Universidad Central "Marta Abreu" de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Telecomunicaciones y Electrónica

TRABAJO DE DIPLOMA

"Evaluación simulada y experimental del algoritmo de reproducción para el control del jitter propuesto por Moon"

AUTOR: Sandy Bolufé Aguila.

TUTOR: MSc. Carlos A. Rodríguez López.

CURSO 2008-2009

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

_____

Firma del Autor

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

_____ _____

Firma del Autor

Firma del Jefe de Departamento donde se defiende el trabajo

_____

Firma del Responsable de Información Científico-Técnica

PENSAMIENTO

Un viaje de mil millas empieza con un paso.

(Anónimo)

DEDICATORIA

A mi madre, padre y hermano, por ayudarme siempre en los momentos difíciles y permitir con sus sacrificios el cumplimiento de este sueño.

AGRADECIMIENTOS

Al profesor Carlos por su tutoría admirable, dedicación y esfuerzo.

A Vanessa por brindarme su apoyo y ayuda incondicional en todo momento.

A José Omar y Erick quienes me aportaron los conocimientos necesarios para trabajar sobre Linux.

A los profesores Samuel e Hiram quienes compartieron el local de trabajo de una forma sencilla y modesta.

A mi amigo Ariel quien desinteresadamente me extendió su mano.

TAREA TÉCNICA

Para lograr los objetivos propuestos en el presente trabajo, la investigación sigue una línea definida por un grupo de tareas, las cuales se muestran a continuación:

- ❖ Revisión bibliográfica de trabajos relacionados con el tema.
- ❖ Estudio del algoritmo de reproducción propuesto por Moon.
- ❖ Determinación de las herramientas de trabajo.
- ❖ Obtención de trazas de paquetes de VoIP simuladas y experimentales.
- ❖ Evaluación del algoritmo ante diferentes condiciones de trabajo.
- ❖ Confección del informe.

Firma del Autor

Firma del Tutor

RESUMEN

En el presente trabajo se realiza la evaluación simulada y experimental del algoritmo de reproducción para el control del jitter propuesto por Moon. Para ello se hace una descripción del sistema de VoIP donde se explican sus principales características, entre ellas el procesamiento de la señal de audio y los factores que afectan la calidad de servicio. Se describe el funcionamiento del algoritmo de Moon y se implementa un paquete de software de código abierto que permite simular completamente la cadena de procesamiento de VoIP. Este incluye dos de los esquemas de codificación más utilizados (G.711, G.729) y dos modelos de calidad de la Unión Internacional de Telecomunicaciones (Modelo-E, PESQ), los cuales se utilizan para predecir la calidad perceptual de la transmisión de extremo a extremo. También permite leer ficheros de trazas de paquetes de VoIP generadas por el software NS-2.33 (Network Simulator), la incorporación de trazas de tráfico real y simular el algoritmo de reproducción descrito por Moon. Por último se evalúa el algoritmo ante diferentes condiciones de trabajo y se analizan los resultados obtenidos.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Internet ha posibilitado el desarrollo de nuevas formas de comunicación, entre ellas se encuentran las aplicaciones en tiempo real las cuales incluyen no solo la transmisión de datos sino también audio y video. Dentro de estas aplicaciones está la telefonía sobre Internet conocida como Voz sobre IP (VoIP) la cual es un servicio multimedia que permite la transmisión de voz en tiempo real de forma interactiva. Los primeros trabajos de investigación como NeVoT (Network Voice Terminal) por Schulzrinne, FreePhone por Bolot y Rat (Robust audio tool) por el proyecto Mice se encaminaron a disminuir los efectos de las pérdidas y de la variabilidad de la demora que se presentan sobre Internet para este tipo de aplicaciones y de esta forma ofrecer la mejor calidad posible para una conversación (Bolot, 1993).

Existen ruteadores en Internet que utilizan memorias intermedias las cuales implementan políticas FIFO (First In First Out), por lo que el primer paquete recibido por el buffer de alguno de ellos es el primero en ser reexpedido. Esto tiene la ventaja de tener un funcionamiento relativamente sencillo, pero la desventaja de que todos los paquetes tienen la misma prioridad: un paquete de voz tiene que esperar en el buffer como cualquier otro paquete, incluso si las restricciones de demora para las aplicaciones en tiempo real son rígidas. Este mecanismo causa primordialmente tres fenómenos: demora, pérdida de paquetes y variabilidad de la demora, los cuales afectan la calidad de servicio de las aplicaciones en tiempo real (Na y Yoo, 2002).

Las aplicaciones de voz generan paquetes en intervalos regulares de tiempo. El tráfico generado por una fuente se divide en períodos de actividad y en períodos de silencio. Al cruzar Internet, los paquetes de audio sufren demoras variables debido al tiempo de espera en las filas de los ruteadores. Esta variabilidad modifica los intervalos regulares de tiempo

en que los paquetes se transmitieron. Para poder reproducir los paquetes recibidos, las aplicaciones deben reducir esta variabilidad, almacenando los paquetes en una memoria intermedia y reproduciéndolos después de un tiempo límite. Los paquetes que llegan después de su correspondiente tiempo límite se consideran perdidos y no se reproducen. Si el tiempo que se espera por los paquetes aumenta, la probabilidad de que estos lleguen antes de su tiempo límite también aumenta. Sin embargo, una gran demora disminuye la interactividad de una sesión de audio. Por esto, existe un compromiso entre demora y pérdidas debido a llegadas tardías para optimizar el tiempo de espera de los paquetes en el buffer el cual constituye el objetivo principal de los algoritmos de control de demora de reproducción para VoIP (Narbutt et al., 2005).

En la actualidad se realizan grandes esfuerzos en el estudio de mecanismos que permitan asegurar una adecuada calidad de servicio en las redes IP pero la solución parece estar todavía distante. Los investigadores centran sus trabajos en los buffers de reproducción los cuales juegan un papel importante en la calidad de servicio percibida por el usuario y donde hasta el momento no existe acuerdo sobre un esquema común estandarizado, por lo cual el objetivo principal de este trabajo es evaluar el algoritmo de reproducción para el control de jitter propuesto por Moon ante diferentes condiciones específicas de trabajo. Con vistas a lograr la idea central de la investigación se trazan los siguientes objetivos específicos:

- Implementar un paquete de software que permita simular completamente la cadena de procesamiento de VoIP.
- Obtener trazas de paquetes de VoIP de forma simulada y experimental que permitan la evaluación del algoritmo propuesto por Moon.

Como resultado de la evaluación se debe determinar la calidad de la voz resultante. Es necesario decir que aunque el algoritmo de Moon es conocido y referenciado resulta útil evaluarlo bajo condiciones específicas de trabajo.

Con el estudio y análisis del algoritmo, y la definición de herramientas para su evaluación bajo condiciones específicas de trabajo se aportan no solo resultados del análisis concreto de su desempeño, sino una forma general para realizar nuevas evaluaciones en escenarios con características diferentes.

Organización del Informe

El informe de la investigación se organiza de la siguiente forma: resumen, introducción, capitulario, conclusiones y recomendaciones, referencias bibliográficas.

Introducción

- Se realiza una reseña donde se define la necesidad, actualidad e importancia del tema que se aborda y se mencionan los elementos del diseño teórico.

Capítulo I

- Se exponen las principales características del sistema de VoIP, se explica el funcionamiento de los modelos para la evaluación de la calidad (PESQ, Modelo-E) y se describe el algoritmo propuesto por Moon.

Capítulo II

- Se describe el proceso de implementación y utilización del paquete de software QofIS_v2, así como la compilación e instalación de los software complementarios.

Capítulo III

- Se realiza la evaluación simulada y experimental del algoritmo propuesto por Moon ante diferentes condiciones de trabajo y se analizan los resultados.

Conclusiones

- Se describen los resultados obtenidos a partir de los objetivos trazados inicialmente.

CAPÍTULO 1. Fundamentos de VoIP

1.1 Sistema básico de VoIP

En un período de tiempo relativamente muy corto, desde que el primer software para lograr la transmisión de voz sobre redes de conmutación de paquetes apareció, el desarrollo de la telefonía a través de Internet se ha incrementado considerablemente.

Normalmente una Red Telefónica Pública Conmutada (PSTN[1]), provee el servicio de telefonía convencional por medio de un canal de comunicación dedicado única y exclusivamente para cada transmisión, por lo que al establecerse una llamada telefónica se crea un circuito que es proporcionado por la red exclusivamente para esa llamada. Por el contrario, la telefonía sobre Internet funciona sobre una red de conmutación de paquetes (Kurose y Ross, 2000), donde el mensaje original se divide en paquetes independientes a los cuales se les agrega información de control y se envían de un extremo a otro, donde en el momento de llegada al destino, se reorganizan para obtener el mensaje original. En las aplicaciones de Voz sobre Internet (VoIP) (Li et al., 2000) se establece un servicio de transmisión de voz entre dos o más puntos sobre una red de datos basada en el Protocolo de Internet (IP), así como el intercambio de información de control requerido para esa transmisión. Para poder transformar la voz en paquetes, la señal de voz humana digitalizada se codifica. La mayoría de los esquemas que codifican la voz comprimen segmentos de la señal de audio y generan paquetes. Uno o múltiples paquetes de voz se concatenan en un paquete al cual se le agrega una cabecera RTP, UDP, IP y se envían a través de Internet a su destino, como cualquier otro tipo de paquetes que se transmiten por este medio (Hoene y Wiethölter, 2004).

[1] Public Switched Telephone Network

En el concepto básico de ejecutar una aplicación de VoIP generalmente se establecen los siguientes pasos**:**

- Conversión de la señal de voz analógica a formato digital.
- Proceso de compresión de la señal digital y confección de paquetes IP.
- Transmisión de los paquetes IP al receptor.
- Recepción en el extremo final donde los paquetes se reorganizan y convierten de señal digital a analógica para la audición.

Las aplicaciones de VoIP se ejecutan en tiempo real, esto provoca que se vean afectadas por diferentes factores que degradan la calidad de la voz, dentro de los cuales se encuentran las pérdidas, la demora y la variabilidad de la demora en la llegada de los paquetes, por esta causa se ha hecho necesario la introducción de una serie de protocolos y mecanismos que permitan asegurar una adecuada calidad de servicio (QoS[2]) para este tipo de aplicaciones.

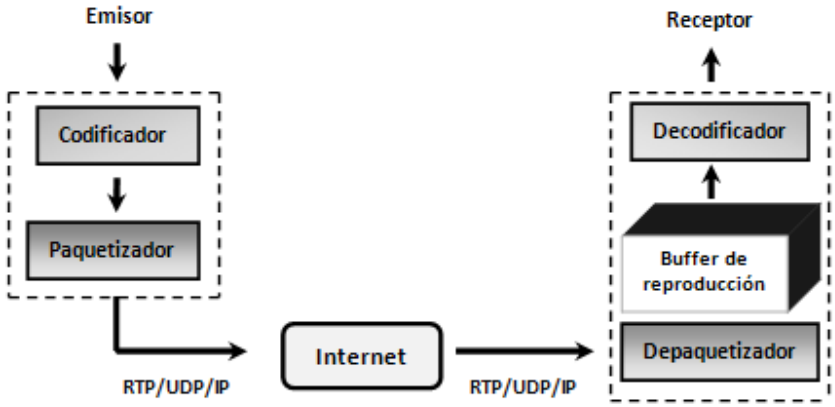

Figura 1. Sistema de VoIP

1.2 Procesamiento de la señal de audio

En las redes de computadoras se transmiten ceros y unos (bits), por esta causa la información se representa como una cadena binaria. Para transmitir el audio sobre Internet,

[2] Quality of Service

el mismo se digitaliza y comprime, el proceso de compresión es muy importante debido a que la señal de audio cuando no se comprime consume una cantidad grande de almacenamiento y ancho de banda. Para optimizar las sesiones de audio se necesita estructurar la señal en paquetes, suprimir los intervalos de silencio y cancelar el eco.

1.2.1 Digitalización de la señal de audio

Para convertir una señal de audio analógica en una señal digital normalmente se siguen los siguientes pasos (Davidson y Peters, 2000):

- La señal analógica se muestrea a una tasa fija. El valor de cada muestra es un número real dado.
- Cada muestra se aproxima a uno de los valores de números finitos, esto se conoce como cuantificación, donde el número de valores finitos es una potencia de 2.
- Cada uno de los niveles de cuantificación se representa por un número fijo de bits.
- Cada muestra se convierte a su representación en bits. Las representaciones en bits de todas las muestras se concatenan para formar la representación digital de la señal.

En el receptor la señal digital se convierte a una señal analógica. Esta señal analógica es usualmente distinta a la señal de audio original debido a que en el proceso cuantificación se aproxima a un valor finito.

1.2.2 Empaquetamiento del audio

En el proceso de empaquetamiento del audio, la señal se divide en segmentos de tiempo donde cada uno se muestrea, codifica y transmite en paquetes. En el emisor se generan los paquetes periódicamente: el tiempo de duración del intervalo en el que se generan depende del tamaño y la cantidad de segmentos de voz que lleva cada paquete del nivel de transporte.

1.2.3 Codificadores de audio

Para alcanzar, en aplicaciones multimedia, una transmisión de señales confiable entre emisor y receptor se usan los codificadores los cuales realizan el proceso de compresión de la señal (Davidson y Peters, 2000). En la Tabla 1 se muestran características y las tasas de calidad de algunos de los más utilizados usando el Modelo-E que se describe en la Sección 1.6.1. Dentro de ellos se encuentra el UIT-T G.711 el cual es del tipo Modulación

por Impulsos Codificados (PCM[3]) y codifica a una tasa de 64 Kbits/s y otro es el UIT-T G.729, donde el intervalo de los paquetes es de 20 milisegundos (ms) que corresponde a dos segmentos de voz de 10 ms cada uno.

Tabla 1. Codificadores de voz

Origen	Estándar	Método	Tasa de bit [Kbps]	Retardo [ms]	Calidad [Modelo-E]
UIT-T	G.711	PCM	64	0.125	94.3
UIT-T	G.726	ADPCM	32	0.125	87.3
UIT-T	G.728	LD-CELP	16	0.625	87.3
UIT-T	G. 729A	CS-ACELP	8	10	84.3
UIT-T	G. 723.1	MP-MLQ	6.3	30	79.3
ETSI	GSM-EFR	ACELP	12.2	20	89.3

1.2.4 Detección de Actividad de Voz

El tráfico que se genera por una transmisión de voz se divide en períodos de actividad (talkspurt en inglés) y en períodos de silencio (Ramos et al., 2003). En la mayoría de los codificadores para disminuir el tráfico generado por el audio, se descartan las muestras obtenidas durante los períodos de silencio y únicamente se codifican las muestras que se generan durante los períodos de actividad (Atzori y Lobina, 2006).

