JUAN JOSÉ RAMOS MUÑOZ

MEJORAS A LA TRANSMISIÓN DE VOZ SOBRE IP CONSIDERANDO CRITERIOS DE CALIDAD EXPERIMENTADA. SELECCIÓN AUTOMÁTICA DE PROTOCOLOS

MEJORAS A LA TRANSMISIÓN DE VOZ SOBRE IP CONSIDERANDO CRITERIOS DE CALIDAD EXPERIMENTADA. SELECCIÓN AUTOMÁTICA DE PROTOCOLOS

JUAN JOSÉ RAMOS MUÑOZ

Escuela Técnica Superior en Ingenierías
Informática y de Telecomunicación

Universidad de Granada

Teoría de la Señal, Telemática y Comunicaciones

2009

.

Editor: Editorial de la Universidad de Granada
Autor: Juan José Ramos Muñoz
D.L.: GR. 2045-2009
ISBN: 978-84-692-2248-5

DECLARACIÓN

D. Juan Manuel López Soler
Profesor titular de Ingeniería Telemática del Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada

CERTIFICA:

Que la presente memoria, titulada *Mejoras a la Transmisión de Voz sobre IP considerando Criterios de Calidad Experimentada. Selección Automática de Protocolos*, ha sido realizada por D. Juan José Ramos Muñoz bajo mi dirección en el Departamento de Teoría de la Señal, TeleMática y Comunicaciones de la Universidad de Granada. Esta memoria constituye la Tesis que D. Juan José Ramos Muñoz presenta para optar al grado de Doctor con Mención Europea por la Universidad de Granada.

Granada, 2009

DIRECTOR: Juan Manuel López Soler

DOCTORANDO: Juan José Ramos Muñoz

Our understanding is correlative to our perception.
— Robert Delaunay

Information's pretty thin stuff unless mixed with experience.
— Clarence Day

Yes, we can
— Barack Hussein Obama

Dedicado a todos los Hombres y Mujeres de la Tierra. Y a sus mascotas.

2009

ABSTRACT

Nowadays the convergence of voice and data networks is a fact. The integration of both types of data enables the development of new types of applications and services, and saves infrastructure and maintenance costs.

However, IP networks provide a best effort service, which means that they can not guarantee that the data will be delivered within a given period. Consequently, IP networks can not meet the requirements of multimedia transmissions consisting of tight time budgets and some degree of reliability. In this sense, the lost of consecutive packets is a critical issue, since it rapidly degrades the quality of the perceived service.

To address these issues by means of solutions which can be deployed in current networks, new application-level schemes have been proposed to alleviate the shortcomings of the IP protocol. Basically, they involve end to end recovery mechanisms which are executed at the sender and receivers equipments. These mechanisms achieve packet loss recovery techniques such as automatic retransmission, forward error correction, etc.

There exist alternative solutions, located at intermediate nodes, which perform these recovery techniques within the network. This new paradigm offers a promising framework to develop new recovery schemes.

To evaluate the performance of the recovery mechanisms, effective quality of service measures are needed. These measures have to assess the subjective quality of experience (QoE) perceived by the final users, and to estimate it from objective network parameters.

As a first contribution, in the present thesis we have developed a non-intrusive measure which assess the intelligibility level of voice over IP (VoIP) flows . This measure enables the online monitoring of the perceptual quality of VoIP flows.

Additionally, we have developed a retransmission-based recovery mechanism which performs early packet loss detection to minimize the delay incurred by the retransmission procedure. Furthermore, we have successfully designed a preventive recovery mechanism based on a low delay packet interleaver which uses packets from several VoIP flows as input.

For both recovery procedures we provide the obtained packet loss patterns and the average packet recovery delay as well. After assessing the performance of the proposed algorithms, we establish heuristics algorithms for selecting their best network locations in terms of perceptual quality and intelligibility evaluations.

Finally, we propose a framework for automatic online protocol selection. It chooses the protocol which best fits a network state. In conjunction to the aforementioned heuristics, this framework provides a complementary method for selecting the multimedia packet loss recovery mechanisms best suited to a given network condition.

RESUMEN

La integración de datos y voz es una realidad constatable. La unificación en la transmisión de ambos tipos de información está permitiendo el desarrollo de nuevos servicios y aplicaciones, abaratando los costes de comunicación y de mantenimiento de las infraestructuras.

Sin embargo, el diseño de la red IP, que ofrece un servicio "best effort" de entrega de datagramas, en general no garantiza que se satisfagan los requisitos de tiempo real (retardos y fluctuaciones de retardos acotados) y de robustez (probabilidad de pérdidas acotadas, disponibilidad de mecanismos de recuperación) que la transmisión de flujos multimedia continuos requiere. Especialmente críticas son las pérdidas consecutivas de paquetes, que producen la rápida degradación de los flujos de voz.

Para mitigar las deficiencias del protocolo IP mediante soluciones implementables en las redes actuales, en otros trabajos se han propuesto soluciones en el nivel de la capa

de aplicación. Básicamente, la idea consiste en que el receptor y el emisor lleven a cabo mecanismos de detección y reparación de pérdidas de paquetes, ya sea solicitando la retransmisión de dichos paquetes, añadiendo redundancia al flujo de paquetes, o regenerando la información dañada. Existen alternativas consistentes en aplicar estas técnicas de reparación en nodos intermedios de la red, en lugar de hacerlo extremo a extremo. Esta nueva aproximación ofrece un marco interesante para el desarrollo de nuevas soluciones de reparación.

Para poder evaluar la efectividad de los mecanismos de reparación, es necesario contar con medidas de la calidad de servicio experimentada por los usuarios (QoE). Dicha medida debe modelar la calidad subjetiva de los usuarios, a partir de parámetros objetivos de la red.

Como resultado del trabajo de la presente tesis, se ha llevado a cabo el diseño de una medida no intrusiva de inteligibilidad para flujos de voz, que permite monitorizar en línea la calidad perceptual de un flujo de voz.

Asimismo, se ha diseñado un servicio reactivo de reparación de pérdidas de paquetes, basado en la retransmisión de los mismos mediante la detección temprana de las pérdidas.

Además se ha realizado con éxito el diseño de un servicio preventivo de reparación de pérdidas de paquetes, basado en un entremezclador de bajo retardo de paquetes de distintos flujos de VoIP.

Para ambos mecanismos se ha obtenido la caracterización del patrón de pérdidas resultante tras aplicar los procedimientos de reparación, así como el retardo medio requerido para la reparación. A partir de la caracterización del rendimiento de estos mecanismos, se han propuesto sendos algoritmo heurísticos para la selección de la mejor ubicación de cada mecanismo en la red, según la estimación de calidad perceptual y la medida de inteligibilidad esperadas que han sido propuestas en esta tesis.

Finalmente se ha llevado a cabo el diseño preliminar de una plataforma general de selección automática de protocolos que escoge de entre un conjunto de candidatos, el protocolo que mejor se ajusta a las condiciones de red existentes.

PUBLICACIONES

Algunas ideas y figuras de esta tesis han aparecido previamente en las siguientes publicaciones:

- Juan J. Ramos-Muñoz and Juan M. Lopez-Soler, *Low Delay Multiflow Block Interleavers for Real-Time Audio Streaming*, ICN 2005, LNCS 3420, pp. 909–916, 2005.
- Juan J. Ramos-Muñoz, Angel M. Gómez, and Juan M. Lopez-Soler, *Intelligibility Evaluation of a VoIP Multiflow Block Interleaver*. IWAN 2005, LNCS 4388, pp. 197–202, 2009.
- Juan J. Ramos-Munoz and Juan M. Lopez-Soler, *Efficient Allocation for VoIP Packet Losses Detection Services in Programmable and Active Networks*, Proceedings of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services (ICAS/ICNS 2005).
- Juan J. Ramos-Munoz, Lidia Yamamoto and Christian Tschudin, *Serial Experiments Online*. ACM SIGCOMM Computer Communication Review, Vol.38, N. 2, April 2008, pp. 33–42.
- Ángel M. Gómez, Juan J. Ramos-Muñoz, Antonio M. Peinado, Victoria Sánchez, *Multi-Flow Block Interleaving Applied to Distributed Speech Recognition Over IP Networks*. Proc. of ICSLP'2006, Pittsburgh (USA), Sep. 2006.

Maybe I didn't treat you
Quite as good as I should have
Maybe I didn't love you
Quite as often as I could have
Little things I should have said and done
I just never took the time
You were always on my mind
You were always on my mind

Johnny Christopher, Mark James y Wayne Carson Thompson.
Always on My Mind, 1972.

AGRADECIMIENTOS

Quiero agradecer en primer lugar a Juan Manuel López Soler, mi director de Tesis y amigo, que ha hecho posible esta tesis. Gracias por tu infinita paciencia.

A mis padres y a mi hermano, por su incondicional apoyo.

A Raquel, por aguantarme a pesar de todo.

A mis compañeros del grupo de investigación de *Teoría de la Señal, Telemática y Comunicaciones*, por las siempre fructíferas charlas científicas, y por las otras. Casi todo lo que sé me lo han enseñado ellos.

Un agradecimiento especial al Prof. Christian Tschudin, Lidia Yamamoto y Christophe Jelger, por acogerme durante mi estancia en el grupo de *Computer Networks* de la Universidad de Basel, por su paciencia conmigo, y por mostrarme su asombrosa capacidad de trabajo.

Por último gracias a todos los que, aunque no os haya nombrado, habéis estado a mi lado a pesar de la distancia, durante todos estos años.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

1 INTRODUCCIÓN

1.1 MOTIVACIÓN

La integración de datos y voz es una realidad constatable. La unificación en la transmisión de ambos tipos de información está permitiendo el desarrollo de nuevos servicios y aplicaciones, abaratando los costes de comunicación y de mantenimiento de instalaciones infraestructuras. La tendencia actual es la integración de todos los servicios de comunicación utilizando IP como tecnología de convergencia.

Las aplicaciones de VoIP (Voz sobre IP) transmiten sobre una red de conmutación de paquetes la señal de voz en tiempo real, convenientemente codificada y empaquetada. Sin embargo, el diseño de la red IP, que ofrece un servicio *best effort* de entrega de datagramas, en general no garantiza que se satisfagan los requisitos de tiempo real (retardos y fluctuación de retardos acotados) y de fiabilidad (probabilidad de pérdidas acotadas, disponibilidad de mecanismos de recuperación) que la transmisión de flujos multimedia continuos requieren.

Para satisfacer las demandas de acotación de retardos, de sus fluctuaciones y de pérdidas o errores en la red, se proponen distintas alternativas sobre el servicio no fiable de *mejor esfuerzo* de IP. Estas nuevas propuestas se aplican a distintos niveles. En este sentido, es constatable el esfuerzo normalizador por parte del IETF (*Internet Engineering Task Force*) que en torno a los años 1990 propuso el modelo de *Internet de Servicios Integrados* [1] y más tarde el modelo de *Internet de Servicios Diferenciados* [2]. Sin embargo, su utilización requiere de la implantación mayoritaria de *routers* que soporten estas tecnologías. Por lo que lejos de conseguir un despliegue global, dichas soluciones han sido adoptadas

en escenarios o contextos muy reducidos.

Con independencia de lo anterior, se han propuesto otras aproximaciones a nivel de la capa de aplicación para mitigar las deficiencias del protocolo IP. Básicamente, la idea es que el receptor y el emisor lleven a cabo mecanismos de detección y reparación de pérdidas de paquetes, ya sea solicitando dicho paquete perdido, añadiéndole al original información redundante, o regenerando la información dañada. Estas técnicas son fácilmente desplegables ya que se implementan en los extremos de la comunicación.

Además de la alternativa extremo a extremo, existen alternativas basadas en aplicar la reparación en nodos intermedios de la red, que ofrecen un marco atractivo para el desarrollo de nuevas soluciones para hacer frente a las carencias de IP en la transmisión de flujos continuos.

Para satisfacer las demandas de calidad exigidas por el usuario, es imprescindible sistematizar una medida de calidad de servicio. A pesar de que la información sobre retardos medios experimentados por los paquetes del flujo, la tasa media de pérdida paquetes, etc. son variables fácilmente observables, la calidad que percibe el usuario, o Calidad de Experiencia de Usuario (QoE), no es fácilmente evaluable. Para mejorar el diseño de los mecanismos involucrados es importante incorporar criterios que tengan en cuenta la QoE. Para tal fin es necesario disponer de funciones que estimen el nivel de satisfacción del usuario del servicio en función de valores objetivos observables en la red.

1.2 OBJETIVOS

El objetivo general de esta tesis es la de proporcionar nuevos mecanismos adaptables que tengan como objeto maximizar la QoE experimentada por el usuario final en flujos de voz sobre IP, y que se puedan desplegar en las redes IP actuales. En concreto, los mecanismos a diseñar deben hacer frente a las pérdidas de paquetes de VoIP en la red, introduciendo el

menor retardo de reparación de cada paquete. Esto afectará indirectamente a la calidad subjetiva percibida, ya que se sabe que ambos parámetros influyen sobre la degradación de la calidad de experiencia de los usuarios.

