervalo tiene conectividad unidireccional instantánea tipo P con Dst.

Conectividad en dos sentidos

Nombre de la métrica

Conectividad-bidireccional-tipo-P

Parámetros de la métrica

- 1. A1: dirección IP de un host.
- 2. A2: dirección IP de otro host.
- 3. T: cierto tiempo.
- 4. dT: duración de tiempo.
- 5. [T,T+dT]: intervalo de tiempo.

Unidades de la métrica

Boolean.

Definición

A1 y A2 tienen conectividad-bidireccional-tipo-P durante el intervalo de tiempo [T,T+dT] si A1 tiene conectividad-unidireccional-tipo-P con A2 durante el intervalo y A2 tiene conectividad-unidireccional-tipo-P con A1 durante el intervalo.

Discusión

Esta métrica no es precisamente lo que uno necesitaría para definir de forma general y útil la conectividad entre dos host, ya que la misma requiere que el envío de un paquete de A1 a A2 tenga su respectiva respuesta de A2 a A1. Con ésta definición, podría darse el caso de que A1 y A2 tuvieran conectividad solo en un tiempo T1 suficientemente pequeño tal que no pudieran responder a la llegada del paquete enviado por el otro. Esta deficiencia motiva la siguiente definición.

Conectividad temporal en dos sentidos

Nombre de la métrica

Conectividad-temporal-tipo-P1-P2

Parámetros de la métrica

- 1. A1: dirección IP de un host.
- 2. A2: dirección IP de otro host.
- 3. T: cierto tiempo.
- 4. dT: duración de tiempo.
- 5. [T,T+dT]: intervalo de tiempo.

Unidades de la métrica

Boolean.

Definición

A1 tiene conectividad-temporal-tipo-P1-P2 con A2 durante el intervalo [T,T+dT] si existen tiempos T1 y T2 e intervalos dT1 y dT2 que verifican:

- T1, T1+dT1, T2, T2+dT2 pertenecen todos al intervalo [T, T+dT].
- T1+dT1 $\leq$ T2.
- En el tiempo T1, A1 tiene conectividad-unidireccional-instantánea-tipo-P1 con A2.
- En el tiempo T2, A2 tiene conectividad-unidireccional-instantánea-tipo-P2 con A1.
- dT1 es el tiempo que le lleva al paquete de tipo P1, enviado por A1 en T1, en llegar a A2.
- dT2 es el tiempo que le lleva al paquete de tipo P2, enviado por A2 en T2, en llegar a A1.

Discusión

Esta métrica define de manera útil y general el concepto de conectividad entre dos hosts. A1 puede enviar un paquete a A2 y recibir respuesta de éste. En éstadefinición se contemple el hecho de que muchas aplicaciones utilizan diferentes tipos de paquete para el tráfico de ida y el de vuelta.

Metodologías

A continuación se presenta una clase de metodologías para estimar la conectividad temporal tipo P1 P2. El término "clase" se basa en que la metodología dependerá del tipo particular de paquete P1 y P2.

Entradas:

- Tipos de paquete P1 y P2, direcciones A1 y A2, intervalo [T,T+dT].
- N, el número de paquetes a enviar como prueba de la conectividad.
- W, el tiempo máximo de espera de respuesta.
- Se requiere que W $\leq$ 255 y que dT $>$ W.
- Valores recomendados:
  - dT=60 segundos.
  - W=10 segundos.
  - N=20 paquetes.

Algoritmo

- Calcular N tiempos de envío uniformemente distribuídos en el intervalo [T,T+dT-W].
- Enviar en cada uno de éstos tiempos un paquete de tipo P1 de A1 a A2.
- Inspeccionar el tráfico que arriba a A1 en busca de una respuesta de A2.Dischа inspección dependerá de los tipos de paquete P1 y P2. Si se recibe una respuesta exitosa, el valo de la étrica es verdadero. En éste punto, la medición termina.
- Si no se recibe respuesta exitosa al tiempo T+dT, el valor de la métrica será falso.

Discusión

El algoritmo es inexacto porque no puede probar conectividad temporal en todo instante en [T,T+dT]. El valor de N marca un equilibrio entre precisión de la medida y sobrecarga de la red. El estado del arte en la investigación de Internet no muestra una forma sólida de tomar N.

H.1.4. Retardo en un sentido (RFC 2679)

Introducción

En esta sección se escribe una de las métricas más representativas y aceptadas en el estudio del desempeño de una red, el *retardo temporal*. El retardo entre un host origen y un host destino resulta útil por varias razones:

- Muchas aplicaciones no se desmpeñan bien (o directamente no funcionan) si el retardo de punta a punta entre hosts supera cierto umbral, particular de dicha aplicación.
- Los cambios erráticos en el retardo hacen difícil o imposible la implementación de muchas aplicaciones de tiempo real.
- Cuanto más grande sea el valor del retardo, más difícil será para los protocolos de capa de transporte mantener altos anchos de banda.
- El valor mínimo del retardo indica una estimación del retardo debido únicamente a propagación y transmisión, siendo también una medida del retardo sufrido en condiciones de baja carga.