En cada emisor se utiliza un Detector de Actividad de Voz (VAD[4]) con el objetivo de diferenciar los períodos de actividad de los períodos de silencio. En este se establece una indicación de la presencia de actividad de voz para facilitar el procesamiento de la misma, además se fija el principio y el fin de las ráfagas de voz. Normalmente cuando el VAD detecta actividad, se espera un tiempo determinado para el empaquetamiento. Existen varios problemas para realizar la detección de los períodos de actividad, uno de ellos es determinar cuando comienza o termina una ráfaga de voz. En la mayoría de los casos se corta el comienzo de la sentencia, este fenómeno se conoce como recorte en el principio y final del discurso (front-end speech clipping en inglés). Otro problema que se presenta es diferenciar entre la actividad de voz y el ruido ambiental, es decir, cuando el VAD es

[3] Pulse Code Modulation
[4] Voice Activity Detection

incapaz de diferenciar entre el ruido generado por el ambiente y la voz, esto se conoce como el umbral Señal a Ruido (Davidson y Peters, 2000).

1.2.5 Eco

En telefonía, se define que estamos en presencia del eco cuando la persona que habla puede escuchar su propia voz durante una sesión de audio, este fenómeno se presenta debido a la reflexión del sonido el cual regresa a la fuente y puede causar tanto interrupciones molestas como pérdida de la interactividad. Por supuesto que mientras más ruidoso y largo es el eco se torna más molesto**.**

Para combatir los efectos del eco, las aplicaciones implementan canceladores de eco dentro de codificadores de baja velocidad y operan estos canceladores dentro del procesador de señal digital. Los canceladores del eco están limitados por el número total de tiempo que ellos esperan para recibir el sonido reflejado, este fenómeno se conoce como final del eco.

1.3 Transporte de Voz sobre IP

Un protocolo de transporte provee una comunicación lógica entre las aplicaciones de diferentes extremos de una red, define el formato y el orden de los segmentos intercambiados entre dos o más entidades de comunicación, así como las acciones tomadas en la transmisión o recepción de un mensaje.

Una red puede ser capaz de utilizar varios protocolos de transporte debido a que en cada uno de ellos se ofrece un modelo de servicio diferente para las aplicaciones. Los protocolos que se definieron con el objetivo de extender el servicio de entrega de IP fueron el Protocolo de Control de la Transmisión (TCP) **y** el Protocolo de Datagrama de Usuario (UDP), los cuales pertenecen a la capa de transporte y permiten garantizar la comunicación.

El protocolo que se usa tradicionalmente para transportar la voz sobre IP es UDP debido a que presenta varias ventajas sobre TCP para las aplicaciones que transcurren en tiempo real. En el protocolo UDP no se establece una conexión antes de enviar los paquetes, no se implementan mecanismos para el control de la congestión y el encabezado de los paquetes es más pequeño. Por el contrario, en TCP se utiliza un mecanismo de control de la congestión que limita cada conexión para compartir el ancho de banda de manera justa, esto

puede tener un efecto negativo en las aplicaciones en tiempo real que tienen un requerimiento mínimo de ancho de banda, también se puede introducir variabilidad de la demora debido a las retransmisiones lo cual se torna perjudicial para este tipo de aplicaciones, los acuses de recibo pueden disminuir la tasa de transmisión de los datos y una cabecera de paquete más grande constituye una mayor demora de procesamiento.

1.3.1 Protocolos

En las aplicaciones que se envía contenido multimedia sobre la red en tiempo real se deben agregar números de secuencia y marcas de tiempo a los segmentos antes de pasarlos a la capa de transporte para poder reconstruir el contenido en el receptor sin tener que esperar a que el mismo se obtenga por completo (Schulzrinne et al., 2003). Por tanto se necesita una estructura estandarizada de paquetes en la que estén presentes campos que indiquen el tipo de codificación, estampas de tiempo y números de secuencia. Los protocolos que se definieron con este propósito fueron el Protocolo de Tiempo Real (RTP) y el Protocolo de Control de Tiempo Real (RTCP) como se muestra en la Figura 2.

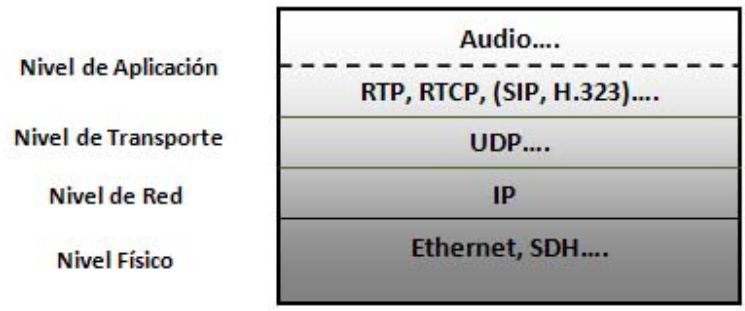

Figura 2. Protocolos para servicios de VoIP

1.3.1.1 RTP

El protocolo RTP se utiliza generalmente sobre UDP, los segmentos de datos de audio se generan por el transmisor de la aplicación multimedia, estos se encapsulan en paquetes RTP, y cada paquete es luego encapsulado en segmentos UDP, en la cabecera RTP se incluye el tipo de codificación, número de secuencia y marca de tiempo. En el lado

receptor, la aplicación extrae los segmentos de audio y utiliza la información de la cabecera para poder decodificarlos y reproducirlos correctamente. En el protocolo RTP no se provee ningún mecanismo para garantizar la entrega, el encapsulamiento solo se ve en los extremos del sistema.

1.3.1.2 RTCP

El protocolo RTCP trabaja conjuntamente con el RTP, su principal función es proveer una retroalimentación sobre la calidad de la distribución de la información. Este 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. En RTCP se usa un puerto diferente al que se utiliza para RTP, por lo general se escogen puertos consecutivos donde el par se asigna al flujo RTP y el impar al flujo RTCP.

1.4 Factores que afectan la calidad de servicio en VoIP

El paradigma de mejor esfuerzo (best-effort en inglés) característico en Internet provoca que se presenten factores que impactan negativamente en la calidad de servicio de las aplicaciones en tiempo real, entre los cuales se encuentran la demora, variabilidad de la demora en la llegada de los paquetes y las pérdidas, por esta causa se trabaja constantemente con el objetivo de mejorar los mecanismos que aseguran la calidad de servicio para este tipo de aplicaciones.

1.4.1 Demora

La demora de boca a oído o de extremo a extremo se mide comúnmente en segundos y se define como el tiempo que transcurre desde el momento en punto que en el emisor se comienza a transmitir una palabra hasta que en el receptor se escucha completamente esa palabra (Na y Yoo, 2002). En una transmisión de voz en tiempo real sobre una red de conmutación de paquetes se pretende superar los efectos causados por la demora de extremo a extremo.

1.4.2 Tipos de demora

La demora de principio a fin o de extremo a extremo se conforma por la demora de empaquetamiento, de red y la demora de reproducción. En la demora de empaquetamiento

se incluye el tiempo de codificación, compresión y decodificación. Los diferentes componentes de la demora de extremo a extremo en VoIP se muestran a continuación (Thomsen y Jani, 2000):

- Demora de empaquetamiento: se refiere al tiempo que toma grabar las muestras de voz y ponerlas en segmentos, el valor promedio normalmente está entre 10 y 30 ms por paquete, en dependencia del hardware utilizado.
  - Demora de codificación: se considera el tiempo necesario para convertir el segmento de voz a su representación binaria, su promedio está entre 5 y 10 ms.
  - Demora de compresión: se estima como el tiempo necesario para comprimir los segmentos de voz, estructurarlos en paquetes e inyectarlos a la red. El tiempo promedio está entre 5 y 10 ms.
  - Demora de decodificación: está entre 5 y 10 ms y se refiere al tiempo que toma descomprimir los paquetes.
- Demora de red: se conforma por la demora de transmisión, propagación, procesamiento y en las filas de espera de los ruteadores.
  - Demora de transmisión: tiempo que se tarda en colocar todos los bits de un paquete en el enlace.
  - Demora de propagación: tiempo que se necesita para que la señal se propague por el medio físico.
  - Demora de procesamiento: tiempo necesario para que el ruteador procese el encabezado de cada paquete.
  - Demora en las filas de espera: tiempo necesario que los paquetes esperan en las filas de los ruteadores para ser procesados.
- Demora de reproducción: tiempo que esperan los paquetes en la memoria intermedia de reproducción (buffer en inglés) del receptor desde que se reciben hasta que se reproducen. Oscila entre 50 y 200 ms**.**

Las aplicaciones en tiempo real se caracterizan por ser sensibles a la demora de extremo a extremo, si la demora es grande los paquetes llegan al receptor pasado su tiempo de reproducción y se afecta la calidad de servicio de la aplicación. La demora de extremo a

extremo debe presentar un promedio total por debajo de 400 ms debido a que valores mayores producen un deterioro en la interactividad de la conversación.

1.4.3 Picos de demora

Un pico de demora (delay spike en inglés) se define como un aumento grande y repentino en la demora de extremo a extremo, el cual es seguido por una serie de llegadas de paquetes casi simultáneas (Ramjee et al., 1994). En (Bolot, 1993) se identifica este fenómeno como resultado de un error de depuración en las operaciones del software de los ruteadores. Los picos de demora afectan severamente la QoS en VoIP representando uno de los problemas principales que se debe resolver, para lograr este objetivo se han creado algoritmos de control de demora de reproducción que son capaces de reaccionar cuando estos picos de demora aparecen.

1.4.4 Variabilidad de la demora

En VoIP el emisor transmite los paquetes de audio igualmente espaciados en el tiempo y los coloca en la red, el tiempo que demoran en llegar al receptor varía por múltiples causas, una de ellas puede ser que el tiempo de espera en la cola de un ruteador para cada paquete sea diferente o que los paquetes se desplacen por diferentes rutas y entonces cuando llegan al receptor se ha perdido el intervalo regular de tiempo que los espaciaba, los paquetes en el peor de los casos pueden no llegar o llegar en desorden, entonces se hace necesario reordenarlos y reespaciarlos en el buffer de reproducción para disminuir las consecuencias de la variabilidad de la demora (jitter en inglés).

1.4.5 Pérdidas

La congestión, los errores de transmisión causados por interferencias cuando se atraviesan las redes inalámbricas y los errores de enlaces que se presentan cuando existen fallas en la gestión de la comunicación, son factores que contribuyen a que los paquetes se pierdan. En las aplicaciones de audio se tolera cierta tasa de pérdidas sin impactar considerablemente la calidad de la voz. En (Buchli y Petit, 2002) se señala que hasta el 5% de pérdidas es tolerable, no obstante en (Narbutt et al., 2005) se indica que las tasas menores al 3% obtienen los más altos niveles de satisfacción, cabe destacar que estos límites dependen del codificador que se utilice.

1.5 Evaluación de la calidad de servicio

Existen diferentes puntos de vista para evaluar la QoS de una sesión de VoIP, uno es a través de la opinión del usuario final, en este caso se tiene una sesión multimedia de referencia, por lo tanto existen dos señales para ser comparadas: la señal original transmitida y la señal recibida deteriorada por la demora, las pérdidas y la variabilidad de la demora. La señal original se reconoce como señal de referencia, entonces un grupo de usuarios escuchan las dos y las comparan de manera subjetiva. Otro punto de vista es manteniendo la razón de pérdidas y la variabilidad de la demora de una aplicación dentro de niveles tolerables, y de esta forma se mide la calidad de servicio a través de la tasa de pérdidas promedio y la demora promedio de reproducción de una sesión determinada, en este caso, se dice que la QoS se mide objetivamente, ya que una cantidad medible es reportada por la aplicación para indicar la calidad vista durante una sesión.

1.5.1 Modelo-E

El Modelo-E es un método estandarizado por la Unión Internacional de Telecomunicaciones (UIT) que permite evaluar los efectos combinados de las variaciones de diversos parámetros de transmisión que afectan la calidad de la conversación telefónica, cuyo principal resultado es una cuantificación escalar de la calidad de transmisión llamado factor de determinación de índices R, que varía linealmente con la calidad de la conversación (ITU-T_G.107, 2003). Una de sus características es el uso de factores de degradación de la transmisión que reflejan los efectos de los dispositivos modernos de procesamiento de señales. En él se estima la calidad de la comunicación de boca a oído percibida por el usuario en el lado receptor.

El factor R es una combinación de los parámetros de transmisión pertinentes para la conexión considerada y se obtiene como se muestra a continuación:

$$R = Ro - Is - Id - Ie\text{-}eff + A \qquad \textbf{(1)}$$

Ro representa en principio la relación señal a ruido básica que incluye fuentes de ruido tales como ruido de circuito y ruido ambiente. El factor Is es una combinación de todas las

degradaciones que aparecen de forma más o menos simultánea con la señal vocal. El factor Id representa las degradaciones producidas por la demora y el factor de degradación efectiva del equipo Ie-eff representa las degradaciones producidas por una baja tasa de codificación, incluye también la degradación debida a pérdidas de paquetes de distribución aleatoria. El factor de mejora A sirve para compensar los factores de degradación cuando existen otras ventajas de acceso para el usuario. El término Ro y los valores Is e Id se subdividen en valores de degradación específicos más detallados.

