Universidad Autónoma Metropolitana Iztapalapa

División de Ciencias Básicas e Ingeniería

Idónea comunicación de resultados

Evaluación de algoritmos de control de retardo en Voz sobre Internet

Presentada para obtener el grado de :
Maestra en Ciencias

Especialidad : Tecnologías de la Información
por

Karen Samara Miranda Campos

Asesor : Víctor Manuel RAMOS RAMOS

En el área : Redes y Telecomunicaciones
del Departamento de Ingeniería Eléctrica

Defendida públicamente en la UAM-Iztapalapa el 29 de enero 2009 a las 11:00 hrs frente al jurado integrado por :

Presidente :	Raúl Teodoro	Aquino Santos	Univ. de Colima, Fac. Telemática
Secretario :	Víctor Manuel	Ramos Ramos	UAM-I, Redes y Telecomunicaciones
Vocal :	Fausto Marcos	Casco Sánchez	UAM-I, Redes y Telecomunicaciones
Vocal :	Marcelo Carlos	Mejía Olvera	ITAM, Depto. Acad. de Computación

Evaluación de algoritmos de control de retardo en Voz sobre Internet

Título en inglés :

Evaluation of playout delay control algorithms for Voice over Internet

Karen Samara Miranda Campos

Enero 2009

A mi madre
y en recuerdo de mi padre...

Agradecimientos

Agradezco con mi más profundo respeto y aprecio al Dr. Víctor Ramos, mi director de tesis, quien me ha guíado dentro del área de redes y telecomunicaciones incluyendo el tema de ésta tesis. Le agradezco por recibirme bajo su tutela, por su apoyo, y por su infinita paciencia para responder todas mis dudas. Gracias por los consejos que me dió, los cuales son invaluables para mí.

Gracias a la Universidad Autónoma Metropolitana por el apoyo otorgado para este proyecto.

A los profesores de la Maestría por todas sus enseñanzas y consejos.

Al Dr. Eitan Altman por aceptarme en su equipo y por su tutoría en mi estancia en Francia. Trabajar con él ha sido un gran experiencia dentro de mi formación profesional.

Muchas gracias al Dr. Raúl Aquino por haber aceptado ser el presidente de mi jurado, así como también, agradezco al Dr. Fausto Casco y Dr. Marcelo Mejía por hacerme el honor de participar en mi jurado de tesis. Gracias por su gran interés y sus comentarios y sugerencias hacia este trabajo.

Les agradezco a mis compañeros de la Maestría, en especial a Ana Ríos, Gustavo Basurto, Alan Lazalde, y Pedro Marcial. Y a mis compañeros en Avignon Silvia Fernández, Pedro González, Luis Valderrama, Rodrigo Acuña y Mariela, que fueron un gran apoyo en mi estancia allí.

Agradezco de todo corazón a mi madre, a mis hermanas, Claudia y Verónica, y mis tíos, Claudia y Felipe, por entender la importancia que tiene para mí este trabajo, por el cariño incondicional y la comprensión que me han tenido durante este tiempo, sin los cuales no habría podido lograrlo.

Resumen

Este proyecto de tesis de maestría trata acerca de los algoritmos de control de retardo en VoIP. Las aplicaciones interactivas sobre Internet se utilizan extensamente en nuestros días. Aplicaciones P2P como Skype, VoIPBuster han incorporado exitosamente la VoIP.

Un algoritmo de control de playout implementa un buffer en el lado del receptor, así guarda los paquetes recibidos. Entonces, el algoritmo calcula un tiempo límite para cada paquete. Si el paquete se perdió en la red o llegó después de su tiempo límite esperado, el paquete se considera perdido en el receptor. Un algoritmo de playout considera el compromiso entre las pérdidas y el retardo de tal forma que optimice la interactividad de la sesión de VoIP.

Nos enfocamos en la clase de algoritmos que actualizan el retardo de playout al inicio de cada frase. Estudiamos un algoritmo NLMS originalmente propuesto por DeLeon y posteriormente modificado por Shallwani para probarlos extensivamente bajo las mismas condiciones de trabajo. Los resultados que obtenemos indican por un lado que el algoritmo de Shallwani puede tener errores de depuración, y por el otro lado, que la detección de picos de retardo puede mejorarse. Así, decidimos mejorar la detección de picos de retardo que propone Shallwani y comparar el desempeño de nuestro algoritmo con los algoritmos de DeLeon y de Shallwani. Encontramos, que para la mayoría de los casos, usando trazas reales de audio, que han sido muy utilizadas en otros trabajos, nuestro algoritmo se desempeña mejor.

Abstract

This masters thesis project is about playout delay control algorithms for Voice over IP (VoIP). Interactive applications on the Internet are extensively used nowadays. P2P applications like Skype, VoIPBuster have implemented very successful VoIP systems.

A playout delay control algorithm implements a buffer on the receiver side so as to store the received packets. Then, the algorithm computes a deadline for each packet. If a packet has been lost in the network or has arrived after its scheduled deadline, the packet is considered lost at the receiver. On one hand, if the deadline is long, the playout algorithm gives more chance to arrive to packets having suffered from a high level of jitter, but at the expense of loosing interactivity during the VoIP call. On the other hand, if the computed deadline is short the playout algorithm is more aggressive but with the risk of loosing more packets even if they arrived at the receiver side, but having arrived after their scheduled deadline. So, a playout algorithm trades-off delay and loss so as to optimize the interactivity of the VoIP call.

In this masters thesis, we focus on the class of playout algorithms that update the playout delay at the beginning of a talkspurt. We choose to study an NLMS algorithm originally proposed by DeLeon and later modified by Shallwani and to extensively stress the working conditions of both algorithms. The results we got till today indicate from one side that the Shallwani's algorithm may be buggy, and from the other side that delay spike detection algorithm may be improved. So, we choose to improve the delay spike detection algorithm proposed by Shallwani and compare the performance of our (so called) NLMS-mod algorithm with the DeLeon's and Shallwani's algorithms. We found that, for most of the cases, with real-audio traces extensively used in other works (kindly offered by Sue Moon) our algorithm performs better than the formers.

Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

Internet ha ayudado a desarrollar nuevas formas de comunicación, entre ellas las aplicaciones en tiempo real que incluyen la transmisión no solo de datos sino también audio y vídeo. Dentro de estas aplicaciones se incluye la telefonía sobre Internet conocida como Voz sobre IP (VoIP). Las aplicaciones interactivas son ampliamente usadas en nuestros días. Aplicaciones P2P como Skype y NetApel han implementado exitosamente sistemas de VoIP. Algunas aplicaciones de mensajería instantánea como MSN Messenger, Yahoo Messenger, AIM Messenger entre otros también ofrecen servicios de VoIP que son gratuitas entre los usuarios de PC a PC. Los primeros trabajos de investigación como NeVoT por Henning Schulzrinne, FreePhone por Bolot y Andrés Vega García, y Rat por el proyecto Mice, invirtieron un gran esfuerzo para optimizar la transmisión de paquetes de audio en tiempo real en Internet[37, 36].

Los ruteadores en la red utilizan buffers que implementan políticas FIFO, lo que quiere decir, que el primer paquete en entrar al buffer es el primer paquete en ser reexpedido. De esta forma, no hay prioridad entre paquetes. Esto tiene la ventaja de tener un funcionamiento sencillo, pero la desventaja de que todos los paquetes tienen la misma prioridad: un paquete de voz tendrá que esperar en el buffer como cualquier otro paquete, incluso si las restricciones del retardo para las aplicaciones en tiempo real son rígidas. Este mecanismo *best-effort* en Internet causa primordialmente tres fenómenos: retardo, pérdida de paquetes y variabilidad en el retardo, los cuales afectan la calidad de servicio de las aplicaciones en tiempo real como la VoIP.

Las aplicaciones en tiempo real generan paquetes en intervalos regulares de tiempo. El tráfico generado por una fuente de audio es divido en periodos de actividad y en periodos de silencio. Al cruzar Internet, los paquetes de audio sufren retardo variable debido al tiempo variable en las filas de espera de los ruteadores. Esta variabilidad modifica los in-

tervalos iguales entre paquetes transmitidos. 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 son considerados 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, un gran retardo disminuye la interactividad de una sesión de audio. Por esto, existe un compromiso entre el retardo y las pérdidas debidas a los paquetes tardíos.

En este trabajo nos enfocamos en ese compromiso entre las pérdidas y el retardo para los algoritmos de control de retardo en VoIP. Usando medidas de retardo de extremo a extremo de los paquetes hechas con NeVoT, presentamos un algoritmo NLMS que ajusta el retardo de playout al inicio de cada frase. Para probar la eficiencia de nuestro algoritmo lo comparamos con dos trabajos presentados previamente por DeLeon[6] y Shallwani[40].

Los filtros NLMS han sido utilizados en diferentes campos, principalmente en la cancelación del eco. Utilizamos el filtro NLMS para estimar el retardo de extremo a extremo. Encontramos, que para la mayoría de los casos, usando trazas reales de audio, muy utilizadas en otros trabajos, que fueron generadas originalmente por Sue Moon, nuestro algoritmo se desempeña mejor.

Este documento está estructurado de la siguiente manera. En el Capítulo 2 abordamos los conceptos necesarios para comprender el funcionamiento de la VoIP. El Capítulo 3 describe los principales fenómenos que afectan la calidad de servicio de las aplicaciones en tiempo real, sus consecuencias, y algunas soluciones. En el Capítulo 4 presentamos el estado del arte de los algoritmos de control de retardo en VoIP. En el Capítulo 5 presentamos el trabajo relacionado, particularmente los algoritmos NLMS, así como también, nuestro trabajo en la mejora del algoritmo NLMS original y el método de comparación. Finalmente en el Capítulo 6 concluimos este trabajo y presentamos nuestro trabajo actual y futuro.

Capítulo 2

Fundamentos de Voz sobre IP

En este capítulo abordamos los conceptos necesarios para comprender las características de la telefonía sobre Internet, además asentamos las bases para comprender los mecanismos de transmisión utilizados en la Voz sobre Internet.

La telefonía convencional o PSTN[1], provee su servicio por medio de un canal dedicado exclusivamente para cada transmisión, y sólo la información referente a esa transmisión correspondiente puede ser enviada por ese canal, es decir, cuando se establece una llamada telefónica se crea un circuito que es otorgado por la red únicamente para esa llamada. En este tipo de conmutación la voz se transporta a una tasa constante (64 kbps) y su calidad se considera como constante [14].

A diferencia de la telefonía convencional, la telefonía sobre Internet funciona sobre una red de conmutación de paquetes, en donde el mensaje original se divide en paquetes independientes entre sí. A estos paquetes se les agrega información de control y se envían de un extremo a otro, y al momento llegar al destino, los paquetes son reagrupados para obtener el mensaje original [14].

La telefonía sobre Internet, también conocida como telefonía IP o Voz sobre Internet: VoIP [39, 32, 15, 44], permite proveer un servicio de transmisión de voz entre dos o más partes sobre una red de datos basada en el protocolo IP, y el intercambio de información de control requerida para esta transmisión. La idea principal de la VoIP es transformar la voz en paquetes, de manera que la señal de voz es digitalizada, codificada y comprimida, para que dichos paquetes sean enviados a través de Internet a su destino, como cualquier otro tipo de paquetes que son transmitidos por este medio [44].

La telefonía sobre Internet se usó inicialmente como una manera simple de dar un

[1] *Public Switched Telephone Network*, Red Telefónica Pública Conmutada

servicio de voz punto a punto entre dos nodos IP, en un principio para reemplazar las costosas llamadas internacionales. Sin embargo, gracias al creciente interés en los servicios de integración de voz, datos y video, el alcance de la "*telefonía sobre Internet*" se ha expandido. Hoy en día el término "*telefonía sobre Internet*" se puede entender como conjunto de servicios, tales como, transporte multimedia, web, movilidad, e-mail, y mensajes instantáneos [39], incluso, con la aparición de las redes inalámbricas se ha acuñado el término *Wi-Fi Internet telephony* [11], pero esto no cambia fundamentalmente el problema de base.

Esta forma de transmisión de voz presenta ventajas sobre la telefonía tradicional y por ello se vuelve más atractiva. Estas ventajas se refieren a:

Precio: Realizar llamadas sobre Internet es mucho más barato que sobre PSTN debido a que evita los cargos de acceso. Permite compartir el ancho de banda y evita los cargos adicionales en las llamadas de larga distancia.

Transmisión simultánea de voz y datos: Internet permite transportar tráfico de voz y datos, por lo que puede ofrecer servicios tanto multimedia como de otro tipo, de manera integral.

Integración: La posibilidad de transmitir simultáneamente voz y datos permite un mejor uso del ancho de banda, ya que no es exclusivo. En PSTN cada conexión tiene un ancho de banda fijo asignado de 64 kbps incluyendo los periodos de silencio, donde no hay transmisión de tráfico, y es en estos periodos cuando es posible la transmisión de otro tipo de tráfico en la Telefonía sobre Internet. Esto último permite la integración de diferentes tipos de tráfico, cambios más baratos, y eliminación de puntos de falla. El uso del protocolo IP para las aplicaciones permite una menor complejidad y más flexibilidad.

Aplicaciones avanzadas: Para compartir fácilmente video, pantallas, o pizarrones, además, ofrece servicios como identificador de llamadas y seguridad a través del cifrado de la información.

2.1. Internet y el fenómeno de la congestión

Internet hace posible el uso de aplicaciones distribuidas entre sus sistemas finales, clientes y servidores, para el intercambio de información. Estas aplicaciones incluyen el acceso remoto, transferencia de datos, streaming de audio y video, y transmisión en tiempo real de audio y video. La filosofía de Internet es dejar que la inteligencia de la red esté en los extremos, asumiendo poco acerca de la red en sí, es decir, lo que pasa dentro de la red Internet lo maneja como una "caja negra", esto sirve para implementar distintos algoritmos sin necesidad de afectar el funcionamiento interno de la red y con ello se asegura una base para las nuevas tecnologías. Esta filosofía de funcionamiento se ha mantenido debido a que aún después de muchos años de investigación en calidad de servicio en el ruteo, además de

las discrepancias entre los proveedores de servicio (ISP) no se ha adoptado todavía un mecanismo único de calidad de servicio, QoS en Internet [14]. La Figura 2.1 presenta un ejemplo de la heterogeneidad de Internet. Este ejemplo está compuesto por diversas redes de área local, inalámbricas, alámbricas, de servidores, y ruteadores alámbricos e inalámbricos.

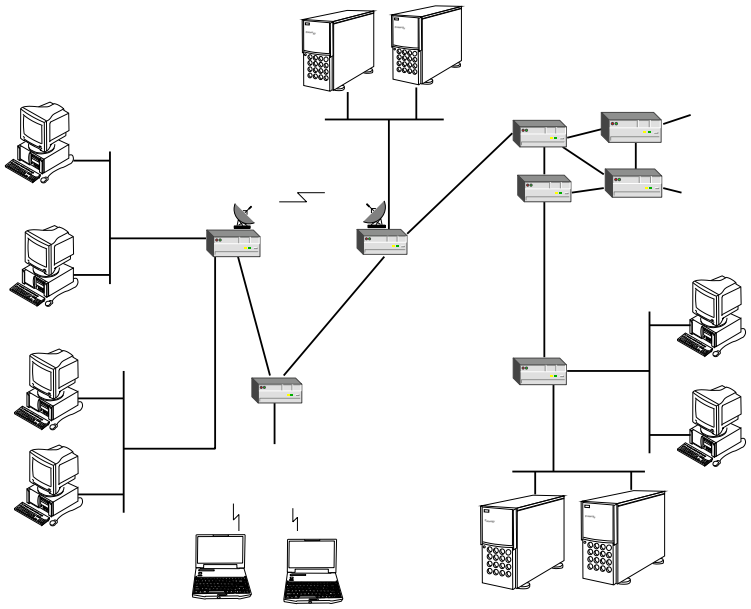

Figura 2.1: Ejemplo de red

Internet comprende protocolos que permiten determinar la manera en que los paquetes se envían y reciben en ella. El protocolo sobre el que se basa Internet es el protocolo IP. IP es un protocolo a nivel de red, que se encarga de las conversiones de direcciones, el formato de datagramas, y la conversión del manejo de paquetes. Su objetivo principal, es permitir la comunicación entre los extremos de la red debido a que estos no se encuentran conectados directamente sino que están conectados a través de ruteadores [14].

Otra característica de IP es reexpedir los paquetes lo más rápido posible con una política de servicio *FIFO*[2] o el Primer paquete en entrar es el primer paquete en salir de la fila; es decir, el primer paquete que llega a la fila de espera de los ruteadores es el primero en ser atendido y el primero en ser reexpedido sin importar si dicho paquete transporta datos, video, o voz, todos son atendidos sin distinciones ni prioridades. Para guardar los paquetes en espera, los ruteadores cuentan con un espacio en memoria, sin embargo, esta memoria es finita y cuando las filas de espera están llenas, éstas no pueden recibir más paquetes. En otras palabras, están saturadas por lo que los paquetes que llegan en ese momento son desechados, a esto se le llama *fenómeno de la congestión.*

Cabe señalar que Internet no monitorea el tamaño de las filas de espera en los ruteadores, por lo que no hay estados en la red ni puede saber si existe congestión o no. Así

[2]First In First Out

IP no ofrece ninguna garantía en la calidad de servicio, de tal suerte que envía los paquetes a través de la red y hace su mejor esfuerzo para que lleguen a su destino, por ello se le conoce como servicio *"Best-effort"* lo cual trae, entre otros, los siguientes fenómenos [14, 32]:

- **Pérdidas**: Los paquetes se consideran perdidos cuando no llegan a su destino, es el caso donde las filas de espera se encuentran llenas en el momento que el paquete llega y por lo tanto, es eliminado. También existen pérdidas por errores de transmisión sobre redes inalámbricas, o por desconexiones en el enlace, por ejemplo.
- **Retardo**: Los paquetes no llegan a su destino en el mismo instante en el que son enviados, dependiendo del tamaño de las filas de espera al momento que cada paquete llega, así como del camino que toman en la red, o de las retransmisiones en la capa de enlace.
- **Variabilidad en el retardo**: Los paquetes no tardan el mismo tiempo en llegar a su destino, por lo que pueden llegar en desorden y sin la misma cadencia con la que son enviados. Esto se debe a que los paquetes pueden tomar distintos caminos en la red, o bien, esperar diferente tiempo en las filas de espera de los ruteadores.

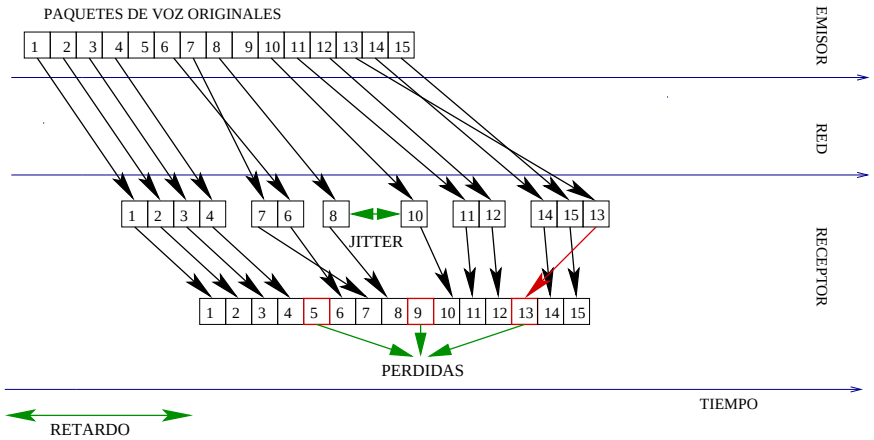

Figura 2.2: Ejemplo de los tres fenómenos provocados por IP

En la Figura 2.2 se observan los tres fenómenos antes descritos. En este ejemplo se aprecia como el paquete 5 se perdió al pasar por Internet, los paquetes 9, 13 y 16 llegan en diferente orden al que fueron enviados, y la inclinación en la flechas representa el retardo de los paquetes.