- Valores de retardo superiores al mínimo brindan una estimación de la congestión del camino.

La medida del retardo en un solo sentido en lugar del retardo de ida y vuelta presenta las siguientes ventajas:

- En la Internet de hoy, el camino de una fuente a un destino es en gral. diferente del camino de regreso del destino al origen ("caminos antisimétricos"), por lo que diferente secuencia de routers han de usarse en cada dirección. Por lo tanto el retardo de ida y vuelta mide en realidad la performance conjunta de dos caminos. Realizar la medida por separado deja de manifiesto la posible diferencia de performance entre ambos caminos, brindando mayor conocimiento de la situación actual.

- Aun teniendo caminos idénticos de ida y vuelta, pueden haber diferencias radicales de retardo debido a la asimetría en las colas de los routers atravesados.

- La performance de cierta aplicación puede depender básicamente de la performance en una sola dirección. Por ejemplo, la performance de FTP sobre TCP depende de la performance del camino en el sentido del flujo de datos y no del camino de regreso de los acuses de recibo.

- En redes con QoS (calidad de servicio), el aprovisionamiento de recursos en un sentido puede ser radicalmente distinta de la dirección reversa, resultando en garantías diferentes de QoS.

Retardo en un sentido

Nombre de la métrica

Retardo-unidireccional-tipo-P

Parámetros de la métrica

- 1. Src: dirección IP de un host.
- 2. Dst: dirección IP de un host.
- 3. T: cierto tiempo.

Unidades de la métrica

Número real o número indefinido (infinito informalmente) de segundos.

Definición

Para un número real dT, decimos que el retardo-unidireccional-tipo-P de Src a Dst en T es dT si Src envió a Dst el primer bit del paquete tipo P en tiempo de cable T y Dst recibió el último bit de dicho paquete en tiempo de cable T+dT. Decimos que el retardo-unidirecconal-tipo-P es indefinido si Src envió el primer bit del paquete tipo P en tiempo de cable T y Dst no recibió dicho paquete.

Discusión

- Todo valor real de retardo será un número positivo, por lo que en principio no tendrá sentido reportar valores negativos de retardo. Sin embargo, un retardo individual que sea negativo o cero como parte de un stream puede ser útil al tratar de descubrir la distribución de los retardos.

- En general los retardos serán pequeños (entre 100 μs y algunos ms), por lo que será de gran importancia una alta sincronización entre fuente y destino. El uso del NTP para proporcionar la sincronización será de poca eficacia si se trata de retardos pequeños, ya que el método está directamente relacionado con el retardo de la red y de los agentes NTP en sí. Por lo tanto se recomienda el uso de sistemas GPS para establecer la sincronización.

- Toda metodología de medida del retardo deberá incluír una forma de diferenciar un valor de retardo infinito (se perdió el paquete) de uno que es simplemente muy largo (pero que e paquete si llegará a destino). La introducción de cotas superiores al retardo deberán estra acompañadas de un buen entendimiento del tiempo de vida de un paquete (por ejemplo, Paxson impone teóricamente 255 s como cota superior para el tiempo de vida de un paquete IP).

- Si el paquete es duplicado en el camino de fuente a destino, el retardo será determinado por la primera aparición del paquete.

- Si el paquete es fragmentado en el camino y por alguna razón no es reconstruído, el paquete será considerado como perdido y el retardo en consecuencia será indefinido (infinito).

Metodologías de medida

Como ya lo hemos mencionado, lametodología dependerá del tipo de paquete involucrado (número de protocolo, número de puerto TCP/UDP, tamaño del paquete, etc..). A continuación se describe una metodología general:

- Asegurarse de que Src y Dst están mutuamente sincronizados y lo mejor posible al tiempo real actual.

- En el host Src, tomar las direcciones IP de Src y Dst y formar un paquete tipo-P de test con dichas direcciones. Cualquier información extra que se deba agregar al paquete para llegar a un cierto tamaño deberá ser generada de manera aleatoria para evitar posibles modificaciones en el retardo por las´técnicas de compresión que puedan ocurrir en el camino.

- Asegurar que el host destino esté en condiciones de recibir el paquete de test.

- En el origen, poner un tiemstamp en el paquete y enviarlo al destino.

- Si el paquete arriba en un tiempo menor a cierta cota establecida, tomar una nueva timestamp lo antes posible. Substrayendo ambas marcas de tiempo se tendrá una estimación del retardo de Src a Dst. Si se conocen los tiempos entre el marcado del tiempo del paquete y su envío/recibo se deberán descontar para corregir la medida.

- Si el paquete no arriba en el tiempo estipulado, el retardo de Src a Dst será indefinido.