Los objetivos particulares de esta tesis se pueden desglosar como sigue:

- El desarrollo de mecanismos de reparación de pérdidas adaptables, ejecutados en nodos intermedios de la red.
- La caracterización del rendimiento de los mecanismos desarrollados, proporcionando una estimación del retardo y la distribución de pérdidas que resulte de su aplicación.
- La provisión de algoritmos que permitan escoger la mejor configuración de cada mecanismo, dadas unas condiciones de red concretas, tal que se maximicen criterios de QoE, una vez conocido el rendimiento del mecanismo de reparación.
- La provisión de algoritmos que permitan seleccionar la ubicación de los mecanismos de reparación diseñados, que maximicen criterios de QoE, una vez conocido el rendimiento del mecanismo de reparación.
- La provisión de medidas de calidad subjetiva de flujos de VoIP que sirvan como criterio para la toma de decisiones.
- La proposición de un procedimiento general para la selección de la mejor técnica de reparación de entre el conjunto de técnicas disponibles en un nodo intermedio.

1.3 CONTRIBUCIONES DE LA TESIS

Del cumplimiento de los objetivos anteriores se han generado las siguientes contribuciones, algunas de ellas parcialmente difundidas en las publicaciones científicas que se detallan para cada punto:

- Diseño de una medida no intrusiva de inteligibilidad para flujos de voz codificada con PCM a 8Khz, basada en un reconocedor automático del habla. Parte de los resultados se han incluido en [3], [4].

- Diseño de un servicio reactivo de reparación de pérdidas de paquetes, basado en la retransmisión de paquetes mediante la detección temprana de las pérdidas ([5]). Adicionalmente han resultado las siguientes contribuciones:
  - Caracterización del patrón de pérdidas resultante de aplicar el servicio de reparación reactivo.
  - Caracterización del retardo medio introducido por el servicio.
  - Algoritmo heurístico para la selección de la mejor ubicación, según la estimación de calidad perceptual ([6], [7]).

- Diseño de un servicio preventivo de reparación de pérdidas de paquetes, basado en un entremezclador de bajo retardo de paquetes de distintos flujos de VoIP ([8], [4]). Adicionalmente han resultado las siguientes contribuciones:
  - Caracterización del patrón de pérdidas resultante de aplicar el servicio de reparación preventivo.
  - Caracterización del retardo medio introducido por el servicio.
  - Algoritmo heurístico para la selección de la mejor configuración y tipo de entremezclador, según la estimación de calidad perceptual y la medida de inteligibilidad propuesta en esta tesis.

- Diseño preliminar de una plataforma general de selección automática de protocolos [9].

Por otro lado, algunas de las contribuciones han formado parte de los resultados de los proyectos nacionales SERVIRA (referencia TIC2002-02798) y GIDRE (referencia TSI2005-08145-C02-02).

1.4 ESTRUCTURA DE LA TESIS

La tesis se estructura en 7 capítulos. En el capítulo 2 se proporciona una revisión del estado del arte de los aspectos y tecnologías más relevantes relacionados con el tema de la tesis. Entre ellas se tratan las técnicas de reparación clásicas extremo a extremo, dos de las cuales se adaptan en los capítulos posteriores para poder ejecutarse en nodos intermedios. También se ofrece una visión general sobre las medidas de calidad perceptual de voz más utilizadas actualmente.

En el capítulo 3 se diseña y evalúa una medida de inteligibilidad de voz que puede aplicarse en tiempo real como medida de calidad perceptual.

El capítulo 4 expone el diseño y evaluación de un mecanismo preventivo de reparación basado en un entremezclador de distintos flujos de voz.

Complementariamente, el capítulo 5 desarrolla el diseño y evaluación de varios mecanismos reactivos de reparación basados en la detección de pérdidas y solicitud de retransmisión de los paquetes perdidos.

El capítulo 6 desarrolla la propuesta de plataforma para la selección automática en línea, de entre un conjunto de candidatos disponibles, del protocolo que mejor se adapte a las condiciones de la red.

Por último, en los capítulos 7 y 8 se exponen las conclusiones finales de la tesis, en español y en inglés respectivamente, tal y como exige la normativa vigente de la Universidad de Granada sobre *Tesis con Mención Europea*. A continuación de las conclusiones se adjunta un resumen en inglés de las principales contribuciones de la tesis, se identifica la bibliografía utilizada, y finalmente se incluyen los apéndices referidos en los distintos capítulos.

2

ANTECEDENTES

2.1 INTRODUCCIÓN

El tráfico multimedia inelástico posee unas características propias que lo diferencian del tráfico elástico. Destacan especialmente su relativa tolerancia a pérdidas, su exigencia de retardos extremo a extremo acotados y su exigencia de fluctuaciones en el retardo acotadas. Por otro lado, es conocido que el protocolo IP se ha erigido como tecnología de convergencia para la transmisión de todo tipo de información. A pesar de ello, el servicio *best effort* de IP, no verifica plenamente los requisitos demandados por este tipo de aplicaciones. Es por ello que se hace necesario el diseño de mecanismos que palíen las deficiencias de las redes IP, las cuales no fueron concebidas para soportar estas aplicaciones con demandas de calidad de servicio.

De entre estos mecanismos, en este capítulo se estudiarán los mecanismos de recuperación de paquetes multimedia perdidos. Para el tráfico elástico, dichos mecanimos de reparación se han llevado a cabo tradicionalmente extremo a extremo. Por lo general, esta aproximación no es factible para tráfico inelástico. En esta tesis se proponen varios procedimientos tal que operado en modo local -es decir, en un nodo intermedio de la red- ayuden a mejorar la calidad percibida. Aunque esta aproximación se podría llevar a cabo mediante nodos dedicados, en la literatura se han propuesto tecnologías que permiten que los nodos intermedios de la red (routers o pasarelas) puedan procesar los paquetes que reenvían. Entre ellas destacan las aproximaciones de *redes programables* y las *redes superpuestas* (*Overlay Networks*).

Las redes programables definen una tecnología que potencialmente puede ser adoptada para mejorar e incrementar los servicios de red ofrecidos. Aunque en los esquemas de redes activas surgen problemas de seguridad y rendimiento (que caen fuera del ámbito de este trabajo), estas ofrecen un

nuevo paradigma para desarrollar nuevos y mejores servicios. Por ejemplo, para aplicaciones multimedia se pueden detectar de forma temprana las pérdidas de paquetes y recuperarlas a tiempo dentro de la red verificando en algunos casos los requisitos impuestos.

De forma alternativa, la aproximación entre iguales (*peer-to-peer*), que ha sido ampliamente utilizada para las aplicaciones de compartición de ficheros, puede ser también aplicada de forma exitosa a los servicios de transmisión de voz, como así hace el popular servicio de VoIP Skype [10]. Para ello, es necesario la cooperación de los sistemas finales de los usuarios de forma tal que operen como nodos intermediarios.

Este capítulo, por tanto, tiene como objetivo fijar el contexto y los antecedentes relacionados con los mecanismos de recuperación en aplicaciones de VoIP.

Para ello, el capítulo se ha organizado en las siguientes secciones. En la sección 2.2 se describen la características propias del tráfico de voz que determinan sus demandas de calidad de servicio, y que lo distinguen de otros tipos de tráfico. En la sección 2.3 se enumeran los problemas que introduce el protocolo IP y que afectan a las aplicaciones de transmisión de voz. A continuación, en la sección 2.4 se comentan varias tecnologías que ofrecen la posibilidad de procesar los paquetes multimedia en nodos intermedios de la red. Dado que en este trabajo se proponen varias técnicas de reparación, en el apartado 2.5 se enumeran las técnicas tanto extremo a extremo como ejecutadas en nodos intermedios para recuperar pérdidas de paquetes. Finalmente, utilizadas como herramientas de evaluación, en la sección 2.6 se estudiarán las medidas de calidad subjetiva más habituales.

2.2 CARACTERÍSTICAS DE LAS APLICACIONES DE AUDIO

Las características y demandas de QoS de los flujos continuos generados por las aplicaciones multimedia, particularmente VoIP, hacen que TCP, protocolos extremo a extremo

fiable, no sea adecuado para transportar este tipo de datos. Principalmente esto se debe a la necesidad de que la información multimedia debe llegar a su destino dentro de un intervalo de tiempo acotado. En concreto, las aplicaciones multimedia, y en particular las de transmisión de voz, se caracterizan por [11]:

- **Requisitos estrictos de temporización.** Si los datos no son entregados al receptor antes de un cierto umbral, d_{max}, en general dichos datos no son útiles para la aplicación. Típicamente, para conversaciones de voz, el retardo máximo admitido es de $300ms$ ([12],[13]). Un retardo superior a ese umbral conllevaría que no se pudiera llevar a cabo una conversación de voz interactiva.

  Como consecuencia directa, las aplicaciones de este tipo suelen utilizar el servicio que ofrece el protocolo UDP [14], pues los mecanismos de retransmisión de TCP junto a la superposición de mecanismos de control de congestión pueden introducir retardos no acotados.

- **Tolerancia a pérdidas.** A diferencia de la transmisión de datos procedentes de aplicaciones en las que a los receptores deben llegar libres de errores todos los datos enviados, como es el caso de la transferencia de ficheros, correo electrónico, etc., el carácter redundante de la señal de voz permite que la pérdida de parte de los datos transmitidos no menoscabe la inteligibilidad de la información en el usuario final. La cantidad de pérdidas que pueden ser toleradas depende del tipo de contenido, de su redundancia y de las técnicas de codificación empleadas en el fragmento de señal correspondiente al paquete perdido.

- **Generación periódica de la información.** La información generada por una aplicación multimedia tiene por lo general un carácter periódico. Esto quiere decir que, suponiendo un codificador con tasa de bits constante, la aplicación generará un cantidad fija de datos cada periodo fijo t_f. Sin embargo, el proceso

de encolado en los *routers* intermedios, y las medidas adoptadas por el control de congestión pueden introducir fluctuaciones en el retardo experimentado paquete a paquete.

Además de las características mencionadas, se deben contemplar otras propiedades relativas al esquema de codificación utilizado. Según el tipo de codificador, la tolerancia a pérdidas será diferente, ya que la calidad dependerá de la dependencia de las tramas vecinas. Por tanto, se ha de tener también en cuenta los siguientes aspectos de los esquemas de codificación:

- **Codificación independiente entre tramas.** Como en el caso de PCM [15], las distintas tramas se codifican de forma independiente. La pérdida de una de estas tramas no tendrá impacto en el proceso de decodificación de las tramas vecinas.

- **Codificación con dependencias entre tramas.** La codificación de una trama se basa en la información de las tramas adyacentes, ya que explota la redundancia que existe entre ellas. La descodificación de dicha trama requiere utilizar información de las tramas de las que dependió en la codificación. Por ello, la pérdida de un paquete que contenga una de esas tramas degradará la señal resultante.

- **Codificación jerárquica** En este caso, en el origen, la información se particiona en varios *flujos constituyentes* o capas. La capa inferior contiene la información esencial para la reconstrucción de la señal en los receptores, mientras que las capas superiores realzan la calidad de la señal decodificada.

Finalmente, según el tipo de interacción de la aplicación, existirán diferentes requisitos de retardos extremo a extremo:

- **Interactivo:** el retardo extremo a extremo está estrictamente acotado. Se ha estudiado el impacto del retardo extremo a extremo durante conversaciones de

voz, concluyendo que para que se pueda llevar a cabo un diálogo interactivo, el intervalo de tiempo que va desde que un usuario emite la señal de voz hasta que el interlocutor la oye, debe ser inferior a 300ms ([12]). En caso contrario, será imposible establecer una conversación fluida, y se pasará a una comunicación *half-dúplex*, en la que cada interlocutor espera su turno de palabra.

- **No interactivo:** en este caso no es un factor tan crítico el retardo extremo a extremo. No obstante, siempre habrá que respetar el retardo máximo de cada paquete, para que la reproducción de la señal en el destino sea continua. En este escenario, el margen de tiempo del que se dispone para realizar tareas de reparación de pérdidas es mucho mayor, pudiéndose emplear mecanismos no aplicables en transmisiones interactivas.

2.3 DIFICULTADES EN LA TRANSMISIÓN DE AUDIO POR INTERNET

Como se ha citado anteriormente, el protocolo IP no fue diseñado para transmitir información con garantías de retardos acotados. De hecho, IP ni siquiera garantiza la entrega del paquete. Los problemas a los que se enfrentan las aplicaciones que transmiten un flujo continuo de voz se resumen en los siguientes subapartados.

2.3.1 *Retardo extremo a extremo*

El retardo extremo a extremo es el resultado de la suma de cada uno de los retardos intermedios sufridos durante el proceso de transmisión de cada paquete. Dichos tiempos se enumeran a continuación.

- El procesamiento en la aplicación también está involucrado en los tiempos de retardo, en operaciones como el muestreo de la señal y la reproducción, ya que el dispositivo de audio posee su propio búfer que ha de ser completado antes de pasar los datos a la apli-

cación. Incluso la carga del sistema puede influir en la planificación de las tareas de la aplicación, retrasando por tanto su ejecución.

- El retardo debido a la *codificación* de la señal se compone de dos elementos. Por un lado, el retardo necesario para segmentar la señal y realizar la extracción de características (parametrización). Por otro lado, el retardo correspondiente al procesamiento requerido por el algoritmo de codificación/descodificación.

- El tiempo de *paquetización* es el que se emplea en encapsular/desencapsular los datos de acuerdo con los distintos protocolos en el emisor y el receptor. Este proceso engloba la adición de las cabeceras y campos de cada protocolo. Estos retardos se pueden reducir diseñando protocolos simples que minimicen el tiempo necesario de procesamiento.

- El *tiempo de transmisión* (t_{tr}) es el tiempo tarda cada interfaz de red en generar la trama completa en el correspondiente enlace.

- El *tiempo de propagación* (t_p) es el tiempo que la señal necesita para atravesar los enlaces que conforman el camino desde el emisor a los receptores. Este retardo impone un límite físico que no puede ser reducido.