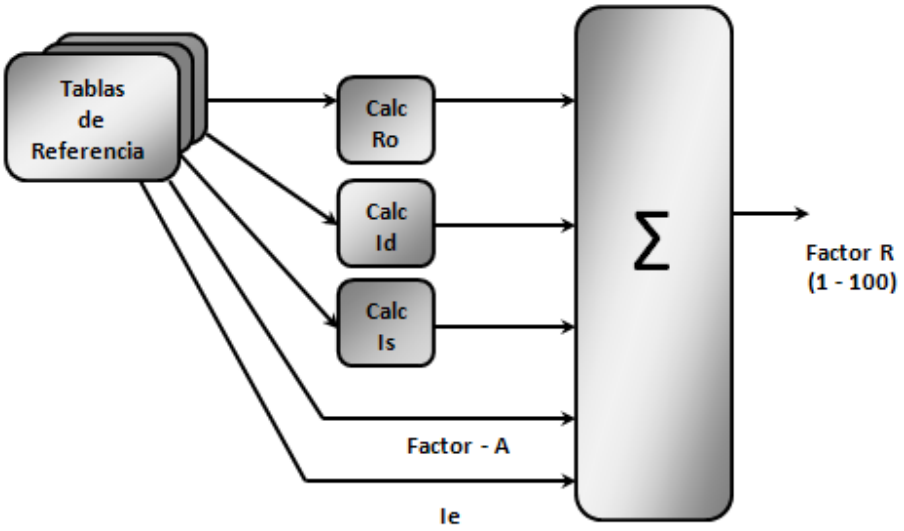

Figura 3. Cálculo del factor R en el Modelo-E

En el Modelo-E se transforman las medidas objetivas en calidad subjetiva, el rango de valores correspondientes a las Categorías de Calidad de la transmisión de voz se muestran en la Tabla 2, donde valores de hasta 70 se consideran aceptables.

Tabla 2. Rangos del valor R en el Modelo-E

Rango del valor R	100 – 90	90 – 80	80 – 70	70 – 60	60 – 0
Categoría de Calidad	La mejor	Alta	Media	Baja	Pobre

1.5.2 Evaluación de la Calidad Vocal por Percepción

La Evaluación de la calidad vocal por percepción (PESQ[5]) es un método para predecir calidad subjetiva de transmisiones de voz tomando en cuenta el filtrado, la variabilidad de la demora y las distorsiones cortas localizadas. PESQ trata estos efectos mediante la ecualización de la función de transferencia, la alineación de tiempo y un algoritmo para promediar distorsiones en función del tiempo.

PESQ compara una señal inicial X(t) con una señal degradada Y(t) que se obtiene como resultado de la transmisión de X(t) a través de un sistema de comunicaciones. La salida de PESQ es una predicción de la calidad percibida por los sujetos en una prueba de escucha subjetiva y que sería atribuida a Y(t). En el primer paso de PESQ se calcula una serie de demoras entre la entrada inicial y la salida degradada, una para cada intervalo de tiempo cuya demora presenta una diferencia significativa en comparación con la del intervalo de tiempo precedente. El algoritmo puede tratar los cambios en la demora durante los períodos de actividad de voz como en los períodos de silencio.

Sobre la base del conjunto de demoras que se encuentran, PESQ compara la señal de entrada con la de salida degradada. Lo esencial en este proceso es la transformación de las dos señales, la inicial y la degradada, en una representación interna que es análoga a la representación psicofísica de señales de audio en el sistema auditivo humano, teniendo en cuenta la frecuencia por percepción y la sonoridad. Esto se realiza en varias etapas: alineación de tiempo, alineación a un nivel de escucha calibrado, correspondencia entre el dominio de la frecuencia y el dominio del tiempo, curvatura de frecuencia y aplicación de la escala compresiva de sonoridad.

La representación interna de las señales se procesa para tener en cuenta efectos tales como las variaciones de la ganancia local y el filtrado lineal que, si no son muy grandes, puede que sólo influyan poco en el aspecto por percepción. Esto se realiza limitando la magnitud de la compensación y haciendo que la compensación se efectúe con posterioridad al efecto.

Por tanto, las diferencias de estado estacionario, de poca magnitud, entre la señal inicial y la degradada quedan compensadas. Los efectos más graves, o las variaciones rápidas, son

[5] Perceptual Evaluation Speech Quality measure

compensadas parcialmente, por lo que queda un efecto residual que contribuye a la perturbación por percepción global. Esto permite utilizar un pequeño número de indicadores de calidad para modelar todos los efectos subjetivos. En PESQ se calculan dos parámetros de error en el modelo cognitivo, los cuales se combinan para dar un valor MOS de calidad. Los resultados proporcionados por PESQ han sido calibrados usando grandes bases de datos de pruebas subjetivas (ITU-T_P.862, 2001a).

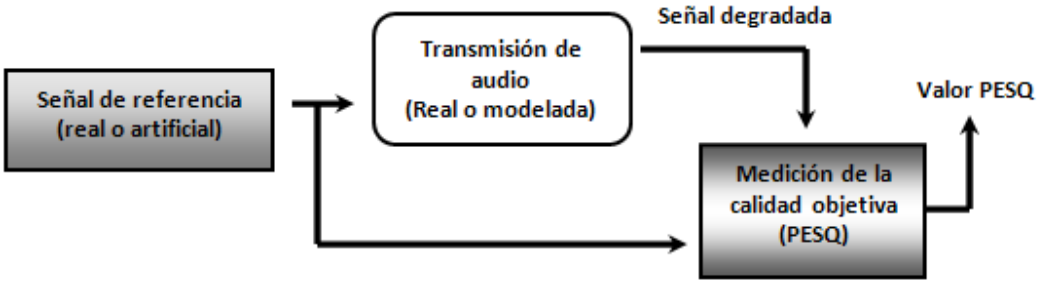

Figura 4. Concepto básico del algoritmo PESQ

1.5.3 Nota Media de Opinión

El método Nota Media de Opinión (MOS[6]) fue desarrollado para determinar de forma subjetiva la calidad de la transmisión, la idea principal es reproducir las condiciones de una conversación telefónica real en un laboratorio, y hacer que un grupo de personas escuche el audio, preguntándoles después el rango de calidad que le asignan a dicho audio. La escala va de 1 a 5 como aparece en la Tabla 3 (ITU-T_P.800, 1996).

Tabla 3**.** Notas de degradación de MOS

Nota	MOS	DMOS	
5	Excelente	Degradación inaudible	No se requiere esfuerzo
4	Bueno	Degradación audible pero no molesta	El esfuerzo no es apreciable
3	Regular	Degradación ligeramente molesta	Esfuerzo moderado
2	Pobre	Degradación molesta	Esfuerzo considerable
1	Malo	Degradación muy molesta	Sin significado

La magnitud de las notas sobre degradaciones se representa por la Nota Media sobre las Degradaciones, DMOS[7].

[6] Mean Opinion Score
[7] Degradation Mean Opinion Score

1.6 Algoritmos de reproducción para el control del jitter

El impacto de los fenómenos que afectan la calidad de servicio en redes de conmutación de paquetes se debe atenuar por mecanismos de control de extremo a extremo. Como los paquetes de audio encuentran una demora variable en las filas de los ruteadores intermedios se modifica la forma periódica original en que se transmitieron. En una aplicación con el fin de utilizar el audio recibido se debe eliminar o por lo menos minimizar esta variabilidad por medio de un tiempo de espera límite en el buffer del receptor. Los paquetes que llegan después de ese tiempo límite se consideran perdidos y por lo tanto no se reproducen.

1.6.1 Relación de compromiso entre demora y pérdidas

Un tiempo de espera muy grande para los paquetes de audio en el buffer del receptor influye negativamente incrementando la demora de extremo a extremo, y por lo tanto con una mayor demora se afecta la interactividad de la aplicación. Asimismo, si el tiempo de espera de los paquetes de audio en el buffer del receptor es pequeño se incrementa el número de pérdidas debido a que se descartan paquetes que demoran en llegar, es por esto que existe un compromiso entre pérdidas y demora.

En los algoritmos de control de demora de reproducción se debe minimizar el tiempo de espera y la razón de pérdidas de los paquetes (Narbutt et al., 2005). Los algoritmos para que sean eficientes deben tomar en cuenta el compromiso entre pérdidas y demora de tal manera que se establezcan estos dos parámetros lo más bajos posibles. La demora de los paquetes debe ser menor a 400 ms y la tasa de pérdidas se debe encontrar entre 5% – 10%**,** en dependencia del codificador para una sesión de audio.

1.6.2 Clasificación

Los algoritmos de control de demora de reproducción se pueden clasificar de acuerdo a la forma en que se calcula y gestiona el tiempo que esperan los paquetes en el buffer del receptor para su reproducción (Liang y Girod, 2003):

- ❖ Algoritmos de control de demora de reproducción fijo
- ❖ Algoritmos de control de demora de reproducción adaptativos
    - Adaptativos por paquete
    - Adaptativos por períodos de actividad de voz
        - Basados en la Calidad de Servicio
        - No tolerantes a Pérdidas
        - Tolerantes a Pérdidas

1.6.3 Algoritmos de control de demora de reproducción fijo

En este tipo de algoritmo se calcula un límite constante para la demora de reproducción mientras dure la sesión de audio. El límite se basa en una estimación del tiempo que tarda un paquete en ir de extremo a extremo. Para optimizar este límite se valora el tiempo máximo de demora que puede existir sin afectar la calidad de servicio y así se espera que lleguen la mayor cantidad de paquetes. La demora de reproducción fija se puede estimar de tal manera que solo una pequeña parte de los paquetes que arriban se pierdan debido a llegadas tardías. Otra manera de calcular la demora es mediante el uso de paquetes de prueba, con los cuales se mide el tiempo que tardan en atravesar la red ida y vuelta, y a partir de esta medición se fija la demora óptima de reproducción. Un tiempo de espera muy pequeño, provoca que se incremente la cantidad de paquetes eliminados, ya que el tiempo de reproducción no se adapta a las condiciones de la red, disminuyendo considerablemente la interactividad de la conversación. Si el límite es demasiado alto, los paquetes que llegan en tiempo esperarían mucho en el buffer antes de ser reproducidos.

1.6.4 Algoritmos de control de demora de reproducción adaptativos

Los algoritmos de control de demora de reproducción adaptativos constituyen una mejora de los algoritmos de control de demora fijo, en ellos se ajusta de mejor forma la demora de reproducción a las condiciones cambiantes de la red utilizando diferentes enfoques, esta se actualiza al comienzo de cada talkspurt, o bien se modifica dentro de cada talkspurt por cada paquete que llega al receptor.

1.6.4.1 Algoritmos adaptativos por paquete

En los algoritmos de control de demora adaptativos por paquete se trabaja ajustando la demora de reproducción dinámicamente a la llegada de cada paquete. El ajuste se realiza mediante estimaciones de demora que se obtienen de cada uno de los paquetes, en cualquier parte de la sesión, incluyendo dentro de los talkspurt.

Dado que los paquetes pueden tener diferentes demoras de reproducción, se necesita una reducción o extensión temporal de la duración de la señal, para evitar interrupciones o traslapes durante la reproducción. Este escalamiento en la tasa de la señal de audio se alcanza por medio de la modificación en la escala del tiempo basada en el algoritmo Solapamiento de Formas de Ondas Similares (WSOLA[8]).

1.6.4.2 Algoritmos adaptativos por talkspurt

Durante la sesión de audio, los algoritmos adaptativos por talkspurt detectan las variaciones en la demora de extremo a extremo y se adaptan a las condiciones de la red. En ellos se modifica la demora de reproducción de los paquetes al inicio de cada talkspurt, por tanto los cambios en el tiempo de reproducción se ajustan en los períodos de silencio. En estos algoritmos se toman muestras de la demora usando comúnmente estampas de tiempo RTP tanto del emisor como del receptor para estimar un promedio de la demora de extremo a extremo.

I. Algoritmos basados en la calidad de servicio

Una de las técnicas que se utiliza para obtener un balance entre las pérdidas y la demora es la calidad percibida por el usuario final. El algoritmo MOS-E en (Fujimoto, 2002) es una de las primeras técnicas para el control de la demora de reproducción que utiliza las medidas subjetivas de la calidad de servicio. La idea central se basa en acotar un modelo matemático con la relación entre la Nota Media de Opinión y la demora de reproducción, de manera que se encuentre el tiempo de espera en el buffer que maximice el valor de MOS**.** La solución se basa en la suposición de que la razón de pérdida de paquetes y la demora de reproducción afectan el índice MOS de manera independiente. Teniendo esto en cuenta, el

[8] Waveform Similarity Overlap-Add

MOS se expresa como una combinación de dos funciones polinomiales de demora y pérdidas. En el algoritmo MOS-E se construye un modelo de opinión simple y se usa para controlar la óptima demora de reproducción.

II. No tolerantes a pérdidas

En las aplicaciones de audio sobre Internet, las pérdidas de los paquetes deben evitarse a toda costa, incluso a expensas de que se aumente la demora de extremo a extremo, basándose en este concepto se crearon algoritmos no tolerantes a pérdidas.

- **Algoritmos basados en estimaciones auto-regresivas**

Los algoritmos de control de demora de reproducción presentados por Ramjee se basan en estimaciones auto-regresivas, en los cuales se trabaja calculando dos variables principales: la estimación de la variabilidad de la demora y la demora de extremo a extremo de cada paquete. En estos se asegura el compromiso entre las pérdidas y la demora, y se proporciona un margen en la demora de reproducción para la llegada de los paquetes, por lo que se asegura que el tiempo de espera en el buffer sea lo suficientemente grande para que una porción pequeña de estos llegue tarde. Mientras se estima la demora de manera auto-regresiva se actualiza por cada paquete y el tiempo de reproducción se inicializa al comienzo de un nuevo talkspurt (Ramjee et al., 1994).

- **Algoritmos basados en filtros adaptativos**

En este tipo de algoritmo se predice la demora y la variabilidad de la demora utilizando un filtro adaptativo Mínimos Cuadrados Normalizado (NLMS[9]) y basándose en esta predicción se determina la demora de reproducción, es necesario señalar que mientras más acerca se encuentre la predicción de la realidad, más efectivo será el ajuste de la demora de reproducción.

III. Algoritmos tolerantes a pérdidas

En las aplicaciones de VoIP se puede soportar una cantidad pequeña de paquetes perdidos, es por esto que los algoritmos de este tipo aprovechan la tolerancia a las pérdidas para optimizar el tiempo de espera de los paquetes en el buffer de reproducción. La idea principal consiste en que se mantenga un seguimiento de la distribución de la demora de los

[9] Normalized Least-Mean-Square

paquetes, y se adapte el tiempo de reproducción en un histograma basado en la tasa de pérdidas tolerada para la sesión. En estos algoritmos se calcula la demora de reproducción por cada nuevo talkspurt y se obtiene la demora actual por cada punto porcentual dado en la función de distribución de las demoras. En la función de distribución se utilizan las demoras de una cantidad determinada de los últimos paquetes que llegaron para formar el histograma. La cantidad de valores de demoras que se guardan en el histograma determina qué tan sensitivo es el algoritmo para adaptarse a los cambios en las condiciones de la red.