Muestra de retardo en un sentido

Dada la medida del retardo en un sentido antes descrita, pasaremos a definir una muestra de dichas medidas. La idea de la muestra será seleccionar un conjunto de parámetros (Src, Dst, tipo de paquete de test) y realizar un conjunto de medidas entre dos tiempos Ti y Tf a una tasa λ. Una posible implementación es seleccionar un proceso Poisson de tasa λ cuyos valores caigan entre Ti y Tf y realizar la medida en cada uno de esos tiempos (el intervalo entre medidas será de $1/\lambda$ segundos).

Nombre de la métrica

Muestra-Poisson-de-retardo-unidireccional-tipo-P

Parámetros de la métrica

- 1. Src: dirección IP de un host.
- 2. Dst: dirección IP de un host.
- 3. Ti: cierto tiempo.
- 4. Tf: cierto tiempo.

- 5. λ: tasa de eventos en recíproco de segundos.

Unidades de la métrica

Una secuencia de pares [T,dT], donde T es el tiempo en el que se realiza la medida y dT es el retardo medido

Definición

Dados Ti,Tf y λ, calculamos un proceso seudo-randómico de Poisson que comience en Ti, finalice en Tf y tenga una tasa promedio de arribos igual a λ. En cada tiempo de éste proceso obtenemos el valor del retardo-unidireccional-tipo-P, formando así la secuencia de parejas tiempo-retardo.

Discusión

Los valores de λ merecen un cuidado especial: se debe notar que un valor grande de lambda puede perturbar la red por introducir demasiados paquetes de test, mientras que un valor de λ pequeño puede no llegar a capturar la información buscada. La muestra es definida en términos de un proceso Poisson para evitar al máximo los posibles efectos de sincronización que un muestreo periódico podría llegar a introducir. El proceso solo marca los tiempos en los cuales se realiza la medida, por lo que en gral. los tiempos de llegada de los paquetes de test al destino no tendrán una distribución Poisson (debido a los efectos introducidos por la red). Por último, es importante destacar el posible problema del arrivo de paquetes en desorden que tendrá que tenerse en cuenta al momento de armar las parejas de la secuencia.

Algunas estadísticas sobre el retardo unidireccional

- **Percentiles**
  Al realizar los cálculos de percentiles los valores indefinidos deberán ser tratados como muy grandes. De ésta forma, un percentil puede llegar a ser indefinido (infinito).

- **Mediana**
  También en el caso de la mediana los valores indefinidos son tratados como muy grandes. La mediana difiere del percentil del %50 solo cuando el número de parejas de la secuencia es par.

- **Retardo mínimo/máximo**

Consideraciones de seguridad

Realizar medidas en Internet trae consigo el tomar consideraciones de seguridad y privacidad.

En lo que respecta a seguridad son dos los aspectos a destacar: daño potencial causado por la medida y sobre la medida. Las medidas pueden causar daño por ser de carácter activo. En situaciones extremas, el tráfico inyectado puede llegar a causar situaciones de congestión y en consecuencia DOS (denial of service).

Las medidas mismas pueden llegar a ser dañadas o alteradas por dos elementos fundamentales: routers que distingan por tipo de tráfico y den a los paquetes de test prioridades determinadas, y atacantes que inyecten tráfico de medida artificial. Si un router puede reconocer un paquete de test y lo trata separadamente, entonces la medida no refleja el tráfico real actual. Al mismo tiempo, si un atacante inyecta tráfico de test artificial y luego es tomado como legítimo, la tasa de pérdida puede ser disminuída artificialmente. Por lo tanto, las técnicas de medida deben hacer que el tráfico de test sea lo más parecido al tráfico "normal" y de alguna forma irreproducible (utilizando, por ejemplo, firmas digitales).

En lo que a privacidad refiere, las medidas de carácter activo no revelan información de usuario que si puede llegar a darse en el caso de las medias pasivas.

H.1.5. Pérdida en un sentido (RFC 2680)

Introducción

Al igual que en el caso del retardo, las pérdidas entre un host origen y un host destino representan uno de los parámetros de calidad mas utilizado y representativo del estado de una red. El conocimiento de las pérdidas resulta útil por varias razones:

- Algunas aplicaciones no pueden ser implementadas correctamente si las pérdidas de punta a punta superan cierto umbral.
- En casos de pérdida excesiva de paquetes se hace extremadamente complicado implementar aplicaciones de tiempo real. El término "excesivo" dependerá del tipo particular de aplicación.
- A mayor pérdida de paquetes, más difícil se hace para la capa de transporte el soportar altos anchos de banda
- La sensibilidad de las aplicaciones de tiempo real y de los protocolos de capa de transporte se ve muy afectada cuando grandes productos de retardo-ancho de banda han de mantenerse.

La medida de las pérdidas en un solo sentido en lugar de las pérdidas de ida y vuelta presenta las siguientes ventajas:

- En la Internet de hoy, el camino de una fuente a un destino es en general diferente del camino de regreso del destino al origen ("caminos antisimétricos"), por lo que diferente secuencia de routers han de usarse en cada dirección. Por lo tanto la medida de las pérdidas de ida y vuelta mide en realidad la performance conjunta de dos caminos. Realizar la medida por separado deja de manifiesto la posible diferencia de performance entre ambos caminos, brindando mayor conocimiento de la situación actual.