- El *tiempo de conmutación*, retardo introducido en los *routers* debido a las tareas de conmutado de los paquetes. Se debe básicamente a la extracción de la dirección de destino del datagrama, la consulta de las tablas de encaminamiento, el reenvío del datagrama por los puertos de salida correspondientes.

2.3.2 *Fluctuación del retardo o* jitter

La fluctuación del retardo o *jitter* se define como el retardo adicional de un paquete con respecto al tiempo necesario para ir desde el emisor al receptor. Por esta definición, este valor sólo puede ser mayor o igual que cero.

El *jitter* se genera básicamente por los retardos sufridos en las colas de los *routers*.

Las aplicaciones multimedia exigen una recepción continua de información, por lo que para paliar el efecto del *jitter* en el receptor se dispone de una memoria temporal que amortigüe las fluctuaciones ([16],[17],[18], [19]).

2.3.3 *Reordenación de paquetes*

Según experimentos realizados [20], en Internet las rutas a veces sufren una alta incidencia de reordenación de datagramas.

La reordenación de paquetes se produce cuando varios paquetes con el mismo origen y destino se propagan por caminos diferentes. A pesar de que los encaminadores en Internet suelen utilizar colas con disciplina FIFO (*First Input First Output*), cuando una ruta cambia y el retardo de ésta es menor, puede ocurrir que algunos datagramas posteriores lleguen antes que los que se enviaron por la ruta antigua.

La solución para detectar este problema se basa en incorporar un número de secuencia en el paquete (por ejemplo como hace RTP [21]) de forma que el receptor pueda identificar el orden cada paquete en el flujo.

2.3.4 *Pérdida de paquetes*

La pérdida de paquetes tiene un gran impacto en la calidad de la señal resultante. Por lo general, cuanto más paquetes consecutivos se pierdan, es decir, se produzca una ráfaga de pérdidas, mayor será el deterioro de la señal reconstruida en

los receptores ([22],[23]). Las causas por las que se pueden producir pérdidas de paquetes son las siguientes ([24]):

- La congestión de los encaminadores pueden producir el descarte de los datagramas que no puedan almacenarse en las colas de salida. Las soluciones más extendidas para aliviar este problema se centran en mecanismos de control de congestión que sean capaces de responder rápidamente a una incipiente congestión (p.e. *Random Early Detection*, [25]).
- Como ya se ha mencionado, las aplicaciones con restricciones de tiempo real desecharán los paquetes correspondientes a información que llegue después del momento en el que debía ser usado. Esta eventualidad también se interpreta como una pérdida.
- La sobrecarga de las estaciones de trabajo puede influir en la planificación de procesos de un sistema multitarea, impidiendo llevar a cabo las tareas de recepción, o incluso retrasar excesivamente la reproducción de una trama.

2.3.5 *Replicación de paquetes*

Este comportamiento anómalo consiste en que la red entrega varias copias del mismo datagrama. Su causa más probable es que se haya producido alguna retransmisión errónea en el nivel de enlace [20]. La numeración secuencial de los paquetes mediante RTP (*Real Time Protocol*, [21]) permite a la aplicación distinguir las copias.

2.3.6 *Corrupción de datos*

La probabilidad de error a nivel de bit es generalmente muy baja ([11]). Sin embargo, en escenarios inalámbricos, estos errores pueden darse con mayor frecuencia, produciéndose la alteración del contenido de algunos paquetes. Estos paquetes corruptos también se consideran paquetes perdidos.

2.4 REDES PROGRAMABLES Y REDES SUPERPUESTAS

En esta tesis se propone proporcionar nuevos mecanismos de reparación ubicados en nodos intermedios de la red. Para ello se requiere que los nodos puedan ejecutar el código de reparación sobre los paquetes que reenvía.

La idea de habilitar a los nodos intermedios de la red para realizar operaciones adicionales a las de clasificación y encaminamiento no es nueva, y se aplica en servicios tales como *cortafuegos* implementados en encaminadores y *puertas de enlace*.

Las tecnologías más maduras que ofrecen estas capacidades son las *redes activas* ([26], [27], [28]), y las *redes superpuestas* (*Overlay Networks*, [29]).

En concreto, las redes activas han demostrado su efectividad para ejecutar mecanismos de reparación para flujos multimedia (p.e. [30], [31], [32], [33]).

2.4.1 *Redes activas*

En 1994 y 1995, la comunidad científica investigadora del DARPA (*Defense Advanced Research Projects Agency*) englobó bajo el nombre de *Redes Activas* varias propuestas para solucionar algunos desafíos que planteaba Internet, como por ejemplo la ejecución de servicios no extremo a extremo. Asimismo se pretendía agilizar la incorporación de nuevas tecnologías y estándares en la infraestructura de red, sin requerir largos procesos de normalización, permitiéndose así incorporar nuevos protocolos que solventaran las deficiencias de TCP/IP.

La tecnología de redes activas dota a los *routers activos* de la capacidad para ejecutar código sobre la información de los paquetes que retransmite, pudiendo acceder a los recursos del nodo, y ofreciendo mecanismos para poder cargar el código de programa de nuevos protocolos.

La principal ventaja de este enfoque reside en su flexibilidad. La implementación de nuevos protocolos, la reparación de errores en el software de red, hasta la posibilidad de ofrecer un procesamiento específico durante una sesión in-

dividual, son posibles con esta nueva tecnología.

Paradigmas de redes activas

En los encaminadores actuales, la información que portan los datagramas IP no son más que una secuencia de bits. La interpretación de dicha información se realiza en las aplicaciones en los nodos finales. Por el contrario, las redes activas dotan a los paquetes de capacidad de definir operaciones a realizar sobre ellos mismos. Dotando a los paquetes de categoría de objetos, y permitiendo a los nodos internos de la red a conocer la semántica de dichos paquetes, la mejora de la flexibilidad en la red para el despliegue de servicios está asegurada.

El concepto de *redes activas* se ha materializado en dos aproximaciones diferenciadas: *conmutadores programables* ([26]) y *cápsulas* ([34], [35]):

- **Conmutadores programables:** la estructura de los mensajes se mantiene, pero los routers se dotan de un mecanismo para descargar programas que procesarán el tráfico que pasa por el conmutador. En esta estrategia, el código a ejecutar en el nodo se envía inicialmente por el usuario, mediante algún mecanismo fiable. *SwitchWare* ([34]) es un ejemplo de esta aproximación.

- **Cápsulas:** en este caso, los paquetes transportan el código que será ejecutado en los nodos. Los datos de usuario se transportan embebidos en estas cápsulas.

  El código transportado puede consistir en instrucciones básicas sobre el contenido de las cápsulas, y en primitivas para el acceso a los recursos del nodo.

  Un ejemplo de implementación de red activa orientada a cápsulas fue *ANTS* ([36]), arquitectura basada en JAVA.

Ventajas y desventajas de los nodos programables

La posibilidad de ejecutar en los routers activos los programas que transporten las cápsulas que reciba, produce nuevos problemas de seguridad que emanan de la ejecución de código que no es propio del nodo ([33]). Este es uno de los problemas principales a los que se enfrenta el diseño del entorno de ejecución de los nodos activos.

El código de los programas activos debe ser portable, seguro y eficiente ([26]). Debe ser portable para que pueda ser ejecutado en el mayor número de plataformas diferentes; seguro en cuanto a la capacidad de controlar el acceso de los programas a los recursos del nodo activo; y eficiente para no degradar el servicio de red ofrecido por el encaminador.

Sin embargo, dichos requisitos implica un compromiso entre los anteriores objetivos. Dado que el código ejecutable debe ser portable, la eficiencia del código es menor que la del código escrito expresamente para el hardware de un encaminador concreto. Para conseguir seguridad en cuanto al acceso a los recursos, deben realizarse comprobaciones que pueden comprometer la eficiencia del programa.

La gestión de recursos es otro problema fundamental que requiere solución. Incluso el código de un mismo flujo puede competir por recursos tanto físicos (memoria, ancho de banda, etc.), como lógicos (tablas de encaminamiento, información de gestión del nodo, etc.). Por tanto, hay que prestar especial atención a la reserva segura de recursos en los nodos activos.

2.5 TÉCNICAS DE REPARACIÓN DE PÉRDIDA DE PAQUETES

Como ya se comentó en el apartado 2.3, uno de los problemas a los que se enfrentan los flujos continuos de contenido audiovisual, y en particular los de voz, es el problema

de la pérdida de paquetes. Si bien los flujos multimedia se caracterizan por su tolerancia a una tasa moderada de pérdidas, la pérdida de tramas consecutivas (o *ráfaga de pérdidas*) degradan rápidamente la calidad del audio o vídeo recibidos.

Para mitigar el efecto de estas pérdidas consecutivas de paquetes, se han propuesto numerosos mecanismos de recuperación de pérdidas, que generalmente se ejecutan extremo a extremo a nivel de aplicación.

Sin embargo, la aparición de las redes activas y redes Overlay han permitido el desarrollo de nuevas estrategias, y la adaptación de técnicas clásicas para ofrecer mayor robustez a los flujos continuos.

Las técnicas de reparación propuestas hasta el momento pueden ser categorizadas en: *técnicas de recuperación basadas en retransmisión*, *técnicas basadas en regeneración*, y *técnicas de protección de paquetes*.

Los mecanismos de reparación pueden ser también clasificados en *técnicas de reparación reactivas*, en las que tras detectar el error o la pérdida se inicia el mecanismo de reparación, y *técnicas de reparación preventivas*, que añaden robustez al flujo en previsión a los errores que puedan aparecer. Dicho esto, la mayoría de las técnicas de reparación basadas en retransmisión y regeneración pertenecen al grupo de técnicas reactivas, y las restantes al grupo de técnicas preventivas.

A continuación se detallan los distintos mecanismos y procedimientos descritos en los trabajos previos sobre la recuperación de paquetes perdidos. En la sección 2.5.1 se describen los mecanismos de detección de pérdidas de paquetes, que son el primer paso en el proceso de reparación en las técnicas de reparación reactivas. A continuación, el punto 2.5.2 detalla las técnicas de recuperación basadas en retransmisión, y el punto 2.5.3 describe las técnicas basadas en regeneración más significativas. La sección 2.5.4 proporciona una visión general de las técnicas de protección más destacadas.

2.5.1 *Detección de pérdidas*

La detección de la pérdida de un paquete es el evento que permite iniciar la reparación de dicho paquete. Según la entidad que realiza la detección, se definen dos tipos de esquemas de detección:

- **Detección en el emisor:** en este caso, el emisor es el responsable de efectuar la detección basándose en confirmaciones positivas manteniendo información de estado de cada receptor. Este esquema no es adecuado para transmisiones en multidifusión, ya que el emisor debe procesar gran número de confirmaciones, y los *routers* cercanos podrían saturarse ([37]).

- **Detección en el receptor:** en esta aproximación los receptores son los responsables de efectuar la detección. Mantienen por ello su propia información de estado. En caso de detectar una pérdida, se envía una confirmación negativa al servidor de reparaciones. Esta aproximación reduce la información que la fuente tiene que procesar, y permite que este esquema sea más escalable que el anterior.

Debido a los problemas de escalabilidad del primer esquema de detección se prefiere utilizar esquemas de detección en el receptor.

Independientemente de si es el emisor o los receptores los que realizan la detección de las pérdidas, cabe la posibilidad de efectuar las detecciones en nodos intermedios que se encuentren entre la fuente y los receptores ([38], [39]). De esta forma, la detección puede producirse con menos retardo, disminuyendo así el tiempo total de reparación. La posibilidad de reducir el retardo de reparación constituye una cuestión crítica en las aplicaciones que requieran un servicio de transporte con restricciones de tiempo.

Los mecanismos de detección de pérdidas identificados en trabajos relacionados se basan en dos esquemas. El primero de ellos consiste en comprobar que los números de

secuencia de los paquetes son correlativos. En caso de que llegue un paquete con un número no correlativo, se supone que se han perdido los paquetes con números de secuencia intermedios. El segundo mecanismo comprueba si tras la expiración de un plazo de tiempo estimado ha llegado el paquete correspondiente.

En el capítulo 5 se realiza un estudio más exhaustivo sobre el funcionamiento de estos esquemas.

2.5.2 *Técnicas de recuperación basadas en retransmisión*

Estas técnicas básicamente consisten en el reenvío de información de paquetes que hayan sido perdidos. Para ello, la entidad que detecta la pérdida del paquete solicita al servidor de retransmisión el reenvío de dicho paquete. Por tanto, se trata de una técnica de reparación reactiva que es iniciada tras el evento de la detección de una pérdida.

La utilización de las técnicas de retransmisión para la recuperación de paquetes perdidos en aplicaciones de transmisión con requisitos de tiempo real ha sido cuestionada en la mayoría de los estudios realizados al respecto (por ejemplo en [11] y [13]) debido a que se trata de un mecanismo que requiere demasiado tiempo para completar su cometido. Esto es debido a que, para obtener el paquete solicitado, hace falta esperar al menos dos veces el retardo que existe entre el nodo receptor y el nodo servidor de recuperaciones para completar el reenvío. Este retardo puede ser inaceptable en algunos casos al incumplir los plazos máximos permitidos para los paquetes de un flujo.

Sin embargo existen propuestas en las que se defiende la utilización de las técnicas de retransmisión para recuperar pérdidas de paquetes de flujos multimedia ([40],[41]) en entornos tales como redes de área local, o mediante la utilización de servidores de recuperación locales ([38], [42]).