Por supuesto que para cada problema existe una solución, para el caso de los retos que conlleva el uso de IP, el protocolo TCP mejora el servicio de transferencia de paquetes.

2.2. Transporte

Los protocolos definen el formato y el orden de los mensajes intercambiados entre dos o más entidades de comunicación, asimismo, las acciones tomadas en la transmisión y/o recepción de un mensaje. Un protocolo de transporte provee una comunicación lógica entre las aplicaciones de diferentes extremos de la red. Una red puede ser capaz de utilizar diferentes protocolos de transporte, ya que cada protocolo ofrece un modelo de servicio diferente para las aplicaciones.

Estos protocolos de transporte están implementados en los extremos de la red pero no en los ruteadores, por esta razón los servicios que un protocolo de transporte puede ofrecer son frecuentemente restringidos por el modelo de servicio de los protocolos de la capa de red. Sin embargo, ciertos servicios pueden ser ofrecidos por la capa de transporte, incluso cuando los protocolos de la capa de red no ofrezcan el correspondiente servicio en su respectiva capa. Por ejemplo, un protocolo de transporte puede ofrecer una transferencia de datos confiable para una aplicación, aunque el protocolo de la capa de red no es confiable.

Otro de los protocolos importantes en Internet, además de IP, es TCP que corrige las desventajas del uso de IP, TCP ofrece un servicio confiable orientado a conexión. TCP también incluye un mecanismo de control de congestión y un servicio para un bienestar general de Internet. Sin embargo, no es el único también existe UDP, que es un protocolo con un funcionamiento mucho más sencillo que TCP. UDP ofrece un servicio no confiable para la aplicación que lo convoca.

La responsabilidad fundamental de UDP y TCP es extender el servicio de entrega de IP entre dos nodos a un servicio de entrega entre dos procesos de los nodos. Además, TCP y UDP proveen la revisión de integridad de datos incluyendo en sus encabezados un campo de detección de errores.

2.2.1. TCP

El protocolo TCP posibilita una transmisión confiable de datos, detectando a los paquetes perdidos y retransmitiéndolos, de tal forma que toda la información sea entregada en el destino. TCP realiza esto creando un circuito virtual que conecta al origen y el destino, manejando la información para las aplicaciones como si fuera una línea continua de bytes sin límites entre paquetes, de tal suerte que la aplicación entienda el contenido como un solo tren de bytes.

TCP al mismo tiempo asegura el control de flujo y el control de congestión usando una estrategia basada en ventana, en otras palabras, implementa un protocolo de ventana deslizante, denotada por w, que se adapta a las condiciones de la red:

Control de flujo: Es la función de adaptar la velocidad de transmisión a la capacidad del receptor, de forma que el emisor no sature el buffer del receptor al transmitir demasiado tráfico en poco tiempo.

Control de congestión: Es la función de adaptar la velocidad de transmisión al ancho de banda disponible en la red. Se enfoca en la cantidad de datos transmitidos sobre la red, de manera que la red pueda manejar ésta cantidad. La congestión se manifiesta cuando existen demasiados paquetes perdidos y el retardo es demasiado largo.

Para asegurarse de que toda la información haya llegado correctamente, TCP pide al destino que envíe un acuse de recibo, *ACK*, por cada paquete que recibe. Así puede saber qué paquetes llegaron correctamente y cuales deben ser reenviados, así comienza enviando un paquete y va duplicando la cantidad de paquetes enviados, crece la ventana w, hasta que llega el acuse de pérdida y la cantidad de paquetes a enviar es reducido a la mitad. El manejo de esta ventana w es muy importante ya que si es constante no se ajusta a la capacidad de la red, si es muy grande la posibilidad de pérdidas es muy grande, y si es muy pequeña subutiliza el canal. De esta forma las diferentes versiones de TCP buscan mejorar la gestión de la ventana de envío w [9].

Para prevenir la congestión en la red, TCP cuenta con dos mecanismos [9]:

Slow start: El crecimiento del tamaño de la ventana del emisor es exponencial.

Congestion avoidance: El crecimiento del tamaño de la ventana del emisor es lineal.

Para una referencia de base sobre el funcionamiento de TCP, el lector puede referirse a [9].

2.2.2. UDP

El protocolo UDP fue inicialmente pensado para aquellas aplicaciones que no requieren la confiabilidad que ofrece TCP.

El principio de UDP es ofrecer un servicio simple, no orientado a conexión, que permita la multiplexión/demultiplexión de paquetes desde diferentes aplicaciones en la misma máquina y una sencilla revisión de errores. La simplicidad de UDP radica en que no establece una conexión antes de comenzar el envío de paquetes, cada segmento de UDP es manejado independientemente de los otros, no necesita recibir un acuse de recibo para enviar el siguiente paquete, el encabezado de los paquetes es más pequeño, la tasa de envío de paquetes es irregular, y no implementa algoritmos para el control de la congestión.

El campo de segmento *checksum* funciona para asegurar la integridad de la información transmitida. El emisor coloca el complemento a 1 de todas la palabras de 16-bits en el segmento UDP y lo coloca en el campo *checksum*. Cuando el paquete es recibido se realiza una suma de todas las palabras de 16-bits, y si el resultado sólo contiene 1's, entonces el segmento llegó sin errores. Una aplicación puede usar UDP y corregir la pérdida de paquetes sin la necesidad de usar las funciones implementadas en TCP.

Algunas aplicaciones utilizan UDP sobre TCP, por ejemplo DNS pasa los mensaje por UDP para las traducciones de los nombres de dominio, RIP lo usa para enviar los mensajes que actualizan las tablas de ruteo, SNMP utiliza UDP para llevar el manejo de la red, y las aplicaciones multimedia para la transmisión de voz y video.

2.3. Procesamiento de señales

Antes de poder transmitir el audio sobre Internet, éste tiene que ser digitalizado y comprimido. La necesidad de estos procedimientos es evidente, las redes de computadoras transmiten bits, por lo que la información debe ser representada como una secuencia de bits. La compresión es importante dado que el audio sin comprimir consume una cantidad enorme de ancho de banda y almacenamiento. La división de la señal en paquetes de audio, así como la cancelación del eco, y la supresión de silencios también son necesarios para optimizar las sesiones de audio.

2.3.1. Empaquetamiento del audio

La señal de audio es dividida en segmentos de tiempo, cada segmento es muestreado, codificado y enviado en un paquete por separado. En [1] se señala que los parámetros de uno o más segmentos de audio se encapsulan en un paquete de nivel transporte para enviarse a través de la red. El emisor genera periódicamente los paquetes; la duración del intervalo de generación de paquetes depende de la cantidad y del tamaño de segmentos de voz que lleva cada paquete de nivel de transporte. En el caso de codificación G.729, este intervalo es de 20 ms, que corresponden a dos segmentos de voz de 10 ms. Por otro lado, más adelante veremos que las experimentaciones que llevamos a cabo en esta tesis, utilizan trazas de audio real que llevan audio codificado en PCM de 8 kHz y la unidad de tiempo de los paquetes es de 20 ms [20]. Estos paquetes se envían periódicamente en el tiempo e inyectados en la red [1, 37]. La Figura 2.3 ilustra como la señal analógica de voz es dividida en segmentos iguales de Δ tamaño.

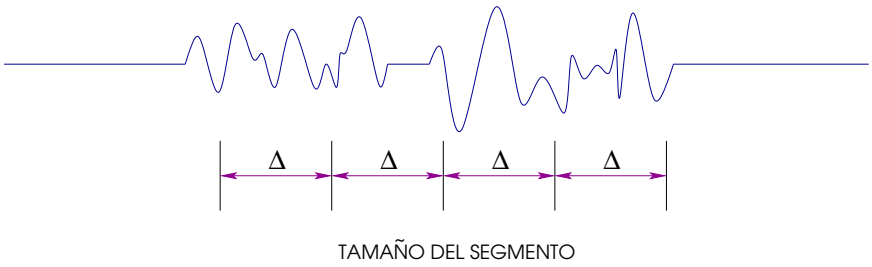

Figura 2.3: La señal analógica de voz se convierte en paquetes de audio

Un codificador puede esperar un poco antes de codificar un paquete con el objetivo de obtener mayor información de la señal de audio, lo cual puede mejorar la codificación, esto se llama *look ahead*. Existen muchos tipos de codificadores con diferentes tasas de codificación y diferentes calidades. Uno de los codificadores más usados es ITU-T G.711, el cual es de tipo PCM y codifica a una tasa de 64 kbits/s, otros ejemplos son ITU-T G.727 y G.729 que son algunos de los codificadores con más alta calidad intrínseca [31].

2.3.2. Detección de actividad de voz

El tráfico generado por una transmisión de voz se divide en periodos de actividad llamados frases[3], y en periodos de silencio [29]. Para reducir la tasa de tráfico generado, la mayoría de los codificadores descartan las muestras obtenidas durante los periodos de silencio y únicamente codifican las muestras que se generan durante los periodos de actividad [1, 38].

La telefonía convencional utiliza un canal bidireccional con una velocidad de transmisión fija de 64 kbps, sin tomar en cuenta si hay actividad o silencio. Esto significa, que existe un porcentaje del total del ancho de banda que se desperdicia. Una de las ventajas de la VoIP es la posibilidad de utilizar ese ancho de banda "desperdiciado" por lo que es necesario detectar los periodos de actividad.

Un detector de actividad de voz, VAD[4], es utilizado por el emisor para diferenciar los periodos de actividad y silencio. Su finalidad es ofrecer una indicación de la presencia de actividad para facilitar el procesamiento de la voz, así como, la posibilidad de proveer delimitadores para el principio y el fin de las frases. Generalmente cuando el VAD detecta actividad, espera un tiempo determinado para el empaquetamiento.

Sin embargo, existen algunos problemas que esto conlleva, como determinar cuando la actividad termina o comienza. En la mayoría de los casos el comienzo de la sentencia es cortada. Este fenómeno se conoce como *front-end speech clipping*. Otro problema es diferenciar entre la actividad de voz y el ruido ambiental, es decir, cuando el VAD es incapaz de diferenciar entre el ruido generado por el ambiente y la voz. Esto es conocido como el umbral Señal a Ruido [5].

2.3.3. Codificadores de audio

En las aplicaciones multimedia los codificadores se usan para comprimir la señal de voz, imágenes o video, y de esta forma alcanzar una transmisión de señales entre fuente y destino confiable.

Uno de los parámetros de los codificadores de audio es su frecuencia de muestreo, que se elige de acuerdo al Teorema de Nyquist. Sea $f(t)$ una señal de audio, con componente máxima de frecuencia f_{max}. Si se toman muestras de $f(t)$ al menos al doble de su frecuencia máxima, entonces el Teorema de Nyquist establece que es posible recuperar la señal a partir de sus muestras [5].

La Unión Internacional de Telecomunicaciones - Estandarización de Telecomunicaciones, ITU-T, estandarizó el ACELP, MPMLQ, PCM, y ADPCM en su recomendaciones de series G. Mostramos las tasas de calidad de los diferentes tipos de codificadores usando el E-model, que detallamos posteriormente en la Sección 2.6. Este modelo evalúa la calidad de la voz usando una medida subjetiva R que tiene un rango de 0 a 100, la mejor tasa equivale a la mejor calidad. La calidad del audio es aceptable si está dentro del rango de 70 a 100,

[3]En inglés *talkspurt*

[4]Voice Activity Detection

esto quiere decir que la calidad es equiparable con la que ofrece PSTN [48, 31]. La Tabla 2.1 resume algunas características de los codificadores comúnmente utilizados.

Tabla 2.1: Codificadores de voz [31]

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

Aplicaciones como Skype, Google Talk, Yahoo!, Messenger, y MSN Messenger utilizan el estándar G.729A, que está pensado para tolerar la pérdida de paquetes [5].

2.3.4. Codificación del audio

Usualmente, una señal de audio se convierte a una señal digital de la siguiente manera [14, 5]:

- La señal analógica es muestreada a una tasa fija. El valor de cada muestra es un número real dado.
- Cada muestra es entonces "redondeada" a uno de los valores de números finitos. Esta operación es referida como cuantificación. 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.

Esta señal digital puede entonces ser convertida a una señal analógica. Sin embargo, esta señal decodificada es usualmente distinta a la señal audio original. Mediante el incremento de la tasa de muestreo y el número de valores de cuantificación la señal decodificada puede aproximarse con mayor exactitud a las señal analógica original. Así, existe un compromiso entre la calidad de la señal decodificada y los requerimientos de ancho de banda y de almacenaje de la señal digital. Cada codificador utiliza un método diferente para la compresión de la voz.

2.3.5. Eco

El eco es un fenómeno relacionado con la reflexión del sonido. En la telefonía, se da cuando el interlocutor puede escuchar su propia voz durante una sesión de audio, es decir, el sonido regresa a la fuente. Este fenómeno es tolerable, siempre y cuando, el retardo de extremo a extremo sea menor a 25 milisegundos, de lo contrario puede causar interrupciones y pérdida de la interactividad. Por supuesto, mientras más ruidoso y largo es el eco más molesto se vuelve [5].

Para combatir los efectos del eco, las aplicaciones implementan canceladores del 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 [5].

2.4. Clases de aplicaciones multimedia

Las aplicaciones multimedia presentan la información empleando una combinación de al menos dos soportes de la comunicación como pueden ser texto, sonidos, voz, imágenes, video, y animación. En la Figura 2.4 se aprecian los elementos de las transmisiones multimedia sobre Internet. Por un lado tenemos el tráfico generado por el contenido en vivo y el contenido a la demanda como música, vídeos, archivos, fotografías, transmisiones radiofónicas, y por supuesto llamadas telefónicas. Todo este tráfico debe pasar por la misma red, desde los servidores hasta los usuarios finales pasando por diversos tipos de redes ya sean alámbricas, inalámbricas, locales, etc.

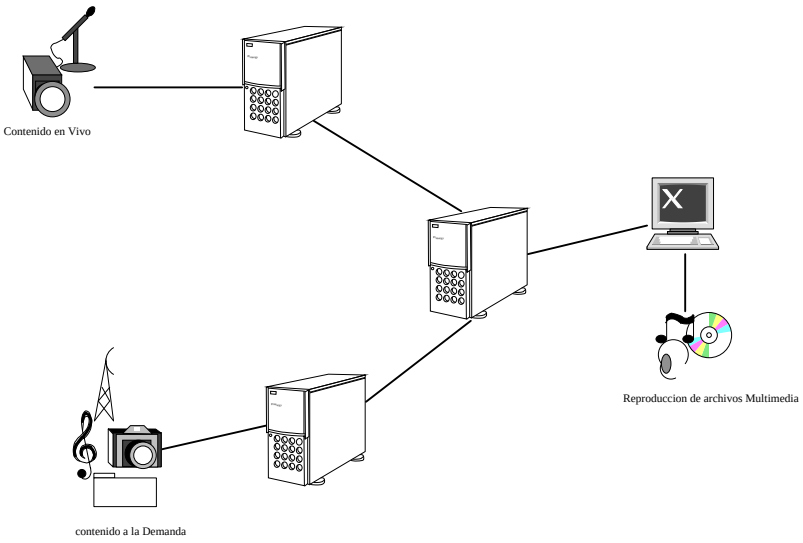

Figura 2.4: Transmisión de Aplicaciones Multimedia sobre Internet

Estas aplicaciones se pueden clasificar en [14]:

- **Streaming de audio y video**: los clientes pueden solicitar en cualquier momento los archivos de audio y video los cuales están almacenados en un servidor. Los clientes sólo reciben mientras se transmite la información. El archivo no necesita descargarse completo para iniciar su reproducción. Ejemplo de estas aplicaciones son los contenidos que se encuentran en sitios de Internet como YouTube donde existe un servidor que contiene una colección de archivos de audio y video que ponen a la disposición del público; así un cliente obtiene el archivo y puede comenzar su reproducción una vez que tiene almacenada suficiente información en la memoria temporal.

- **Unidireccionales en tiempo real**: similar a una transmisión de radio o televisión sólo que se transmiten sobre Internet, por lo que la transmisión puede venir de cualquier lado del mundo, donde un servidor envía la información y los clientes se abonan para recibirla. Existen diferentes radiodifusoras que ofrecen la opción de escuchar la *radio en línea*. Una vez que el cliente desea escuchar va a la página de la estación deseada y comienza a escuchar el contenido que se transmite en ese momento, es decir, si desea escuchar un programa que ya empezó, no hay posibilidad de regresar al principio del programa o de adelantar para escuchar el siguiente programa.

- **Interactivas**: permiten a los usuarios comunicarse entre ellos en tiempo real o con el servidor. En este tipo de aplicaciones se encuentra la voz sobre Internet, también conocida como telefonía sobre Internet, debido a que existe reciprocidad en la comunicación tal y como existe en la telefonía convencional.

2.5. Transporte de Voz sobre IP

El mecanismo de control de congestión de TCP intenta limitar cada conexión TCP para compartir el ancho de banda de manera justa. El problema de la tasa de transmisión puede tener un efecto dañino en las aplicaciones de tiempo real de audio y video que tienen un requerimiento mínimo de ancho de banda. Sobre todo, las aplicaciones en tiempo real son tolerantes a pérdidas y no necesitan un servicio de transporte totalmente confiable. De hecho, los ACK y las retransmisiones de TCP pueden disminuir la tasa de transmisión de los datos en tiempo real. Por estas razones, las aplicaciones en tiempo real usan tradicionalmente el protocolo UDP ya que estas aplicaciones no pueden transmitir con TCP.

Además de lo anterior, existen otros motivos por los que las aplicaciones de tiempo real no utilizan TCP:

- TCP introduce variabilidad en el retardo debido a las retransmisiones y al reordenamiento de los paquetes recibidos.

- TCP introduce también, una tasa de variabilidad debido al control de congestión

- Las aplicaciones de audio son sensibles a la pérdida de paquetes y al retardo de extremo a extremo ya que ambos fenómenos impactan directamente en la QoS.

Así, UDP es una mejor opción ya que no se establece una conexión antes de enviar los paquetes, no se necesita recibir un acuse de recibo para enviar el siguiente paquete, no implementa algoritmos para el control de la congestión, el encabezado de los paquetes es más pequeño. Estas claras ventajas también provocan que sea necesario implementar otro tipo de mecanismos para tratar las pérdidas, el retardo, y la variabilidad en el retardo, porque UDP no ofrece el servicio para detectar si hay paquetes perdidos ni una indicación de cuales paquetes no fueron recibidos.