1.7 Algoritmo propuesto por Moon

El algoritmo propuesto en (Moon et al., 1998) considera el problema de ajustar adecuadamente la demora de reproducción y mantenerla tan pequeña como sea posible, al mismo tiempo trata de evitar pérdidas excesivas debido a la llegada de paquetes al receptor después de que su tiempo de reproducción haya expirado. Este se basa en los algoritmos 1 y 4 descritos por Ramjee los cuales se usan en la teoría de estimación y control, y operan estimando estadísticas que caracterizan la demora de red incurrida por paquetes de audio. Cada una de estas estimaciones es recalculada cada vez que un nuevo paquete llega.

Los algoritmos 1 y 4 se presentan por completo en (Ramjee et al., 1994), en ambos se calcula una estimación de la variabilidad en la demora $\hat{v}_k^i$ y la demora de extremo a extremo $\hat{u}_k^i$ de un paquete ***i*** en el talkspurt ***k***.

Al principio de un nuevo talkspurt, la demora de reproducción $\hat{p}_k$ se estima como sigue:

$$\hat{p}_k = \hat{u}_{k-1}^{n_{k-1}} + \beta \hat{v}_{k-1}^{n_{k-1}} \qquad \textbf{(2)}$$

Aquí β es la constante que asegura el compromiso entre las pérdidas y la demora, su valor proporciona una holgura para la demora de reproducción de los paquetes que llegan, mientras más grande es el coeficiente, más paquetes se reproducen en su fin a expensas de demoras de reproducción más largas. En los algoritmos 1 y 4 se usa la Ecuación (2) para determinar la demora de reproducción para un talkspurt.

El algoritmo 1 utiliza las Ecuaciones (3) y (4), este describe un filtro lineal que es lento en alcanzar un cambio en la demora, pero es bueno en mantener un valor estable, cuando la ganancia de estimación (1 – α), se selecciona para ser muy baja.

$$\hat{u}_k^i = \alpha \hat{u}_k^{i-1} + (1-\alpha)\hat{d}_k^i \qquad \textbf{(3)}$$

$$\hat{v}_k^i = \alpha \hat{v}_k^{i-1} + (1-\alpha)\,|\,\hat{u}_k^i - \hat{d}_k^i\,| \qquad \textbf{(4)}$$

Si se supone que una señal de audio consiste en *M* talkspurts se pueden definir las siguientes variables:

- t_k^i : estampa de tiempo de emisión del paquete ***i*** en el talkspurt *k*

- a_k^i : estampa de tiempo de recepción del paquete ***i*** en el talkspurt *k*

- $\hat{d}_k^i$: intervalo de tiempo desde que el paquete ***i*** en el talkspurt *k* se genera por el emisor hasta que se recibe por el receptor, se calcula como:

$\hat{d}_k^i = a_k^i - t_k^i$, sin asumir que los relojes del emisor y el receptor están sincronizados.

El Algoritmo 4, que se muestra en la Figura 5 tiene dos modos de operación, los cuales dependen de si se descubre un pico de demora o no. En el modo normal, este opera como el Algoritmo 1 con una ganancia distinta, pero en el modo de descubrimiento de pico, el $\hat{u}_k^i$ se actualiza de forma diferente, usando la Ecuación (5).

$$\hat{u}_k^i = \hat{u}_k^{i-1} + \hat{d}_k^i - \hat{d}_k^{i-1} \qquad \textbf{(5)}$$

El algoritmo descrito por Moon colecciona estadísticas de paquetes que han llegado y las usa para estimar la demora de reproducción. El tiempo que demora en llegar cada paquete se registra y con este dato se actualiza una función de distribución de demora en cada llegada de paquete. Cuando un nuevo talkspurt comienza, en el algoritmo se calcula un punto de porcentaje q en particular en la función de distribución de las demoras para los últimos w paquetes y se usa como la demora de reproducción para el nuevo talkspurt.

Como en el Algoritmo 4, este descubre picos y se comporta en consecuencia: cuando un pico es descubierto, deja de coleccionar las demoras de los paquetes y sigue el pico hasta

que descubra el final, una vez que descubre el final, reanuda su operación normal. Las demoras de los paquetes en un pico disminuyen en forma lineal, por tanto si un nuevo talkspurt comienza durante un pico se hace razonable usar la demora del primer paquete del talkspurt como la demora de reproducción para el talkspurt.

```
IF (mode == NORMAL)
    IF (|d̂_k^i − d̂_k^{i−1}| > |v̂_k^{i−1}| * 2 + 800)
      var = 0;
      mode = SPIKE;
  ELSE
      var = var/2 + |(d̂_k^i − d̂_k^{i−1})/8 + (d̂_k^i − d̂_k^{i−2})/8|;
    IF (var ≤ 63)
      mode = NORMAL;
      d̂_k^{i−2} = d̂_k^{i−1}
      d̂_k^{i−1} = d̂_k^i
      return;
IF (mode == NORMAL)
    û_k^i = 0.125 * d̂_k^i + 0.875 * û_k^{i−1};
ELSE
    û_k^i = û_k^{i−1} + d̂_k^i − d̂_k^{i−1};
v̂_k^i = 0.125 * |d̂_k^i − û_k^i| + 0.875 * v̂_k^{i−1};
d̂_k^{i−2} = d̂_k^{i−1}
d̂_k^{i−1} = d̂_k^i
return;
```

Figura 5. Algoritmo 4

El algoritmo se presenta en pseudocódigo lenguaje-C en la Figura 6, este opera en dos modos. Para cada paquete que llega al receptor, el algoritmo comprueba el modo en curso y, si es necesario, cambia de modo en las líneas 1 - 7. En las líneas 9 - 22 se actualiza la distribución de demora en el modo normal. Si un paquete llega con una demora que es más grande que algún múltiplo de la demora de reproducción en curso, el algoritmo cambia a modo de detección de picos o modo de descubrimiento de un pico de demora. El final de un pico se descubre de un modo similar.

```
( 1) IF (mode == SPIKE)
( 2)   IF (d̂_i^k ≤ tail * old_d) /* the end of a spike */
( 3)     mode == NORMAL;
( 4) ELSE
( 5)   IF (d̂_k^i > head * p̂_k)     /* the beginning of a spike */
( 6)     mode = SPIKE;
( 7)     old_d = p̂_k;          /* save p̂_k to detect the end
                                  of a spike later */
( 8)   ELSE
( 9)     IF (delays[curr_pos] ≤ curr_delay)
(10)       count -= 1;
(11)     distr_fcn[delays[curr_pos]] -= 1;
(12)     delays[curr_pos] = d̂_k^i;
(13)     curr_pos = (curr_pos+1) % w;
(14)     distr_fcn[d̂_k^i] += 1;
(15)     IF (delays[curr_pos ] < curr_delay)
(16)       count += 1;
(17)     WHILE (count < w × q)
(18)       curr_delay += unit;
(19)       count += distr_fcn[curr_pos];
(20)     WHILE (count > w × q)
(21)       curr_delay -= unit;
(22)       count -= distr_fcn[curr_pos];
```

Figura 6. Algoritmo propuesto por Moon

Si la demora de un paquete recién llegado es menor que algún múltiplo de la demora de reproducción antes del pico en curso, el modo se regresa a estado normal. Dos parámetros heat y tail se usan en las líneas 5 y 2 para lograr el descubrimiento del principio y final de un pico.

```
(1)   IF (mode == SPIKE)
(2)     p̂_k = d̂_k^1;
(3)   ELSE (mode == NORMAL)
(4)     p̂_k = curr_delay;
```

Figura 7. Estimación de la demora de reproducción

Según el modo en curso, la demora de reproducción para el siguiente talkspurt se estima de forma diferente como se muestra en la Figura 7. En el modo de descubrimiento de pico, la

demora del primer paquete de un talkspurt se toma como la demora de reproducción estimada para el talkspurt. En el modo normal se utiliza **curr_delay** que es un punto de porcentaje de demora dado basado en las estadísticas previas de las demoras de paquetes.

1.8 Conclusiones del Capítulo

En el contenido presentado en el Capítulo1 se hace una descripción de las principales características de los sistemas de Voz sobre redes IP, en particular el procesamiento de la señal de audio, los factores que afectan la calidad de servicio y algoritmos de control de demora de reproducción. También se realiza una explicación del funcionamiento de los modelos para la evaluación de la calidad (PESQ, Modelo-E) y del algoritmo de reproducción para el control del jitter propuesto por Moon.

CAPÍTULO 2. Implementación del paquete de software QofIS_v2

2.1 Descripción General

En este capítulo se describe la implementación de un paquete de software de código abierto llamado QofIS versión 2, el cual permite la simulación de diferentes algoritmos de reproducción para VoIP. El paquete de software fue creado por C.Hoene y S.Wiethölter y presentado en el Quinto Taller Internacional sobre la Calidad de los Futuros Servicios de Internet en el año 2004, que se desarrolló en Barcelona, España.

El QofIS_v2 está disponible en su código fuente en la página web del Instituto Tecnológico de Berlín (Hoene y Wiethölter, 2004a), este se creó para ser utilizado sobre plataformas Linux y está compuesto por un software nombrado Playout y un conjunto de ficheros los cuales permiten la interacción entre Playout y el software Network Simulator (NS).

En el software Playout se incluyen los algoritmos de reproducción descritos por Moon y Ramjee, dos de los esquemas más utilizados para codificar voz (G.711 y G.729), y dos modelos de calidad de la Unión Internacional de Telecomunicaciones PESQ y Modelo-E, los cuales se utilizan para predecir la calidad perceptual de la transmisión de extremo a extremo (Hoene y Wiethölter, 2004b).

Para su implementación se utilizó una Computadora Personal, con un microprocesador Celeron ® CPU 2.66Ghz, 256MB de memoria RAM DDR1, 80GB de disco duro y el sistema operativo Debian GNU/ Linux 5.0.0 _Lenny_- Official i386.

Las direcciones de los repositorios utilizados por el sistema operativo se muestran a continuación:

- deb http://debian.mes.edu.cu/debian/ sid main contrib non-free
- # deb http://10.12.1.5/debian/ lenny main contrib non-free
- # deb http://10.12.1.8/debian/ etch main contrib non-free

Estas permiten instalar actualizaciones, librerías y paquetes de software necesarios para realizar la implementación.

Dentro de los software utilizados para lograr la instalación y el funcionamiento de QofIS_v2 se encuentran el Sox versión 14.2.0, Eclipse Ganymede, el NS en su versión 2.33 y los códigos fuentes de los software PESQ y G.729 los cuales están estandarizados por la UIT. Estos últimos a pesar de que son esenciales para el funcionamiento de QofIS_v2 no se brindan en el paquete debido a que no son software de distribución libre (Hoene et al., 2004).

2.2 Sox

El software Sox versión 14.2.0 es una herramienta multiplataforma que trabaja a base de líneas de comandos y permite realizar conversiones entre distintos formatos de audio, aplicar filtros, añadir efectos especiales a los ficheros de sonido, modificar características de un archivo de sonido como la tasa de muestreo o el tamaño de la muestra, ajustar la velocidad, concatenarlo con otro o mezclar dos ficheros en uno.

El programa no presenta interfaz gráfica y la relación de formatos que puede procesar es extensa, entre ellos se encuentran Ogg Vorbis, MP3, WAV, AIFF, VOC, SND, AU, GSM y varios más. El software se apoya en diferentes librerías las cuales le proporcionan paquetes de datos básicos para su funcionamiento. La librería Libsox1 contiene los paquetes de datos necesarios para habilitar la conversión a diferentes formatos, se encarga del procesamiento y de los efectos que se pueden aplicar a los diferentes archivos de audio. En la librería Libsox-fmt-base se encuentran los formatos básicos de entrada y salida del software, es importante destacar que cualquier soporte de formato proporcionado por Sox requiere al menos libsox-fmt-base, algunos formatos tienen su propio paquete como por ejemplo el proceso de leer un archivo mp3 es soportado por la librería libsox-fmt-mp3.

La instalación del Sox, en este trabajo, se realizó a través del software Gestor de Paquetes Synaptic que presenta Linux el cual se utiliza en el sistema operativo para realizar la instalación de diferentes programas y librerías que se encuentran distribuidos en los repositorios. Sox también se puede instalar colocando en el terminal de Linux el comando apt-get install sox, el cual es un método fácil de ejecutar y solo requiere privilegios de superusuario (root).

Cuando Sox se emplea para cambiar de formato un fichero de audio, se escribe el comando cd (cambiar directorio) en el terminal para direccionar dicho fichero, después se llama al software con el comando sox y se le pasan los parámetros necesarios para la conversión. Una representación general se muestra a continuación:

sox <fichero de audio de entrada> <opciones> <fichero de audio de salida. formato>

Por ejemplo si el fichero de audio XXX.wav se quiere convertir en un fichero de extensión sw entonces se debe escribir en el terminal el siguiente comando:

debian: /home/linux/Desktop# sox XXX.wav -r 8000 -c 1 XXX.sw

donde el 8000 corresponde a la frecuencia de muestreo en Hz y el 1 significa la cantidad de canales de audio que presenta el fichero resultante.

2.3 Eclipse Ganymede

El Eclipse es un software multiplataforma con un Entorno Integrado de Desarrollo (IDE) que permite la programación y compilación de componentes para aplicaciones Java, sitios webs y programas basados en C/C++, al cual se le puede añadir nuevas funcionalidades a través de módulos (plugins), para programar en diferentes lenguajes.

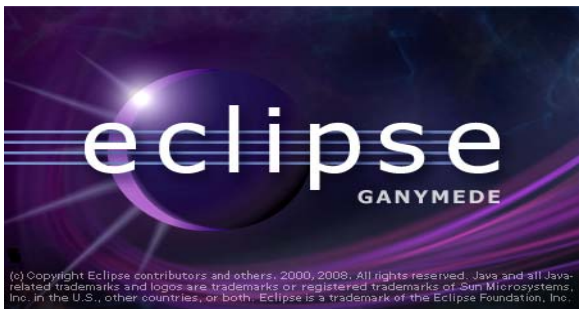

Figura 8. Logotipo del Eclipse Ganymede

La versión Ganymede se encuentra disponible en Internet (Foundation, 2008), esta presenta una serie de mejoras con respecto a versiones anteriores, dentro de las cuales se destacan un aumento en la robustez y rendimiento del software. En esta versión el compilador se hizo más funcional y eficiente, y aunque la interfaz del programa no varió sustancialmente, se realizaron algunos cambios que permiten una mejor utilización del software. El Eclipse requiere para su funcionamiento la librería Java Runtime Environment (JRE), la cual en este trabajo se instaló a través del Gestor de Paquetes Synaptic utilizando las direcciones de los repositorios anteriormente mencionados.