- Aun teniendo caminos idénticos de ida y vuelta, pueden haber diferencias radicales en las pérdidas debido a la asimetría en las colas de los routers atravesados.

- La performance de cierta aplicación puede depender básicamente de la performance en una sola dirección. Por ejemplo, la performance de FTP sobre TCP depende de la performance del camino en el sentido del flujo de datos y no del camino de regreso de los acuses de recibo.

- En redes con QoS (calidad de servicio), el aprovisionamiento de recursos en un sentido puede ser radicalmente distinto de la dirección reversa, resultando en garantías diferentes de QoS.

Pérdidas en un sentido

Nombre de la métrica

Pérdida-de-paquetes-unidireccional-tipo-P

Parámetros de la métrica

- 1. Src: dirección IP de un host.
- 2. Dst: dirección IP de un host.
- 3. T: cierto tiempo.

Unidades de la métrica

El valor de la pérdida en un sentido es 0 (transmisión exitosa) o 1 (se perdió el paquete)

Definición

La pérdida de un paquete de Src a Dst en tiempo T es 0 si Src envió el primer bit del paquete tipo-P a Dst en tiempo de cable T y Dst recibió el paquete. Por el contrario, la

pérdida es 1 si Dst no recibió el paquete.

Observaciones

Las nociones de pérdida y retardo unidireccional están intimamente ligadas. En efecto, la pérdida será 0 si el reatrdo es un valor finito, y 1 si el retardo resulta indefinido (infinito).

Metodología de medida

Como en todos los casos anteriores, los detalles de una metodología de medida dependerán del tipo de paquete que se utilice en la misma. Sin embargo, a continuación se presenta un método general de medición:

- Asegurar que Src y Dst estén sincronizados temporalmente. El grado de sincronización es un parámetro de la metodología y dependerá del umbral que se tome para considerar que un paquete se ha perdido.
- Formar el paquete de test con las direcciones IP de Src y Dst.
- Poner una estampa de tiempo en el origen y enviar el paquete.
- Si el paquete arriba en un período razonable de tiempo, el valor de la medida será 0. El umbral de lo "razonable" será también un parámetro de la metodología.
- De no arribar el paquete, el valor de la medida será 1.

Muestra de pérdida de paquetes en un sentido

Como en el caso anterior, definimos una muestra de la pérdida de paquetes en un sentido entre Src y Dst como una secuencia de parejas tiempo-pérdida, en el intervalo Ti,Tf. Para seleccionar los tiempos en los que se realizará cada medida individual se utilizará un proceso Poisson de parámetro λ (se realiza una medida cada $1/\lambda$ s en promedio) cuyos valores estén restringidos al intervalo [Ti,Tf]

Nombre de la métrica

Muestra-Poisson-de-pérdida-unidireccional

Parámetros de la métrica

- 1. Src: dirección IP de un host.

- 2. Dst: dirección IP de un host.
- 3. Ti: cierto tiempo.
- 4. Tf: cierto tiempo.
- 5. λ: tasa de eventos en recíproco de segundos.

Unidades de la métrica

Una secuencia de pares [T,L], donde T es el tiempo en el que se realiza la medida y L es el resultado de la medida de pérdida individual (0 o 1).

Definición

Dados Ti,Tf y λ, calculamos un proceso seudo-randómico de Poisson que comience en Ti, finalice en Tf y tenga una tasa promedio de arribos igual a λ. En cada tiempo de éste proceso obtenemos el valor de la pérdida -unidireccional-tipo-P, formando así la secuencia de parejas tiempo-pérdida.

Algúnas estadísticas

- **Pérdidas promedio**
  Es importante observar que en general lás pérdidas serán menores al 1 %, por lo que el tamaño de las muestras deberá ser mayor a lo que uno quisiera para obtener resultados representativos.

H.1.6. Retardo de ida y vuelta (RFC 2681)

Introducción

El retardo de ida y vuelta entre dos host puede resultar de importancia por las siguientes razones:

- El valor mínimo de ésta métrica da una indicación del retardo debido exclusivamente a la transmisión y propagación, siendo además una estimación del retardo sufrido en condiciones de baja carga.
- Valores de retardo por arriba de éste valor mínimo dan una indicación del nivel de congestión de la red.

A su vez, si bien la medida del retardo de ida y vuelta tiene muchas deficiencias frente a la medida del retardo unidireccional, se pueden distinguir un par de ventajas:

- Facilidad de implementación: en gral, la medidas de retardo unidireccional necesitan de algún tipo específico de software en ambas puntas para realizar la medida. Para las medidas de retardo de ida y vuelta ya hay implementadas herramientas estándar en casi todos los sistemas operativos (como los paquetes de ICMP echo-request implementados por el "ping") que trabajan con servicios de eco en puertos bien conocidos.
- Las medidas de retardo de ida y vuelta son inmunes a los problemas de sincronización entre loa equipos de origen y destino.
- Facilidad de interpretación: en algúnas casos, el retardo de ida y vuelta es en efecto la cantidad de interés. En éstos casos la deducción de dicho tiempo en función de los retardos unidireccionales y del conocimiento del tiempo de procesamiento del destino es más complicada e inexacta.