2.5.1. Protocolos y estándares

2.5.1.1. RTP

El RTP, *Real Time Protocol* definido en el RFC 1889 y actualizado en el RFC 3550, es un protocolo que se encarga de ofrecer un servicio de entrega para el tráfico de las aplicaciones con características de tiempo real, como los formatos WAV o GSM para sonido y MPEG1 o MPEG2 para video. Estos servicios incluyen la identificación del tipo de carga, número de secuencia, estampas de tiempo, sincronización Intra-media, Fuente única/Identificador de sesión, cifrado, y monitoreo de entrega [38].

Los primeros 12 octetos del encabezado RTP existen en todos los paquetes RTP, estos campos tienen el siguiente significado,

- *version (V)*: 2 bits. Identifica la versión de RTP utilizada. El valor 1 para la versión, es usado para la primera versión de RTP y el valor 0 es usado por el protocolo usado inicialmente en la herramienta de audio VAT [38].

- *padding (P)*: 1 bit. Pude necesitarse para algunos tipos de algoritmos de cifrado con tamaño de bloques fijo o para llevar algunos paquetes RTP en una capa más baja de protocolo.

- *extension (X)*: 1 bit. El encabezado fijo debe seguirse por exactamente una extensión de encabezado con un formato específico.

- *CSRC count (CC)*: 4 bits. El contador CSRC contiene el número de identificador CSRC que sigue al encabezado fijo. Si la cuenta CSRC es cero, entonces la fuente de sincronización es la fuente de la carga útil.

- *marker (M)*: 1 bit. El segmento actual es el inicio de una frase (ajuste del retardo de playout).

- *payload type (PT)*: 7 bits. Identifica el formato de la carga RTP y determina su interpretación mediante la aplicación. Provee 128 tipos diferentes de codificación como PCM, MPEG2, etc.

- *sequence number*: 16 bits. El número de secuencia se incrementa en uno por cada paquete RTP enviado, y puede ser usado por el receptor para detectar paquetes perdidos y para restaurar la secuencia de los paquetes.

- *timestamp*: 32 bits. Las estampas de tiempo reflejan el instante en el que el primer octeto de un paquete RTP es muestreado. Se usa para remover la variabilidad en el retardo introducido por la red y para la sincronización Inter-media.

- *Synchronization Source identifier (SSRC)*: 32 bits. Identifica la fuente de sincronización. Este identificador debe ser escogido aleatoriamente para evitar la duplicación de identificadores. En caso de asignar un identificador ya existente, lo detecta y anuncia la baja del viejo identificador para después escoger uno nuevo. El campo SSRC ayuda a distinguir una fuente de audio dentro de una sesión.

- *Contributing Source identifiers (CSRC) list*: 0 a 15 objetos de 32 bits cada uno. Identifica la contribución de la fuente por el contenido de la carga en el paquete.

RTP sólo provee un encabezado que contiene información acerca del contenido de un paquete de audio, no asegura la Calidad de Servicio. RTP se puede ver como una mejora al protocolo UDP para las aplicaciones de audio.

2.5.1.2. RTCP

RTP trabaja conjuntamente con RTCP, *Real Time Control Protocol*, cuya principal función es proveer una retroalimentación sobre la calidad de la distribución de la información. RTCP lleva un identificador persistente para una fuente RTP a nivel de la capa de transporte, llamado *CNAME*. Además monitorea el número de participantes, utilizando este número para calcular la tasa a la cual se envían los paquetes. Una cuarta función es hacer llegar la información mínima de control de la sesión[14, 38].

Los tipos de reporte que genera RTCP son [38]:

- *Sender report (SR)*: Número de bytes enviados, así el receptor puede calcular la tasa de recepción, y Tiempo universal, posibilita al receptor la sincronización entre emisores.

- *Receptor report (RR)*: Número de paquetes recibidos, para estimar la tasa de paquetes perdidos y la variabilidad en el retardo. Permite al emisor calcular el RTT a través de las estampas de tiempo SR y RR.

- *Source description (SDES)*: Contiene el CNAME que identifica al usuario en la sesiones multimedia.

- *Explicit leave (BYE)*: Una aplicación envía un mensaje BYE cuando deja la sesión, también se usa cuando existe una colisión de identificadores.

2.6. Evaluación de la calidad de servicio

Para evaluar la calidad de servicio en una sesión de VoIP existen diferentes enfoques. El primer enfoque es por medio de la reducción de la tasa de pérdidas y de la variabilidad en el retardo de una aplicación, y manteniendo estos parámetros dentro de los niveles tolerables; de esta manera podemos medir, por ejemplo, la calidad de servicio a través de la tasa de pérdidas promedio y el retardo de playout promedio de una sesión dada. En este caso, decimos que la QoS es medida *objetivamente*, ya que una cantidad medible es reportada por la aplicación para indicar la calidad vista durante una sesión. Otro enfoque es por medio de la opinión del usuario final. En este caso una sesión multimedia es grabada. Hay dos señales para ser comparadas: la transmisión de la señal original, y la señal recibida deteriorada por las pérdidas, el retardo y la variabilidad en el retardo. La señal original es conocida como señal de referencia. Entonces, un grupo de usuarios, escuchan las dos señales y deben compararlas de manera *subjetiva* [32]. En esta sección describimos los enfoques más utilizados.

2.6.1. E-model

El E-model es un modelo común del ITU-T para evaluar los efectos combinados de las variaciones de diversos parámetros de transmisión que afectan a 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. 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 [48].

Este modelo estima la calidad de la comunicación de la conversación de boca a oído percibida por el usuario en el lado de la recepción, como oyente y como hablante.

Existen tres parámetros diferentes asociados con el tiempo de transmisión. El retardo absoluto Ta representa el retardo total en un sentido entre el emisor y el receptor y se utiliza para estimar la degradación debida a retardos demasiado largos. El parámetro de retardo promedio en un sentido, T, representa el retardo entre el lado de recepción, en estado hablante, y el punto de una conexión en el que aparece un acoplamiento de señales como una fuente de eco. El retardo de ida y vuelta, Tr, representa degradaciones debidas al eco para el oyente.

El cálculo del factor de determinación de índices de transmisión, R, es una combinación de todos los parámetros de transmisión pertinentes para la conexión considerada. Este factor R está constituido por:

$$R = Ro - Is - Id - Ie - eff + A(3-1)$$

donde 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 el retardo y el factor de degradación efectiva del

equipo, *Ie-eff*, representa las degradaciones producidas por 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[48]. El E-model transforma las medidas objetivas en calidad subjetiva, el rango de valores correspondientes a las Categorías de Calidad de la transmisión de voz, como aparece en la Tabla 2.2 [31].

Tabla 2.2: Rangos del valor R en el E-model [31]

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

2.6.2. PESQ

La Evaluación de la calidad vocal por percepción, PESQ[5], es un método para predecir la calidad subjetiva de microteléfonos y codificadores de banda estrecha, tomando en cuenta el filtrado, la variabilidad en el retardo 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 [47].

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 retardos entre la entrada inicial y la salida degradada, uno para cada intervalo de tiempo cuyo retardo presenta una diferencia significativa en comparación con el del intervalo de tiempo precedente. Para cada uno de estos intervalos se calcula un punto de arranque y un punto de parada correspondientes. El algoritmo de alineación se basa en el principio de comparación entre la confianza de dos retardos en un cierto intervalo de tiempo y la confianza de tener un solo retardo para ese intervalo. El algoritmo puede tratar los cambios en el retardo tanto durante periodos de silencio como durante periodos de habla activa [47].

Sobre la base del conjunto de retardos que se encuentran, PESQ compara la señal de entrada con la salida degradada alineada del dispositivo sometido a prueba, utilizando un modelo por percepción. 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 del nivel a un nivel de escucha calibrado, correspondencia entre el dominio de la

[5] *Perceptual evaluation speech quality measure*

frecuencia y el dominio del tiempo, curvatura de frecuencia, y aplicación de escala compresiva de la sonoridad [47].

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 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 una MOS de calidad de escucha objetiva [47].

2.6.3. MOS

El método de Nota Media de Opinión, MOS[6], fue creado para determinar subjetivamente la calidad de transmisión. La idea principal es reproducir las condiciones de una conversación telefónica real en un laboratorio, y hacer que un grupo de gente 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 el Tabla 2.3 [46].

Tabla 2.3: Notas de degradación de MOS [31, 46]

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].

2.7. Conclusiones preliminares

En este capítulo hemos estudiado los conceptos necesarios para comprender la telefonía IP, así como también, las principales diferencias entre la telefonía convencional y la

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

telefonía sobre Internet. Abordamos el funcionamiento de Internet, los tres fenómenos que conlleva el uso del servicio "best-effort" del protocolo IP y la utilización del protocolo UDP sobre TCP para las aplicaciones en tiempo real. Además describimos el procesamiento de la señal analógica de voz para transmitirla digitalmente por Internet, las clases de aplicaciones multimedia y algunos protocolos y estándares utilizados en la telefonía IP. Por último, hemos presentado en este capítulo tres formas de evaluar la calidad de servicio en las aplicaciones de audio sobre IP como son el E-model, PESQ y MOS.

En el siguiente capítulo describiremos como los fenómenos de retardo, pérdidas y variabilidad en el retardo, afectan la QoS en la VoIP y algunos de los mecanismos más utilizados para mitigarlos.

Capítulo 3

Fenómenos que afectan la calidad de servicio en VoIP

Dos características de la señal de voz son la claridad y la fidelidad. La claridad desde el punto de vista del usuario se refiere a una neta comprensión de la voz o el audio recibido, mientras que la fidelidad representa qué tanto se parece la voz o el audio recibido con respecto al original. Esta definición de claridad se ve entonces como un parámetro subjetivo en la calidad de la conversación, en tanto que la fidelidad se ve como un parámetro objetivo. En la telefonía sobre Internet el retardo, las pérdidas y la variabilidad en el retardo, son los tres factores más importantes que afectan estas características y por lo tanto impactan directamente en la calidad de servicio, por lo que, las redes IP deberían mejorarse con mecanismos que aseguren la calidad de servicio requerida para el transporte de voz [22]. A continuación introducimos cómo estos fenómenos impactan a las aplicaciones de audio sobre Internet y algunos mecanismos para mitigar sus consecuencias.

3.1. Retardo

El retardo es comúnmente medido en segundos desde el momento en que el emisor comienza una palabra hasta que el receptor escucha totalmente esa palabra. Esto se conoce como retardo *de boca a oído* [22] o *retardo de extremo a extremo*. Uno de los principales desafíos de la transmisión en tiempo real de voz en una red conmutada por paquetes es como se pretende superar los efectos causados por el retardo de extremo a extremo encontrado.

3.1.1. Tipos de retardo

El retardo de extremo a extremo consiste en tres principales componentes, el retardo de empaquetamiento y playout, retardo de transporte, y el retardo de propagación. El retardo de empaquetamiento incluye el tiempo que toma el retardo de codificación, de decodificación, compresión, y decompresión. Los diferentes componentes del retardo de extremo a extremo de VoIP son [22, 44, 14]:

- **Retardo de empaquetamiento**: El tiempo que toma grabar las muestras de voz y ponerlas en segmentos. El valor promedio de este retardo está entre 10 y 30 milisegundos por paquete en función del hardware.
- **Retardo de codificación**: El tiempo necesario para transformar el segmento de voz a su representación bits, entre 5 y 10 milisegundos es el tiempo promedio.
- **Retardo de compresión**: El tiempo requerido para comprimir los segmentos de voz, colocarlos en paquetes, e inyectarlos en la red. Entre 5 y 10 milisegundos.
- **Retardo de red**: El intervalo de valores está entre 70 y 120 milisegundos. Este retardo está compuesto por:
  - **Retardo de procesamiento**: El tiempo que le toma al ruteador procesar el encabezado de cada paquete.
  - **Retardo de las filas de espera**: El tiempo que los paquetes esperan en las filas de los ruteadores para ser atendidos, cabe recordar que política que manejan los ruteadores es FIFO.
  - **Retardo de transmisión**: El tiempo que tarda en poner todos los bits de un paquete en el enlace.
  - **Retardo de propagación**: El tiempo que le toma a la señal a nivel físico propagarse por el medio.
- **Retardo de decodificación**: El tiempo que toma descomprimir los paquetes entre 5 y 10 milisegundos
- **Retardo de playout**: El tiempo que esperan los paquetes en el buffer del playout desde que son recibidos hasta que son reproducidos. Entre 50 y 200 milisegundos

En total tenemos un promedio de entre 150 y 400 milisegundos para los valores del retardo de extremo a extremo [44]. La Figura 3.1 muestra los componentes del retardo de extremo a extremo.

Las aplicaciones en tiempo real son sensibles al retardo de extremo a extremo; si este retardo es muy grande la interactividad comienza a ser pobre, ya que el retardo trae consigo el problema del eco. Si tomamos en cuenta que en una red convencional PSTN el retardo está virtualmente siempre debajo de 150 ms y que el oído humano puede tolerar un retardo de

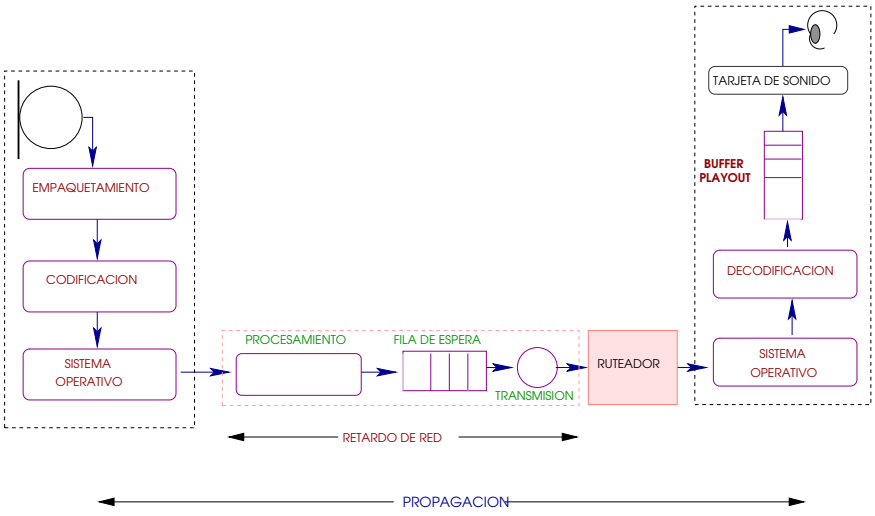

Figura 3.1: Fuentes del retardo

hasta 400 ms [34, 20, 10], entonces el valor promedio del retardo de extremo a extremo antes mencionado se encuentra dentro de este intervalo. En [24], los autores realizan un estudio reciente donde muestran que los niveles de satisfacción más altos se obtienen cuando el retardo de extremo a extremo promedio es menor a 100 milisegundos, además en [48], la ITU-T presenta la relación entre el retardo y el nivel de satisfacción del usuario. Aunado a esto los algoritmos para la cancelación del eco son necesarios, estos algoritmos son similares a los utilizados para incorporar enlaces satelitales y transoceánicos en PSTN. El retardo sólo es un problema para las aplicaciones en tiempo real, y no así, para las aplicaciones no interactivas.

Para el retardo en las filas de espera de los ruteadores es posible tener una solución, agregar más ancho de banda para absorber el tráfico y hacer desaparecer las filas de espera para todos y dar prioridad a los paquetes de audio sobre los paquetes de datos en los ruteadores [31].

3.1.2. Picos de Retardo

Un *pico de retardo* constituye un repentino y gran aumento en el retardo de extremo a extremo, seguido de una serie de llegadas de paquetes casi simultáneas [30]. En [2], Bolot identifica este fenómeno como resultado de un error de depuración en las operaciones del software de los ruteadores. En la Figura 3.2 observamos como la mayor concentración de paquetes tiene un retardo menor a 0.5 segundos, excepto algunos, cuyo retardo es repentinamente mucho mayor, y después de ellos los paquetes vuelven a tener un retardo promedio.

Los picos de retardo afectan severamente la QoS en VoIP, junto a la variabilidad en el retardo, son los problemas principales que deben resolver los algoritmos de control de retardo de playout. En el caso de los picos de retardo, es posible modificar los algoritmos de control de retardo de playout para que sean capaces de reaccionar cuando estos picos

aparecen, como abordamos en el siguiente capítulo.

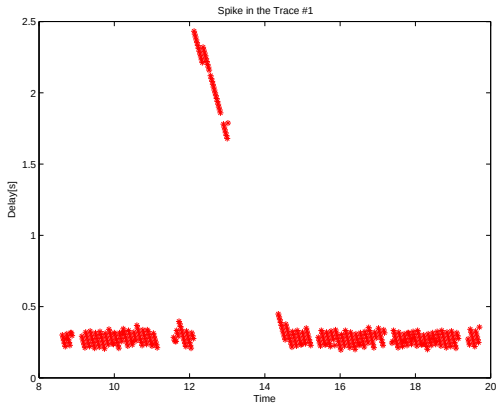

Figura 3.2: Ejemplo de un pico de retardo

3.2. Pérdidas

Los paquetes pueden perderse por diferentes razones como la congestión, los errores de transmisión (e. g. los causados por interferencias cuando se atraviesan redes inalámbricas) y los errores de enlace (e. g. cuando hay errores en la gestión de la comunicación). Además de este tipo de pérdidas, existe otro tipo llamado *pérdidas artificiales*, que se presentan cuando el retardo extremo a extremo de los paquetes es excesivo, por lo que estos paquetes son inútiles para la sesión de audio, sin importar que lleguen correctamente al receptor.

La aplicaciones de audio toleran cierta tasa de pérdidas sin impactar la calidad de la voz. En [34, 20, 10] señalan que hasta el 5 % de pérdidas es tolerable, no obstante en [24], los autores indican que las tasas menores al 3 % obtienen los más altos niveles de satisfacción. Cabe destacar que estos límites dependen del codificador que se utilice.

Los mecanismos para tratar las pérdidas en VoIP se clasifican como Corrección de Error Proactiva, FEC[1], y Corrección de Errores Reactiva que son los más utilizados en las aplicaciones en audio en tiempo real [27].

3.2.1. Corrección de Errores Proactiva

Aunque existen varios tipos de FEC, en general, este tipo de mecanismos consisten en enviar información redundante de menor calidad del paquete n el paquete $n+1$, tal como ilustra la Figura 3.3. De ésta manera en caso de perderse el paquete n se puede recuperar la información al llegar el paquete $n+1$, lo cual disminuye la tasa de pérdidas de paquetes. Observamos también que la redundancia está codificada a una tasa menor que el paquete

[1]Por sus siglas en inglés *Foward Error Correction*

original. Si bien es cierto que no se recupera el paquete original, el reproducir un paquete a una menor tasa de codificación es mejor que no reproducir nada.

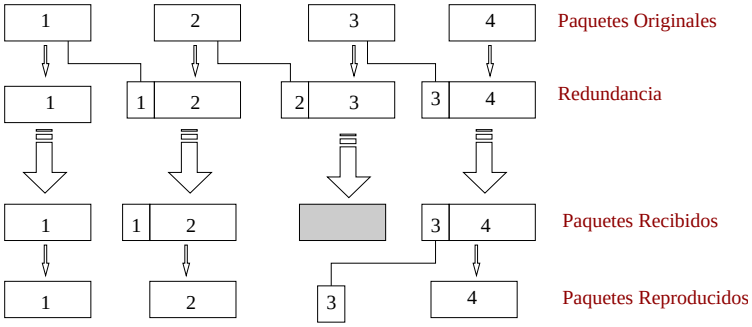

Figura 3.3: Ejemplo del mecanismo FEC

Si bien FEC disminuye las pérdidas, se debe tomar en cuenta que el uso del FEC conlleva aumentar el tamaño de cada paquete y con esto se incrementa el ancho de banda que se consume para la transmisión. Asimismo, en caso de perder el paquete *n* se tendría que esperar a que llegue el paquete *n+1* para recuperarse de la pérdida, por lo tanto se necesita que el tiempo de espera aumente provocando un incremento del retardo de extremo a extremo [3].