2.4 NS-2.33

La simulación, en las investigaciones sobre redes de computadoras, es una técnica donde un programa modela el comportamiento de una red calculando o estimando la interacción entre las diferentes entidades que la componen (computadoras, ruteadores, enlaces para transmisión de datos, paquetes, etc.) utilizando fórmulas matemáticas o modelando a través de capturas reales y observaciones de la distribución de tráfico que se presenta en diferentes redes.

El NS es un software de código abierto que se utiliza para simular redes, el cual fue construido en C++ y permite programar ejemplos de redes cableadas o inalámbricas, este soporta los protocolos IPv4, UDP, TCP, RTP y permite que el usuario describa una topología de red con parámetros específicos y luego el programa principal de NS simula esa topología.

El código fuente de la versión allinone-2.33, la cual presenta todos los componentes necesarios se puede descargar en Internet (Berkeley et al., 2002).

Para lograr su funcionamiento es necesario instalar algunas librerías específicas, las cuales se mencionan a continuación: gcc-2.95, g++-2.95, libstdc++2.10-glibc2.2, libx11-dev, libxt-dev.

El proceso de instalación de las librerías se realiza escribiendo en el terminal apt-get install <nombre de la librería>.

Generalmente cuando el software se descarga de Internet aparece compactado, para lograr su instalación se descompacta y se direcciona la carpeta ns 2.33 utilizando el comando cd y

el terminal, después se ejecuta el comando **./install** y por último **validate**, esto permite compilar y obtener el ejecutable del software.

Después de terminada la instalación, en este trabajo, se realizó un enlace simbólico entre el ejecutable y la dirección usr/local/bin con el objetivo de mejorar la utilización del software. Para realizar el enlace simbólico se utilizó el software MC que se puede encontrar en las direcciones de los repositorios.

Para instalar el NS existe bibliografía en Internet. El proceso de instalación utilizado en este trabajo se basó en los pasos descritos por Miller en su página web, los cuales a pesar de que son para la versión 2.26 se ajustan perfectamente a la 2.33 (Miller, 2006).

2.5 Playout

El software Playout constituye la herramienta fundamental dentro del paquete QofIS, este codifica muestras de voz aplicadas a un fichero de trazas de paquetes de VoIP, simula múltiples algoritmos de reproducción y finalmente nos proporciona la calidad del servicio telefónico (pérdida de paquetes, demora de transmisión e información del algoritmo de reproducción utilizado). Así, se puede determinar la razón de pérdida de paquetes final, la demora de transmisión media y la calidad de la conversación.

El software cubre completamente la cadena de procesamiento de VoIP, este permite:

- Codificar muestras de audio con G.711 y G.729.
- Análisis sintáctico de ficheros de trazas de paquetes generados por los software Snuffle (Hoene, 2001) o NS-2.33 permitiendo recopilar datos sobre demoras de paquetes o pérdidas.
- Generar ficheros de trazas artificialmente para estudiar el impacto de los picos de demora.
- Simular los algoritmos de reproducción descritos por Moon, Ramjee y otros.
- Decodificar un fichero de audio y reestructurar el tiempo de reproducción.
- Valorar la calidad usando el Modelo-E y PESQ.

El software Playout permite leer ficheros de trazas de paquetes que contienen demoras, cuando este modo de operación se ejecuta el software basa su funcionamiento en la siguiente figura.

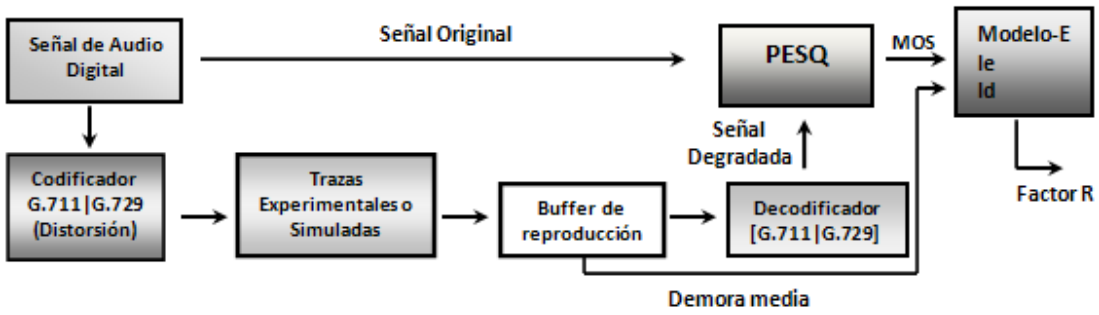

Figura 9. Descripción del funcionamiento del software Playout

Primero la señal de voz digital se comprime utilizando los codificadores G.711 o G.729. Posteriormente la señal de audio comprimida se combina con trazas de paquetes de VoIP simuladas o experimentales que presentan demoras, en este paso se debe agregar una demora del sistema debido al procesamiento introducido por los sistemas operativos, codificadores, etc. Después los paquetes de audio pasan al buffer de reproducción donde el software Playout tiene implementados algoritmos de reproducción fijo y adaptativos descritos por Van Jacobsen, Mills, Schulzrinne, Ramjee y Moon. En este bloque se estiman los tiempos de reproducción de los paquetes y la demora de transmisión media, solo se consideran los paquetes que coinciden con la actividad de voz, esto se debe a que durante los períodos de silencio las personas no identifican la demora de transmisión. En el siguiente paso el decodificador descomprime la señal de voz y genera un fichero, el cual contiene la señal de voz afectada por pérdidas y demora que se conoce como señal degradada, aquí el decodificador puede ocultar paquetes perdidos debido a llegadas tardías o errores de trasmisión. Por último se calcula la calidad a través de un valor MOS proporcionado por PESQ a partir de la muestra de audio degradada y el fichero original, después la demora de transmisión media y el valor de calidad MOS se utilizan por el Modelo-E para determinar el valor R (Hoene, 2004).

Es importante señalar que el buffer del receptor adapta el tiempo de reproducción durante la transmisión y esta replanificación podría dañar la calidad de la señal de audio debido a discontinuidades temporales. Sin embargo, el Modelo-E no tiene en cuenta la dinámica de una transmisión, pero confía en parámetros de transmisión estáticos. El PESQ en cambio considera la adaptación de reproducción, pero no incluye el retraso absoluto en su algoritmo de posición.

Para vencer estas dificultades en el software Playout se combinan ambos modelos. El PESQ estima la calidad de la voz y proporciona un valor MOS al Modelo-E. El proceso de combinar ambos modelos matemáticamente se describe en (Hoene et al., 2003).

Cuando el software Playout se ejecuta, este realiza varias corridas en dependencia de los algoritmos de reproducción seleccionados y obtiene varios resultados, estos se mencionan a continuación:

- La salida del terminal o consola contiene bastante información sobre el proceso de compresión, algoritmo de reproducción y valoración de la calidad. Con el propósito de verificación puede ser salvada.
- Un fichero "playout_result.txt" que presenta los resultados de la simulación del algoritmo de reproducción escogido, los cuales están formados por la demora de transmisión media, el valor MOS (entre 1 y 5) proporcionado por PESQ, el factor R (entre 0 y 100) que brinda el Modelo-E, el fichero de audio producido (si alguno), la razón de pérdidas (entre 0-1) y el número de paquetes perdidos debido a las pérdidas de la red.
- Un par de ficheros "resultXXX.sw" que presentan el contenido de audio transmitido, incluso con la distorsión debido a la codificación, pérdida de paquetes y esquema de reproducción. Ellos se pueden utilizar para verificaciones auditivas.

2.5.1 Modificaciones en el código fuente de Playout

En el software Playout se realizaron algunas modificaciones con el objetivo de garantizar el funcionamiento del paquete de software QofIS_v2, esto es importante para lograr la compatibilidad de su código fuente con los distintos programas que no se proporcionan en el paquete. Playout durante su funcionamiento debe leer unos resultados que se encuentran en un fichero nombrado "pesq_itu_result.txt" generado por el software PESQ. Como se ha mencionado anteriormente en este trabajo se utiliza una versión de prueba descargada de la UIT (la cual no es la misma que utilizó el fabricante) y debido a esto fue necesario modificar el código fuente original escrito en C++ del fichero pesq.cpp de Playout para garantizar que el proceso de lectura pueda ser realizado, de esta forma el valor de la variable MOS no se lee en una posición errónea y así no ocurre que se detenga la ejecución del programa.

La solución proporcionada a este problema consiste en la recolocación del segmento de código ***&mos*** en una posición que le permita a Playout leer el valor necesario de "pesq_itu_result.txt" y debido a esto cambiar el valor con el cual se compara la variable ***res***, como se muestra en la Figura 10. De esta forma Playout lee el valor MOS generado por PESQ y continúa con el cálculo del valor R que se utiliza en el Modelo-E.

pesq.cpp

```
/* read pesq result file */
fhd3 = fopen(pesqres, "r");
if(fhd3 == NULL) {
fprintf(stderr,"Could not find file %s\n%s\n", pesqres, strerror(errno));
exit(3);
}
printf("valor: \n%d \n", res);
res =  fscanf(fhd3, "%255s%255s%255s%255s%255s%255s", dummy,
dummy,dummy, dummy, dummy, dummy);
printf("valor: \n%d \n", res);
res += fscanf(fhd3, "%255s%255s%255s%255s%255s", dummy,dummy,
dummy, dummy, dummy);
printf("valor: \n%d \n", res);
res += fscanf(fhd3,
"%lf%255s%255s%255s%lf%lf%255s%lf",&mos,dummy,dummy,dummy,
&subjmos,&cond,dummy,&crudedelay);
printf("valor: \n%d \n", res);
if(res < 16) {
fprintf(stderr, "coult not read pesq itu result file\n%s", strerror(errno));
exit(3);
}
```

Figura 10. Modificación en el código fuente del fichero pesq.cpp

2.5.2 Instalación

Para lograr la implementación del software Playout se necesita la instalación de los compiladores g++ y gcc, en este trabajo se utilizó la versión 4.1 la cual fue instalada de los repositorios utilizados por el sistema operativo. Una vez instalados los compiladores se utiliza el terminal para direccionar la carpeta playout presente dentro del paquete de software y como root se escriben los siguientes comandos:

debian: /home/linux/Desktop/qofis_v2/playout#./configure

debian: /home/linux/Desktop/qofis_v2/playout# make

Con estos pasos se crea el ejecutable del software dentro de la carpeta nombrada playout2. En este trabajo se creó un enlace simbólico entre el ejecutable y la dirección usr/local/bin para facilitar el trabajo con el mismo.

En el código fuente de Playout también se proporciona un fichero de proyecto llamado playout2.kdevprj que permite la compilación del software a través de la herramienta Kdevelop. Es importante aclarar que esto es posible debido a que el desarrollo de Playout se realizó con esta herramienta.

2.5.3 Ejemplos de uso

Playout es un software que no presenta interfaz gráfica y se puede utilizar de diferentes formas. Una de ellas es a través del terminal escribiendo el comando playout2 y pasándole varios parámetros específicos. Una representación general se muestra a continuación:

playout2 <parámetros> <fichero fuente>

Donde el fichero fuente representa el fichero de audio que se quiere evaluar, playout2 el nombre del ejecutable del software y los parámetros se describen en la Tabla 4.

El fichero de audio de entrada requerido por el software debe estar codificado con 16 bit por muestra PCM, frecuencia de muestreo 8000 Hz, mono y almacenado con un orden de byte Little-endian (Hoene y Wiethölter, 2004b). Para convertir el fichero de audio a las condiciones de entrada requeridas se utiliza el software Sox de la siguiente manera:

sox <entrada> -r 8000 -c 1 <salida.sw>

donde el formato sw cumple con las especificaciones de entrada requeridas por Playout.

La otra forma de utilización se basa en ficheros llamados (scripts), los cuales se proporcionan en la carpeta examples que se encuentra dentro del software y se pueden modificar de acuerdo a las condiciones que se desean simular. Uno de ellos se nombra example1 el cual se ejecuta a través del terminal utilizando el comando que se muestra a continuación:

debian: /home/linux/Desktop/qofis_v2/playout/examples#**./**example1

La carpeta examples también contiene dos ficheros de trazas en formato RXSTAT (único formato soportado por Playout) y dos ficheros de audio en formato sw, esto se proporciona para que el software se utilice de primera instancia.

Tabla 4. Parámetros soportados por el software Playout

Parámetros	Descripción	Por defecto
-c [G.711|G.729]	Selecciona el esquema de codificación.	[G.711]
-t <Archivo de trazas> xxxx.out	Lee un archivo de trazas que contiene demoras. Las trazas tiene que estar en formato RXSTAT.	n/a
-T <Valor aleatorio> <No. de picos> <Alto> <Ancho>	Genera un archivo de trazas que contiene picos de demora.	n/a
-d <Demora del Sistema>	Demora de transmisión de extremo a extremo menos la demora introducida por el buffer de reproducción. (Su valor tiene que estar entre 0-1s)	0.15s
-o <Desplazamiento en la traza>	Permite que el programa deseche el principio del del archivo de trazas.	0s
-f <Grado de Desplazamiento>	Simula un buffer de reproducción fijo que usa una demora absoluta.	n/a
-F <Grado de Desplazamiento>	Simula un buffer de reproducción fijo que usa una demora relativa.	n/a
-m	Simula el algoritmo propuesto por Moon.	Ver código fuente
-r	Simula el algoritmo propuesto por Ramjee. (7 variantes diferentes)	Ver código fuente
-S	Planifica tres formas de reproducción después de alcanzado un pico de demora.	n/a

El ejemplo que se muestra a continuación nos permite simular la cadena de transmisión completa de VoIP y a la vez valorar la calidad. Para poder ejecutarlo se debe direccionar el lugar donde se encuentra el fichero de trazas stations10_rxstats_w8_wl8.out y el fichero de audio 3-a_f01s13.sw.

playout2 -c G711 -t stations10_rxstats_w8_wl8.out -d 0.150 -f 0.01 -m -r 3-a_f01s13.sw

En dicho ejemplo se asume una demora del sistema de 150 ms la cual representa la demora de transmisión de extremo a extremo menos la demora introducida por el buffer de reproducción. Se elige un buffer de longitud fija con una demora absoluta de 10 ms y se escogen los algoritmos de Moon y Ramjee. El resultado se proporciona en la carpeta donde

se direccionó para la simulación y está compuesto principalmente por el fichero "playout_result.txt" y los ficheros resultantes de audio.