Retardo de ida y vuelta

Nombre de la métrica

Retardo-de-ida-y-vuelta-tipo-P

Parámetros de la métrica

- 1. Src: dirección IP de un host.
- 2. Dst: dirección IP de un host.
- 3. T: cierto tiempo.

Unidades de la métrica

Número real o indefinido (informalmente infinito) de segundos.

Definición

Dado un número real dT de segundos, decimos que el retardo de ida y vuelta de tipo P entre Src y Dst en el tiempo T es dT si Src envió el primer bit del paquete de tipo P al destino en tiempo de cable T, Dst recibió dicho paquete e inmediatamente envió otro paquete de tipo P de vuelta a Src, y Src recibió el último bit de éste paquete en tiempo de cable T+dT.

El retardo de ida y vuelta será indefinido si en alguno de los puntos antes definidos Src o Dst no recibiese el paquete, o bien Dst no enviase el paquete de vuelta.

Observaciones

Los valores del tiempo T en el cual se realiza la medida deben ser lo más exacto posibles para obtener conclusiones del estado de la red en ese instante. Por lo tanto, Src debe tener un conocimiento del tiempo real actual. En general se utiliza el NTP con éste fin, por lo que, dado la instrumentación del protocolo, se debe tomar en cuenta que si entre el momento de envío del paquete y la recepción ocurrió alguna actualización de tiempo la medida se verá afectada.

Metodología general

- En el origen, armar el paquete de test de tipo P con las direcciones IP de Src y Dst, teniendo en cuenta que de agregar carga útil la misma deberá ser generada aleatoriamente. El paquete debe tener algún tipo de información de identificación para poder identificar el paquete de respuesta desde Dst (por ejemplo, incluyendo en la carga útil el valor de la estampa de tiempo al enviarse el paquete).

- El destino debe ser capaz de recibir el paquete y de enviar una respuesta.

- Enviar el paquete con la estampa de tiempo correspondiente. Si el paquete llega al destino, enviar la correspondiente respuesta lo antes posible (para medir efectivamente el retardo de la red y no involucrar el retardo de procesamiento de los hosts).

- Al recibir el paquete de respuesta, poner una estampa del tiempo de arribo y, restando las estampas inicial y final se obtiene una estimación del retardo de ida y vuelta.

- Si el paquete no arriba en un tiempo razonable (donde "razonable" dependerá de la metodología en sí) se marcará el resultado como indefinido.

H.1.7. Patrón de pérdidas en un sentido (RFC 3357)

Introducción

Utilizando como base la métrica de pérdida definida en H.1.5, se definen dos métricas derivadas: la "distancia de pérdida" o LD (loss distance) y el "período de pérdida" o LP

(loss period). El LP registra la frecuencia y el largo de la pérdida una vez que comienza, mientras que el LD registra el espaciamiento entre LPs. El fin de éstas nuevas métricas es la de caracterizar los patrones de las pérdidas sufridas por los flujos de paquetes en la red. La Internet presenta ciertos comportamientos específicos (como la pérdida de paquetes en ráfagas ante situaciones de congestión) que pueden afectar la performance percibida por un operador o usuario final. El patrón o distribución de las pérdidas es un factor clave que influye directamente en ciertas aplicaciones de tiempo real, como voz y video. Para una misma tasa promedio de pérdidas, dos distribuciones diferentes pueden llegar a tener impactos muy distintos sobre la percepción de la performance desde las puntas del servicio.

Definiciones Básicas

- **Número de secuencia:** Se asignan números enteros consecutivos a paquetes consecutivos temporalmente.
- **Pérdidas en ráfaga:** Pérdidas que abarcan paquetes consecutivos de un flujo de paquetes.
- **Distancia de pérdida o LD:** La diferencia, en números de secuencia, entre dos paquetes perdidos que pueden o no estar separados por paquetes recibidos.
- **Período de pérdida o LP:** Sea P_i el i-ésimo paquete; definamos $f(P_i) = 1$ si P_i se perdió y 0 en caso contrario. Se define el comienzo de un LP si $f(P_i) = 1$ y $f(P_{i-1}) = 0$, considerándose el final cuando arribe exitosamente un paquete.

  Ejemplo: consideremos la siguiente secuencia de paquetes (donde x indica paquete perdido y r paquete recibido) con número de secuencia igual al lugar de aparición (comenzando por 0) y su respectiva $f(P_i)$

  r r r x r r x x x r x r r x x x
  0 0 0 1 0 0 1 1 1 0 1 0 0 1 1 1

  Se observan 4 LP en 3, 6, 10 y 13.