Los mecanismos FEC son más eficientes cuando la cantidad de pérdidas es pequeña y cuando son independientes, la razón de esto es que cuando hay pérdidas continuas, por ejemplo, si se pierde el paquete *n* y se pierde el paquete *n+1*, no hay manera de recuperar el paquete original o su copia. Para esto se puede modificar el mecanismo básico de FEC aumentando el número de redundancias en los paquetes subsecuentes e. g., se envía redundancia de *n* en los paquetes *n+1, n+2, n+3*, etc [3].

Muchos codificadores pueden recuperarse al perder algunos pocos paquetes, sin embargo, cuando se presentan *ráfagas de pérdidas*[2], el deterioro en la calidad de la transmisión es muy notable, y el mecanismo FEC ofrece menos beneficios comparado con el mecanismo *Interleaving* [20].

Para minimizar el daño por las ráfagas de pérdidas, los paquetes pueden intercalarse. De esta manera si existe una ráfaga de pérdidas, las pérdidas en el receptor no serían consecutivas y por lo tanto serían más sencillas de ocultar, así, la probabilidad de recuperarse de las pérdidas aumentaría [14, 27]. Herramientas de investigación como RAT [36] del proyecto MICE implementan esta técnica. Observamos en la Figura 3.4 como los paquetes son intercalados en el emisor, enviados, y reordenados en el receptor.

3.2.2. Corrección de Errores Reactiva

ARQ, Automatic Repeat reQuest, es un mecanismo usado para recuperarse de las pérdidas haciendo una solicitud explícita a la fuente. Este tipo de mecanismos reactivos no es usado

[2] *burst*

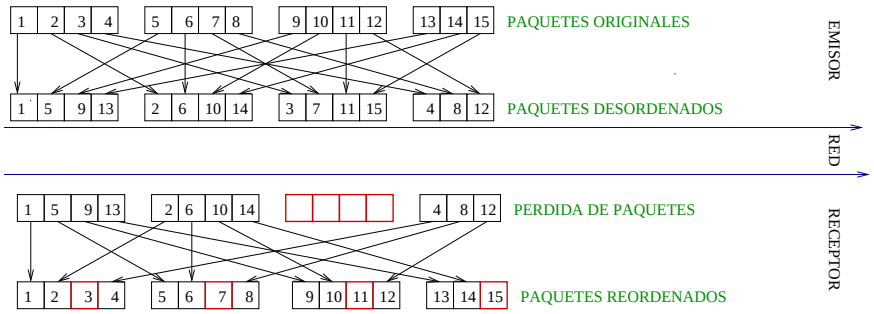

Figura 3.4: Los paquetes son intercalados en el emisor y reordenados en el receptor

conjuntamente en VoIP dado que introduce retardo adicional, para lo cual este tipo de aplicaciones son casi intolerantes. Otro tipo de corrección reactiva consiste en tratar de ocultar los espacios que quedan con la pérdida de paquetes. Existen diversas técnicas para lograr esto, de las cuales se habla detalladamente en [27]. Entre las cuales se destacan:

- **Esquema basado en la inserción**: el espacio que deja el paquete perdido se corrige agregando silencio o ruido. También se puede repetir la reproducción del último paquete recibido o simplemente se juntan los paquetes anterior y posterior al paquete perdido.
- **Esquema basado en interpolación**: incluye la modificación de la escala de tiempo donde se expanden los paquetes para ocultar la pérdida.

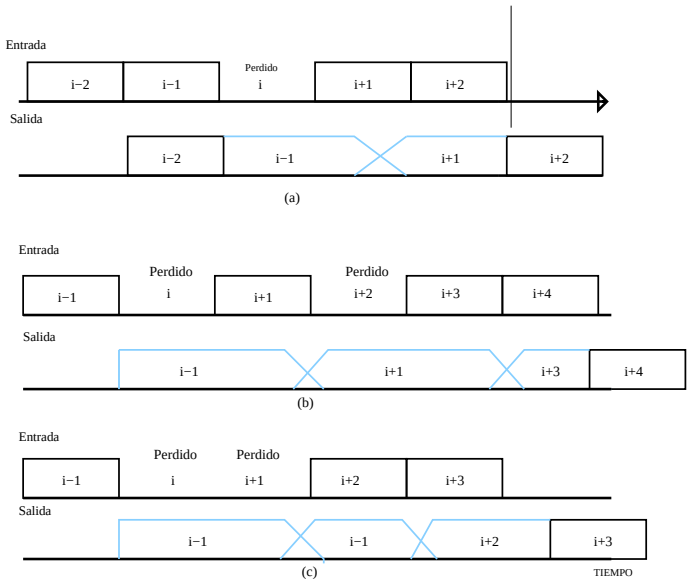

Figura 3.5: Ejemplo de Correción de Errores Reactiva

En la Figura 3.5 se pueden apreciar algunos ejemplos de los esquemas antes descritos, donde observamos la modificación en la escala de tiempo de los paquetes *i-1* e *i+1* que se expanden para corregir el espacio dejado por *i*. En la Figura 3.5b se utiliza el mismo mecanismo para reparar pérdidas no consecutivas, y en la Figura 3.5c se usa el mismo mecanismo más la repetición del paquete *i-1* para corregir pérdidas consecutivas [35].

3.3. Variabilidad en el retardo

La fuente envía los paquetes de audio igualmente espaciados en el tiempo y los inyecta en la red; sin embargo, el retardo en Internet es variable por diferentes causas entre ellas que puede ser distinto el tiempo en las filas de espera para cada paquete, los paquetes pueden tomar distintas rutas, pueden haber retransmisiones en los enlaces, etc. Entonces, al momento que los paquetes llegan al receptor, la mayoría de ellos han perdido el intervalo regular que había entre ellos al momento de salir del emisor.

En otros casos, los paquetes puede llegar en desorden o en el peor de los casos no llegar, por estas razones los paquetes de las sesiones de VoIP recibidos no pueden ser reproducidos en el momento que llegan. Los paquetes deben esperar en el receptor para su reordenamiento y reespaciamiento en una memoria intermedia conocida como *buffer*, y así atenuar las consecuencias de la variabilidad en el retardo.

La diferencia entre el espacio del tiempo de transmisión y el espacio del tiempo de recepción se le conoce como variabilidad en el retardo o *jitter*, esto se ilustra en la Figura 3.6. Este problema se puede resolver, al agregar retardo a los paquetes, este retardo se le conoce con el nombre de retardo de *playout* o *dejittering delay*. El retardo de *playout* de un paquete recibido i, es hasta el tiempo $S_i + C$ donde C es una constante. Si el paquete llega después de su tiempo límite, entonces se descarta, y se espera por el siguiente paquete.

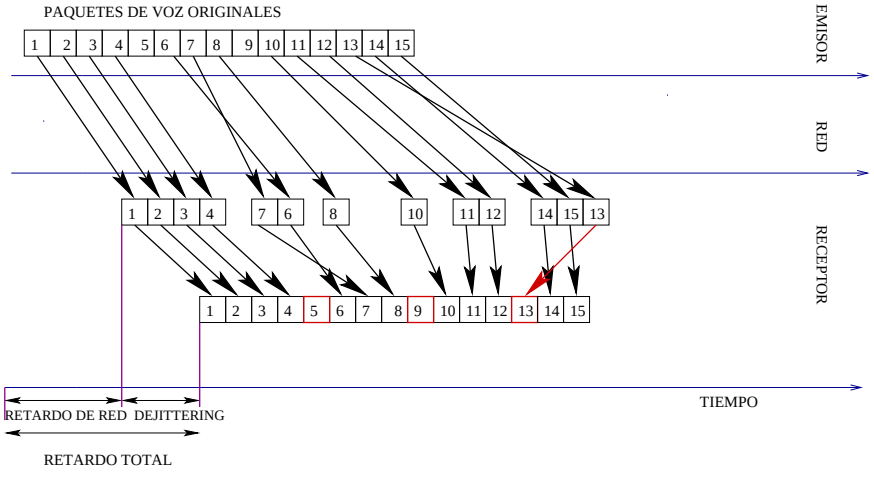

Figura 3.6: Ejemplo de variabilidad en el retardo

3.4. Conclusiones preliminares

En este capítulo se estudiaron las razones por las que el retardo, las pérdidas, y la variabilidad en el retardo son los principales fenómenos que afectan la calidad de servicio en las sesiones de VoIP. Estos tres fenómenos impactan directamente a la interactividad, a la claridad y a la fidelidad. Por estas razones, se deben implementar mecanismos que aseguren la calidad requerida para las aplicaciones de VoIP.

Además se estudiaron algunos mecanismos para disminuir o eliminar los problemas causados por ellos, principalmente los mecanismos implementados de extremo a extremo más utilizados en la aplicaciones multimedia.

En el siguiente capítulo veremos los diferentes tipos de algoritmos de control de retardo de playout, sus características, clasificación, y funcionamiento, para luego tratar en el Capítulo 5 la implementación, comparación, propuesta de un nuevo algoritmo y los resultados obtenidos.

Capítulo 4

Algoritmos de control de retardo de *play-out* en VoIP

El retardo, la variabilidad en el retardo y la pérdida de paquetes en las redes de conmutación de paquetes, como Internet, son los factores que impactan la calidad del audio de las aplicaciones multimedia interactivas. El impacto causado por estos fenómenos debe ser minimizado a través del empleo de mecanismos de control de extremo a extremo. La Figura 4.1 ilustra estos tres factores y su efecto en las aplicaciones de audio, en donde podemos observar la razón por la que los paquetes no pueden ser reproducidos en el momento que llegan al receptor.

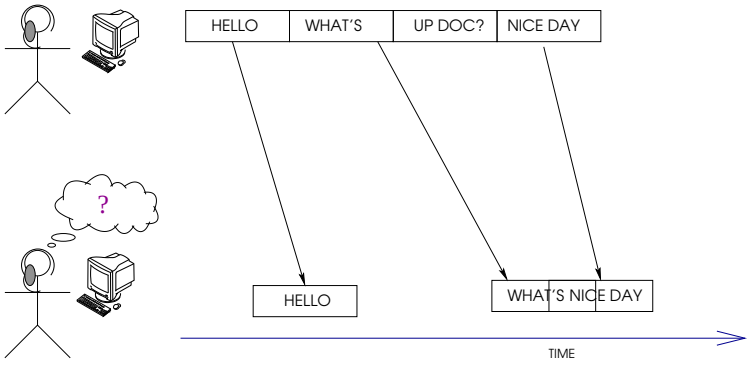

Figura 4.1: Efecto del *jitter*: los paquetes no pueden ser reproducidos en el momento que llegan

Los paquetes de audio encuentran retardo variable cuando cruzan Internet, esto es debido principalmente a que el tiempo es variable en las filas de espera de los ruteadores. Así, la variabilidad en el retardo modifica la forma periódica original del audio transmitido. Con

el fin de utilizar el audio recibido, una aplicación debe reducir o eliminar esta variabilidad, por medio de un tiempo de espera para los paquetes en el receptor y utilizándolos después de cierto tiempo límite. Los paquetes que llegan después de ese tiempo límite son considerados perdidos y por lo tanto no son reproducidos. Si el tiempo de espera en el receptor aumenta, la probabilidad de que los paquetes lleguen antes de su tiempo límite programado también aumenta. Esto reduce el número de paquetes eliminados artificialmente en el receptor.

El tiempo que tarda cada paquete desde que se genera en el emisor hasta que se reproduce en el receptor se le conoce como **retardo de playout**. El tiempo de *playout* es el momento en que cada paquete se reproduce.

Las aplicaciones de audio emplean Algoritmos de Control de Retardo de *Playout* para reducir o eliminar los efectos de la variabilidad en el retardo. Estos algoritmos funcionan en el lado del receptor como un mecanismo de control de extremo a extremo. Los algoritmos de control de retardo de *playout* implementan una memoria intermedia o *buffer* donde los paquetes esperan hasta su tiempo de *playout*. Los algoritmos deben calcular el tiempo límite para cada paquete, así como, el tiempo de *playout* de los paquetes. De esta forma, se busca aumentar el número de paquetes recibidos y dar tiempo para el reordenamiento de los paquetes y restablecimiento del espacio temporal entre ellos.

Sin embargo, un retardo de *playout* excesivamente grande tiene un impacto negativo en la interactividad de la sesión de audio. Obviamente, existe un compromiso entre el retardo y la tasa de pérdidas debida a los paquetes tardíos.

4.1. Compromiso entre pérdidas y retardo

El atenuar la variabilidad en el retardo añadiendo retardo parecería una idea contradictoria, ya que tanto el retardo como la variabilidad en el retardo afectan directamente la calidad en la transmisión de voz. Si el tiempo de espera de los paquetes en el buffer del receptor es muy grande, con el propósito de tener el menor número de pérdidas posible, se incrementa el retardo de extremo a extremo y, como hemos mencionado, un mayor retardo afecta la interactividad. Asimismo, si el tiempo de espera de los paquetes en el buffer es muy pequeño se incrementa el número de pérdidas en el receptor. Es por eso que existe un compromiso entre las pérdidas y el retardo.

Un buen algoritmo de control de retardo de *playout* en Voz sobre Internet, debe ser capaz de minimizar el tiempo de espera de los paquetes y la tasa de pérdidas de paquetes [24]. En otras palabras, si un algoritmo de control de retardo de playout es eficiente, éste toma en cuenta el compromiso entre las pérdidas y el retardo, de tal manera, que mantenga estos dos parámetros lo más bajos posibles. Para una sesión de audio interactiva, el retardo de los paquetes debe ser menor a 400ms y la tasa de pérdidas debe encontrarse entre 5 %–10 % [26], dependiendo del codificador utilizado.

Para optimizar este compromiso, existen varios enfoques de algoritmos de control de retardo de playout, los cuales podemos clasificar por la manera en la que calculan y gestionan el retardo de playout [17, 1, 13]:

- Algoritmos de control de retardo fijo
- Algoritmos de control de retardo adaptativo
  - Adaptativo por paquete
  - Adaptativo por frase
    - No tolerantes a Pérdidas
    - Tolerantes a Pérdidas
    - Basados en la Calidad del Servicio (QoS)

Estos algoritmos los detallamos a continuación.

4.2. Algoritmos de control del retardo fijo

Esta clase de algoritmos determinan un límite para el retardo de *playout* constante todo el tiempo que dure la sesión. El límite se basa en una estimación del tiempo que tarda un paquete en ir de extremo a extremo q, por lo tanto, si un paquete i se envía en el tiempo t, éste llega aproximadamente en el tiempo $t+q$. Para optimizar este límite se valora el tiempo máximo de retardo que puede haber sin afectar la calidad del servicio y así, se espera que lleguen la mayor cantidad de paquetes. El retardo de *playout* fijo se puede calcular tal que no más que una porción dada de los paquetes que llegan se pierdan debido al arribo tardío.

Otra manera de calcular el retardo 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 ésta medición fijar el retardo de *playout* óptimo.

Sin embargo, si el tiempo de espera es muy pequeño, la cantidad de paquetes eliminados artificialmente incrementa ya que el tiempo de *playout* no se adapta a las condiciones de la red disminuyendo considerablemente la interactividad en la conversación. Por otra parte, si se tiene un límite demasiado alto, los paquetes que lleguen en un tiempo adecuado, esperarían mucho tiempo en el buffer antes de ser reproducidos, aumentando así el retardo de *playout*. La Figura 4.2 muestra un ejemplo de la distribución del retardo de extremo a extremo de los paquetes; observamos que el tiempo de *playout* se mantiene constante durante toda la sesión. También observamos que los paquetes que llegan antes del tiempo límite son reproducidos al llegar su tiempo de *playout* y que los paquetes que llegan después del tiempo límite son considerados como perdidos.

En [12], se presenta un algoritmo que pertenece a esta clase. El retardo de playout de la llamada se determina en la fase inicial de la llamada escogiendo el valor del retardo más bajo y controlando la relación retardo de playout/pérdidas. El algoritmo por llamada también estima el retardo de red promedio. Este enfoque guarda información relacionada a las llamadas previas y la usa para calcular el retardo de playout fijo en las llamadas futuras. El objetivo de este algoritmo por llamada es no reaccionar demasiado rápido a las pequeñas variaciones experimentadas en el retardo de red y evitar el costo de un retardo de *playout* grande.

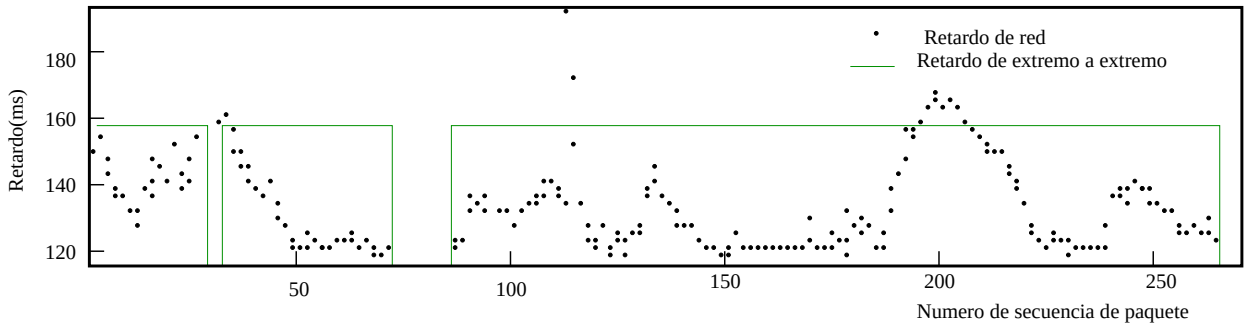

Figura 4.2: Ejemplo Algoritmos de retardo de playout fijo [17]

4.3. Algoritmos adaptativos de control del retardo

Los algoritmos de control de retardo adaptativo buscan mejorar las limitaciones de los algoritmos de control de retardo fijo, ajustando el retardo de *playout* de mejor forma a las condiciones que presenta la red.

Estos algoritmos utilizan diferentes enfoques para estimar el retardo de *playout*, actualizándolo al comienzo de cada frase, o bien modificándolo dentro de cada frase por cada paquete que llega al receptor, y los presentamos a continuación.

4.3.1. Algoritmos adaptativos por paquete

Los algoritmos de control de retardo adaptativo por paquete trabajan ajustando el retardo de *playout* dinámicamente a la llegada de cada paquete por medio de estimaciones de retardo obtenidas de cada uno de los paquetes. El ajuste se hace en cualquier periodo de la sesión incluso dentro de las frases; de esta forma puede adecuarse a cualquier condición que la red presente de manera precisa [1]. La Figura 4.3 muestra como el retardo de *playout* calculado sigue de cerca la distribución del retardo de los paquetes y como la cantidad de paquetes perdidos debido al arribo después de su tiempo límite es pequeña.