El programa también puede generar trazas de paquetes de VoIP que estén afectadas por el impacto de picos de demora. En el siguiente ejemplo se programa la presencia de cinco picos de demoras uniformes en el archivo de audio, cada uno con una anchura de 200 ms y una altura de 100 ms:

playout2 -c G711 -T $RANDOM, 5, 0.1, 0.2 -d 0.150 -S 3-a_f01s13.sw

La posición de los picos de demora se obliga a que estén en un área donde exista actividad de voz dentro del fichero de audio (ningún pico durante el silencio) y depende del primer parámetro, el cual es un valor de distribución aleatorio escogido por la función $RANDOM.

El parámetro -S programa la reproducción con tres algoritmos diferentes. Primero, se dejan perder todos los paquetes que son retrasados por el pico. Segundo, se retrasa la reproducción de acuerdo a la altura del pico. Último, se retrasa la reproducción de acuerdo a la altura del pico, pero se reprograma el tiempo de reproducción para el comienzo del próximo talkspurt.

2.5.4 Implementación del software PESQ

Para lograr la implementación del software PESQ se descargó su código fuente de las recomendaciones de la UIT (ITU-T_P.862, 2001b) la cual brinda una versión de prueba para experimentos, esto fue necesario debido a que el software no se proporciona en el QofIS_v2 debido a que su distribución no es libre. Para lograr la compilación del código fuente y así poder obtener el ejecutable que necesita Playout se utilizó el Eclipse, este permitió crear un nuevo proyecto en C/C++ y adicionar los ficheros que conforman el código fuente de PESQ:

Dsp.c, Dsp.h, Pesq.h, Pesqdsp.c, Pesqio.c, Pesqmain.c, Pesqmod.c, Pesqpar.h.

A continuación se brinda una descripción de los pasos necesarios a seguir:

- Se crea un nuevo proyecto en la opción C/C ++ Proyect del Eclipse y se nombra pesq.
- Se adicionan los ficheros descargados que conforman el código fuente.

- Una vez todos los ficheros adicionados se construye el proyecto y el ejecutable se obtiene en la carpeta pesq donde se creó el proyecto.

En este trabajo se le realizó al ejecutable un enlace simbólico a la dirección usr/local/bin con el objetivo de que pueda ser utilizado por Playout.

2.5.5 Implementación del software G.729

El software G.729 no se proporciona dentro del paquete QofIS debido a que se encuentra protegido por derechos de autor y su distribución no es libre, por esto se descargó de la UIT (ITU-T_G.729, 2007), en la cual se brinda el código fuente de una distribución solo para pruebas. Dentro de todos los anexos que aparecen solo se pueden utilizar los que presentan Detección de Actividad de Voz (VAD), esto se debe a que el software Playout utiliza esta función. El escogido para este trabajo fue el Anexo B, específicamente el nombrado c_codeB.

La instalación del software se realiza a través del terminal y direccionando la carpeta c_codeB, una vez dentro de ella se escribe el comando make acompañado de la función –f como se muestra a continuación:

debian: /home/Linux/Desktop/c_codeB#make –f coder.mak

debian: /home/Linux/Desktop/c_codeB#make –f decoder.mak

De esta forma se crean dos ejecutables los cuales corresponden uno al codificador y otro al decodificador, a estos se les debe cambiar el nombre y colocarles g729encoder y g729decoder respectivamente con el objetivo de lograr la compatibilidad con Playout, después en este trabajo se les realizó un enlace simbólico a la dirección usr/local/bin para facilitar el trabajo con los mismos.

2.6 Simulaciones de VoIP con NS-2.33

El software Playout puede utilizar trazas de paquetes de VoIP generadas por el NS-2.33, para lograr esto se brinda en el paquete de software QofIS_v2 unos ficheros llamados fuentes de NS compuestos por una aplicación de VoIP, un nuevo agente RTP y un ejemplo de red programado en lenguaje TCL.

Para obtener el paquete de trazas apropiado para evaluar el software Playout, se conducen las simulaciones mediante el Cifrado Estándar de Datos (DES) proporcionando datos predefinidos a la instancia de simulación.

El desarrollo de un modelo de simulación para NS-2.33 fue necesario debido a que este todavía no soporta el procesamiento de flujo de bit (bit stream) o trazas de paquetes de VoIP y los estándares de trazas del NS no incluyen la información suficiente que necesita el software Playout.

2.6.1 Aplicación de VoIP

Durante la simulación, la aplicación trabaja con el archivo de flujo de bit en el lado emisor y construye paquetes de audio según las configuraciones predefinidas (intervalo de generación y tamaño de paquete). En el lado receptor, la aplicación guarda la pista de los paquetes recibidos y obtiene información sobre el tiempo de generación, tiempo de llegada, número de secuencia, versión y banderas, y todo esto es imprimido en el fichero de trazas de salida. Este fichero se puede proporcionar al software Playout que finalmente tasa la calidad del flujo de VoIP. La aplicación puede representar a un remitente o a un proceso de recepción así como una combinación de ambos. El código fuente de la aplicación de VoIP se encuentra en el fichero voip_traffic.cc.

2.6.2 Agente RTP avanzado

El agente RTP estándar del NS es sólo capaz de generar paquetes por sí mismo, por esta causa se proporciona un nuevo agente RTP el cual constituye una instancia que maneja los paquetes de la aplicación de VoIP hacia abajo. Básicamente, se introducen interfaces en el proceso de aplicación descrito. La implementación se incluye en rtpdata.cc /.h.

2.6.3 Modificaciones para la instalación de las fuentes de NS

Para poder generar ficheros de trazas de paquetes de VoIP que puedan ser utilizados por Playout, primero se deben instalar las fuentes proporcionadas por el QofIS en el software NS-2.33. La instalación requiere varios pasos los cuales se describen detalladamente en el fichero readme.txt de las fuentes y dos cambios que se mencionan a continuación.

El primero se realiza en el fichero voip_traffic.cc presente en las fuentes de NS y que describe la aplicación de VoIP. En el código original del fichero se debe quitar el nombre

de la clase VOIP_Traffic presente en la definición de la función command, esto se debe a que la definición de la función no lleva el nombre de la clase y de lo contrario provoca un error. En la figura siguiente se muestra el código fuente corregido.

voip_traffic.cc

VOIP_Traffic class

```
class VOIP_Traffic : public TrafficGenerator {
public:
    VOIP_Traffic();
 //read in of tcl-commands
int  command(int argc, const char*const* argv);
```

Figura 11. Código fuente corregido del archivo voip_traffic.cc

El segundo cambio que se debe realizar consiste en la modificación del fichero Makefile.in el cual se encuentra dentro del software NS-2.33, aquí se debe colocar en INCLUDES la línea de código siguiente -I/usr/include/g++-3/ \ .

Makefile.in

```
INCLUDES = \
    -I. @V_INCLUDE_X11@ \
    -I. \
    @V_INCLUDES@ \
    -I./tcp -I./sctp -I./common -I./link -I./queue \
    -I./adc -I./apps -I./mac -I./mobile -I./trace \
    -I./routing -I./tools -I./classifier -I./mcast \
    -I./diffusion3/lib/main -I./diffusion3/lib \
    -I./diffusion3/lib/nr -I./diffusion3/ns \
    -I./diffusion3/filter_core -I./asim/ -I./qs \
    -I./diffserv -I./satellite \
    -I/usr/include/g++-3/ \
    -I./wpan
```

Figura 12. Modificación del archivo Makefile.in

Esto se hace con el objetivo de incluir una ruta de acceso hacia las librerías que se encuentran en la dirección usr/include/g++-3 las cuales son necesarias para el

reconocimiento de funciones que se encuentran definidas dentro de dichas librerías y que se verifican en el proceso de instalación de las fuentes de NS.

2.6.4 Generación de ficheros de trazas

Después de instaladas las fuentes correctamente se procede a generar trazas utilizando el NS. Para realizar esto se necesitan tres ficheros los cuales se mencionan a continuación:

- 1VOIP_802_11.tcl: ejemplo de red proporcionado en las fuentes de NS.
- in.bit: archivo de audio en formato de cadena binaria.
- in.vuv: archivo de audio.

El archivo de audio en formato de cadena binaria se puede crear a través del codificador G.729 Anexo B, en una de sus dos variantes. La escogida para el trabajo fue el c_codeBA el cual se instaló como se describe en la sección 2.5.5 y se utiliza como se muestra a continuación:

Usage: **./**coder <archivo de audio.formato> <archivo de audio.bit> < bandera VAD>

Por ejemplo si el archivo de audio XXX.sw se quiere convertir en una cadena binaria se direcciona el ejecutable del codificador y se escribe el siguiente comando:

debian: /home/linux/Desktop/c_codeBA# **./**coder XXX.sw XXX.bit 0

Aquí hay que señalar que el ejecutable del codificador y el fichero de audio XXX.sw tienen que estar en el mismo lugar, que la bandera de VAD siempre tiene que estar desactivada porque de lo contrario altera los datos predefinidos que se le entregan a la instancia de simulación y por último una vez obtenido el archivo XXX.bit se le cambia el nombre por in.bit.

El archivo in.vuv se obtiene cambiándole el nombre y la extensión al fichero de audio.

Después utilizando el terminal se direcciona el lugar que contiene los tres ficheros y se escribe el comando siguiente:

debian: /home/linux/Desktop/gener-trazas# ns 1VOIP_802_11.tcl

Cuando el ejemplo de red 1VOIP_802_11.tcl se ejecuta uno o dos ficheros de trazas se generan en dependencia de la configuración que se le asigne a su código fuente. Esto es posible debido a que la aplicación puede representar a un emisor, receptor o una combinación entre ambos.

2.7 Conclusiones del Capítulo

En este capítulo se describe el proceso de implementación del paquete de software de código abierto QofIS_v2, la compilación e instalación de los software PESQ y G.729, el proceso de obtención de ficheros de trazas de paquetes de VoIP utilizando NS-2.33, así como la forma de utilización del paquete de software y las diferentes modificaciones que se realizaron para garantizar su funcionamiento.

La implementación del QofIS constituye un mecanismo de gran importancia para evaluar la cadena de transmisión completa de VoIP, este permite simular diferentes esquemas de reproducción como el descrito por Moon y probarlo bajo diferentes condiciones, por lo que resulta de gran utilidad para determinar la calidad tanto objetiva como subjetiva con que responde este tipo de algoritmo.

CAPÍTULO 3. Resultados de los Experimentos

En este capítulo se analizan los resultados obtenidos de la evaluación del algoritmo de reproducción para el control de jitter propuesto por Moon bajo condiciones específicas de trabajo. La implementación del paquete de software de código abierto QofIS_v2 permite obtener los resultados que reflejan el comportamiento del algoritmo ante ficheros de trazas simuladas generadas por el NS las cuales contienen diferentes distribuciones de jitter, capturas de trazas reales de paquetes de VoIP, variación de parámetros específicos del algoritmo, y diferentes situaciones de picos de demoras. Las gráficas que se muestran contienen los valores de los índices de calidad resultantes de las simulaciones (MOS y R) pertenecientes a los modelos de calidad (PESQ y Modelo-E) respectivamente, y permiten observar el comportamiento del algoritmo en los diferentes ambientes.

3.1 Evaluación del algoritmo con trazas simuladas generadas por el NS

El algoritmo propuesto por Moon se evaluó con trazas simuladas generadas por el NS las cuales se obtuvieron como resultado de la implementación de una red programada en lenguaje TCL. La red se conformó con cinco nodos alámbricos, un ruteador, una estación base y cuatro nodos inalámbricos como se muestra en la Figura 13.

El ejemplo de red se programó para simular cuatro llamadas bidireccionales que ocurren al mismo tiempo entre los teléfonos móviles y los fijos utilizando diferentes distribuciones de jitter generadas en la fuente. En cada llamada se establece una conexión que proporciona como resultado ficheros de trazas de salida correspondientes a los paquetes de voz que van del nodo alámbrico al inalámbrico y viceversa.

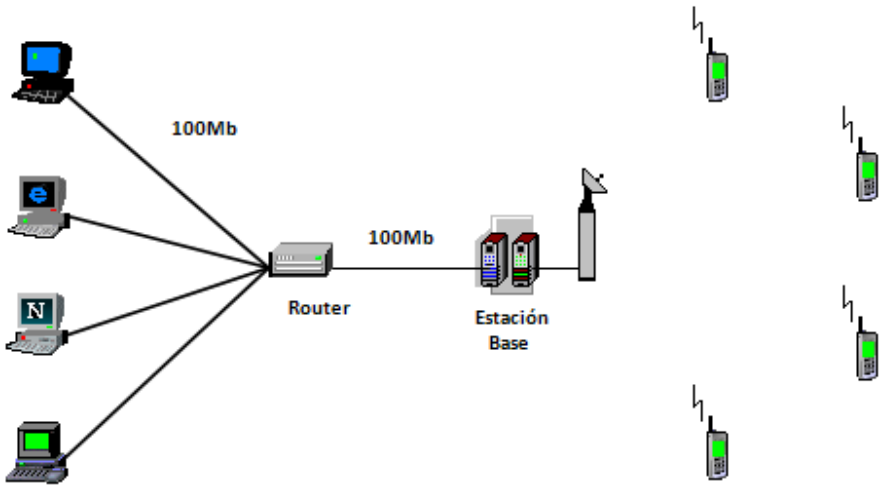

Figura 13. Arquitectura de la red

Las distribuciones de jitter pertenecientes a cada conexión se establecieron de dos tipos:

- Uniforme [mín. - máx.]: representa una distribución uniforme de un valor mínimo a un máximo.
- Exponencial [x]: representa una distribución exponencial con un valor promedio x.

Como resultado de las simulaciones con NS se obtuvieron los ficheros de trazas equivalentes a tres grupos de valores de jitter utilizados para cada conexión los cuales se muestran a continuación:

Tabla 5. Valores de las distribuciones de jitter

Tipo de Distrib.	Trazas Simul. 1	Trazas Simul. 2	Trazas Simul. 3
Distrib. Unifor	0 - 0.03	0 - 0.04	0 - 0.05
Distrib. Expon	0.01	0.03	0.04
Distrib. Expon	0.04	0.05	0.06
Distrib. Unifor	0 - 0.02	0 - 0.03	0 - 0.04

El algoritmo de Moon se simuló con los tres ficheros de trazas utilizando los codificadores G.711 y G.729, un fichero de audio de aproximadamente 39 segundos y una demora del sistema de 40 ms.