Muestras de LD y LP en un sentido

Nombres de las métricas

- Flujo de tipo P de LD unidireccional
- Flujo de tipo P de LP unidireccional

Parámetros

- 1. Src: dirección IP de un host.
- 2. Dst: dirección IP de un host.
- 3. Ti: cierto tiempo.
- 4. Tf: cierto tiempo
- 5. λ, la tas del método de muestreo.

Unidades

- Secuencia de pares de la forma [LD,pérdida], donde pérdida se deriva de la muestra de "pérdida unidireccional" y LD es un entero positivo o 0.
- Secuencia de pares de la forma [LP,pérdida], donde pérdida se deriva de la muestra de "pérdida unidireccional" y LP es un entero positivo o 0

Definiciones

- **Distancia de pérdida**: cuando se pierde un paquete, miramos su número de secuencia y lo comparamos con el número de secuencia del último paquete perdido. La diferencia entre estos números de secuencia es la distancia de pérdida. Se obtiene de esta forma la secuencia de pares de la forma [LD,pérdida] antes definida. La LD del primer paquete perdido se define como 0. Esta definición asume números de secuencia consecutivos y crecientes para los paquetes de prueba.

- **Período de pérdida**: comenzamos un contador n en cero, y lo incrementamos en uno cada vez que un paquete cumple con la definición previa de período de pérdida. Se tiene así una secuencia de la forma [LP,pérdida], donde nuevamente pérdida es derivada de la muestra de "pérdida unidireccional" y LP es 0 (si la pérdida es 0) o n (si la pérdida es 1). De ésta forma, cuando un paquete se pierde n indica el LP al cual éste pertenece.

Ejemplos

Consideremos la siguiente muestra de pérdida unidireccional:

$$[T_1, 0], [T_2, 1], [T_3, 0], [T_4, 0], [T_5, 1], [T_6, 0], [T_7, 1], [T_8, 0], [T_9, 1], [T_{10}, 1]$$

Los paquetes enviados en T_2, T_5, T_7, T_9 y T_{10} se pierden. Obtengamos las dos muestras antes definidas:

Como el paquete 2 es el primero en perderse, la distancia de pérdida asociada vale 0. Para el próximo paquete perdido (el 5) la distancia de pérdida es 3. De ésta forma se obtiene la siguiente muestra de distancia de pérdida:

[0,0],[0,1],[0,0],[0,0],[3,1],[0,0],[2,1],[0,0],[2,1],[1,1]

El paquete 2 pone el contador n en 1, que es luego incrementado por los paquetes 5, 7 y 9 (el paquete 10 no incrementa el contador según la definición anterior). La muestra de período de pérdida resulta:

[0,0],[1,1],[0,0],[0,0],[2,1],[0,0],[3,1],[0,0],[4,1],[4,1]

Metodologías

En lo que respecta a una metodología general a seguir, es importante observar que tanto la muestra del LP como la de LD se derivan de la muestra de pérdida unidireccional, por lo que el procedimiento general para obtener la muestra es básicamente el mismo.

Observaciones

La muestra de distancia de pérdida permite el estudio de la separación entre pérdidas de paquetes. Esto puede resultar útil en la determinación de cierto factor de disperción asociado con la tasa de pérdida. Conjuntamente, la muestra del período de pérdida permite estudiar las pérdidas en ráfaga. Un único período de pérdida de largo n puede significar una porción importante de la tasa de pérdida. Observar que por su propia definición, dos pérdidas en ráfaga pueden estar separadas por uno o más paquetes recibidos.

Estadísticas

Tasa de pérdida perceptible

Se dice que la pérdida de un paquete es perceptible si la distancia entre dicha pérdida y la pérdida anterior es menor a cierto número entero positivo δ. Dada una muestra de LD unidireccional, la tasa de pérdida perceptible puede ser calculada cómo el cociente entre el número

de pérdidas perceptibles y el número total de paquetes recibidos (también se puede calcular cómo el cociente entre número de pérdidas que violan la restricción δ y el número total de pérdidas). La estadística es muy útil en aquellos casos donde las pérdidas cercanas afectan la calidad del servicio. Por ejemplo, muchos codecs utilizados en servicios multimedia soportan niveles de pérdidas bajos mediante el uso de algoritmos de interpolación de la información pasada. Claramente la eficiencia de estos algoritmos decae al tener pérdidas separadas por distancias pequeñas. Tomando δ en función de la sensibilidad del algoritmo de reconstrucción es posible utilizar la estadística para medir cuan influyente será la tasa de pérdida sobre el servicio.

Largo de período de pérdida

Representa una secuencia de pares (período de pérdida-largo), donde el largo del período de pérdida se obtiene mediante conteo de la muestra (LP, pérdida). Está estadística es un indicador de pérdidas en ráfagas, las cuales claramente influyen en varios servicios de tiempo real.

Largo de período entre pérdidas

Esta estadística mide la distancia entre períodos sucesivos de pérdida. Distancias pequeñas resultaran en pérdidas de mayor frecuencia y degradación de los posibles algoritmos de reconstrucción.