Según la entidad que realice la retransmisión en el proceso de recuperación, los esquemas de retransmisión pueden categorizarse como:

- **Basados en el emisor:** el emisor es el que retransmite los paquetes de recuperación. La repetición de paquetes puede inducir la congestión en los routers entre el emisor y los receptores. De hecho, si se da una alta probabilidad de pérdidas en la red, este esquema es ineficiente. Como ejemplos de protocolos que adoptan este enfoque cabe destacar a XTP (*Xpress Transfer Protocol*, [43]) y RMTP (*Reliable Multicast Transport Protocol*, [44]).

- **Descentralizados:** para lograr escalar el esquema de recuperación, este enfoque utiliza servidores dedicados en lugar de efectuar las retransmisiones desde el emisor. Estos servidores se pueden utilizar para el procesamiento de los mensajes de control, y para las recuperaciones locales, mejorando la utilización del ancho de banda y la latencia. También permiten transformaciones entre diferentes esquemas de control de error.

  Generalmente, estos servidores no sólo se encargan de las retransmisiones locales, sino que llevan a cabo funciones adicionales como el control de realimentación. Por ejemplo, el protocolo SRM [45] lleva a cabo recuperaciones distribuidas, permitiendo que los receptores puedan retransmitir paquetes. Las peticiones son notificadas mediante confirmaciones negativas en multidifusión. Cualquiera de los receptores de esta solicitud que hayan recibido el paquete requerido podrá contestar retransmitiendo dicho paquete.

Los servidores de recuperación locales son nodos de la red cuya función consiste en mantener copias locales de los paquetes y retransmitirlos en caso de que se solicite la recuperación del mismo.

El empleo de servidores de recuperación local, cuya distancia y retardo hasta el receptor es mucho menor que

desde el origen del flujo, es la base de protocolos de transmisión en multidifusión fiable como SRM ([45]) o multidifusión robusta como STORM ([42]). La posibilidad de implementar servidores de este tipo en encaminadores gracias a las redes activas hace viable la utilización de técnicas de retransmisión de paquetes para la recuperación de pérdidas en flujos multimedia, debido a que el retardo entre servidor de retransmisiones y solicitante puede ser tolerable ([46]).

2.5.3 *Técnicas de reparación basadas en la regeneración de la señal*

Estas técnicas intentan reemplazar las tramas perdidas por otras similares a la original. Se suelen ejecutar en el receptor, como técnicas de ocultación de errores (*Packet Loss Concealment*, PLC [13]). Su efectividad depende de la duración de la pérdida. Las pérdidas de entre $4-40ms$ son las que se pueden beneficiar más de estas técnicas, ya que las señales de audio, y especialmente las de voz, poseen gran correlación temporal.

Estos esquemas son efectivos cuando el porcentaje de error es inferior al 15%, y las pérdidas aparecen aisladamente. Estos esquemas no excluyen el uso de otros mecanismos de reparación. Los métodos básicos de regeneración son los de *ensamblado*, *inserción*, *interpolación* y *emparejamiento de modelos*:

Ensamblado

El fragmento de señal correspondiente a la pérdida no se reemplaza con ninguna muestra. Es decir, se ensamblan las tramas anterior y posterior directamente. Esto produce una discontinuidad en la temporización de la señal recibida. Su aplicación afecta al uso del búfer de reproducción del receptor, ya que se extraen tramas sin seguir la cadencia original de la reproducción. Puede funcionar para pérdidas de $4-16ms$ y porcentajes de pérdidas menores al 3%, aunque

se ha comprobado que los resultados son deficientes [13]. Por ello, no se aconseja su utilización.

Basados en inserción

Las pérdidas se reparan mediante tramas de relleno que se insertan en el lugar correspondiente a las tramas perdidas. De esta forma, la señal obtenida tiene la misma duración que la original. Dependiendo del contenido de la trama insertada, se distinguen distintos tipos de técnicas:

- **Sustitución por silencio.** En este caso se inserta un segmento de silencio de igual duración que la trama perdida. Se puede aplicar en pérdidas de menos de $4ms$ y para porcentajes de pérdidas inferiores al 2%. La calidad resultante se degrada rápidamente conforme crece la duración de la trama, que suele ser generamente de $20-40ms$. Dada la simplicidad de su implementación, es una técnica muy utilizada ([13],[24]).

- **Sustitución por ruido.** Se ha comprobado que la sustitución de la trama perdida por ruido de fondo ofrece mejores resultados de inteligibilidad y calidad de la señal que el anterior esquema. Estudios sobre la percepción humana muestran que el cerebro humano repara de forma inconsciente los segmentos de habla perdidos en el caso de que se sustituyan por ruido ([24]).

- **Repetición.** En este esquema las tramas perdidas son reemplazadas por copias de la última trama recibida previa a las perdidas. La complejidad de este esquema es muy bajo, y su efectividad es razonablemente buena. La calidad resultante mejora los resultados de los anteriores esquemas, y su sencillez y bajo coste computacional lo hacen uno de los esquemas presentes en la mayoría de las aplicaciones de transmisión de voz sobre IP.

  Como mejora del esquema, se puede aplicar un desvanecimiento progresivo de la señal reconstruida, como ocurre en GSM.

- **Sustitución de la forma de onda.** En este esquema se utilizan las muestras de audio previas, y opcionalmente posteriores, para hallar una señal adecuada y cubrir la ausencia provocada por la pérdida. Se utilizan plantillas para encontrar los patrones de pitch adecuados a ambos lados de la pérdida.

- **Replicación del *pitch* de la forma de onda.** Se trata de un refinamiento de la técnica anterior. Consiste en realizar la detección de la frecuencia de *pitch* a ambos lados de la pérdida. Las pérdidas que correspondan a segmentos sordos se reparan mediante repetición de tramas anteriores, y si corresponden a segmentos sonoros, entonces se repite una señal de voz con el periodo de *pitch* apropiado ([47]).

Basados en interpolación

Este grupo de técnicas consisten en interpolar la tramas perdidas a partir de las adyacentes. La ventaja que presenta esta técnica es que tiene en cuenta las características cambiantes de la señal. Algunas de estas técnicas aprovechan el conocimiento sobre el algoritmo de compresión empleado, para derivar los parámetros del codificador. De esta forma, el fragmento de audio perdido puede ser sintetizado de nuevo.

Estos esquemas son más efectivos, pero más complejos, que los basados en inserción, requiriendo mayor potencia de cálculo.

Se han identificado los siguientes esquemas de regeneración por interpolación:

- **Empaquetado adaptativo para la regeneración.** Esta técnica, denominada AP/C (Adaptive Packetization / Concealment, [48], [31]), explota las propiedades de la voz estableciendo el tamaño de los paquetes generados para que el receptor obtenga una señal de calidad tras el proceso de regeneración.

La tasa de pérdidas a ráfagas es mayor si se mide en los extremos, que medida en los nodos intermedios. Por ello, AP/C es más efectivo si se lleva a cabo en nodos intermedios.
Los experimentos realizados en [32] muestran cómo para altas probabilidades de error, la mejora introducida por el empleo del esquema en nodos intermedios frente a su empleo extremo a extremo es significativo.

- **Reconstrución basada en transformaciones.** Este esquema utiliza el mecanismo de entremezclado a nivel de muestras para hacer frente a las pérdidas que se experimenten en la red ([49]). Se utiliza para codificadores que trabajan muestra a muestra. En caso de pérdidas, en el receptor se lleva a cabo la interpolación de las muestras perdidas mediante un promediado de las tramas adyacentes.

- **Escalado en el dominio del tiempo.** Este esquema de regeneración consiste en extender la señal de audio desde ambos lados de la pérdida, cubriendo así el espacio correspondiente a las tramas perdidas ([50]). Esta aproximación es computacionalmente compleja, pero obtiene mejores resultados que las técnicas de sustitución de la forma de onda y de replicación del periodo fundamental de la forma de onda.

- **Interpolación del estado del codificador.** Para codificadores basados en predicción lineal, el decodificador puede interpolar entre estados. Por ejemplo, el propio vocóder G.723.1 de la *ITU* realiza interpolación a ambos lados de pérdidas cortas los coeficientes del predictor lineal y utiliza, según se trate de un segmento de voz sonoro o sordo, la misma excitación periódica de la trama anterior, o la ganancia de un generador de números aleatorios.

La principal desventaja de este tipo de codificadores es que requieren una capacidad de cálculo relativamente elevada.

Recuperación basada en modelos

En este esquema, los segmentos de voz anterior y posterior a la pérdida se utilizan para ajustar un modelo que se utiliza para sintetizar la señal de voz correspondiente a la trama perdida.

En [51] se propone una reparación para la codificación ley μ entremezclada de voz consistente en la combinación de los resultados del análisis de autorregresión del último conjunto de muestras con una estimación de la excitación para el periodo de pérdida. Este esquema funciona correctamente debido a que los bloques entremezclados son de $8-16ms$, con lo que se asegura que el bloque de una trama anterior contiene características relevantes para el bloque siguiente.

2.5.4 *Técnicas de protección de flujos*

Otro enfoque propuesto para reducir el número de pérdidas de paquetes, lo constituyen las estrategias de protección del flujo. Estas técnicas preventivas ofrecen robustez a los paquetes del flujo mediante la adición de información redundante o la redistribución de las muestras de la señal. De esta forma se minimiza el impacto de la presencia de pérdidas sobre la señal reconstruida.

A continuación se describen las diferentes técnicas de protección identificadas.

Entremezclado

La idea principal de esta técnica se resume en obtener un flujo en el que la pérdida de paquetes consecutivos no implique la pérdida de demasiadas muestras consecutivas de la señal. Recuérdese que la reparación de pérdidas aisladas

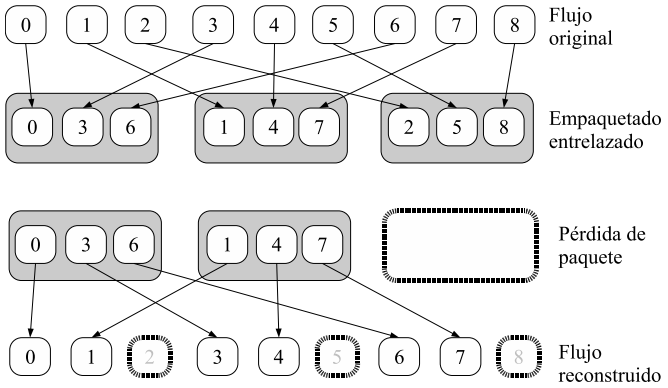


Figura 1: Empaquetado con entremezclado de tramas.

puede realizarse con un porcentaje de éxito muy alto simplemente con técnicas de regeneración. Sin embargo, las ráfagas de pérdidas degradan la calidad de la señal muy rápidamente.

El procedimiento general consiste en reordenar las tramas y empaquetarlas de forma que las pérdidas en caso de ráfagas estén dispersas, obteniendo una distorsión menos severa e incluso reparable en el receptor.

El entremezclado puede llevarse a cabo a distintos niveles:

- **entremezclado a nivel de muestras.** Previo a la codificación, las muestras son desordenadas teniendo en cuenta las características de pérdidas de la red. En el receptor se reordenarán tras la decodificación. Como resultado, las pérdidas se aislarán en la señal reconstruida, de forma que el impacto perceptual será menor ([52],[53]).

- **entremezclado a nivel de tramas.** Se altera el orden en el que se envían las tramas, de forma que las ráfagas de pérdidas afecten a tramas no adyacentes.

Como ventajas de este tipo de estrategias cabe destacar la ausencia de consumo de ancho de banda adicional. Pero la reordenación de las tramas o las muestras implica que en el receptor se experimenta un mayor retardo para que

se pueda obtener la señal reconstruida. Por ello no es adecuado para aplicaciones interactivas si se necesita hacer frente a ráfagas largas.

Adición de redundancia

Esta aproximación consiste en introducir algún tipo de redundancia en el flujo para que la información original enviada (o una versión de inferior calidad) pueda ser recuperada en caso de pérdidas.

La redundancia que se puede incorporar a un flujo no sólo consiste en añadir información de respaldo, sino que también puede consistir, por ejemplo, en la utilización de rutas alternativas de respaldo.

- **Diversidad de rutas** ([54], [55], [56]). En este esquema, distintas representaciones de la señal de voz se envían por distintas rutas entre el emisor y el receptor.

  La recepción del conjunto completo de representaciones enviadas por las distintas rutas permite reconstruir la señal con la máxima calidad.

- **Corrección de errores hacia adelante (FEC).** Esta técnica consiste en añadir al flujo información redundante para así reparar las pérdidas que puedan producirse. La cantidad de información de respaldo que se genera a partir del conjunto de datos originales depende de la probabilidad de pérdidas que caracterice a la ruta utilizada.

  Se puede distinguir entre dos tipos de FEC, según el tipo de redundancia que se genera. Si la información adicional consiste en versiones de diferente calidad de la trama a proteger, se denomina *FEC específico*. En el caso de añadir información generada a partir de la secuencia de bits que representa a los datos, se habla de *FEC independiente del contenido*.

- *FEC específico*:

  En paquetes posteriores al envío de la trama original, se incorpora información relativa a dicha trama o *unidad de datos*. La codificación de la primera copia de la trama se denomina *codificación primaria*, mientras que la codificación de la información redundante se denomina *codificación secundaria*. La *codificación secundaria* puede coincidir con la de la trama original, en el caso de codificadores con suficiente calidad y baja tasa de bits (Ej.: G.723.1, con una tasa de bits de $5.3 - 6.3\text{kbps}$), aunque generalmente serán de menor tasa de bits y calidad (Ej.: codificación LPC, $2.4 - 5.6\text{kbps}$).
  Así, en caso de perderse el paquete correspondiente a la codificación primaria de la trama, existe la posibilidad de recibir alguna de las descripciones de la misma en paquetes posteriores. Las codificaciones secundarias del paquete n se encapsularán en los paquetes $n + k, k \geqslant 1$, para hacer frente a las ráfagas de tamaño k. Sin embargo, a mayor valor de k, mayor es el retraso que se introduce para la reparación de la pérdida.