Dado que los paquetes pueden tener diferentes retardos de *playout*, es necesario 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*[1] basada en el algoritmo WSOLA, Waveform Similarity Overlap-Add. El algoritmo WSOLA descompone la entrada en segmentos sobrepuestos de igual tamaño; estas entradas son realineadas y sobrepuestas en las salidas [17]. El algoritmo expande los paquetes y desaceleran la velocidad del playout para adaptarse a los incrementos repentinos en el retardo de la red.

En [16], Liang, Fäber y Girod trabajan con esta estrategia en una primera instancia, para calcular el tiempo de *playout* de cada paquete. Los mismos autores expandieron este

[1]Time Scale Modification

trabajo en [17], agregando una técnica para ocultar las pérdidas y con ello la falta de los paquetes.

Este mismo trabajo considera la detección de picos de retardo. El algoritmo funciona en dos modos: adaptación rápida y normal; cuando el algoritmo entra en modo de adaptación rápida descarta el primer paquete que tenga un gran retardo, el cual no había sido estimado, y utiliza la estadística del retardo que obtuvo con el último pico registrado, para los tiempos de *playout* de los paquetes subsecuentes. El algoritmo regresa a modo normal cuando el retardo vuelve al nivel que tenía antes del pico [17].

Dentro de esta misma técnica, Ranganathan y Kilmartin introducen en [33] un algoritmo que utiliza un estimador del retardo de red que se basa en redes neuronales y sistemas difusos. En este trabajo ellos presentan el diseño llamado Sistema Analizador Difuso para calcular el retardo de red, este diseño analiza la tendencia y usa esto para una adaptación del retardo de *playout* por paquete. Utilizando los algoritmos aplicados principalmente en el campo de Inteligencia Artificial, como Redes Neuronales Multicapas Perceptron, Lógica Difusa, y Tiempo de retardo de Redes Neuronales, los autores caracterizan los valores de retardo como una aplicación de las técnicas de análisis de series de tiempo generales. El análisis de series de tiempo usualmente involucra la estimación de valores futuros de series basadas en el conocimiento de valores previos. Asimismo, los valores futuros de una serie de tiempo pueden ser modelados como una función del valor actual y de los valores pasados de la misma serie, así una aproximación muy cercana de la función puede ser alcanzada usando un conjunto de valores pasados y futuros de las series de tiempo.

El algoritmo de tiempo de *playout* adaptativo por expansión de segmentos se presenta en [23]. Este algoritmo expande el tamaño del segmento usando el algoritmo Overlap-Add Sincronizado, SOLA, en el receptor, la cual es una técnica para la modificación de la escala de tiempo de la señal de audio. El algoritmo usa esta técnica para reducir la cantidad de paquetes perdidos artificialmente. El receptor decide si aplica el algoritmo por expansión de segmentos o no de acuerdo a la condición del tiempo de *playout*. Al recibir un paquete el algoritmo examina si llegó a tiempo o no, si es considerado a tiempo, expande su tamaño usando SOLA, sólo para satisfacer las condiciones del tiempo de *playout*; es decir, no se aplica el algoritmo SOLA a todos los paquetes por lo que la calidad de la voz mejora ya que

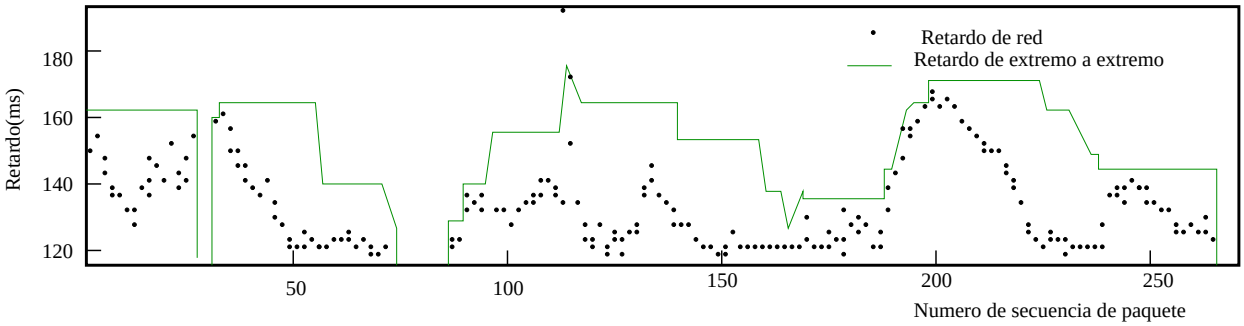

Figura 4.3: Ejemplo Algoritmo para el retardo fijo [17]

se reduce la probabilidad de perder los paquetes. El algoritmo calcula el tiempo de *playout* usando el retardo de extremo a extremo y la variabilidad en el retardo de los paquetes previos para satisfacer el tiempo límite del primer paquete en la frase, y un tiempo de *playout* para cada uno de los paquetes subsecuentes en la frase se calcula como un desplazamiento del tiempo en que el primer paquete es reproducido.

4.3.2. Algoritmos adaptativos por frase

Este tipo de algoritmos, detectan las variaciones en el retardo de extremo a extremo durante la sesión, adaptándose a las condiciones de la red. Estos modifican el tiempo de espera de los paquetes en el buffer al inicio de cada frase, por lo que los cambios en el tiempo de *playout* se ajustan en los periodos de silencio. Estos tipos de algoritmos toman muestras del retardo usando comúnmente estampas de tiempo RTP tanto del emisor como del receptor para estimar un promedio del retardo de extremo a extremo. En la Figura 4.4 se ilustra como este tipo de algoritmos ajustan el tiempo de *playout* únicamente al inicio de cada frase. Los paquetes que llegan después de su tiempo límite se consideran perdidos. Los periodos de silencio delimitan cada frase.

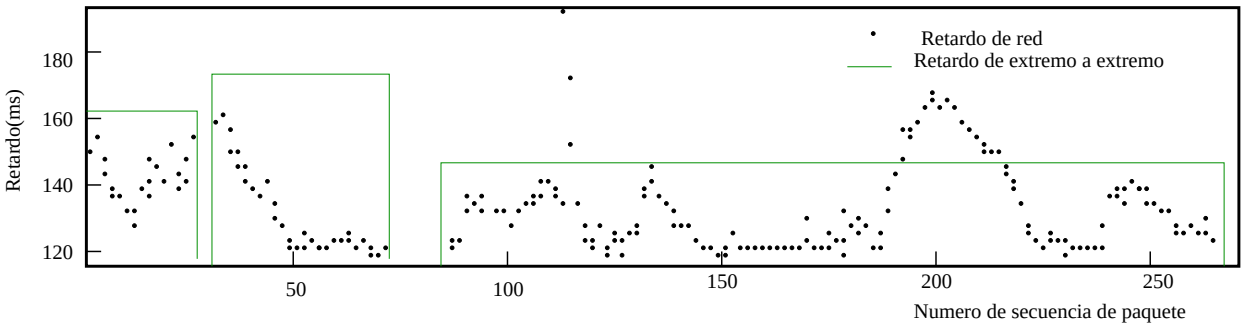

Figura 4.4: Ejemplo Algoritmo de retardo de adaptativo por frase [17]

4.3.2.1. Algoritmos no tolerantes a pérdidas

En las sesiones de audio sobre Internet, las pérdidas artificiales no se desean por lo que deben evitarse como prioridad, incluso a expensas de tener un gran retardo de extremo a extremo. Bajo este compromiso algunos autores han propuesto diferentes algoritmos no tolerantes a pérdidas.

I. *Algoritmos basados en estimaciones auto-regresivas*

Uno de los primeros trabajos publicados, basados con este enfoque, sobre algoritmos de control del retardo de *playout*, es el presentado por Ramjee et al. en [30]. En este trabajo se describen cuatro algoritmos básicos, estos algoritmos trabajan con dos pasos

principales: la estimación de la variabilidad del retardo, $\hat{v}_k^i$, y del retardo de extremo a extremo de cada paquete, $\hat{d}_k^i$.

Con la siguiente ecuación, se estima el retardo de extremo a extremo para el *i-ésimo* paquete:

$$\hat{d}_k^i = \alpha \hat{d}_k^{i-1} + (1-\alpha) n_k^i$$

Y, se calcula la variabilidad estimada con una ecuación similar a la de Van Jacobson en [9]:

$$\hat{v}_k^i = \alpha \hat{v}_k^{i-1} + (1-\alpha)|\hat{d}_k^i - n_k^i| \tag{4.1}$$

donde n_k^i es el retardo real de cada paquete i en cada frase k, α es igual a 0.98802 y es el factor de peso que se usa para el control de la tasa de adaptación del algoritmo, en otras palabras, es un filtro recursivo lineal[30].

Con los valores anteriores se obtiene el tiempo de playout del p_k^i paquete de cada frase:

$$p_k^i = \begin{cases} t_k^i + \hat{d}_k^i + \beta \hat{v}_k^i, & \text{para } i{=}1 \\ p_k^1 + (t_k^i - t_k^1), & \text{para } 1 < i \leq N_k \end{cases} \tag{4.2}$$

donde β es la constante que asegura el compromiso entre las pérdidas y el retardo, y proporciona una holgura en el retardo de playout para la llegada de los paquetes. En [30] $\beta = 4$. El termino $\beta \hat{v}_k^i$, asegura que el tiempo de espera en el buffer sea lo suficientemente grande para que una porción pequeña de los paquetes llegue tarde. Mientras se estima el retardo, $\hat{d}_k^i$, de manera auto-regresiva y actualizada por cada paquete, el tiempo de *playout* es inicializado al inicio de una nueva frase.

El primer algoritmo en [30] hace uso de un filtro recursivo lineal caracterizado por el factor de peso α. El segundo es similar pero con una modificación menor basada en el uso de dos valores para α. Por un lado, un valor para incrementar rápidamente la tendencia del retardo, usando un valor de 0.75, cuando el valor real del retardo es mayor que el estimado. Por otro lado, un valor para decrementar la tendencia en el retardo, de esta manera es posible dar seguimiento a las pequeñas ráfagas de los paquetes que incurren en grandes retardos.

El tercer procedimiento inicializa el valor de retardo, $\hat{d}^i$, como el mínimo valor experimentado en la frase anterior

$$\hat{d}^i = \min_{j \in S_i} \{n_j\}$$

donde S_i is el conjunto de todos los paquetes recibidos en la frase anterior.

En [30], los autores proponen un cuarto algoritmo que incluye la detección de estos picos de retardo. De esta manera el algoritmo trabaja de manera diferente si detecta

un pico de retardo o no. El inicio del pico es detectado al verificar si la diferencia entre el retardo de paquetes consecutivos es mucho más grande que el umbral dado.

Durante el modo pico, la estimación auto-regresiva del retardo es abandonada y la estimación del retardo del playout está basada únicamente en los valores más recientes del retardo.

Cabe destacar que este es el trabajo más referenciado en algoritmos de control de retardo de *playout* y la mayoría de esta clase de algoritmos utiliza estas ecuaciones para calcular el tiempo de *playout* y para estimar la variabilidad.

II. *Algoritmos basados en filtros adaptativos*

Esta clase de algoritmos, determinan el retardo de *playout* prediciendo el retardo y la variabilidad del retardo. Así una predicción precisa del retardo puede rápidamente cambiar y ajustarse de manera efectiva.

En [6], Phillip DeLeon y Cormac J. Sreenan, presentan un algoritmo basado en un filtro adaptativo *Mínimos cuadrados normalizado*[2] (NLMS). Este tipo de filtros han sido utilizados en diversas áreas como la ecualización, la cancelación del eco y la predicción. El filtro NLMS minimiza el error cuadrado medio entre los datos reales y los estimados. Los datos previos pasan a través de un filtro de respuesta de impulso finito, FIR, para calcular el estimado actual. El error cuadrado medio se usa para ajustar el factor de peso del filtro adaptativo.

Este algoritmo se profundizará en el siguiente Capítulo.

4.3.2.2. Algoritmos tolerantes a pérdidas

Los algoritmos de la sección anterior tratan de mantener la tasa de paquetes perdidos muy pequeña, sin embargo, las aplicaciones de VoIP pueden tolerar u ocultar una pequeña cantidad de paquetes perdidos. Los algoritmos que se describen a continuación, aprovechan la tolerancia a las pérdidas que se acaba de mencionar para optimizar el tiempo de espera de los paquetes de audio en el buffer del receptor. Los parámetros de entrada para esta clase de algoritmos permiten entonces optimizar la calidad de servicio en términos de una tasa máxima tolerada de pérdidas en el lado receptor.

La información de los retardos en red de los paquetes es utilizada para construir una función de densidad de probabilidad (pdf). En [20], la idea principal es mantener un seguimiento de la distribución del retardo de los paquetes, y adaptar el tiempo de playout en un tipo histograma basado en la tasa de pérdidas tolerada para la sesión. El algoritmo registra el retardo de cada paquete y actualiza un histograma de estos retardos por cada paquete que llega. El algoritmo calcula el retardo de *playout*, por cada nueva frase, encontrando el retardo actual por cada punto porcentual dado en la función de distribución. La

[2]Normalized Least-Mean-Square

función de distribución usa los retardos de los paquetes por los últimos w paquetes para formar el histograma.

El número de retardos de paquetes, guardados en el histograma determina que tan sensitivo es el algoritmo para adaptarse a los cambios en las condiciones de la red. Si w es muy pequeña, el algoritmo tendrá una visión miope del pasado, y probablemente producirá una pobre estimación del retardo de playout. Por otro lado, si w es muy grande, el algoritmo mantendrá registro de una gran cantidad de historia, y no será capaz de reaccionar a los cambios rápidos en las condiciones de la red.

En caso de no haber picos de retardo, el retardo de *playout* se inicializa en un punto percentil q en la pdf. En caso de detectar un pico de retardo, el retardo de red del primer paquete de una frase se convierte en el retardo de playout para esa frase.

El algoritmo Concord en [41], fija dos umbrales para la máxima tasa de paquetes retrasados y para el retardo máximo aceptado. Una parte crucial de esta estrategia es el uso de un histograma de los retardos de la red observados por los últimos paquetes; a partir del histograma, una versión aproximada y muestreada de la distribución del retardo de los paquetes (PDD) es calculada. Usando la PDD el algoritmo puede inicializar los valores del retardo de los paquetes, así con las dos restricciones en el máximo porcentaje de paquetes perdidos y del máximo retardo son satisfechos. Los autores muestran la importancia de mantener actualizados los PDD aplicando los pesos apropiados a los retardos observados de los paquetes anteriores en base a la antigüedad de la información recolectada.

El trabajo presentado en [7] se enfoca en la estimación de la última parte de la distribución de los retardos de red. De hecho, sólo esta parte es importante cuando se calcula la pérdida de paquetes potencial. Por medio de un pequeño test, los autores determinan que la distribución de Pareto proporciona mejores resultados con respecto a los otras funciones. Este resultado es usado como un punto de inicio para la inicialización del tiempo de playout, el cual se desempeña de la siguiente manera:

- Un valor meta para la tasa total de paquetes perdidos es fijado de antemano.
- La información acerca del retardo de red por cada paquete es guardado y usado para actualizar la estimación de los parámetros de la distribución de Pareto.
- El mínimo valor del retardo de paquete es buscado por lo que la tasa de paquetes perdidos esperada es menos que el umbral fijado.

Obviamente, esto funciona dado que asume que la cantidad de paquetes perdidos dentro de la red no excede el umbral.

El porcentaje promedio de pérdidas puede cambiar de una sesión de audio a otra, incluso si este parámetro se mantiene. En [29], presentan un algoritmo basado en un Predictor de promedio móvil, que ajusta el retardo de *playout* de una frase a otra, dado un objetivo dado de un porcentaje promedio de pérdidas p. El algoritmo asegura que el porcentaje promedio de pérdidas obtenido durante la sesión está muy cerca o es casi igual al deseado.

Al mismo tiempo, en la mayoría de los casos, este algoritmo alcanza este objetivo con un tiempo promedio de playout menor que se necesita obtener.

Otro tipo de algoritmo correspondientes a esta técnica es presentado en [34]. La principal contribución consiste en la evaluación de las interacciones entre la Corrección de Errores Reactiva, FEC, y el retardo de playout. En particular, los autores evalúan la capacidad de reparar del esquema FEC y su influencia tanto en el retardo de playout como en la tasa total de pérdidas. Para este fin, definen el concepto de retardo virtual como la diferencia entre el tiempo en que llega el paquete original o la primera copia que llegue, y el tiempo en que se generó el paquete. Algo muy importante de destacar es que no hace diferencia entre la llegada del paquete original o la recuperación de una de las copias, simplemente escoge el menor, así el retardo no es el real sino el virtual, sólo le interesa que algún paquete con esa información llegue. El uso del retardo virtual mejora el desempeño del buffer del playout, reduciendo el retardo de extremo a extremo promedio para una probabilidad de pérdidas deseada. Para calcular el retardo de playout en [34], utilizan los algoritmos propuestos por Ramjee et. al en [30] y Moon et. al en [20]. Para más referencias sobre algoritmos tolerantes a pérdidas se encuentran en [28] y [18].

4.3.2.3. Algoritmos basados en la calidad de servicio

Otra de las técnicas utilizadas para obtener un balance entre las pérdidas y el retardo es la calidad percibida por el usuario final. Esto requiere un análisis de la importancia de cómo las pérdidas y el retardo afectan la calidad de la conversación, y encontrar un compromiso óptimo de acuerdo a un modelo de opinión como los presentados en la Sección 2.6.

El algoritmo E-MOS en [7] es una de las primeras técnicas para el control de retardo de *playout* que utiliza las medidas subjetivas de calidad de servicio. La idea básica es acotar un modelo matemático de la relación entre MOS y el retardo de playout, de manera que sea posible encontrar el tiempo de espera en el buffer que maximice el valor de MOS. La solución está fundamentada en el supuesto de que la tasa de pérdida de paquetes y el retardo de playout afectan el índice MOS de manera independiente. Dado este supuesto, el MOS es expresado como una combinación de dos funciones polinomiales de retardo y pérdida. Entonces, haciendo uso de la función de densidad acumulativa de Pareto, los autores construyen la relación entre la tasa de pérdidas artificiales y el retardo de playout, Además, los autores del algoritmo E-MOS construyen un modelo de opinión simple y lo usan para controlar el óptimo retardo de playout. El E-MOS utiliza la propiedad aditiva del retardo y factores de equipamiento, pero no puede ser tan preciso como el modelo ITU-T. En particular, los autores de [7] han desarrollado su algoritmo en base a una limitada gama de curvas de calidad MOS, las cuales refieren al codificador G.711. De acuerdo a eso, puede llevar a algunos resultados imprecisos dado que la curva difiere de un codificador a otro, especialmente de un codificador PCM a una tasa baja de bit.

El trabajo en [4], se enfoca hacia el problema de conjuntar el mecanismo FEC con el retardo de playout. Se proponen dos soluciones principales, llamadas *parcial* y *completa*.

De acuerdo con la solución parcial, el retardo de playout es supuestamente conocido por la fuente antes de utilizar la operación de FEC, el cual busca la asignación de los bits disponibles para la fuente y el canal de codificación basado en la perspectiva de calidad del usuario. De acuerdo con la solución completa, los parámetros de FEC y del retardo del playout son conjuntamente optimizados. La calidad es evaluada haciendo uso del E-Model con algunos cambios adicionales. Utilizan diferentes curvas de degradación, dependiendo del grado de interactividad deseada por el usuario final. Además, una expresión analítica del factor de equipamiento es derivada como una función de la tasa de codificación de la fuente y la tasa de paquetes perdidos.