Ejemplos

Consideremos la siguiente muestra de pérdidas:

$$[T_1, 0], [T_2, 1], [T_3, 0], [T_4, 0], [T_5, 1], [T_6, 0], [T_7, 1], [T_8, 0], [T_9, 1], [T_{10}, 1]$$

Sea $\delta = 2$. La muestra de LD será la siguiente:

$$[0, 0], [0, 1], [0, 0], [0, 0], [3, 1], [0, 0], [2, 1], [0, 0], [2, 1], [1, 1]$$

En el ejemplo hay 3 distancias de pérdida menores o iguales a δ ,por lo que la tasa de pérdida perceptible será 3/5.

Consideremos ahora la siguiente muestra de LD:

$$[0, 0], [1, 1], [0, 0], [0, 0], [2, 1], [0, 0], [3, 1], [0, 0], [4, 1], [4, 1]$$

Los largos de los períodos de pérdida individuales serán 1, 1, 1 y 2 respectivamente, por lo que la muestra de los largos de los períodos de pérdida será:

$$[1, 1], [2, 1], [3, 1], [4, 2]$$

En la misma muestra de LD los períodos 1 y 2 están separados por distancia 3, los períodos 2 y 3 por distancia 2 y los períodos 3 y 4 por distancia 2, por lo que la muestra de los largos de los períodos entre pérdida será:

$$[1, 0], [2, 3], [3, 2], [4, 2]$$

H.1.8. Medición de variación del retardo (RFC 3393)

1. Introducción

La variación del retardo entre paquetes constituye una de las métricas fundamentales en los servicios de tiempo real, en particular en el servicio de VoIP y VideoIP on demand. La definición del ipdv (IP Packet Delay Variation) o "jitter" tiene sentido solo para paquetes dentro de un flujo de paquetes.

Sabido es que varias aplicaciones no soportan altos niveles de jitter, principalmente por requerimientos de los sistemas de transmisión y recepción. La implementación de buffers de recepción permite mitigar los efectos de la variación del retardo entre paquetes, pero a costa del aumento del retardo global de punta a punta. El dimensionado del tamaño de dichos buffers solo puede lograrse mediante la estimación de esta variación. El conocimiento del jitter es también importante a la hora de estudiar la dinámica de las colas de los enlaces de la red.

La ventaja que tiene esta métrica sobre otras referidas a tiempos es su robustez frente a problemas de sincronización de los relojes de los equipos de las puntas. Esto permite el uso de la métrica sin restricciones de sincronización, factor por demás influyente en la medida de retardos unidireccionales.

2. Variación del retardo en un sentido

Nombre de la métrica

Variación-del-retardo-unidireccional-tipo-P

Parámetros de la métrica

- 1. Src: dirección IP de un host.
- 2. Dst: dirección IP de un host.
- 3. T_1: cierto tiempo.
- 4. T_2: cierto tiempo.
- 5. L: largo de paquete en bits (todos los paquetes del flujo deben tener el mismo tamaño).
- 6. F: función de seleccción de los dos paquetes involucrados en la métrica (por ejemplo, tomar siempre dos paquetes seguidos) .
- 7. I_1 e I_2: tiempos que indican el comienzo y el fin del intervalo de medida.

Unidades de la métrica

Número real o indefinido (informalmente infinito) de segundos.

Definición

Sean I_1 e I_2 dos tiempos tales que el primer paquete que pasa por el punto de medida MP_1 luego de I_1 tiene asignado el índice 0 y el último paquete en pasar por MP_1 antes de I_2 tiene asignado el mayor índice. La variación del retardo para dos paquetes de Src a Dst seleccionados por la función F se define como la diferencia entre el retardo unidireccional del paquete de índice mayor en tiempo de cable T_2 y el retardo del paquete 0 en tiempo de cable T_1. Por lo tanto, decir que ddT es una medida de la variación del retardo unidireccional entre Src y Dst en T_1 , T_2 significa que la fuente Src envió 2 paquetes, el primero en tiempo de cable T_1 (primer bit) y el segundo en tiempo de cable T_2 y que los mismos fueron recibidos en Dst en tiempos de cable $dT_1 + T_1$ (úlitmo bit) y $dT_2 + T_2$, dando como resultado $ddT = dT_2 - dT_1$. Si alguno de los paquetes no llega al destino la medida queda indefinida.

La figura H.1 ilustra la definición de la métrica. Supongamos que los paquetes i y k son seleccionados:

En este caso, $ddT = dT_k - dT_i$

3. Definición de muestras de variación del retardo

La muestra permite el análisis de la variación del retardo individual. Se realizan las medidas individuales trabajando sobre un flujo de test generado mediante un proceso de Poisson de tasa λ.