  Existen numerosas propuestas sobre el tipo de codificaciones secundarias a utilizar. El tipo más extendido es el de codificadores de baja tasa de bits (LPC, a $2.4 - 5.6\text{kb/s}$ o GSM a 13.2kb/s), ([24], [57]). También se ha propuesto el uso de codificadores jerárquicos para obtener flujos de respaldo de menor tasa de bits ([58]). El codificador de la fuente se encarga de producir todas las subcapas.

  Como ventaja de la utilización de este tipo de FEC resalta el bajo retardo que requiere para la recuperación, especialmente comparado con las técnicas basadas en retransmisión, con lo que es adecuado para aplicaciones interactivas. Si el retardo extremo a extremo permitido lo admite, la

dispersión de la información redundante puede permitir mayor robustez frente a ráfagas.
Además, la adición de redundancia puede ser adaptable, según las condiciones de pérdidas de la red, por lo que la cantidad de redundancia puede regularse.

- *FEC independiente del contenido*:

  La técnica de FEC independiente del medio añade paquetes de reparación redundantes generados a partir de la secuencia de bits de los paquetes originales. Mediante estos paquetes adicionales se puede recuperar los datos de algunos paquetes eliminados, según las características del tipo de código utilizado para generar la redundancia.

  Existe un amplio abanico de códigos corrección de eliminaciones, de los que destacan por su utilización los *Códigos Reed-Solomon*, y los códigos Tornado [59]. Otros códigos que han sido empleados para generar FEC independiente del medio son los *códigos Turbo* [60], *códigos de Reed-Muller* [61], *Códigos BCH*, *códigos expandibles*, y *códigos LDPC* (*Low Density Parity Check*) [62].

  Este tipo de FEC se ha utilizado con éxito en aplicaciones de transmisión multimedia ([63], [64], [65]).

El mecanismo de corrección de errores hacia adelante se suele utilizar combinado con otras técnicas de reparación, como la retransmisión selectiva, lo que se denomina *ARQ híbrido*, y sobre todo se acompaña de mecanismos de conformación del flujo adaptables de la tasa de bits.

Control híbrido de errores (ARQ/FEC)

Una de las dificultades principales en la utilización de FEC (*Forward Error Correction*) es la elección de la cantidad adecuada de redundancia a incluir para hacer frente a las condiciones de la red. Uno de los enfoques para abordar este problema consiste en combinar ARQ y FEC:

- **ARQ híbrido de tipo I.** En este caso, el paquete redundante con paridad se envía inmediatamente, y las pérdidas que se detecten serán recuperadas utilizando ARQ. De esta forma se consigue que en un gran número de receptores se recuperan las pérdidas sin utilizar retransmisiones, como es deseable en transmisiones audiovisuales en tiempo real.

- **ARQ híbrido de tipo II.** Se envían los bloques de paridad sólo cuando se requiere una retransmisión. De esta forma se ahorra ancho de banda, y en casos de gran número de receptores con pérdidas no correlacionadas, se aumenta la media de reparaciones que puede llevar a cabo un paquete. Con un solo paquete de paridad se pueden reparar las diferentes pérdidas de los distintos receptores, en vez de tener que reenviar los distintos paquetes a cada uno de ellos.

Existen propuestas para realizar labores de adición de redundancia en encaminadores intermedios, según las condiciones de pérdidas de los enlaces siguientes. El objetivo de esta adaptación es la de añadir en cada tramo de la ruta del emisor al receptor la cantidad óptima de FEC, ahorrando así ancho de banda y adaptándose a las condiciones cambiantes de la red. Por ejemplo, en los servicios activos de codificación de paridad *APES* (*Active Parity Encoding Services*, [66]), se combina el uso de almacenamiento y recuperación local con técnicas de adición de redundancia FEC. El uso de los protocolos APES reduce los requerimientos de búferes y de ancho de de banda en los servidores de reparación en regiones con alta probabilidad de pérdidas o gran número de receptores. Los protocolos APES envían reparaciones basadas en FEC en lugar de retransmitir los paquetes originales.

En [33] se describe un ejemplo de esquema de inclusión de FEC salto a salto en lugar de extremo a extremo. Cada encaminador añade sólo la información redundante necesaria para hacer frente a las pérdidas experimentadas en el enlace siguiente.

Técnicas basadas en fuente y canal combinados

Este tipo de técnicas requiere que la red ofrezca serviciosde priorización de paquetes. La fuente clasifica la información del flujo por su importancia para la reconstrucción de la señal, la prioriza y la envía utilizando dicho servicio diferenciado. Aunque su despliegue en Internet no es factible, es posible implementarlo parcialmente usando redes superpuestas.

Para redes que cumplan el requisito anteriormente descrito, han sido propuestas algunas estrategias:

- **Priorización del código.** Las palabras código se segmentan en bits de mayor y menor prioridad, y se encapsulan en diferentes paquetes [24]. De esta forma, se prima la entrega de paquetes cuyo contenido corresponda a la información más significativa de la codificación.

- **Priorización de paquetes.** Se priorizan los paquetes que portan tramas más importantes (o más sensibles a las pérdidas) para la inteligibilidad perceptual y recuperación de la señal.

2.6 MEDIDAS DE CALIDAD DE VOZ

El criterio básico para evaluar la calidad de un flujo de voz es la percepción que tiene un usuario de la calidad que recibe. La *calidad de experiencia de usuario* (*QoE*) está influida fundamentalmente por las degradaciones que introducen tanto la distorsión introducida por la codificación de la señal original, los retardos de transmisión, el retardo variable (jitter), etc. Sin embargo, la correlación entre estas variables, que pueden ser medidas objetivamente, y la calidad percibida por el usuario es muy compleja, por lo que no se pueden utilizar dichas variables directamente para evaluar la calidad de una transmisión de voz sobre IP.

Para medir la calidad subjetiva se realizan tests de evaluación. Estos tests dan como resultado un valor subjetivo, la *nota media de opinión* (*Mean Opinion Score*, *MOS*), obtenida

como la media de los resultados individuales del test sobre un conjunto numeroso de sujetos. Este procedimiento está normalizado en la recomendación P.800 [67], donde se especifica la metodología a seguir para llevar a cabo los tests.

La evaluación de los participantes del test puede darse en términos absolutos, mediante los *índices de categorías absolutas (ACR)*, o mediante la valoración del sistema en relación a otro de referencia en la escala de *determinación de índices por categoría de degradación* (DCR).

En la escala ACR se definen 5 puntuaciones de calidad: Excelente (5), Buena (4), Regular (3), Mediocre (2), Mala (1). El valor MOS se obtiene como la media de las puntuaciones de los oyentes. En la escala DCR se definen los siguientes niveles de calidad: Degradación inaudible (5), audible pero no molesta (4), ligeramente molesta (3), molesta (2), y muy molesta (1). La media de las valoraciones dan como resultado la *nota media de opinión sobre las degradaciones* o DMOS.

A pesar de que la valoración subjetiva de la calidad es un procedimiento extendido, también es el más costoso, pues cada vez que se introduzca una modificación del sistema evaluado hay que repetir los tests de opinión, y puede que poco reproducible. Por esta razón se diseñan métodos para mapear las características de la red y de los terminales utilizados a la calidad subjetiva experimentada por los usuarios. Este precedimiento se denomina *evaluación objetiva de la calidad*.

Las metodologías de evaluación objetiva de la calidad pueden categorizarse según varios criterios. En [68] se ofrece una clasificación que considera cuál es la entrada requerida para producir el valor MOS correspondiente. Así, los métodos que se basan en obtener el valor MOS a partir de los parámetros de calidad de la red y de los terminales utilizados los denomina *modelos de opinión*. Por otro lado, los métodos que requieren de las señales de voz como entrada para calcular el valor MOS se denominan *modelos objetivos del nivel de voz* (*speech-layer objective models*). Por último, los métodos que tienen como entrada las características de los datagramas IP, son clasificados como *modelos objetivos del nivel de paquete* (*packet-layer objective models*).

Por otro lado, en [69] se ofrece otra clasificación más acorde con los objetivos de esta tesis. El criterio de clasificación consiste en distinguir entre los métodos que requieren de la señal de referencia para calcular el valor MOS correspondiente (*métodos intrusivos*) o no (*métodos no intrusivos*). Mientras que los métodos intrusivos suelen ser más precisos, y normalmente no pueden ejecutarse en tiempo real, los no intrusivos son adecuados para la monitorización y la predicción de la calidad de la transmisión de voz, a partir de los parámetros de la red, de las características de los paquetes, o de la señal degradada final. Por lo general, los *modelos de opinión* y *modelos objetivos del nivel de paquete* suelen entrar en la categoría de métodos no intrusivos, mientras que los *modelos objetivos del nivel de voz* suelen ser métodos intrusivos.

En [70] se definen varias subcategorías para los *métodos intrusivos* y *no intrusivos*. A continuación se detalla dicha categorización, combinada con las de [68]:

- **Métodos intrusivos**
  - *Medidas en el dominio del tiempo*. Se basan en comparar la señal original con la señal degradada. Son fáciles de implementar, pero no son adecuados para evaluar códecs de baja tasa de bits. Ejemplos de este tipo de medidas lo constituyen la relación señal-ruido (SNR, *Signal to Noise Ratio*) y la SNR segmentada (SNRSeg, *Segmental Signal to Noise Ratio*).
  - *Medidas del dominio espectral*. Esta medida, basada en comparar segmentos de las señales original y degradada en el dominio espectral, es más fiable y no depende de tanto de la alineación temporal de ambas señales. Como ejemplo de medidas de este tipo destacan la *codificación lineal predictiva* (LPC, *Linear Predictive Coding*) y la *medida de distancia Cepstral* (CD, *Cepstral Distance*, [71]).
  - *Medidas del dominio perceptual*. Se basan en un modelo del sistema humano de percepción del sonido, y se caracterizan por su gran precisión.

De entre estos métodos destacan: MNB (*Measuring Normalizing Block*), PSQM (*Perceptal Speech Quality Measure* [72]), y PSQM+ ; PAMS (*Perceptual Analysis Measurement System*, [73]); EMBSD (*Enhanced Modified Bark Spectral Distortion*, [74]); por último, sobre la base de PAMS y PSQM+, la ITU-T estandariza PESQ (*Perceptual Evaluation of Speech Quality*, Rec. P.862 [75]).

- **Métodos no intrusivos**
  - *Modelos de opinión*. El INMD (*In Service Non-intrusive Measurement Device*,[76]) fue estandarizado por la ITU en 2000 como recomendación P.562. Aunque tiene parámetros de entrada similares a los del *e-model*, sólo se han tabulado los parámetros referentes a impedimentos individuales. El CCI (Call Clarity Index, [77]), desarrollado por la British Telecom, se integra con INMD, y mediante un modelo de percepción humana, obtiene el *índice de claridad* de una llamada individual. Por su parte, la France Telecom R&D desarrolló su propia medida PSOM (*Perceptual Single ended Objective Measure*). La NIQA (*Non-intrusive Quality Assessment*) se definió como extensión de CCI para considerar todos los tipos de distorsión. De entre todas las medidas comentadas, destaca el e-model, estandarizado como recomendación G.107 [78].
  - *Modelos objetivos del nivel de la voz*. Una de las medidas de este tipo más representativas es la recomendación P.563 de la ITU-T [79]. Este método intenta estimar la calidad analizando la señal de voz recibida, sin requerir la señal original.
  - *Modelos objetivos del nivel de paquetes*. La ITU-T intenta estandarizar con el nombre de P.VTQ [80], una medida de calidad basada sólo en paquetes IP, que no necesita procesar el contenido del paquete. P.VTQ estima la calidad subjetiva mediante la información de fluctuación de retardos,

pérdidas, etc. disponibles en de las cabeceras RTP y RTCP. PsyVoIP fue uno de los algoritmos candidatos a utilizar en P.VTQ. PsyVoIP es similar a INMD, con la salvedad de que sí está diseñado para evaluar VoIP. El otro algoritmo candidato para P.TVQ era VQmon [81]. Finalmente, las recomendación ITU-T P.564 [82] también proporciona un método para medir flujos VoIP con codificadores para banda estrecha, a partir de los paquetes RTP y RTCP-XR recibidos.

Fuera de la clasificación anterior, caben destacar las propuestas de medidas de calidad alternativas, basadas en el nivel de inteligibilidad del flujo de voz como medida de la calidad subjetiva del servicio, obtenidas mediante máquinas de reconocimiento automático del habla [83], [84].

De entre las medidas más utilizadas en la bibliografía relacionada destacan PESQ como medida intrusiva y el e-model como medida no intrusiva.

2.6.1 *PESQ*

La recomendación de ITU-T P.862 (PESQ, [75]), al igual que PAMS, mide la distorsión de la señal obtenida y la señal de referencia. Como salida proporciona una nota MOS en una escala propia. En la recomendación P.862.1 se define la corresponencia entre la salida de PESQ y el MOS estándar.

A pesar de no poder aplicar PESQ en línea, ya que requiere la señal original para realizar la evaluación, PESQ se utiliza frecuentemente para sustituir los test subjetivos MOS para validar estimaciones de calidad de otros modelos, debido a la posibilidad de automatizar estos tests [85].