En [42], los autores introducen un nuevo método para predecir la calidad de la voz transmitida por Internet. En este método, un modelo de regresión no lineal es derivado para cada codificador con la ayuda de PESQ y del E-model. Después, proponen el uso del mínimo impedimento como un criterio para la optimización del tiempo de espera en el buffer. Este criterio es comparado con el uso tradicional de MOS. Por último, presentan que las caraterísticas del retardo del tráfico de VoIP, es mejor modelado por una distribución Weibull que por una distribución Exponencial o de Pareto. Basados en todas estas contribuciones, los autores proponen un algoritmo para la optimización perceptual del retardo de playout. Para profundizar en este tema consultar [45] y [43].

4.4. Conclusiones preliminares

En este capítulo se estudiaron los tipos más importantes de algoritmos de control de retardo de playout y su empleo para la reducción de los efectos de la variabilidad en el retardo. Asimismo, se estudió la importancia del compromiso entre las pérdidas y el retardo, y la manera en que este compromiso influye en los algoritmos

Así, se presentó un estado del arte de acuerdo a la clasificación de estos algoritmos, mencionando los trabajos más representativos de cada tipo de algoritmos para el control de retardo de playout. De acuerdo con este estado del arte determinamos que los algoritmos NLMS son un objeto interesante y atractivo de estudio para su uso en una herramienta real de VoIP ya que el costo computacional no es alto como los propuestos en [41, 29].

En el siguiente capítulo describiremos el método de evaluación utilizado para la comparación de los algoritmos, y presentaremos un nuevo algoritmo NLMS y la comparación con respecto a los algoritmos del mismo tipo.

Capítulo 5

Estimación NLMS para control del retardo de *playout*

Hasta ahora, hemos revisado los efectos de los tres fenómenos que impactan directamente la calidad de servicio de las aplicaciones de audio en tiempo real y algunos mecanismos para reducir ese impacto.

La variabilidad en el retardo es uno de esos tres fenómenos, que se debe principalmente a que el tiempo en las filas de espera de los ruteadores es variable. La variabilidad en el retardo modifica los intervalos periódicos con los que se envían los paquetes. Por lo que una aplicación de audio debe reducir o eliminar esta variabilidad en el retardo implementando un buffer en el receptor para almacenar los paquetes que llegan y reproducirlos después de cierto tiempo límite. Si los paquetes llegan después de ese límite se descartan y por lo tanto no se reproducen. Podemos aumentar la probabilidad de que los paquetes lleguen antes de su tiempo límite, aumentando el retardo de playout. Esto reduce el número de paquetes perdidos artificialmente en el receptor; sin embargo, al aumentar el retardo de playout, se puede ver comprometida la interactividad.

En este trabajo comparamos dos algoritmos previamente propuestos en la literatura que utilizan una aproximación auto-regresiva; ambos algoritmos usan un predictor adaptativo de error cuadrado medio normalizado, NLMS. La diferencia es que el segundo es una extensión del primero. Finalmente, proponemos una mejora con la detección de picos de retardo. Este tipo de algoritmo son una opción interesante y atractiva para su uso en una herramienta en tiempo real de VoIP debido a su bajo costo computacional. Debemos aclarar que este trabajo de investigación no es, en ninguna manera, un trabajo que aborde el procesamiento de señales. La estimación NLMS que se presenta no se utiliza como una estimación espectral, sino como un *algoritmo* de estimación de series de tiempo.

5.1. Antecedentes

Los receptores utilizan el buffer de playout para las aplicaciones de audio en tiempo real. Al retardo de los paquetes se le agrega más tiempo para compensar la variabilidad en el retardo. El retardo de *playout* puede ser constante durante toda la sesión de audio, o puede ser ajustado entre frases. En un trabajo reciente [17], se muestra que usando una técnica llamada *packet scaling*, es posible cambiar el retardo de *playout* de paquete en paquete mientras se mantiene la distorsión resultante en niveles tolerables. En este trabajo nos centramos en los algoritmos adaptativos por frase.

La Figura 5.1 muestra diferentes etapas de una sesión de audio. El *i-ésimo* paquete de la frase k se envía en el tiempo t_k^i, el paquete llega al receptor en el tiempo a_k^i, y se mantiene en el buffer del receptor hasta el tiempo p_k^i cuando el paquete es reproducido. Dentro de una frase los paquetes son generados en intervalos iguales en el receptor, estos intervalos tienen un tamaño de Δ segundos.

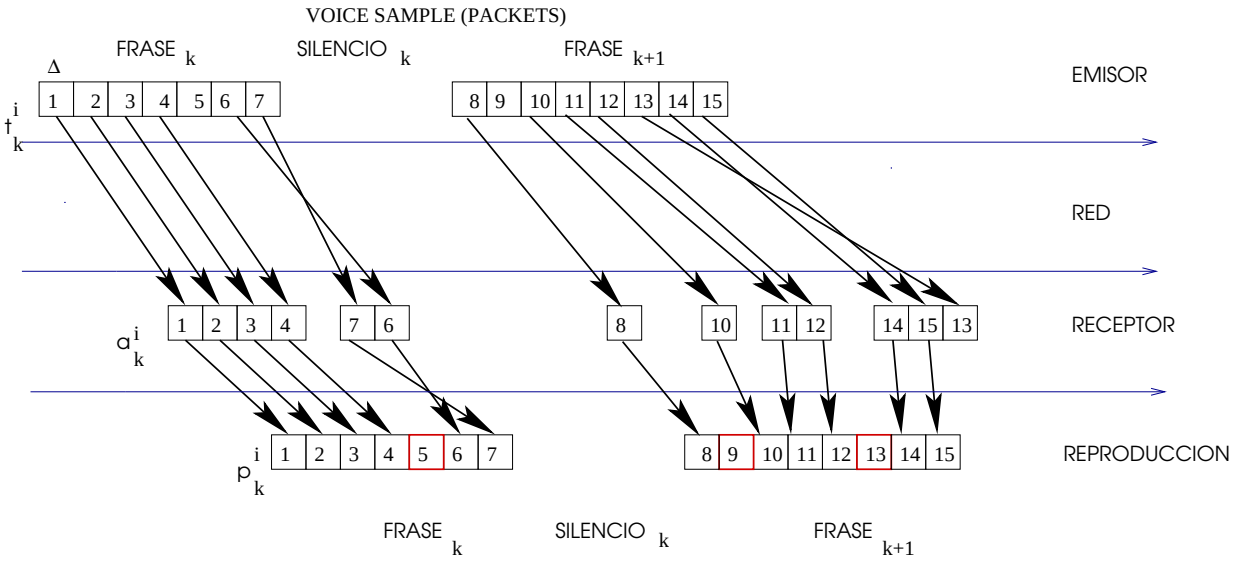

Figura 5.1

Por medio del retardo de *playout* y eliminando los paquetes que llegan después de su tiempo límite, somos capaces de reconstruir la forma periódica de los paquetes de audio. En la Figura 5.1, el paquete 13 fue eliminado debido a su arribo tardío. Un paquete es una pérdida artificial si llega después de su tiempo límite p_k^i. La tasa de pérdidas puede reducirse incrementando el tiempo que los paquetes esperan en el buffer de playout. Un algoritmo de control de playout eficiente debe tener en cuenta este compromiso entre pérdidas y retardo, manteniendo los dos parámetros lo más bajos posible.

Para validar nuestro algoritmo utilizamos trazas de audio generadas durante una sesión de audio real. Los detalles de estas trazas son ampliamente explicados en la Sección 5.2.1.

Una muestra típica de los retardos de extremo a extremo se ilustra en la Figura 5.2.

Cada paquete es representado por un diamante y los límites de cada frase por los rectángulos de guiones. El eje x representa el tiempo del receptor desde que inicia la sesión de audio. Sólo la porción variable del retardo de extremo a extremo es representada en el eje y de la Figura 5.2.

Observamos en la Figura 5.2 la presencia de picos de retardo. Este fenómeno en el retardo de extremo a extremo ha sido reportado previamente en [2, 30] y en este trabajo en la Sección 3.1.2. Los picos de retardo representan un problema serio para las aplicaciones de audio debido a que afectan el desempeño de los algoritmos de control de retardo de playout adaptativo.

Un pico de retardo puede ser contenido dentro de una sola frase o puede expandirse a varias frases. La Figura 5.2a muestra un pico de retardo que se extiende sobre dos frases consecutivas. La Figura 5.2b muestra un pico de retardo que se extiende sobre tres frases consecutivas.

El retardo de playout es generalmente actualizado entre frases, un algoritmo de este tipo se comporta mejor cuando los picos de retardo se expanden sobre más de una frase. Sólo de esta manera los algoritmos de control de retardo de playout pueden reaccionar adecuadamente al pico, al actualizar el retardo de playout de acuerdo al retardo experimentado. Si el pico desaparece antes de que termine la frase, el algoritmo de control no tendrá suficiente tiempo para actualizar el retardo de playout de acuerdo a eso.

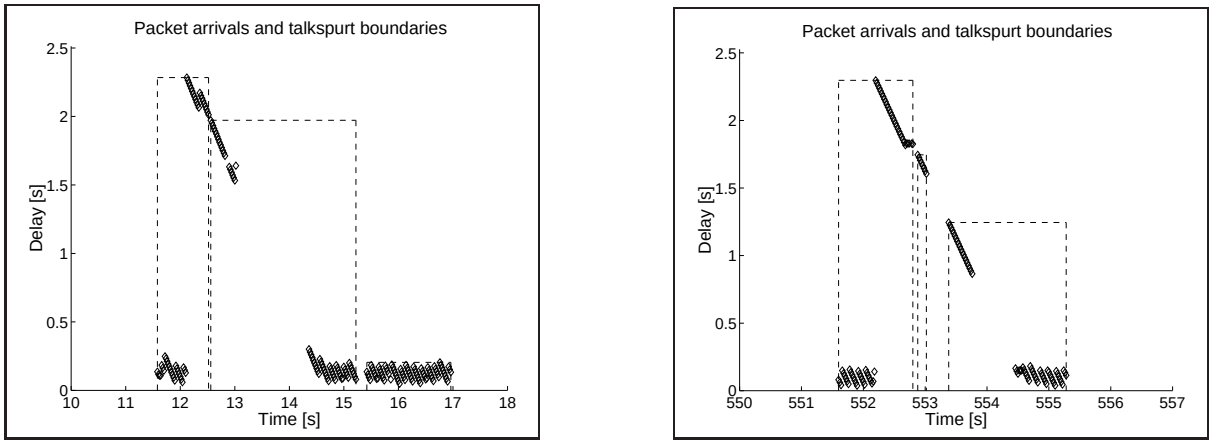

Figura 5.2: Picos de retardo en el retardo de extremo a extremo.

5.2. Método de evaluación

5.2.1. Trazas

Es posible comparar el desempeño de distintos algoritmos a partir de resultados experimentales. Para ello, la simulación basada en trazas es una de las mejores opciones. Una traza es un conjunto de resultados experimentales de los cuales se extraen parámetros de entrada a

dichos algoritmos. La ventaja de este tipo de simulación es que permite comparar diferentes algoritmos bajo condiciones idénticas de red. En este trabajo, elegimos la simulación basada en trazas para comparar los algoritmos que estudiamos.

Varios trabajos sobre algoritmos de control del retardo de playout utilizan las trazas proporcionadas por Sue B. Moon en [20], generadas con NeVoT[1]. NeVoT es una herramienta de audio, que transmite voz empaquetada en Internet. Funciona en ambientes punto a punto, broadcasting, o IP multicast, utilizando los protocolos RTP. NeVoT contiene un mecanismo de rastreo que colecta las estampas de tiempo de emisor y el receptor, así como, números de secuencia RTP. Obtiene las muestras de audio del emisor y las envía como paquetes a través de la red. Asimismo, recibe los paquetes y los reproduce en una aplicación del receptor. Cada sesión de NeVoT puede trabajar con una o más conferencias.

En estas simulaciones se considera que NeVoT transmite 160 bytes de audio en un paquete, los paquetes fueron codificados en modo PCM de 8-KHz, y la unidad de tiempo de empaquetamiento es de 20 milisegundos. Las transmisiones contienen voces tanto femeninas como masculinas [37, 20].

Estas trazas se inyectan a los algoritmos a comparar, para evaluar el desempeño de los algoritmos que estudiamos. Para manipular las trazas, utilizamos el lenguaje de programación `awk`, diseñado para manipular datos en modo texto y realizar operaciones como suma, resta, multiplicación, conteo, etc., entre columnas o dato por dato. Una de sus más grandes ventajas es su eficiencia en el procesamiento de grandes cantidades de datos en texto plano.

Las trazas utilizadas contienen en la primera columna un identificador "D" para los paquetes recibidos en una frase, y "!" en el caso que exista un silencio. De ésta forma podemos contar cuántas frases existen dentro de la traza, y cuántos paquetes en total para cada frase dentro de cada sesión de audio. Operaciones fundamentales para los fines de nuestro trabajo.

Los códigos fuentes empleados para contar el número de frases, y para contar el número de paquetes, se encuentran en el Apéndice.

En la segunda columna de las trazas de audio se encuentran las estampas de tiempo del receptor para cada paquete recibido dentro de la sesión, y en la tercera columna se encuentra la estampa de tiempo de emisor.

Ejemplo de valores contenidos en las trazas

```
D   134634815   135274479
D   134634815   135274639
D   134634815   135274799
D   134634815   135274959
D   134634815   135275119
D   134634815   135275279
!   135277519
D   134637887   135277519
D   134637887   135277679
D   134637887   135277839
D   134637887   135277999
D   134637887   135278159
```

[1]Network Voice Terminal

Como las estampas de tiempo se encuentran en el formato de protocolo vat[2], no están medidas en segundos. Para nuestro trabajo las procesamos para emplear el tiempo en segundos; el código fuente del Apéndice 6 sirve a ese fin. Un pequeño ejemplo del resultado que obtenemos después de procesar las trazas es el siguiente:

Ejemplo de valores en segundos

```
talkspurt = cell(818, 1);
talkspurt1 = [ ...
0 0.216
0.02 0.216
0.04 0.304
0.06 0.284
0.08 0.264
0.1 0.244
0.12 0.352
0.14 0.332
0.16 0.312
0.18 0.292
0.2 0.272
];
talkspurt{1}=talkspurt1;
clear talkspurt1;
talkspurt2 = [ ...
```

Los detalles de cada traza se encuentran en el Tabla 5.1. En la primera columna tenemos el número de traza. En la segunda y tercera columna, están los nombres de los sitios emisores y receptores respectivamente. Las siglas que corresponden a cada uno de ellos son, UMass para la University of Massachusetts Amherst, Estados Unidos, GMD Fokus para Institut für Oggene Kommunikationssysteme, Belín, Alemania, INRIA para Institut National de Recherche en Informatique et en Automatique, Francia, UCI para University of California, Irvine, Estados Unidos, y Osaka Univ. para Osaka University, Japón. En la cuarta columna tenemos la hora del inicio de la transmisión, la fecha y el día de la semana. En la quinta columna está la duración en segundos de cada sesión de audio registrada en cada traza.

Tabla 5.1: Detalles de las trazas [20]

No. traza	Emisor	Receptor	Tiempo de Inicio(Emisor)	Duración[s]
1	UMass	GMD Fokus	08:41 pm 6/27/95(Tu)	1348
2	UMass	GMD Fokus	09:58 am 7/21/95(Fr)	1323
3	UMass	GMD Fokus	11:05 am 7/21/95(Fr)	1040
4	INRIA	UMass	09:20 pm 8/26/93(Th)	580
5	UCI	INRIA	09:00 pm 9/18/93(Sa)	1091
6	UMass	Osaka Univ.	00:35 am 9/24/93(Fr)	649

[2]NeVoT implementa el protocolo vat, visual audio tool, que fue desarrollado por Van Jacobson y Steven McCanne para proveer audio a las teleconferencias sobre Internet.

Las estadísticas de las trazas se muestran en el Tabla 5.2. La primera columna muestra el número de traza. En la segunda y tercera columna, tenemos el número de total frases y el número total de paquetes, respectivamente, obtenidos con un script `awk`. El retardo mínimo de cada traza se presenta en la cuarta columna, este retardo está expresado con las estampas de tiempo proporcionadas por el protocolo vat. La quinta columna muestra la estampa de tiempo base para cada traza, también está expresada en el formato del protocolo vat, y ambas son necesarias para la conversión de las estampas de tiempo a segundo como mencionamos. En la sexta columna tenemos el retardo de extremo a extremo promedio para cada una de las trazas, y el cual está expresado en segundos. Por último tenemos el número de picos de retardo que tiene cada traza, para obtener este número implementamos el Algoritmo 4 que propone Ramjee et al. en [30].

Tabla 5.2: Estadísticas de las trazas

Traza	Frases	Paquetes	Ret. mín[vat]	Est. de tiempo base[vat]	Ret. e-e prom[s]	No.Picos
1	818	56979	-640080	131271999	0.2100	59
2	406	24490	-14643186	119211478	0.5059	59
3	536	37640	-774206	135895789	0.5110	43
4	252	27814	-807010	460510397	0.4890	56
5	540	52836	944917	416127014	0.0307	10
6	299	23293	-1489551	501927541	0.08201	109

Salta a la vista que la mayoría de los retardos mínimos son números negativos, esto se debe a que los relojes de emisor y el receptor no están sincronizados. Este fenómeno lo detallamos en la siguiente Sección.

5.2.2. Sincronización de relojes

Para realizar mediciones del retardo en las aplicaciones, el emisor y el receptor cuentan con relojes respectivamente; sin embargo, dichos relojes no están sincronizados entre sí y, en muchos casos, los relojes no avanzan con la misma cadencia. Estos dos fenómenos provocan que los relojes tengan discrepancias, y pueden ocultar verdaderos cambios experimentados por lo paquetes en Internet, e. g. los cambios que experimentan las filas de espera al crecer o decrecer [21, 49, 7, 25].

Algunos ejemplos de los efectos de la desincronización y cadencia de relojes los observamos cuando los paquetes "viajan en el tiempo", es decir, si el reloj del receptor tiene un adelanto en la hora con respecto al emisor, pareciera que los paquetes llegan al receptor antes de que los envíe el emisor, así, los paquetes tienen retardos negativos. Otro ejemplo es el caso, donde, el reloj del emisor lleva una cadencia más lenta que el reloj del receptor, y da la impresión que el retardo de extremo a extremo va en aumento [21, 25].

Las mediciones del retardo de extremo a extremo son importantes para las aplicaciones como la VoIP, ya que con estas mediciones se calcula el tiempo de playout, además de ser un parámetro para la evaluación de los algoritmos de control del playout. Por lo anterior la precisión de las mediciones y el análisis de desempeño del retardo son fundamentales [21, 49, 7], así como también, para el diseño de algoritmos de control de tráfico[49].

Dos de las formas de medir el retardo son:

- **Retardo de extremo a extremo (OWT)**[3]: Es la diferencia entre el tiempo en el que la fuente inyecta el primer bit de un paquete en la red y el tiempo en el que el destino recibe el último bit de ese paquete.