Los índices de calidad MOS y R resultantes de las simulaciones se muestran en la siguiente tabla:

Tabla 6. Índices de calidad resultantes de las simulaciones

	Trazas Simuladas 1		Trazas Simuladas 2		Trazas Simuladas 3	
	G.711	G.729	G.711	G.729	G.711	G.729
MOS	3.87148	3.65631	3.72028	3.53089	3.58736	3.42259
R	76.5838	70.2996	72.7107	66.5473	69.9517	64.9398

En la Figura 14 se grafican los índices de calidad MOS obtenidos por PESQ para las trazas simuladas. Los mejores resultados, como se esperaba, se presentaron para las trazas simuladas 1 utilizando el codificador G.711 mientras que los valores más bajos se obtuvieron para las trazas simuladas 3 usando G.729 ya que el codificador G.711 presenta menor demora de compresión y mayor tasa de calidad. En el primer caso el valor obtenido fue de 3.87 lo cual en la escala de MOS se considera como calidad media mientras que para el segundo caso el valor fue de 3.42 lo cual se califica como calidad baja.

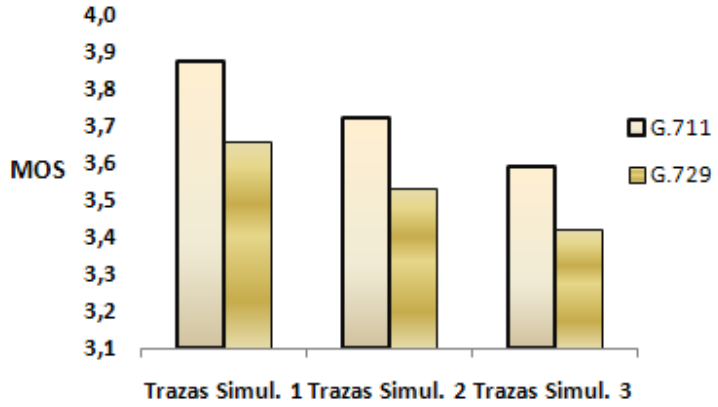

Figura 14. Valor MOS obtenido en las simulaciones

En la Figura 15 se muestran los valores R proporcionados por el Modelo-E los cuales corresponden a las simulaciones. Se observa que para las trazas simuladas 1 usando ambos codificadores se obtienen resultados considerados según la escala del factor R como calidad

media, mientras que para las trazas 3 utilizando G.729 los resultados se clasifican como calidad baja.

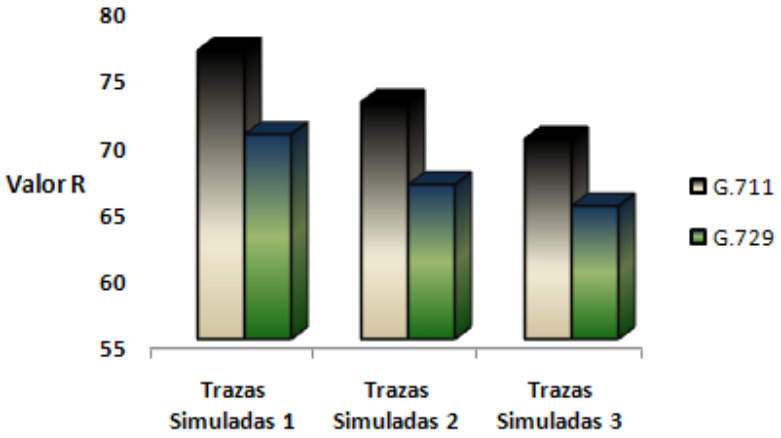

Figura 15. Factor R obtenido en las simulaciones

3.2 Evaluación del algoritmo con trazas reales

La simulación con trazas reales permite conocer cómo responde el algoritmo de Moon a las condiciones existentes en la red donde las trazas fueron capturadas, de esta forma se puede determinar la calidad perceptual de la transmisión de extremo a extremo que resulta de la evaluación del algoritmo.

Los experimentos que permitieron capturar las trazas se realizaron en la Facultad de Ingeniería Eléctrica de la Universidad Central Marta Abreu de Las Villas. Las herramientas que se utilizaron fueron el Softphone Ekiga presente en Linux que permitió establecer la comunicación de voz y el analizador de tráfico de red Wireshark, al cual en un primer paso se le configuró el filtro de captura para obtener los paquetes RTP generados por la comunicación y luego se le modificó el filtro de display para proporcionar los datos de las trazas necesarios para la simulación, de esta forma se capturaron tres ficheros de trazas pertenecientes a tres llamadas. Los tres ficheros de trazas se convirtieron a formato RXSTAT con el objetivo de que fueran compatibles con el formato de ficheros de trazas de entrada que soporta QofIS_v2.

El algoritmo de reproducción para el control del jitter propuesto por Moon se simuló con los tres ficheros de trazas reales utilizando los codificadores (G.711, G.729), un fichero de audio de aproximadamente 29 segundos y una demora del sistema de 50 ms.

Los índices de calidad MOS y R resultantes de las simulaciones se muestran a continuación:

Tabla 7. Índices de calidad resultantes de las simulaciones con trazas reales

	Trazas Reales 1		Trazas Reales 2		Trazas Reales 3	
	G.711	G.729	G.711	G.729	G.711	G.729
MOS	3.94019	3.68775	3.95909	3.70274	3.81722	3.57304
R	77.9060	71.5198	78.4629	72.9270	74.8201	69.4196

Los resultados son satisfactorios en general por lo que el algoritmo de Moon se desempeña exitosamente bajo estas condiciones.

En la Figura 16 se observa que los índices de calidad MOS obtenidos por PESQ para el codificador G.711 están en el rango de 3.8 a 4.0, estos valores se aproximan en las notas de degradación de MOS a una degradación audible pero no molesta lo cual no requiere un esfuerzo apreciable para la audición, mientras que para el G.729 evidentemente los valores son inferiores pero calificados de aceptables.

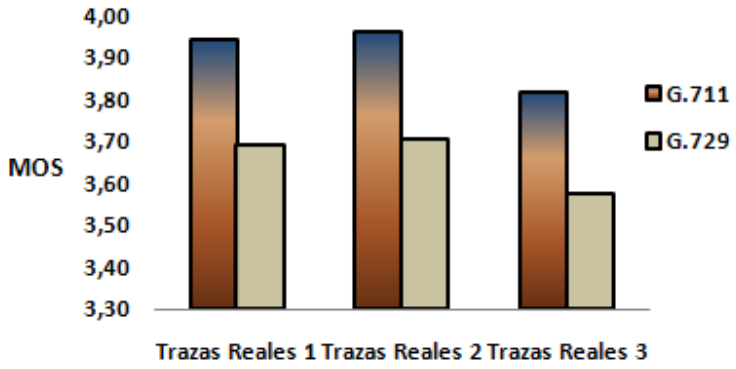

Figura 16. Valor MOS obtenido en las simulaciones

En la Figura 17 se grafican los valores de R proporcionados por el Modelo-E para las simulaciones con los tres ficheros de trazas. En ella se observa que el desempeño del algoritmo para los dos codificadores es satisfactorio, los valores para G.711 se acercan a 80 lo cual en la escala del factor R equivale a una calidad alta mientras que los valores para G.729 se aproximan a 70 que se considera como calidad media.

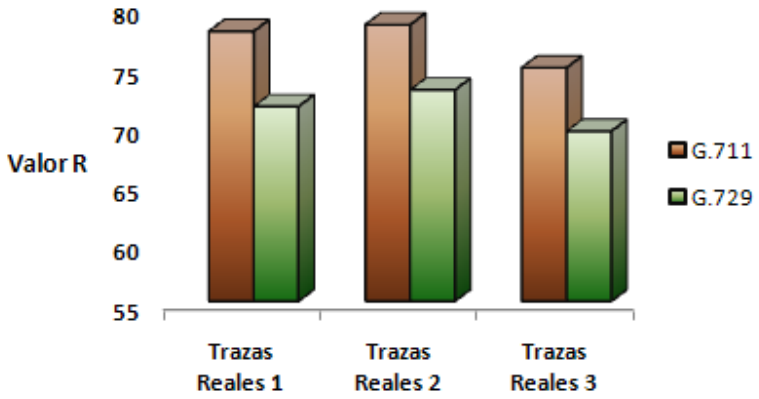

Figura 17: Factor R obtenido en las simulaciones.

3.3 Respuesta del algoritmo a picos de demoras

El paquete de software QofIS_v2 permite generar diferentes situaciones de picos de demoras y simular el comportamiento de varios algoritmos cuando se enfrentan a ellas. En este trabajo conjuntamente con el algoritmo de Moon se evaluó un algoritmo de buffer Fijo que deja perder todos los paquetes retrasados por el pico, esto se hizo con el objetivo de comparar el desempeño entre ambos y así conocer los valores de calidad resultantes para estas condiciones. Los dos algoritmos se sometieron a tres simulaciones correspondientes a 3, 5 y 7 picos de demoras de un alto de 100 ms y ancho de 200 ms utilizando los codificadores (G.711, G.729), una demora del sistema de 40 ms y un audio de 39 segundos. Los índices de calidad MOS y R resultantes de las simulaciones se muestran en la siguiente tabla:

Tabla 8. Índices de calidad resultantes de las simulaciones con picos de demoras

# de Picos de Demora (alto 100 ms, ancho 200 ms)							
		3		5		7	
		Moon	Fijo	Moon	Fijo	Moon	Fijo
G.711	MOS	3.943653	3.737363	3.903092	3.542791	3.775925	3.407807
	R	77.3478	73.0673	74.2129	68.8553	72.6746	66.0721
G.729	MOS	3.630092	3.482035	3.567804	3.338006	3.454548	3.140621
	R	70.0659	68.1419	69.2059	64.2684	66.9868	61.7859

En la Figura 18 se grafican los índices de calidad MOS obtenidos por PESQ para los dos algoritmos, se observa que a medida que aumentan los picos de demoras disminuye el desempeño de ambos y que la disminución más brusca ocurre en el algoritmo Fijo.

Los mayores índices de calidad MOS entre ambos algoritmos para un mismo codificador se obtuvieron con Moon el cual para G.711 arrojó resultados comprendidos entre 3.75 y 3.95 los cuales se clasifican atendiendo a la escala de MOS como calidad media.

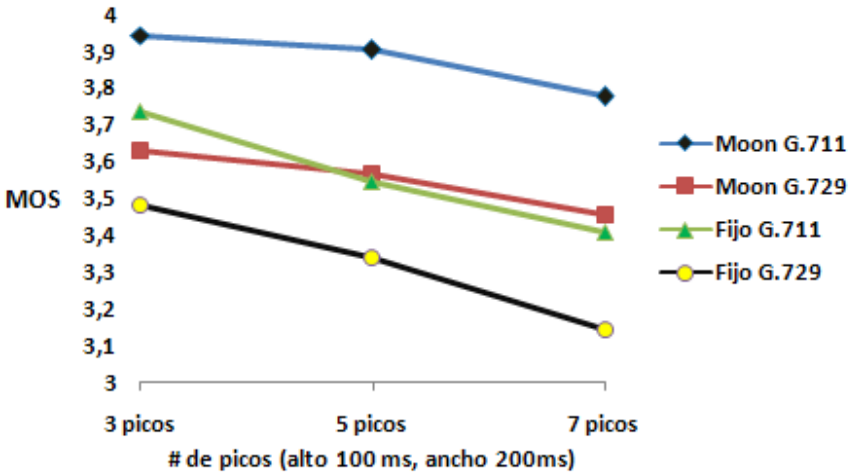

Figura 18. Valor MOS obtenido en las simulaciones

La Figura 19 contiene los valores R proporcionados por el Modelo-E para ambos algoritmos. Moon con G.711 obtuvo resultados que se encuentran entre 70 y 80 a lo cual le corresponde una calidad considerada como media mientras que para el Fijo utilizando G.729 los valores están en el rango de 60 a 70 que se clasifica como calidad baja.

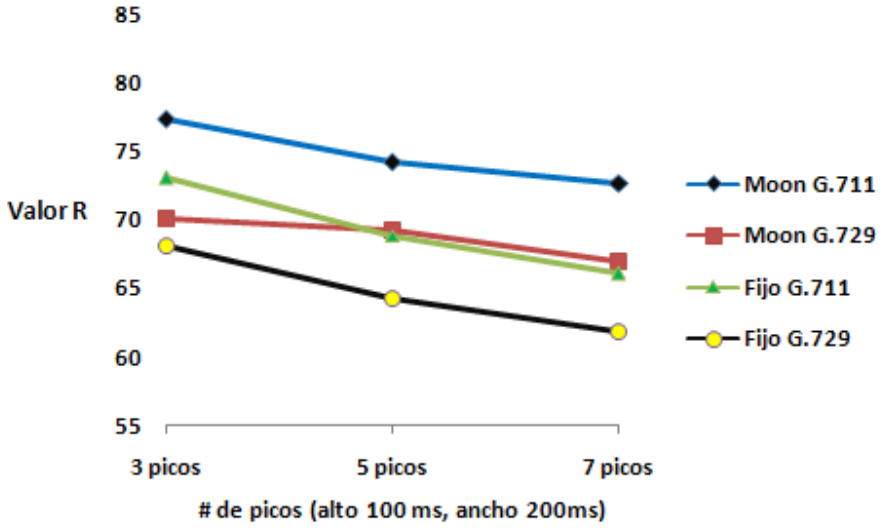

Figura 19. Factor R obtenido en las simulaciones

3.4 Variación del tamaño máximo del buffer de historial de demoras

El algoritmo descrito por Moon colecciona estadísticas de paquetes que han llegado y las usa para estimar la demora de reproducción. El parámetro que permite realizar esta acción se conoce como tamaño del buffer de historial de demoras (w) y define la cantidad máxima de demoras de paquetes que pueden ser almacenadas por el algoritmo.

El tamaño del parámetro w determina cuan sensible es el algoritmo en adaptarse a los cambios de demoras de la red, si el tamaño de w es muy corto el algoritmo tendrá una visión miope del pasado y producirá una estimación pobre de la demora de reproducción lo cual repercutirá en un deterioro de la calidad, si el valor de w es muy largo el algoritmo mantendrá la pista de un número innecesario de demoras almacenadas en el histograma.