Sin embargo, algunos estudios han revelado que PESQ no es adecuado para VoIP, pues no evalúa correctamente en escenarios con pérdidas consecutivas ([86]).

2.6.2 *El modelo e-model para la estimación de la medida de calidad de la voz*

Aunque fue inicialmente concebida como una herramienta de planificación, el denominado *e-model*, especificado en la recomendación G.107 de la ITU-T [78], ha sido ampliamente utilizada como una función de medida de la calidad perceptual de voz en los esquemas de transmisión.

Dado un escenario de VoIP, básicamente compuesto por un emisor, el codificador, el decodificador, la red y el receptor, el emodel proporciona una estimación numérica $R \in [0, 100]$ de la calidad de la transmisión de voz. Los parámetros de la función analítica consiste en un conjunto de impedimentos de la transmisión, tales como el eco, el ruido de fondo, las pérdidas de los paquetes, los retardos, la distorsión de la codificación, etc. Todos ellos se modelan analíticamente mediante la definición de 4 factores (I_s,I_d,I_e,A).

El *factor de simultaneidad* I_s está relacionada con la SNR del canal de transmisión. I_d es el *factor de retardo*, y modela los impedimentos relacionados con el retardo. El *factor de equipamiento* I_e incluye las distorsiones introducidas por el algoritmo de codificación/descodificación y por las pérdidas de paquetes. Finalmente, el *factor de expectación* A representa un factor de corrección que modela las deficiencias que los usuarios están dispuestos a tolerar a cambio de las ventajas de usar el servicio (por ejemplo, la movilidad). En resumen, dado un escenario para una transmisión de voz, el modelo *emodel* define el factor R como sigue:

$$R = 100 - I_s - I_d - I_e + A \tag{2.1}$$

En la definición original del e-model no se proporcionaban los los factores sintonizados para las transmisiones de VoIP. Sin embargo, algunos trabajos han calibrado los valores correspondientes para este tipo de comunicaciones ([87], [88]). En concreto, en [88] se presenta una simplificación del modelo, asumiendo ciertos valores por defecto, que permite monitorizar en línea flujos de VoIP. En concreto,

esta simplificación proporciona la siguiente expresión para calcular R:

$$R \approx 94.2 - 0.024 \cdot d + 0.11 \cdot (d - 177.3) \cdot H(d - 177.3) - I_e \quad (2.2)$$

donde d es el retardo extremo a extremo expresado en milisegundos. El cálculo de I_e depende, entre otros, del códec utilizado y el patrón de pérdidas. Si suponemos un códec PCM G.711 con ocultación de errores y un patrón de pérdidas aleatorio, la expresión para I_e puede estimarse como:

$$I_e(G.711, random) \approx 30 \cdot \ln(1 + 15 \cdot e) \quad (2.3)$$

En la expresión 2.3, e se define como la probabilidad de pérdidas global. Se define la función $H(x)$ como:

$$H(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geqslant 0 \end{cases} \quad (2.4)$$

Finalmente, la relación entre el factor R (expresión 2.2) y la puntuación MOS puede expresarse como se muestra en la ecuación 2.5:

$$MOS = 1 + 0.035 \cdot R + R \cdot (R - 60) \cdot (100 - R) \cdot 7 \cdot 10^{-6} \quad (2.5)$$

Deficiencias del modelo

Cabe apuntar que, aunque el modelo *emodel* estima la calidad de la voz, no modela completamente todos los factores psicológicos involucrados. De hecho, el *emodel* está actualmente bajo la revisión del Grupo de Estudio 12 de la ITU-T. No obstante, el modelo *emodel* fue inicialmente diseñado sólo como herramienta de planificación.

Otro de los problemas del modelo consiste en que sólo ha sido definido para algunos codificadores estándares. Esto limita su utilización, por lo que se depende de que el órgano de estandarización provea los parámetros correspondientes a cada nuevo codificador, o calibrar los parámetros mediante costosos tests MOS sobre grupos de oyentes.

Por otro lado, aunque el modelo considera el impacto de los retardos de los paquetes en la calidad percibida

mediante el factor I_e, definido para un conjunto limitado de condiciones de pérdidas, de esquemas de ocultación de errores y de distribución de tramas por paquete, el efecto de la distribución de pérdidas que utiliza es demasiado simple, y no refleja adecuadamente las consecuencias de las pérdidas consecutivas. No obstante, recientemente se han introducido modificaciones en la recomendación para solucionar algunas de estas deficiencias [89].

De cualquier manera, dada su simplicidad, podemos asumir alguna inexactitud a cambio de introducir un criterio de calidad perceptual en el desarrollo de los servicios de VoIP proporcionados.

Extensiones del modelo

Para paliar las deficiencias del *emodel*, trabajos posteriores han propuesto extensiones al modelo para mejorar su precisión y su aplicabilidad.

En [90] se proporciona un nuevo factor que sustituye a la probabilidad de pérdidas p en el cálculo del valor R. Este nuevo factor considera el nivel de agrupación de las pérdidas haciendo que el valor de la variable p de la expresión 2.1 dependa de una medida de agrupación de las pérdidas (*Packet Loss Burstiness Measure*).

En [91] proporcionan un nuevo factor I_j que considera el efecto del jitter, y ajustan I_e para cubrir un mayor rango de condiciones de pérdidas, sirviéndose como datos de referencia el valor generado por PAMS. Sin embargo, dichos parámetros se definen sólo para algunos codificadores y técnicas de ocultación de errores.

Adicionalmente, en [92] se calibran los parámetros del modelo para considerar además distintos tamaños de tramas.

Por último, [93] se propone una metodologá general para calibrar los parámetros del modelo para caulquier codificador mediante PESQ.

2.7 CONCLUSIONES

Las deficiencias del servicio de *mejor esfuerzo* ofrecido por IP, el cual no garantiza ningún tipo de mecanismo para asegurar la calidad de servicio requerida por los flujos continuos con restricciones de tiempo, hacen necesario el diseño de técnicas aplicadas a nivel de aplicación.

La transmisión de voz sobre redes de conmutación de paquetes difiere en muchos sentidos de las transmisiones de otro tipo de datos. Por ejemplo, en este tipo de transmisiones es posible tolerar ciertos niveles de pérdidas sin degradar excesivamente la calidad de la señal recibida. Además, la naturaleza perceptual de la información transmitida confiere a este tipo de flujos la capacidad de emplear diferentes representaciones de la misma información original y permite que se puedan llevar a cabo mecanismos de recuperación. Por otro lado, las restricciones impuestas en el retardo máximo tolerable para la entrega de paquetes restringen el ámbito de aplicación de muchos de los mecanismos de reparación utilizados en aplicaciones de otra índole.

Frente a soluciones extreo a extremo, el empleo de estas técnicas en los nodos intermedios puede reportar mejoras substanciales en términos de tiempos de respuesta y escalabilidad.

En este capítulo se han descrito las principales técnicas de control de errores para flujos de audio propuestos en los trabajos relacionados sobre la transmisión de audio. Estas técnicas consisten tanto en prevenir la aparición de pérdidas de tramas durante la reconstrucción de la señal como en introducir mecanismos de recuperación una vez que se haya producido dicha pérdida.

Los diferentes tipos de técnicas de reparación ofrecen un amplio abanico de posibilidades, si bien se encuentran limitadas por los requerimientos de tiempo de los flujos de aplicaciones interactivas.

Parte de las distintas alternativas para reparar pérdidas de paquetes, tradicionalmente llevadas a cabo extremo a extremo, pueden modificarse y adaptarse para funcionar en nodos intermedios.

Por último, de las medidas de calidad subjetiva de voz disponibles actualmente, se han identificando aquéllas que pueden utilizarse para evaluar la calidad de los flujos de voz en línea. Tales medidas se obtienen a partir de parámetros de los paquetes IP recibidos, o incluso de la señal de voz resultante en el receptor. La utilización de medidas de este tipo permite que los mecanismos de reparación puedan adaptarse en tiempo de ejecución, guiados por la estimación de la calidad subjetiva que el usuario final percibe.

3

I-MODEL: MODELO DE INTELIGIBILIDAD PARA FLUJOS DE VOZ SOBRE IP

Para ajustar en línea mecanismos adaptables de reparación de flujos de voz es necesario contar con funciones de utilidad que consideren los aspectos que afectan a la calidad subjetiva de los usuarios. Como se describió en la sección 2.6, aunque en la actualidad existen varias medidas de calidad perceptual de la voz (p.e. [78],[75]), sólo unas pocas permiten mapear parámetros objetivos de red (probabilidad de pérdidas, retardos, etc.) a calidad subjetiva de usuario en tiempo real. Sin embargo, generalmente estas medidas no contemplan adecuadamente el efecto de las ráfagas de pérdidas sobre la calidad final del flujo.

Adicionalmente, nótese que para que las alicaciones de transmisión de voz en tiempo real sean de utilidad es prioritario proporcionar un nivel de inteligibilidad mínimo, aunque la fidelidad con la señal no sea maximizada.

En este capítulo se propone un modelo y una metodología para obtener una medida de calidad de un flujo VoIP que pueda utilizarse en línea. La medida propuesta estima la inteligibilidad esperada a partir de una representación sencilla de la distribución de las pérdidas experimentadas en la red.

3.1 INTRODUCCIÓN

Tradicionalmente, la calidad de la transmisión de voz se ha evaluado llevando a cabo costosos test MOS (Mean Opinion Score, [67]) sobre un conjunto numeroso de oyentes.

En muchos casos, no es viable llevar a cabo estos test en términos de tiempo y esfuerzo, especialmente debido a la necesidad de repetir cada test cada vez que cambian las condiciones experimentales.

A fin de evaluar la calidad percibida por un usuario de forma automática, se han propuesto distintos algoritmos

tales como *PESQ* (recomendación P.862 de la ITU-T [75]) o el *e-model* (recomendación G.107 de la ITU-T [78]). Algunos trabajos han expuesto deficiencias en los anteriores modelos, a la hora de medir la calidad de los flujos VoIP. En particular, se ha mostrado que PESQ no evalúa correctamente la degradación que producen las ráfagas de pérdidas sobre los flujos de VoIP ([86]). Por otro lado, Aunque en las recientes revisiones de la recomendación G.107 se ha propuesto una extensión del *e-model* para contemplar escenarios con ráfagas de pérdidas, las definiciones iniciales del *e-model* distinguen escasos escenarios de pérdidas, lo cual no es suficiente para abarcar todas las condiciones de red ([88]).

Como se ha demostrado en [84], los sistemas automáticos de reconocimiento del habla (ASR) ofrecen una medida alternativa efectiva para poder evaluar la calidad perceptual de un flujo de voz. Además, como se mostrará en secciones posteriores, la medida obtenida es sensible tanto al porcentaje de paquetes perdidos como al agrupamiento pérdidas consecutivas.

En este trabajo adoptaremos un sistema de reconocimiento automático de la voz para proporcionar un índice de calidad de las señales de voz resultantes. Dado que la salida del sistema de reconocimiento automático está altamente correlacionada con la calidad que perciben los usuarios, la utilización de esta herramienta está justificada no sólo por su bajo coste comparado con el de los tests MOS sobre grupos de oyentes, sino también por su comportamiento determinista y reproducible.

El resto del capítulo se organiza como sigue: en la sección 3.2 se describe cómo los sitemas de reconocimiento automático del habla pueden utilizarse como medida de calidad de flujos VoIP. Una vez justificado su uso, en la sección 3.3 se propone un modelo que estima el funcionamiento de un reconocedor automático de referencia. A continuación, en la sección 3.4 se llevan a cabo varios experimentos para validar el uso del estimador propuesto. Finalmente, en la sección 3.5 se presenta una breve discusión sobre el uso del modelo propuesto, y finalmente en 3.6 se resumen las principales conclusiones extraídas.

3.2 EL RECONOCIMIENTO AUTOMÁTICO DEL HABLA COMO MEDIDA DE INTELIGIBILIDAD

La metodología de evaluación aplicada utiliza un sistema de reconocimiento de voz continua que mide las tasas de palabras y frases erróneas totales de un corpus de frases dado. Estas tasas de error están directamente relacionadas con la inteligibilidad final que un usuario percibiría ([94]). Según [84], este resultado, se puede establecer una correspondencia entre el valor de inteligibilidad y la calidad MOS percibida.

Ya que la tarea de reconocimiento automático del ASR utilizada en este estudio es diferente al presentado en [95], no es posible aplicar a nuestros resultados la misma función de correspondencia entre el índice de reconocimiento y calidad subjetiva derivada en dicha referencia. En su lugar, las mejoras de calidad se evaluarán comparando los rendimientos del reconocimiento automático de la voz para cada solución.

El rendimiento del reconocimiento de la voz se mide mediante la tasa de error de palabras (*Word Error-Rate*, WER) definida como:

$$WER = \frac{n_i + n_s + n_d}{n_t} \tag{3.1}$$

donde n_s es el número de palabras sustituidas por el reconocedor, n_i es el número de palabras insertadas, n_d es el número de palabras eliminadas, y n_t es el número total de palabras.

Alternativamente se utilizan también como índices de reconocimiento la precisión de palabras (*Word Accuracy*, W_{acc}), calculada como $(1 - WER) \cdot 100$, y el *porcentaje de oraciones correctas* (Corr), que corresponde al porcentaje de sentencias en las que no se ha insertado, sustituido o borrado ninguna palabra en el reconocimiento.