- **Retardo de ida y vuelta (RTT)**[4]: Es la suma de los tiempos que necesita un paquete para viajar desde un nodo A hasta su destino B y del nodo B de regreso al A.

El *one way time* se obtiene colocando una estampa de tiempo a cada paquete en el momento de ser enviado y otra estampa de tiempo en el momento de ser recibido. Como es nuestro caso para VoIP, al obtener la diferencia entre la estampa de tiempo del receptor y la de emisor el resultado es la porción variable del retardo de extremo a extremo, que le corresponde al *i-ésimo* paquete d^i [7]. En la Figura 5.3 trazamos en el eje de las x el tiempo transcurrido en segundos y el eje de las y el retardo correspondiente a cada paquete, podemos observar que la gráfica presenta una pendiente o *skew*, que es la diferencia en las frecuencias de un reloj y el "verdadero" reloj, ocasionada por la diferencia de cadencias entre los relojes del emisor y receptor[21].

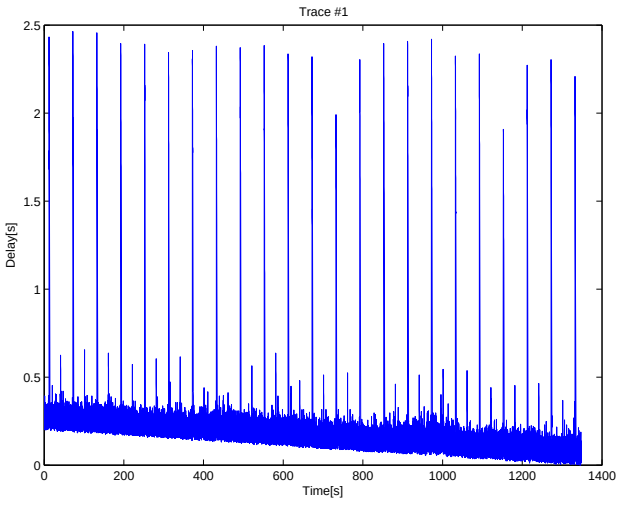

Figura 5.3: Pendiente en la traza 1

[3]One-Way Time
[4]Round Trip Time

Para remover la pendiente es necesario un tratamiento de las trazas, y para ello existen diversos algoritmos como el propuesto por Moon, Skelly y Towsley en [21], donde utiliza la programación lineal para dibujar una línea que determina la inclinación de la pendiente. En [25], Paxson utiliza la regresión lineal; la idea principal de este algoritmo es la técnica de buscar una línea adecuada que trace la pendiente basada en estadísticas robustas. Este algoritmo, usa la media como un estimado para la pendiente, sin embargo, comienza a tener fallas en la medida que el tiempo de extremo a extremo tiene variaciones. Por último, el algoritmo "Convex Hull", una propuesta de Li Zhang en [49], este algoritmo examina los puntos dentro de un conjunto dado de izquierda a derecha. Por cada punto, determina si debe ser empujado hacia el montón o si debe sacar algunos puntos del montón. Al terminar el algoritmo, todos los puntos en el montón son vértices de un límite más bajo.

El Sistema de posicionamiento Global, GPS[5], ofrece una sincronización precisa entre nodos de la red; desafortunadamente por su costo, este sistema es escaso en las redes computacionales. Además, GPS requiere una continua recepción de múltiples satélites lo cual es difícil de acoplar, en centros de seguridad de datos.

Para lidiar con la sincronización de relojes, también es posible utilizar el Protocolo de Tiempo en Red, NTP [6]. NTP mide el RTT y usa un procedimiento de reducción para estimar las diferencias entre el tiempo reportado por un reloj y el tiempo "verdadero". Un método de sincronización reciente basado en sistemas Pentium, es una alternativa a la sincronización GPS. Sin embargo, este método se llama a través de un nivel GPS sincronizado a un servidor NTP en la proximidad de la medición de los extremos, una requisición que no es práctica para los nodos remotos. Todos estos tipos de sincronizaciones para los relojes trabajan correctamente sólo cuando el retardo es simétrico [8].

En la Figura 5.4, presentamos dos ejemplos con las trazas 1 y 2 usando el algoritmo propuesto por Moon para remover la pendiente.

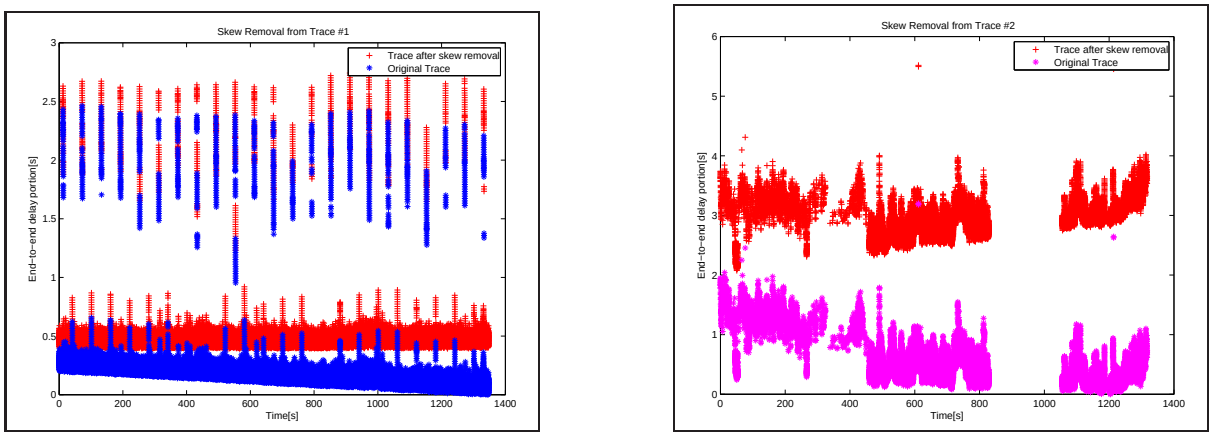

Figura 5.4: Comparación de las trazas 1 y 2 antes y después de remover la pendiente

[5]Global Positioning System
[6]Network Time Protocol

Para nuestro fin, suponemos que los relojes están sincronizados considerando la porción variable del retardo de extremo a extremo, pero es necesario remover la componente constante del retardo de extremo a extremo. Por esto, el retardo debe normalizarse quitando el mínimo valor de retardo de extremo a extremo, d_k^i, de toda traza a los retardos de cada uno de los paquetes, tal como indican los autores en [30] y en [20]:

$$d_k^i = a_k^i - t_k^i - min(a_k^i - t_k^i)$$

La mayoría de los trabajos relacionados con los algoritmos de control de retardo de playout, asumen esta misma suposición debido principalmente a que no es el objeto de estudio de este tipo de trabajos y a que esta suposición no afecta las estimaciones.

5.2.3. Medidas de desempeño

Antes de poder probar los algoritmos propuestos, definimos el conjunto de medidas de desempeño que usamos en nuestro trabajo. Para medir el desempeño de un algoritmo adaptativo de playout, nos enfocamos en el número total de paquetes que son reproducidos durante la sesión de audio, así como también, en el retardo de extremo a extremo promedio que experimentan los paquetes [29]. Las variables p_k^i, N, L, N_k, t_k^i, y a_k^i se definen como en la Tabla 5.3.

Tabla 5.3: Definición de variables

Variables	Definición
L	El número de paquetes que llegan al receptor durante la sesión
N	El número de frases durante la sesión
N_{k}	El número de paquetes durante una frase
$t_{\mathrm{k}}^{\mathrm{i}}$	Estampa de tiempo del emisor de i-ésimo paquete de la k-ésima frase
$a_{\mathrm{k}}^{\mathrm{i}}$	Estampa de tiempo del receptor de i-ésimo paquete de la k-ésima frase
$d_{\mathrm{k}}^{\mathrm{i}}$	La porción variable del retardo de extremo a extremo que corresponde al i-ésimo paquete en la k-ésima frase
$p_{\mathrm{k}}^{\mathrm{i}}$	El tiempo en que se reproduce el i-ésimo paquete de la k-ésima frase

Como en [20], definimos r_k^i como una variable indicadora si un paquete i de una frase k es reproducido o no. Así, r_k^i se define como:

$$r_k^i = \begin{cases} 0, & \text{si } p_k^i < a_k^i \\ 1, & \text{en otro caso} \end{cases}$$

donde el valor 0 corresponde al caso de que el paquete no llegue a tiempo para su reproducción en el receptor y el valor 1 al caso de que el paquete llegue a tiempo para su reproducción. Esto se hace para cada frase dentro de la sesión, por lo que el número total de paquetes reproducidos, representado por T, en una sesión de audio es dado por:

$$T = \sum_{k=1}^{N} \sum_{i=1}^{N_k} r_k^i$$

donde se suma el número de paquetes reproducidos de cada frase, de todas las frases para cada sesión, así con esta información podemos obtener el porcentaje de pérdidas total en una sesión de la siguiente manera:

$$l = \frac{L - T}{L} * 100 \tag{5.1}$$

Además observamos el tiempo que le toma a un paquete llegar de un extremo al otro extremo en la red, con esto calculamos el retardo de extremo a extremo, D_{prom}, con la siguiente ecuación:

$$D_{prom} = \frac{1}{T} \sum_{k=1}^{N} \sum_{i=1}^{N_k} r_k^i [p_k^i - t_k^i] \tag{5.2}$$

donde se calcula sumando el retardo experimentado por cada paquete que llegó a tiempo para su reproducción de todas las frases durante la sesión entre el número total de paquetes reproducidos.

Como mencionamos en el Capítulo 4, el compromiso entre las pérdidas y el retardo es fundamental para determinar entre los algoritmos de control de retardo de playout cual funciona mejor que los otros. De esta forma, con las ecuaciones anteriores podemos graficar una curva que represente este compromiso para un algoritmo dado, con el retardo promedio y la tasa de pérdidas [24].

5.3. Comparación del desempeño de algoritmos NLMS

5.3.1. Trabajo relacionado

Desde aquí, nos enfocamos sólo en los algoritmos propuestos por DeLeon (NLMS) en [6] y después modificado por Shallwani (E-NLMS) en [40]. El algoritmo de Shallwani trata de mejorar el algoritmo NLMS propuesto por DeLeon introduciendo la detección de picos de retardo.

El algoritmo propuesto por DeLeon utiliza la ecuación NLMS para estimar el retardo de extremo a extremo:

$$\mathbf{h}_{i+1} = \mathbf{h}_i + \frac{\tilde{\mu}}{\mathbf{n}_i^T \mathbf{n}_i + a} \mathbf{n}_i e_i \tag{5.3}$$

donde $\mathbf{n}_{i+1}$ es el vector **N**x1 de coeficientes del filtro adaptativo, $\tilde{\mu}$ es el tamaño de paso, $\mathbf{n}_i$ es el vector de **N**x1 que contiene los más recientes N retardos, T es el vector transpuesto y e_i es el error estimado dado por:

$$e_i = \mathbf{h}_i^T \mathbf{n}_i - \hat{n}_i \tag{5.4}$$

donde $\hat{n}_i$ es el retardo estimado para el paquete i.

Este algoritmo NLMS minimiza el error cuadrado medio (e) entre la muestra del retardo actual y su estimado. Las muestras previas pasan a través de un filtro FIR para calcular el estimado actual. El error cuadrado medio se usa para actualizar los pesos del filtro adaptativo.

Para estimar la variabilidad del retardo, se utiliza la ecuación 4.1 que propone Ramjee en [30] y para escoger el tiempo de playout que le corresponde a cada paquete utilizan la ecuación 4.2 para calcular el p_k^i correspondiente.

Estas estimaciones se hacen por cada paquete que llega. Si bien se puede detectar si hay un pico y adaptar el tamaño del buffer para esperar a los paquetes subsecuentes, esto sólo lo pueden hacer al inicio de cada frase y, en caso de haber picos dentro de una frase, los algoritmos no hacen cambios sino hasta el siguiente inicio de frase, lo cual provoca que los paquetes de ese pico se pierdan.

El algoritmo E-NLMS agrega un modo de detección de picos de retardo al original NMLS en [40]. En modo normal el algoritmo opera con las funciones de NLMS en [6]. Dentro del modo pico, el retardo de playout es todavía basado en el predictor NLMS, pero no deja que el retardo estimado sea menor a un estimado de la variabilidad utilizando la ecuación propuesta en [30]. En el Algoritmo 1 mostramos el pseudocódigo del E-NLMS. El E-NLMS utiliza el retardo estimado con la ecuación NLMS durante el modo "NORMAL" y durante el modo "SPIKE" abandona esta estimación y utiliza la ecuación propueta por Ramjee para estimar el retardo.

En ambos trabajos los autores utilizan trazas generadas con un programa que usa el protocolo *ping* para medir el tiempo que tarda en llegar un paquete de emisor al receptor, estas trazas son generadas entre tres nodos en EU y uno en Inglaterra. Esas trazas no están disponibles para todos y por lo tanto no podemos verificar el comportamiento del retardo en ellas.

5.3.2. Algoritmo NLMS modificado

Para nuestros fines utilizamos los valores de desempeño de los algoritmos propuestos por Ramjee en [30], específicamente el denominado *Algoritmo 4*, que contempla la detección de picos de retardo.

Dado el estudio de los algoritmos NLMS, decidimos reemplazar la detección de picos de retardo del algoritmo de Shallwani modificando la detección de picos de retardo de Ramjee por ser una opción que no se había explorado. Escogimos ajustar el parámetro VAR en la línea 8 del Algoritmo 4 a un valor adaptativo como función de los picos de retardo previos

1: $ARdelay_i = ARdelay_{i-1} + (1-\alpha)n_{i-1}$;
2: $\hat{v}_i = \hat{v}_{i-1} + (1-\alpha)|\hat{d}_{i-1} - n_{i-1}|$;
3: **if** (mode == SPIKE) **then**
4: $varfactor_i = \beta/4\hat{v}_i$;
5: $D_i = \hat{d}_i + varfactor_i$;
6: $D_i = \hat{d}_i + varfactor_i$;
7: **if** $(D_i < ARdelay_i + \hat{v}_i)$ **then**
8: $D_i = ARdelay_i + \hat{v}_i$;
9: **end if**
10: **else**
11: $varfactor_i = \hat{v}_i$;
12: $D_i = \hat{d}_i + varfactor_i$;
13: **end if**
14: **if** $(D_i < n_i)$ **then**
15: $packet_i = LOST$;
16: **else**
17: $packet_i = INTIME$;
18: **end if**
19: **if** $(n_i > \hat{d}_i)$ **then**
20: mode = NORMAL ;
21: **end if**
22: **if** $((n_i > \hat{d}_i + 5\hat{v}_i) OR (packet_i == LOST))$ **then**
23: mode = SPIKE;
24: **end if**

ALG 1: Algoritmo E-NLMS

y recientes. El algoritmo NLMS original con las ecuaciones 5.3 y 5.4 se mantiene durante el modo "NORMAL", en Algoritmo 2 mostramos el pseudocódigo de nuestro algoritmo.

Llamamos a nuestro nuevo algoritmo NLMS, **NLMS-mod**. Así, probamos nuestro algoritmo con las trazas de audio provistas por Sue B. Moon y lo comparamos con los algoritmos de DeLeon y Shallwani.

1: $n_i = Receiver\_timestamp - Sender\_timestamp$;
2: **if** $(mode == NORMAL)$ **then**
3: **if**$(abs(n_i - n_{i-1}) > abs(\hat{v}) * 2 + 800)$;
4: $var = 0$;
5: $mode = IMPULSE$;
6: **else**
7: $var = var/2 + abs((2n_i - n_{i-1} - n_{i-2})/8)$;
8: **if** $(var <= 63)$; **then**
9: $mode = NORMAL$;
10: $n_{i-2} = n_{i-1}$;
11: $n_{i-1} = n_i$;
12: **return**;
13: **end if**
14: **end if**
15: **if** $mode == NORMAL$ **then**
16: $\hat{d}_i = \mathbf{h}_i^t = n_i$;
17: **else**
18: $\hat{d}_i = \hat{d}_{i-1} = n_i = n_{i-1}$;
19: **end if**
20: $\hat{v}_i = 0{,}125 * abs(n_i - \hat{d}_i) + 0{,}875 * \hat{d}_{i-1}$;
21: $n_{i-2} = n_{i-1}$;
22: $n_{i-1} = n_i$;
23: **return**;

Alg 2: Algoritmo NLMS-mod

Además, los experimentos realizados por Ramjee y posteriormente por Moon utilizaron trazas de audio reales, las cuales son totalmente representativas de conversaciones VoIP comparadas con trazas artificiales generadas por DeLeon y Shallwani.

5.3.3. Resultados

La comparación la hacemos entre los algoritmos NLMS propuesto por DeLeon[6], E-NLMS por Shallwani[40], y el algoritmo NLMS-mod que presentamos en [19]. Estos algoritmos son simulados *offline* con la herramienta de cálculo numérico MATLAB para cada una de las trazas descritas anteriormente, cada traza representa una sesión de audio. Para los algoritmos NLMS y NLMS-mod utilizamos los valores de los parámetros como aparecen en la Tabla 5.4.

Tabla 5.4: Parámetros NLMS

Parámetro	Valor
$\mathbf{h}_0$	$[10 \ldots 0]^T$
N_{vector}	18
$\tilde{\mu}$	0.01

Para el algoritmo E-NLMS utilizamos los valores descritos en la Tabla 5.5. Cada algoritmo se ejecuta, y obtenemos el retardo de playout promedio y la tasa de pérdidas con las ecuaciones (5.1) y (5.2). Una vez que obtenemos estos valores graficamos los resultados de cada traza variando los valores de β desde 4 hasta 0, de esta manera variamos el compromiso entre la tasa de pérdidas y el retardo, de tal forma que podemos observar como los algoritmos reaccionan ante tasas más bajas de pérdidas.

Tabla 5.5: Parámetros E-NLMS

Parámetro	Valor
$\mathbf{h}_0$	$[10 \ldots 0]^T$
N_{vector}	20
$\tilde{\mu}$	0.001

Una vez que tenemos todos los valores y parámetros, comparamos nuestro algoritmo NLMS, NLMS-mod con cada una de las seis trazas de audio.

En la etapa de pruebas, utilizando las seis trazas proporcionadas por Sue B. Moon obtuvimos los resultados mostrados en la Figura 5.5, donde el desempeño de nuestro algoritmo, en la mayoría de los casos, es mejor que el NLMS y el E-NLMS en las tasas de pérdida que nos interesa, entre 5 % y 10 % dependiendo del codificador utilizado, especialmente con las trazas 2, 3 y 6. Si bien no mejora en todos los casos al algoritmo NLMS original, sí mejora en todos los casos al algoritmo E-NLMS.

Un resultado muy importante es que el algoritmo E-NLMS no siempre se desempeña mejor que el algoritmo NLMS de De Leon. Este resultado es importante porque demuestra que el proceso de retardo de paquetes enviados con el protocolo ping no modela bien un proceso de retardo obtenido con una conversación de voz real.