Nombre de la métrica

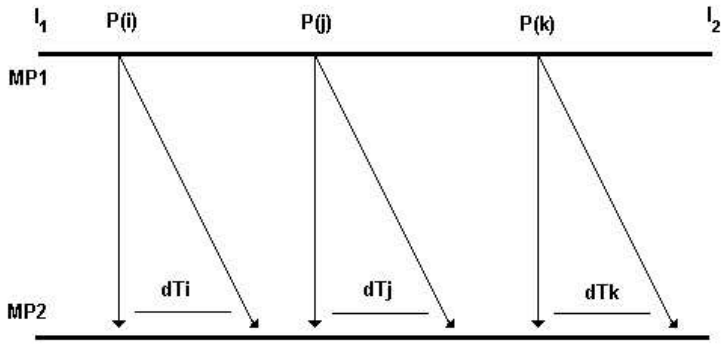


Figura H.1: Ejemplo de medida

Muestra-Poisson-de-la-variación-del-retardo-unidireccional

Parámetros de la métrica

- 1. Src: dirección IP de un host.
- 2. Dst: dirección IP de un host.
- 3. T_0: cierto tiempo.
- 4. T_f: cierto tiempo.
- 5. λ: tasa de generación de paquetes de test.
- 6. L: largo de paquete en bits (todos los paquetes del flujo deben tener el mismo tamaño).
- 7. F: función de seleccción de los dos paquetes involucrados en la métrica (por ejemplo, tomar siempre dos paquetes seguidos).
- 8. I_i e I_{i+1}: tiempos que indican el comienzo y el fin del intervalo de medida.

Unidades de la métrica

Secuencia de tuplas de la forma $[T_1, T_2, dT]$ donde T_1 y T_2 son los tiempos de envío de los paquetes y dT es la diferencia de los retardos.

Definición

Se define un proceso de generación de paquetes de test de tipo Poisson que comienza antes de T_0 y termina después de T_f. Todos aquellos tiempos T_i mayores a T_0 y menores a T_f son seleccionados para enviar paquetes. Todo paquete que caiga dentro del sub-intervalo

$[I_i, I_{i+1}]$ (intervalo que deberá ser suficientemente grande como para la muestra tenga valor estadístico) es analizado por la función F, obteniendo como resultado los dos paquetes necesarios para la medida. Cómo los paquetes pueden perderse, duplicarse o arribar en distinto orden que el enviado, los paquetes de test deben identificarse con números de secuencia. En caso de paquetes duplicados se debe tomar en cuenta la primera instancia recibida. En aquellos casos donde la medida sea indeterminada (si alguno de los dos paquetes se pierde) se deberá eliminar la medida para poder realizar análisis estadísticos (teniendo en cuenta que formalmente será un análisis condicional).

4. Estadísticas

Distribución de la variación del retardo

La variación del retardo (en lo que sigue ipdv) en un sentido tiene cotas superior e inferior por el simple hecho de que el retardo en un sentido también las tiene. Sea U la cota superior para el retardo (tiempo máximo admisible para considerar un paquete como no perdido) y L la cota inferior. De esta forma las cotas para la variación del retardo serán [L-U,U-L]. De esta forma los valores de ipdv podrán ser positivos y/o negativos. Dado que el rango de variación es acotado (y que los valores infinitos fueron descartados) los valores del ipdv no pueden aumentar o decrementar indefinidamente. En consecuencia se podrá representar la distribución de los valores del ipdv de una muestra de las dos formas clásicas: una pdf empírica y/o una cdf empírica. La pdf empírica es representada mediante un histograma. La cdf empírica es simplemente la proporción de las medidas de ipdv menores que un cierto valor, para todo valor posible adoptado por la ipdv.

Percentiles

Dada la muestra de ipdv y fijado un porcentaje x, el percentil es el elemento y de la muestra tal que el x % de los elementos de la muestra son mayores que y.

Percentiles inversos

Dada la muestra de ipdv y fijado un porcentaje x, el percentil es el elemento y de la muestra tal que el x % de los elementos de la muestra son menores o iguales que y.

Apéndice I

Contenido del CD

En este anexo se detallan las carpetas y archivos contenidos dentro del CD que se incluye en la contratapa de esta documentación.

I.1. Carpetas y archivos incluidos

1. Archivo **pqos.pdf**: documentación del proyecto en formato PDF (Portable Document Format) de Adobe ©.
2. Archivo **PQoSWin32.exe**: instalador del programa desarrollado (PQoS - Perceived Quality of Service) para OS Windows (2000 y XP).
3. Archivo **manual.pdf**: manual del programa en formato PDF (Portable Document Format) de Adobe ©.
4. Carpeta **documentacion**: incluye la documentación de las clases de software del programa desarrollado.
5. Carpeta **fuentes**: incluye los archivos fuente del programa desarrollado.
6. Carpeta **fuentes/herramientas**: incluye programas auxiliares utilizados por el programa.
7. Carpeta **fuentes/multimedia/secuencias**: incluye secuencias de audio y de video utilizadas por el programa.

NOTA: para poder acceder a los archivos en formato PDF es necesario disponer del programa ***Adobe Reader*** ©, versión 3.0 o superior. Al momento de escribir esta documetación se utilizó un computador personal con las siguientes características:

- Procesador AMD-XP 2.6 GHz.
- Memoria DDR 1GB.