3.2.1 *Impacto de las ráfagas de pérdidas en la inteligibilidad*

Es bien conocido que las ráfagas de pérdidas degradan la calidad perceptual del usuario. Las pérdidas aisladas, sin

embargo, pueden ser reparadas fácilmente por medio de técnicas de ocultación de errores [96]. Ésta es la principal razón para utilizar esquemas de recuperación tales como el entremezclado, cuyo objetivo consiste en aislar pérdidas consecutivas, en lugar de recuperar los paquetes perdidos.

En trabajos previos se ha evidenciado la incidencia que tienen las ráfagas de pérdidas en la calidad que percibe el usuario. Por lo general, se trata de una relación no lineal, que depende de la probabilidad final de pérdidas, del tamaño medio de las ráfagas [4], y del codificador de voz utilizado [83].

En la figura 2 se presenta la degradación de W_{acc} que introducen distintas distribuciones de pérdidas para un flujo de voz codificado con G.711, encapsulado en paquetes de 20ms de audio, y que utiliza una técnica de ocultación de pérdidas basadas en repetición [96].

Dado que la codificación PCM (Pulse Code Modulation) no introduce distorsiones significativas para voz, y que las tramas codificadas son independientes entre sí, los valores mostrados ofrecen los resultados más optimistas que pueden obtenerse utilizando cualquier códec.

Los puntos representados en la superficie corresponden al valor W_{acc} obtenido tras la ejecución del proceso automático de reconocimiento. En concreto, en la figura se muestran los resultados para un conjunto de test al que se le aplicó, por cada prueba, un patrón con una probabilidad de pérdidas $p \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$, y una longitud de ráfaga $L_i \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 14, 17, 20, 23, 26, 29\}$ sobre un flujo VoIP.

En la figura 2 puede también observarse la dependencia entre la precisión de palabras W_{acc}, la tasa de error (p) y la longitud media de ráfagas (L_i). En concreto, puede contemplarse que el número de pérdidas consecutivas afecta notablemente a la degradación de la tasa de reconocimiento dado un p fijo. De hecho, la pendiente de la superficie de W_{acc} para un valor de L_i fijo frente a la probabilidad de pérdidas se hace más notable según crece la longitud de las ráfagas. Para un valor fijo de L_i, el índice W_{acc} disminuye linealmente con p. La pendiente de esta degradación es más acusada conforme aumenta el tamaño de ráfaga media L_i.

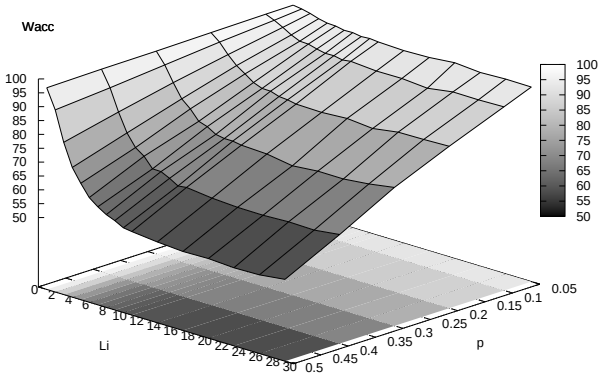


Figura 2: Resultados preliminares de inteligibilidad para la voz codificada con PCM y un modelo de error de Gilbert

Se puede observar además que la degradación del índice de inteligibilidad W_{acc} es mucho más acusada cuando el valor de ráfaga media es inferior a $L_i = 10$. A partir de este valor, las variaciones de la longitud media apenas introducen variaciones en el valor de W_{acc}.

Por tanto, del anterior análisis se puede extraer que las mejoras que mayor impacto tienen, consisten en reducir el tamaño de las ráfagas cuando $L_i \leqslant 10$. En los casos en que $L_i > 10$, tiene mayor influencia reducir la probabilidad de pérdidas p que reducir el tamaño medio L_i.

3.3 CONSTRUCCIÓN DEL MODELO DE INTELIGIBILIDAD

Una vez identificada la relación entre p, L_i y W_{acc}, es necesario modelar analíticamente dicha correspondencia de forma que no sea necesario realizar el procedimiento de reconocimiento automático. El principal objetivo de esta sección es ofrecer un modelo analítico que se ajuste al comportamiento del reconocedor. De esta manera, para un flujo de VoIP, dado un patrón de pérdidas, será posible predecir en tiempo real su impacto en términos de inteligibilidad.

Para muestro propósito es conveniente adoptar un modelo sencillo que pueda representar el comportamiento a

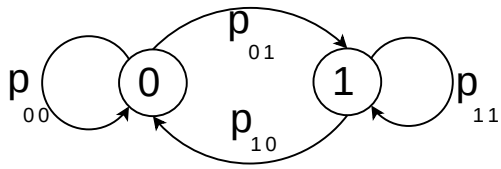


Figura 3: Modelo de Gilbert

ráfagas de la red. Para ello, se adopta un modelado del canal mediante el modelo de Gilbert.

3.3.1 *Modelado del canal mediante Gilbert*

El modelo de Gilbert [97] ha sido ampliamente empleado para representar las pérdidas de paquetes en ráfagas en distintos estudios. El modelo de Gilbert se puede representar mediante una cadena de Markov de dos estados, representando uno de ellos el estado libre de pérdidas, y el otro el estado de pérdidas a ráfagas. En la figura 3 se muestra el modelo de Gilbert de dos estados. Del estado libre de errores (etiquetado con 0) pasa al estado que representa el periodo de pérdidas (etiquetado con 1) con una probabilidad p_{01}. La transición inversa se da con una probabilidad p_{10}. Con probabilidad $p_{11} = 1 - p_{10}$ se mantiene en el estado de pérdidas, y con probabilidad $p_{00} = 1 - p_{01}$ en el estado libre de pérdidas.

Una caraterización más próxima a las medidas de pérdidas más utilizadas e intuitivas puede obtenerse mediante los parámetros p y L_i (probabilidad de pérdidas y tamaño medio de ráfagas, respectivamente) mediante las siguientes expresiones:

$$p = \frac{p_{01}}{p_{01} + p_{10}} \tag{3.2}$$

$$L_i = \frac{1}{p_{10}} \tag{3.3}$$

De esta forma se puede construir el modelo de Gilbert que se ajuste a unos parámetros de ráfagas medias y probabilidad de pérdidas dados.

3.3.2 *Descripción del ASR de referencia*

Para la tarea de reconocimiento de los experimentos realizados se ha utilizado la base de datos de *AURORA2*. Esta base de datos consiste en secuencias de dígitos conectados para interlocutores de habla inglesa-americana. Un extractor de características segmenta la señal de voz recibida en tramas solapadas de 25ms cada 10ms. Cada trama de voz representa un vector de características de dimensión 14 que contiene 13 coeficientes *Mel Frequency Cepstrum (MFCCs)* más el logaritmo de la energía *log-Energy*. Finalmente, los vectores de características se extienden con sus primeras y segundas derivadas.

El reconocedor de voz se basa en modelos ocultos de Markov (HMM). Emplea 11 modelos de palabra HMM continuos de 16 estados, (más silencio y pausa, que tienen 3 y 1 estado, respectivamente), con 3 gausianas por estado (excepto el silencio, con 6 gausianas por estado). Estos modelos HMM se entrenan con un conjunto de 8440 frases sin ruido, mientras que el test comprende 4004 frases libres de ruido.

Previamente al recuento de errores de sustituciones, eliminaciones e inserciones, se utiliza programación dinámica para alinear la frase reconocida con su transcripción correcta.

3.3.3 *Muestreo de las salidas del reconocedor*

Para poder modelar el comportamiento del reconocedor, se realizan un total de 3957 ejecuciones del reconocedor de referencia sobre un conjunto de trazas que representen las condiciones de pérdidas más significativas.

En concreto, se generan 2836 patrones de pérdidas mediante un modelo de Gilbert con parámetros dentro de los intervalos $p \in [0.01, 0.20]$ en pasos de 0.01, y $L_i \in [1, 10]$ en incrementos de 1. Para cada patrón, se ejecuta el reconocimiento sobre la traza obtenida. Para cada dupla (p, L_i) se efectúan una media de 11.04 ejecuciones (con

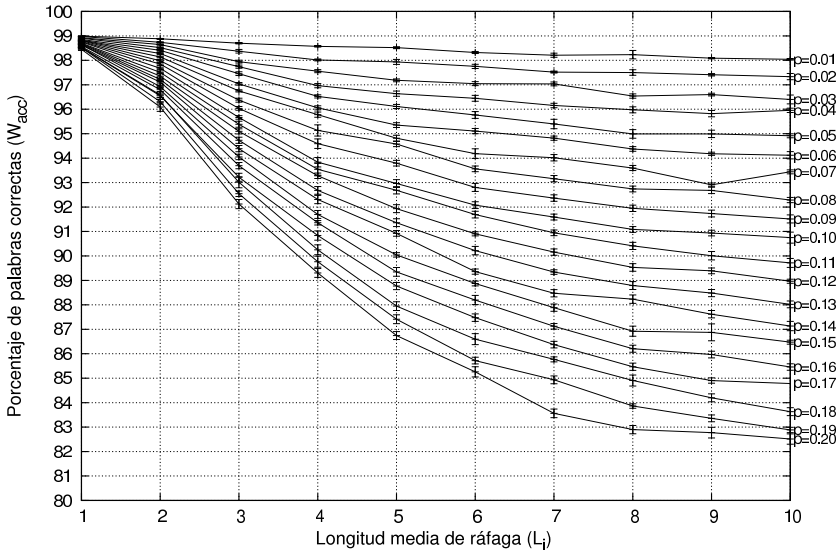


Figura 4: Medias para cada p desde p=0.01 hasta p=0.20 en incrementos de 0.01, respecto a L_i=1,2,3,4,5,6,7,8,9,10. Para cada valor, se muestra el intervalo de confianza del 95%.

un mínimo de 6 ejecuciones por dupla, y un máximo de 33). El número de repeticiones del reconocedor, determinará el intervalo de confianza del valor de W_{acc} obtenido. Adicionalmente se generan y evalúan mediante el reconocedor un total de 1121 trazas mediante varios modelos de Gilbert con valores de $p \in 0.20, 0.25$, y $L_i = 15, 20, 25, 30$.

Como resultado de las ejecuciones del reconocedor para los valores de interés de p y L_i se obtienen los valores mostrados en la figura 4.

Resumidamente, se vuelve a constatar las tendencias identificadas en los resultados preliminares, comprobando el efecto que tiene el tamaño medio de ráfagas en el nivel de inteligibilidad que se obtiene. Recuérdese que otras medidas tales como el *e-model* no tienen este comportamiento diferenciado, sólo atienden al valor de la probabilidad de pérdida p.

La media de intervalos de confianza del 95% es de 0.161424, con un valor máximo de 1.1172. Se observa que el intervalo de confianza crece con L_i. Para obtener estimaciones más precisas, se podría aumentar el número de muestras.

Sin embargo, los intervalos obtenidos son suficientemente estrechos para una estimación razonable.

3.3.4 *Función de utilidad*

A partir de los datos obtenidos mediante las ejecuciones anteriormente descritas, se obtiene una función que se ajusta al comportamiento descrito mediante los resultados de los tests de reconocimiento.

Como resultado, tras diversos tests de ajuste, la función con el menor error cuadrático medio corresponde a la expresión 3.4.

$$\mathtt{imodel}_{p,L_i} = 99.036 - p \cdot \left(\frac{217.908}{1 + e^{-\frac{L_i}{132.775}}} - 2.429 \right) \quad (3.4)$$

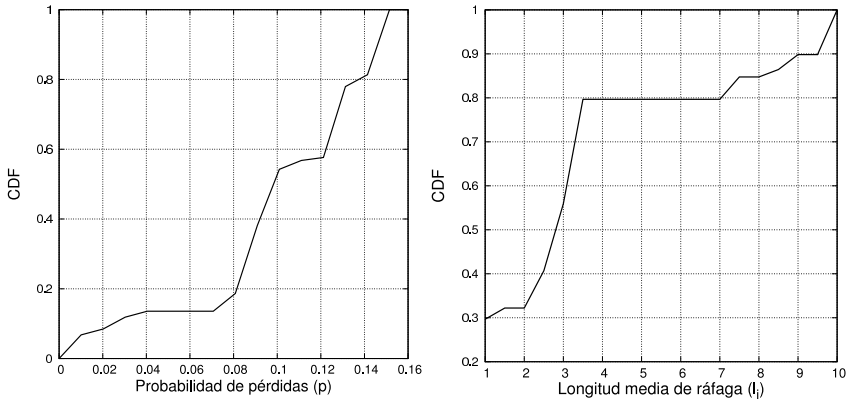
El error medio absoluto cometido sobre la estimación de la función $\mathtt{imodel}(p, L_i)$ sobre los valores utilizados para ajustar el modelo es igual a 0.34196 puntos en media, con una desviación estádar de 0.007938. Por lo tanto, el modelo se ajusta con suficiente precisión al comportamiento del reconocedor ante trazas generadas por modelos de Gilbert.

3.3.5 *Utilización del modelo sobre distribuciones de pérdidas arbitrarias*

El modelo generado se ha diseñado asumiendo que las pérdidas generadas siguen un patrón ajustado por un modelo de Gilbert. Sin embargo, en escenarios reales, estos patrones pueden obedecer cualquier otra distribución.

Para poder adoptar el modelo $\mathtt{imodel}$ propuesto, es necesario identificar los parámetros p y L_i que corresponderían a un modelo de Gilbert equivalente. Para ello se propone la siguiente metodología: tras obtener un histograma representativo de pérdidas de paquetes del flujo de voz, se busca el modelo de Gilbert que mejor se ajuste a dicho histograma.