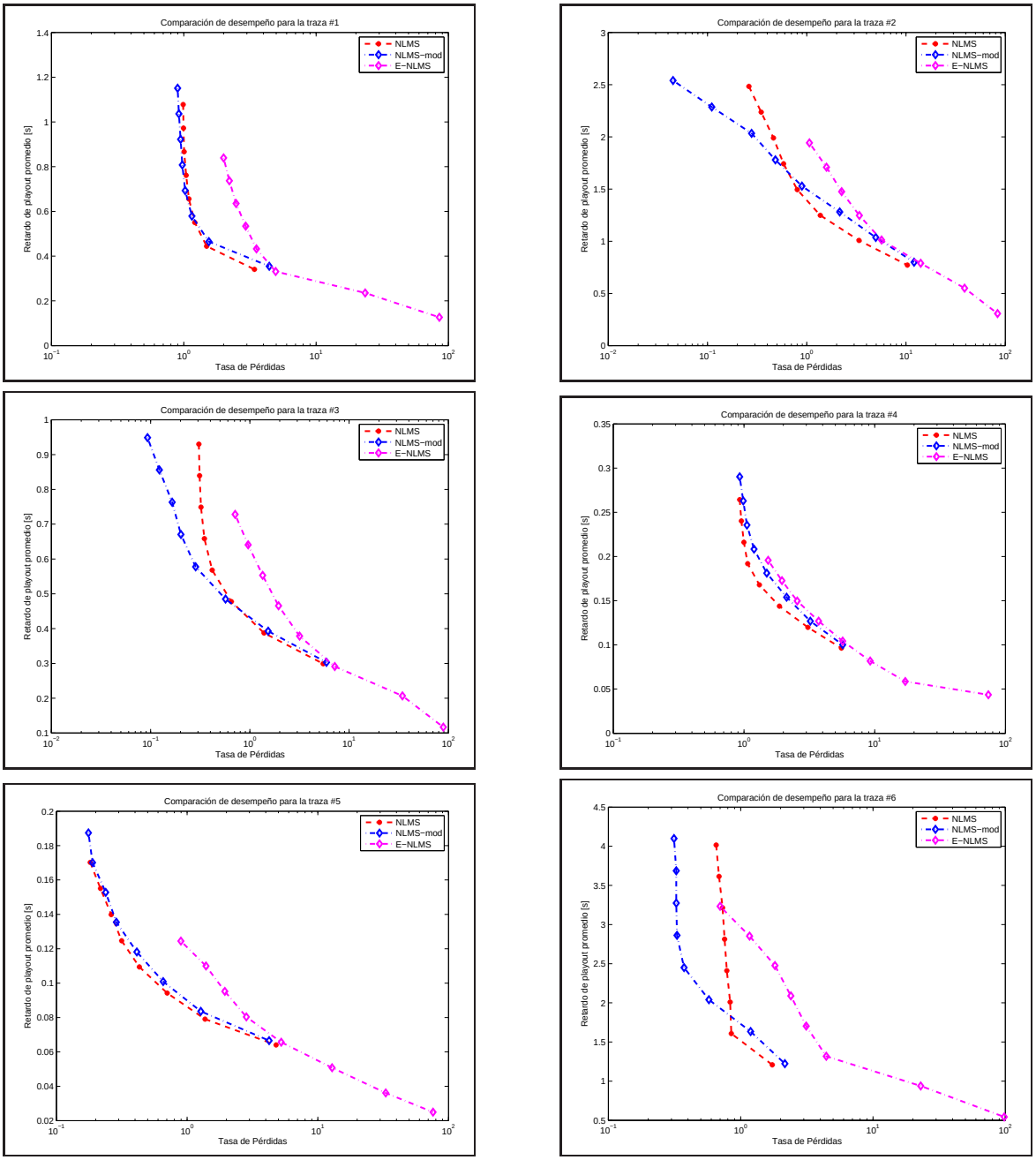

Figura 5.5: Comparación de desempeño

Capítulo 6

Conclusiones y Perspectivas

En este trabajo hemos descrito un algoritmo NLMS que ajusta el retardo de playout al inicio de cada frase. Para evaluar el desempeño de nuestro algoritmo, lo comparamos con esquemas similares ya existentes.

Para lo anterior, se presentó primero un recorrido extenso en el estado del arte de la investigación de VoIP. Las pérdidas, el retardo y la variabilidad en el retardo son los fenómenos en Internet que más impactan la QoS en las aplicaciones de VoIP.

También, se presentó una mejora al algoritmo NLMS original para el control de retardo propuesto por DeLeon[6]. Con simulaciones basadas en trazas, comparamos nuestro algoritmo con los algoritmos de DeLeon y de Shallwani, y probamos que nuestro algoritmo NLMS modificado es un método eficiente para el control del retardo del playout. Cuando la variabilidad en el retardo y las pérdidas son muy altos, como en las trazas 2 y 6, el algoritmo NLMS-mod se desempeña eficientemente y demuestra una mejora en comparación con los algoritmos NLMS y E-NLMS respecto a la tasa de pérdidas de interés en aplicaciones interactivas como la VoIP. Esto muestra que usar estimación NLMS podría ser mejor que otros algoritmos propuestos previamente que usan promedios móviles para su estimación. En [29], el algoritmo propuesto por los autores no implementa detección de picos de retardo y sufre de bajo desempeño con las trazas 2 y 6, que son las trazas con valores de retardo y la variabilidad en el retardo más extremos del conjunto de trazas evaluadas.

Un resultado adicional, es que probamos que contrariamente a lo dicho en el artículo de Shallwani, el algoritmo E-NLMS en la mayoría de los casos no mejora el desempeño del propuesto por DeLeon.

Nuestro algoritmo estima el retardo de playout para los paquetes por cada frase, usando un vector que guarda los valores de retardo de los paquetes. Para reconstruir los intervalos iguales, el retardo del playout de los paquetes en una frase está basado en el

tiempo de playout del primer paquete de la frase.

Nuestro trabajo futuro se enfocará en adaptar el tamaño de la memoria del algoritmo NLMS en una propuesta dinámica, de tal forma que el tamaño de la memoria del predictor se adapte al estado de la red. Otra dirección de investigación en la que estamos interesados es explorar el comportamiento de diferentes estimadores auto-regresivos para el control de retardo en voz sobre Internet.

Apéndice

Código fuente para contar el número de frases en una traza

```
awk 'BEGIN { count = 0} /!/ { count++; }
END { print (count+1);}' traza
```

Código para contar el número de paquetes recibidos en una traza

```
awk 'BEGIN { count = 0} /D/ { count++; }
END { print (count);}' traza
```

Código fuente para obtener el retardo de extremo a extremo

```
 awk '/D/ {dif = $2 - $3; print $1, dif }' traza | sort -n > tr
```

Código fuente para conversión de las estampas de tiempo a segundos

```
    awk 'BEGIN {
   count = 1;              # Number of talkspurts counter
   tksps = "talkspurt";    # Vector name for matlab

   basetimestamp;          # Sender timestamp of the first packet in a trace
   minimum;                # The minimum delay found by findMinimum.awk
   numTksps;               # Total number of talkspurts (known in advance)

   # Put the string "talkspurt = cells(numTksps, 1)"
   printf("%s = cells(%s, 1);\n", tksps, numTksps);

   # Put the string "talkspurt1 = [ ..."
   printf("%s%s = [ ...\n", tksps, count)
}

$1 == "D" { print ($3-basetimestamp)/8000, ($2-$3-minimum)*0.000125 }
$1 == "!" {printf("];\n");
           printf("talkspurt{%s}=talkspurt%s;\n", count, count);
           printf("clear talkspurt%s;\n", count);
           printf("%s%s = [ ...\n", tksps, ++count);
}

END {printf("];\ntalkspurt{%s}=talkspurt%s;", count, count);
     printf("\nclear talskpurt%s", count)}' traza > tr
```

Código fuente para remover la pendiente

```
clear all

load traza1.dat;
d = traza1(1,:);

t_s = traza1(:,2);

N         = length(d);
n         = [1, 2, 0];
k         = 2;

for i=3:N
    for j=k:-1:2
        sol1  = [t_s(i), -1; t_s(n(j)), -1] \ [d(i); d(n(j))];
        sol2  = [t_s(n(j)), -1; t_s(n(j-1)), -1] \ [d(n(j)); d(n(j-1))];

        if sol1(1) > sol2(1)
                                            break
        end
    end

    k = j+1; n(k) = i;
end

opts = mean(t_s);

for i=1:k-1
   if (t_s(i) < opts) & (opts < t_s(n(i+1)))
      mysol = [t_s(n(i)), -1; t_s(n(i+1)), -1] \ [d(n(i)); d(n(i+1))];
      fprintf('%f and %f\n', mysol(1), mysol(2))
   end
end
```

Acrónimos

PSTN	Public Switched Telephone Network
IP	Internet Protocol
VoIP	Voice over Internet Protocol
QoS	Quality of Service
TCP	Connection Oriented Transport Protocol
UDP	User Datagram Protocol
ACK	Acknowledgment
DNS	Domain Name System
RIP	Routing Protocol
SNMP	Simple Network Protocol
ITU-T	International Telecommunication Union-Telecommunication Standardization Sector
PCM	Pulse Code Modulation
ADPCM	Adaptive Differential Pulse Code Modulated
LD-CELP	Low Delay Code Excited Linear Prediction
CS-ACELP	Conjugate Structure Algebraic Code Excited Linear Predictive
MP-MLQ	Multi-Pulse - Maximum Likelihood Quantizer
ACELP	Algebraic code excited linear prediction
RTP	Real Time Protocol
RFC	Request For Comments
WAV	WAVEform audio format
GSM	Groupe Spécial Mobile
MPEG	Moving Pictures Experts Group
RTCP	Real Time Control Protocol
PESQ	Perceptual Evaluation Speech Quality Measure
MOS	Mean Opinion Score
FEC	Forward Error Correction
RTT	Round Trip Time
NLMS	Normalized Least-Mean-Square
E-NLMS	Enhanced Normalized Least-Mean-Square
NLMS-mod	Normalized Least-Mean-Square modificado

Referencias

[1] Luigi Atzori and Mirko L. Lobina. Playout buffering in IP Telephony: A Survey Discussing Problems and Approaches. *IEEE Communications Surveys & Tutorials*, 8:36–46, 2006.

[2] Jean-Chrysostome Bolot. Characterizing end-to-end packet delay and loss in the Internet. *Journal of High Speed Networks*, 2:305–323, 1993.

[3] Jean-Crysostome Bolot and Andrés Vega-García. The case for FEC-based error control for packet audio in the Internet. *In ACM Multimedia Systems*, 1997.

[4] Catherine Boutremans and Jean-Yves Le Boudec. Adaptive joint playout buffer and FEC adjustment for Internet telephony. In *Proceedings IEEE INFOCOM*, pages 652–662, 2003.

[5] Jonathan Davidson and James Peters. *Voice Over IP Fundamentals*. Cisco Press, Indianapolis, 2000.

[6] Phillip DeLeon and Cormac J. Sreenan. An adaptive predictor for media playout buffering. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 6, pages 3097–3100, Phoenix, AZ, March 1999.

[7] Kouhei Fujimoto. *Adaptive Playout Buffer Algorithm for Enhancing Perceived Quality of Streaming Applications*. PhD thesis, Osaka University, Departament of Informatics and Mathematical Science, 2002.

[8] Omer Gurewitz, Israel Cidon, and Moshe Sidi. One-way delay estimation using network-wide measurements. *IEEE/ACM Transactions on Networking (TON) Special issue on Networking and Information Theory*, 52:2710–2724, June 2006.

[9] Van Jacobson. Congestion avoidance and control. In *ACM SIGCOMM '88*, pages 314–329, Stanford, CA, August 1988.

[10] Maarten Buchli Jan Janssen, Danny De Vleeschauwer and Guido H.Petit. Assessing voice quality in packet-based telephony. *IEEE Internet Computing*, 6:48–56, May/June 2002.

[11] Sixto Ortiz Jr. Internet Telephony Jumps off the Wires. *IEEE Computer*, 37(12):16–19, 2004.

[12] Younchan Jung and J. William Atwood. Switching between fixed and call-adaptive playout: A per-call playout algorithm. *IEEE Internet Computing*, 9(4):22–27, 2005.

[13] Aman Kansal and Abhay Karandikar. Jitter-free audio playout over best effort packet networks. In ATM Forum International Symposium, August 2001.

[14] James F. Kurose and Keith W. Ross. *Computer Networking A Top-Down Approach Featuring the Internet*. Addison Wesley Professional, Boston, MA, USA, November 2000.

[15] Bo Li, Mounir Hamdi, Dongyi Jiang, Xi-Ren Cao, and Y. Thomas Hou. QoS enabled Voice Support in the next generation Internet: issues, existing approaches and challenges. *IEEE Communications Magazine*, 38:54–61, April 2000.

[16] Yi J. Liang, Nikolaus Färber, and Bernd Girod. Adaptive playout scheduling using time-scale modification in packet voice communications. In *Proceedings of ICASSP*, pages 1445–1448, 2001.

[17] Yi J. Liang, Nikolaus Färber, and Bernd Girod. Adaptive playout scheduling and loss concealment for voice communications over IP networks. *IEEE Transactions on Multimedia*, 5:532–543, December 2003.

[18] Hugh Melvin and Liam Murphy. An Evaluation of the Potential of Synchronized Time to Improve Voice over IP Quality. *IEEE International Conference on Communications*, 3:1922–1926, May 2003.

[19] Karen S. Miranda-Campos and Víctor M. Ramos R. On NLMS estimation for VoIP playout delay algorithms. *In Proceedings of the International Conference on Signal Processing and Multimedia Applications (SIGMAP)*, pages 342–347, July 2008.

[20] Sue B. Moon, Jim Kurose, and Don Towsley. Packet audio playout delay adjustment: Performance bounds and algorithms. *ACM/Springer Multimedia Systems*, 6:17–28, January 1998.

[21] Sue B. Moon, Paul Skelly, and Don Towsley. Estimation and removal of clock skew from network delay measurements. In *Proceedings of the IEEE Infocom*, pages 227–234, New York, NY, USA, March 1999.

[22] Songun Na and Seungwha Yoo. Allowable Propagation Delay for VoIP calls of Acceptable Quality. In *Proceedings of the First International Workshop on Advanced Internet Services and Applications*, pages 47–56, London, UK, 2002. Springer-Verlag.

[23] Jae-Hyun Nam, Won-Joo Hwang, Jong-Gyu Kim, Soong-Hee Lee, Jong-Wook Jang, Kyo-Hong Jin, and Jung-Tae Lee. Adaptive playout algorithm using packet expansion for the VoIP. *Lecture notes in Computer Science International Conference on Information Networking, Korea*, 2662:563–572, February 2003.

[24] Miroslaw Narbutt, Andrew Kelly, Philip Perry, and Liam Murphy. Adaptive VoIP playout scheduling: assessing user satisfaction. *IEEE Internet Computing*, 9:28–34, July-August 2005.

[25] Vern Paxson. On calibrating measurements of packet transit times. In *Proceedings of the ACM SIGMETRICS*, pages 11–21, Madison, WI, June 1998.

[26] Colin Perkins and Orion Hodson. RFC 2354: Options for repair of streaming media, June 1998.

[27] Colin Perkins, Orion Hodson, and Vicky Hardman. A survey of packet loss recovery techniques for streaming audio. *IEEE Network*, 12:40–48, September/October 1998.

[28] Jesús Pinto and Kenneth J. Christensen. An algorithm for playout of packet voice based on adaptive adjustment of talkspurt silence periods. In *Proceedings of the 24th IEEE Conference on Local Computer Networks*, pages 224–231, Lowell, Massachusetts, October 1999.

[29] Víctor M. Ramos R., Chadi Barakat, and Eitan Altman. A moving average predictor for playout delay control in VoIP. In *Proceedings of the 11th International Workshop on Quality of Service*, pages 155–173, Berkeley, CA, USA, June 2003.

[30] Ramachandran Ramjee, Jim Kurose, Don Towsley, and Henning Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *13th Proceedings of the Conference on Computer Communications (IEEE Infocom)*, volume 2, pages 680–688, Toronto, Canada, June 1994.

[31] Víctor Ramos and Chadi Barakat. Audio over the Internet, 2004. Course taught at Institut of Advanced Internet Applications at Marseille, France.

[32] Víctor Manuel Ramos Ramos. *Thesis: Robust and reliable multimedia transmission over the Internet*. PhD thesis, Université de Nice, Sophia-Antipolis, France, 2004.

[33] Mohan Krishna Ranganathan and Liam Kilmartin. Neural and fuzzy computation techniques for playout delay adaptation in VoIP networks. *IEEE Transactions on Neural Networks*, 16:1174–1194, September 2005.

[34] Jonathan Rosenberg, Lili Qiu, and Henning Schulzrinne. Integrating packet FEC into adaptive voice playout buffer algorithms on the Internet. In *Proceedings of the IEEE Infocom*, volume 3, pages 1705–1714, Tel Aviv, Israel, March 2000.

[35] H. Sanneck, A. Stenger, K. Younes, and B. Girod. A new technique for audio packet loss concealment. In *Proceedings of the IEEE Global Internet*, pages 48–52, London, England, November 1996.

[36] A. S. Sasse and Vicky Hardman. Multi-way multicast speech for multimedia conferencing over heterogeneous shared packet networks. RAT-robust audio tool. Technical report, EPSRC Project GRIK72780, 1996 February.

[37] Henning Schulzrinne. Guide to NeVoT 3.28. Technical report, 1995. GMD Fokus.

[38] Henning Schulzrinne, Steven Casner, R. Frederick, and Van Jacobson. RTP: A transport protocol for real-time applications. RFC 3550, July 2003.

[39] Henning Schulzrinne and Jonathan Rosenberg. The IETF Internet telephony: Architecture and protocols. *IEEE Network*, 13:18–23, May/June 1999.

[40] Aziz Shallwani and Peter Kabal. An adaptive playout algorithm with delay spike detection for real-time VoIP. In *Proceedings of the IEEE Canadian Conference on Electrical Computer Engineering*, pages 997–1000, May 2003.

[41] Cormac J. Sreenan, Jyh-Cheng Chen, and Prathima Agrawal. Delay reduction techniques for playout buffering. *IEEE Transactions on Multimedia*, 2(2):88–100, June 2000.

[42] Lingfen Sun and Emmanuel C. Ifeachor. New models for perceived voice quality prediction and their applications in playout buffer optimization for VoIP networks. *IEEE International Conference Communications*, 3:1478–1483, June 2004.

[43] Lingfen Sun and Emmanuel C. Ifeachor. Voice quality prediction models and their application in VoIP networks. *IEEE Transactions Multimedia*, 8:809–820, August 2006.

[44] Guy Thomsen and Yashvant Jani. Internet telephony: going like crazy. *IEEE Spectrum*, 37:52–58, May 2000.

[45] Kuo-Kun Tseng and Yuan-Cheng Laiand Ying-Dar Lin. Perceptual codec and interaction aware playout algorithms and quality measurements for VoIP systems. *IEEE Transactions on Consumer Electronics*, 50(SI):297–305, February 2004.

[46] International Telecommunicaion Union. Métodos de determinación subjetiva de la calidad de transmisión, August 1996. ITU-T Recommendation P.800.

[47] International Telecommunicaion Union. Objective measuring apparatus, February 1998. Appendix 1: Test signals, ITU-T Recommendation.

[48] International Telecommunicaion Union. El modelo E, un modelo informático para utilización en planificación de la transmisión, May 2005. ITU-T Recommendation G.107.

[49] Li Zhang, Zhen Liu, and Cathy Honghui Xia. Clock synchronization algorithms for network measurements. In *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 160–169, 2002.