El algoritmo de Moon se simuló utilizando los codificadores (G.711, G.729), una demora del sistema de 40 ms, las trazas simuladas 2, un fichero de audio de aproximadamente 39 segundos y un tamaño de w que varió entre 10000 y 100 paquetes.

Los índices de calidad MOS y R resultantes de las simulaciones se muestran en la siguiente tabla:

Tabla 9. Índices de calidad resultantes de las simulaciones

Tamaño W	G.711		G.729	
	MOS	R	MOS	R
10000	3.730282	72.7107	3.440886	66.5473
2000	3.757404	73.2493	3.450415	66.6704
1500	3.778332	73.6501	3.491857	67.4480
1200	3.768298	73.3923	3.496383	67.5126
800	3.625690	70.0849	3.444890	66.2974
400	3.484645	67.2683	3.187204	61.3163
100	3.051449	59.0549	2.917735	56.4986

En la Figura 20 se muestran los índices de calidad MOS proporcionados por PESQ para los distintos valores de w, se observa que para valores mayores, o menores cercanos a la cantidad de paquetes en que se convierte la señal de audio los índices de calidad no sufren una variación pronunciada, mientras que para valores menores a 1200 paquetes los resultados para ambos codificadores caen drásticamente, hasta el punto que para 100 paquetes el MOS no sobrepasa el valor de 3.1, esto equivale en las notas de degradación de MOS a degradación ligeramente molesta que requiere un esfuerzo moderado para la audición.

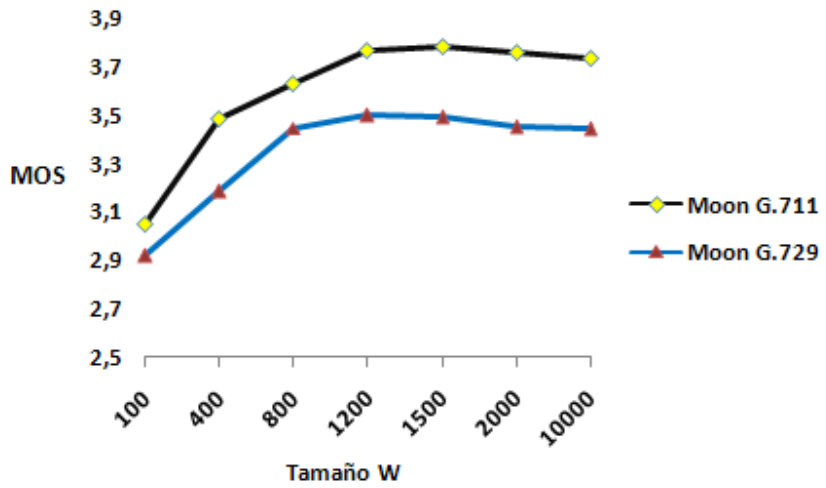

Figura 20. Valor MOS obtenido en las simulaciones

En la Figura 21 se presentan los valores R obtenidos en las simulaciones, como se esperaba las características de la gráfica coincidieron con la Figura 20. Se observa que una disminución del tamaño máximo del buffer de historial de demoras afecta el resultado del valor R. Para un tamaño de w igual a 100 paquetes los valores de R para ambos codificadores están en el rango de 55 a 60, lo cual equivale en la escala del factor R a una calidad pobre.

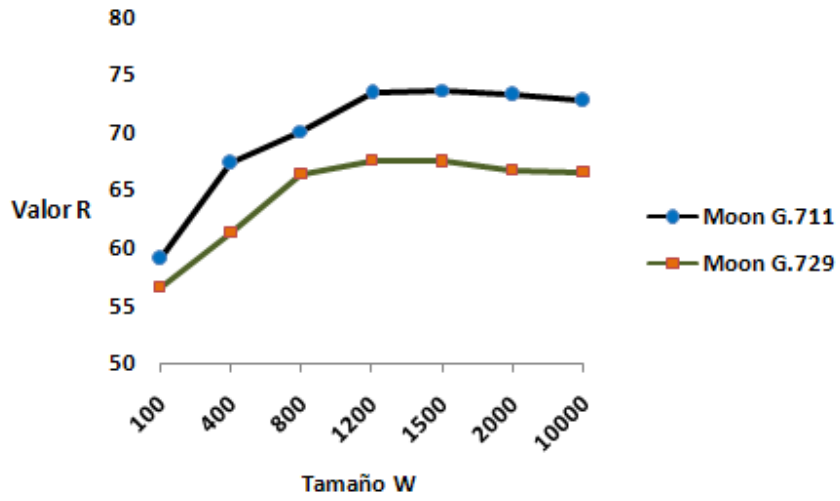

Figura 21. Factor R obtenido en las simulaciones

3.5 Conclusiones del Capítulo

En este capítulo se analizan los resultados obtenidos de la evaluación del algoritmo de reproducción para el control del jitter propuesto por Moon bajo diferentes condiciones específicas de trabajo. Con los resultados de los experimentos se observa cómo reacciona el algoritmo ante diferentes tipos de trazas, codificadores, demoras del sistema, picos de demoras y variación de parámetros. Los índices de calidad resultantes de las simulaciones evidencian la superioridad del codificador G.711 con respecto al G.729, que los resultados con trazas reales fueron satisfactorios en general y que a medida que disminuye el tamaño máximo del buffer de historial de demoras se produce un deterioro en la calidad arrojada por el algoritmo.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

En el trabajo se realizó la evaluación simulada y experimental del algoritmo de reproducción para el control del jitter propuesto por Moon. En particular:

- Se implementó un paquete de software de código abierto que permitió simular completamente la cadena de procesamiento de VoIP, el cual resultó esencial para el desarrollo del trabajo debido a las numerosas funciones que brinda. El paquete ofrece la posibilidad de trabajar con dos de los esquemas de codificación más utilizados (G.711, G.729) así como simular diferentes algoritmos de reproducción tanto de buffer fijo como adaptativos, también permite leer ficheros de trazas de paquetes de VoIP generadas por el software NS-2.33, la incorporación de trazas de tráfico real, generar artificialmente ficheros de trazas para estudiar el impacto de los picos de demora, valorar la calidad usando el Modelo-E y PESQ, cambiar los valores de los diferentes parámetros específicos de los algoritmos y analizar el desempeño de los mismos ante diferentes condiciones de trabajo.
- Se obtuvieron trazas simuladas y reales de paquetes de VoIP que permitieron la evaluación del algoritmo de Moon y conocer el desempeño del mismo ante las diferentes condiciones de red existentes donde las trazas fueron adquiridas. La interacción entre el NS-2.33 y el paquete de software permite explorar el comportamiento de los algoritmos en diferentes arquitecturas de redes brindando así nuevos escenarios para evaluaciones con características diferentes.

Con los resultados de los experimentos se observa que:

- Los índices de calidad resultantes de la evaluación del algoritmo de Moon ante las trazas simuladas utilizando G.711 se consideran según las categorías de calidad (MOS, R) como calidad media mientras que para G.729 como calidad baja y que los valores más altos se corresponden con menores distribuciones de jitter.
- Los resultados obtenidos de la evaluación ante trazas reales para ambos codificadores fueron satisfactorios, lo que evidencia el buen desempeño del algoritmo ante las condiciones existentes en la red donde las trazas fueron capturadas.
- A medida que aumenta el número de picos de demoras se produce una disminución en el desempeño tanto del algoritmo de Moon como en el Fijo y que la disminución más brusca ocurre en este último.
- A medida que disminuye el tamaño máximo del buffer de historial de demoras el algoritmo produce una estimación pobre de la demora de reproducción lo que repercute en un deterioro de la calidad.

Teniendo en cuenta los resultados obtenidos se considera adecuada la plataforma para la realización de estudios posteriores.

Recomendaciones

Atendiendo a las funcionalidades del paquete de software, se recomienda el empleo del mismo en futuros trabajos de investigación que permitan analizar el desempeño de los algoritmos utilizando trazas simuladas generadas por el NS en diferentes topologías de redes y trazas reales capturadas en distintos escenarios.

Debido a que el QofIS_v2 es un paquete de software de código abierto, se recomienda valorar la implementación de nuevos módulos para la simulación de VoIP y al mismo tiempo mantener una actualización en el seguimiento de trabajos similares desarrollados por parte de otros autores e instituciones.

REFERENCIAS BIBLIOGRÁFICAS

ATZORI, L. y LOBINA, M. L., (2006) "Playout buffering in IP Telephony: A Survey Disscussing Problems and Approaches". *IEEE Communications Surveys and Tutorials.* vol.8: pp.36-46.

BERKELEY, U., et al. (2002). "The Network Simulator"[En línea]. disponible en: http://www.isi.edu/nsnam [10 de marzo de 2009]

BOLOT, J. C., (1993) "Characterizing end-to-end packet delay and loss in the Internet". *Journal of High Speed Networks.* vol.2: pp.305-323.

BUCHLI, M. y PETIT, G. H., (2002) "Assessing voice quality in packet-based telephony". *IEEE Internet Computing.* vol.6: pp.48-56.

DAVIDSON, J. y PETERS, J., (2000) *Voice Over IP Fundamentals.* Indianapolis, Cisco Press.

DELEON, P. y SREENAN, C. J., (1999) "An adaptive predictor for media playout buffering" *In Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Phoenix, USA.

FOUNDATION, E. (2008). "Software Eclipse Ganymede"[En línea]. disponible en: http://www.eclipse.org/ganymede [3 de marzo de 2009]

FUJIMOTO, K., (2002) Adaptive Playout Buffer Algorithm for Enhancing Perceived Quality of Streaming Applications. PhD thesis. Departament of Informatics and Mathematical Science, Osaka University.

HOENE, C. (2001). "Easysnuffle - a tool to measure the performance of multimedia flows over IEEE 802.11b"[En línea]. disponible en: http://www.tkn.tu-berlin.de/research/easysnuffle [1 de abril de 2009]

HOENE, C., (2004) "Predicting Performance of PESQ in Case of Single Frame Losses" *In Proccedings Measurement of Speech and Audio Quality in Networks (MESAQIN)*, Prague.

HOENE, C., et al., (2003) "On the importance of a voip packet" *In Proceedings of ISCA Tutorial and Research Workshop on Auditory Quality of Systems*, Mont-Cenis, Germany.

HOENE, C. y WIETHÖLTER, S. (2004a). "QofIS - a software package to simulate playout schedulers for VoIP"[En línea]. disponible en: http://www.tkn.tu-berlin.de/research/qofis/ [22 de enero de 2009]

HOENE, C. y WIETHÖLTER, S. (2004b). "Simulating Playout Schedulers for VoIP-Software Manual"[En línea]. Berlin, disponible en: http://www.citeseer.nj.nec.com/ [25 de enero de 2009]

HOENE, C., et al., (2004) "Predicting the Perceptual Service Quality Using a Trace of VoIP Packets" *In Proceedings of Fifth International Workshop on Quality of Future Internet Services*, Barcelona, Spain.

ITU-T_G.107 (2003). "E-model, a computational model for use in transmission planning"[En línea]. disponible en: http://www.itu.org [7 de marzo de 2009]

ITU-T_G.729 (1996). "Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear-prediction (CS-ACELP)"[En línea]. disponible en: http://www.itu.org [20 de marzo de 2009]

ITU-T_G.729 (2007). "G.729 source code release 2"[En línea]. disponible en: http://www.itu.int/rec/T-REC-G.729-200701-l/en [1 de abril de 2009]

ITU-T_P.800 (1996). "Métodos de Determinación Subjetiva de la Calidad de Transmisión"[En línea]. disponible en: http://www.itu.org [11 de marzo de 2009]

ITU-T_P.862 (2001a). "Perceptual evaluation of speech quality (PESQ): An Objective Method for End-to-end Speech Quality Assessment of Narrow-band Telephone Networks and Speech Codecs"[En línea]. disponible en: http://www.itu.org [10 de febrero de 2009]

ITU-T_P.862 (2001b). "PESQ source code release 1.2"[En línea]. disponible en: http://www.itu.int/rec/T-REC-P.862/en [25 de marzo de 2009]

KANSAR, A. y KARANDIKAR, A. (2001). "Jitter-free audio playout over Best Effort Packet Networks"[En línea]. disponible en: http://www.ee.iitb.ac.in/ [7 de marzo de 2009]

KUROSE, J. F. y ROSS, K. W., (2000) *Computer Networking A Top-Down Approach Featuring the Internet.* Boston, USA, Addison Wesley Professional.

LI, B., et al., (2000) "QoS enabled Voice Support in the next generation Internet: issues, existing approaches and challenges ". *IEEE Communications Magazine.* vol.38: pp.54-61.

LIANG, Y. J. y GIROD, B., (2003) "Adaptive playout scheduling and loss concealment for voice communications over IP networks". *IEEE Transactions on Multimedia.* vol.5: pp.532-543.

MILLER, M. J. (2006). "Installing NS-2.26 in Debian"[En línea]. disponible en: http://www.matthewjmiller.net/howtos/ [16 de marzo de 2009]

MOON, S. B., et al., (1998) "Packet audio playout delay adjustment: Performance bounds and algorithms". *ACM/Springer Multimedia Systems.* vol.6: pp.17-18.

NA, S. y YOO, S., (2002) "Allowable Propagation Delay for VoIP calls of Acceptable Quality" *In Proceedings of the First International Workshop on Advanced Internet Services and Applications*, London.

NARBUTT, M., et al., (2005) "Adaptive VoIP playout scheduling: assessing user satisfaction". *IEEE Internet Computing.* vol.9: pp.28-34.

RAMJEE, R., et al., (1994) "Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks" *In Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Toronto,Canada.

RAMOS, V. M., et al., (2003) "A moving average predictor for playout delay control in VoIP" *In Proceedings of the 11th International Workshop on Quality of Service*, Berkeley, USA.

ROSENBERG, J., et al., (2000) "Integrating packet FEC into adaptive voice playout buffer algorithms on the Internet" *In Proceedings of the IEEE Infocom*, Israel.

SCHULZRINNE, H., et al. (2003). "RTP: A transport protocol for real-time applications"[En línea]. disponible en: http://www.citeseer.nj.nec.com/ [5 de febrero de 2009]

SCHULZRINNE, H. y ROSENBERG, J., (1999) "The IETF Internet telephony: Architecture and protocols". *IEEE Network.* vol.13: pp.18-23.

THOMSEN, G. y JANI, Y., (2000) "Internet telephony: going like crazy". *IEEE Spectrum.* vol.37: pp.52-58.