Para ello se puede emplear cualquier procedimiento de ajuste que pueda ejecutarse en tiempo real. Por ejemplo, en el apéndice A se propone un procedimiento basado en la optimización por mínimos cuadrados.



a) b)

Figura 5: Histogramas acumulados de probabilidad de pérdidas (a) y tamaño medio de ráfagas (b) para las trazas evaluadas.

Una vez ajustado el histograma de ráfagas experimentado con un modelo de Gilbert, los parámetros del modelo de Gilbert obtenido son los que se utilizarían para calcular el valor de W_{acc} estimado, calculado mediante la expresión 3.4.

3.4 VALIDACIÓN EXPERIMENTAL

Para validar el modelo y medir el error cometido en distintos tipos de trazas, se ejecuta el reconocimiento sobre trazas obtenidas mediante simulación y mediante la captura de trazas reales. Dichas trazas no siguen un modelo de Gilbert de generación de pérdidas, por lo que es necesario ajustarlo previamente para que ofrezca una estimación precisa.

En concreto, se realizan las estimaciones de un total de 118 trazas diferentes, de las cuales 20 corresponden a trazas reales. Las características de todas las trazas evaluadas se resumen en la figura 5, donde se muestran los histogramas de las probabilidades de pérdidas y de las ráfagas que exhiben las trazas evaluadas.

Como se puede observar, las características de las trazas evaluadas se encuentran distribuidas entre los intervalos $p = [0.01, 0.16]$, y $L_i = [1, 10]$.

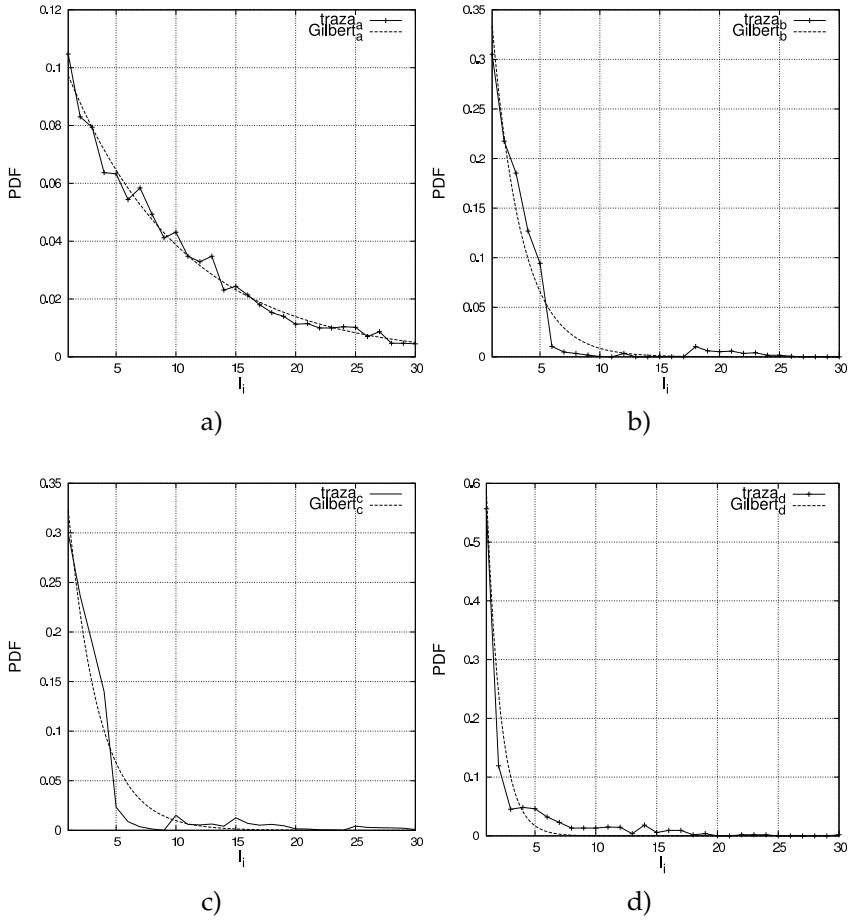


Figura 6: Ajuste de las PDF de distintas trazas sintéticas. Los errores de predicción para a), b), c) y d) son, respectivamente 0.0278, 1.1078, 2.0068, 3.7395.

Para estimar el valor del imodel de cada traza, es necesario como paso previo ajustar a un modelo de Gilbert los histogramas de las trazas evaluadas. De forma ilustrativa, en las figuras 6 y 7 se detallan los histogramas originales y los ajustados a un modelo de Gilbert de las trazas más representativas del conjunto evaluado. En concreto, la figura 6 muestra los histogramas de varias trazas generadas mediante simulaciones de escenarios de red con topologías aleatorias, y en la figura 7 se muestran las de varias trazas de flujos reales.

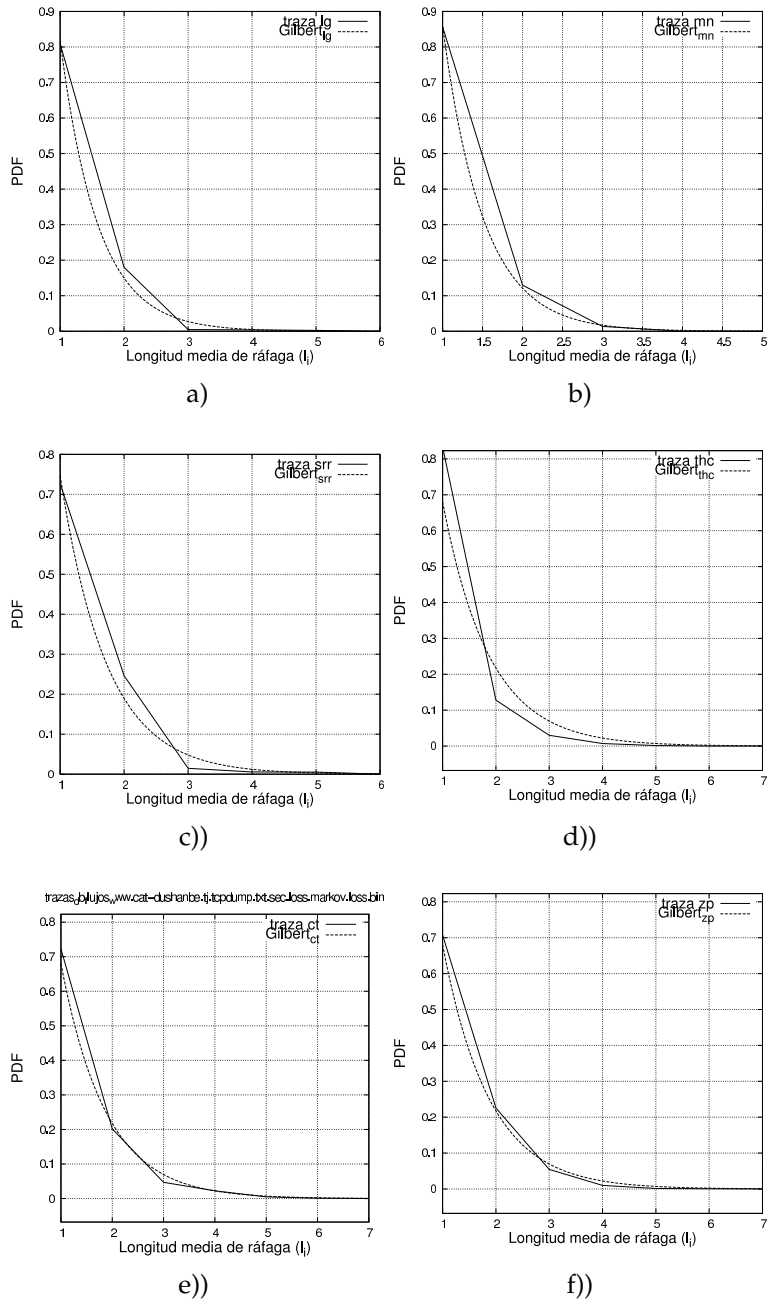


Figura 7: Ajuste de las PDF de distintas trazas capturadas. Los errores de predicción para a, b, c,d,e y f son, respectivamente 0.1021, 0.1683, 0.07, 0.4564, 0.01, y 0.2328

Tras utilizar la expresión 3.4 del imodel propuesto, se obtiene que el error de predicción de las trazas sintetizadas

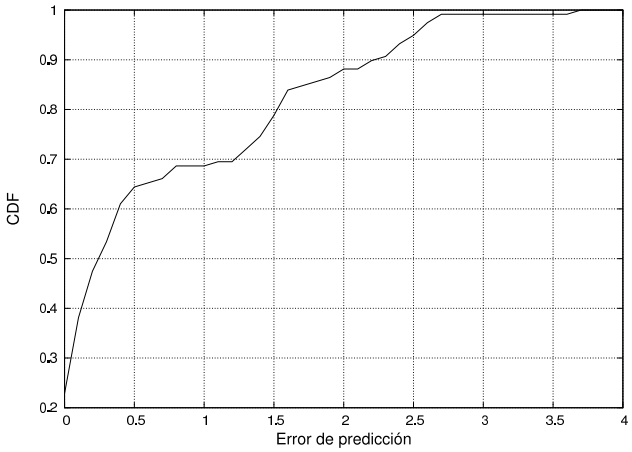


Figura 8: Histograma acumulado de errores de predicción sobre la evaluación de las trazas.

tienen una media 0.8704, con una desviación estándar de 0.0022. El error de predicción medio para las trazas reales es de 0.3384 puntos, con una desviación estándar de 0.0267. La precisión que exhibe el modelo para las trazas reales se justifica porque las probabilidades de pérdidas y tamaños medios de ráfagas de estas trazas son muy bajos, rango en el cual el modelo imodel obtenido presentaba un intervalo de confianza más estrecho. Por último, la media global de error de estimación contemplando todo el conjunto de trazas es de 0.7847, con una desviación estándar de 0.0059.

Para resumir los resultados de las estimaciones del modelo imodel, la figura 8 muestra un histograma de los errores de predicción cometidos sobre el conjunto de tramas de evaluación descrito. Como puede comprobarse, el 80% de los errores cometidos son inferiores a 1.5 puntos, y el 90% es inferior a 2.5.

3.5 DISCUSIÓN

Aunque el modelo presentado considera tanto la distribución de pérdidas como el efecto de las pérdidas en la inteligibilidad del flujo de voz, éste no contempla la naturaleza temporal de los flujos de voz. Esto implica que es insensible a los retardos que puedan experimentar los paquetes de los flujos de voz.

Es por ello que, como trabajo futuro, se estudiará la integración del presente modelo con funciones de utilidad que contemplen el efecto del retardo sobre el nivel de interactividad del flujo de voz.

Después de analizar los mecanismos de evaluación de calidad de voz existentes podemos esperar que surjan nuevos modelos que contemplen las distintas dimensiones que supone la evaluación de la calidad de voz. Parece lógico pensar que para distintos contextos, la calidad de voz deba ser evaluada mediante distintos criterios. Por ejemplo, para comunicaciones en entornos hostiles, la calidad de voz puede ser entendida como el nivel de inteligibilidad obtenido, mientras que en aplicaciones de ocio, como puede ser un chat de voz o el sistema de comunicación en un juego en línea, la calidad de voz se entiende como la fidelidad de la señal recibida a la original.

Por ello vislumbramos nuevos modelos de calidad, evaluables en tiempo real, en los que se pueda ponderar las distintas facetas a medir (interactividad, inteligibilidad, fidelidad, fluidez, etc.), ajustadas según la demanda de la aplicación de transmisión de voz de que se trate.

Por último, indicar que el modelo se ha obtenido exclusivamente para un único códec, por lo que se prevé la necesidad de extenderlo a otros codificadores estándares, mediante la metodología automática adoptada en este trabajo.

3.6 CONCLUSIONES

Como se ha demostrado en la bibliografía relacionada, el reconocimiento automático de voz es una efectiva medida de calidad de flujos de VoIP. Sin embargo, su utilización depende del empleo de un reconocedor, lo que puede dificultar su ejecución en tiempo real.

Por ello, en este capítulo se ha propuesto una expresión analítica que modela el comportamiento de un reconocedor de referencia, que utiliza una base de datos de palabras y una tarea estandarizadas. La tarea de reconocimiento consiste en identificar números de varios dígitos, pronunciados

en inglés. Dicha tarea garantiza una gran correlación entre el valor obtenido mediante el reconocmiento automático y el de un oyente humano, ya que la determinación de cada dígito no puede inferirse del contexto ni de los dígitos adyacentes (en caso contrario el oyente humano tendría ventaja).

El modelado se ha realizado sobre el resultado de ejecutar el reconocedor sobre 3957 distintos patrones de pérdidas, generados mediante modelos de Gilbert, obteniendo un error de ajuste medio inferior a 0.35 puntos porcentuales.

La precisión del modelo se evalúa asimismo sobre trazas que no siguen un patrón de pérdidas ideal según el modelo, es decir, generado por un modelo de Gilbert. Como resultado, se obtiene un error medio de predicción inferior a 1.05 puntos.

Como resultado final, se ha obtenido un estimador de la inteligibilidad (o función de utilidad, expresión 3.4) de un flujo de voz codificado con PCM, que puede ejecutarse en línea y en tiempo real, tal que, conociendo la probabilidad de pérdidas p, y la longitud media de ráfagas L_i experimentadas, proporciona una estimación del W_{acc}.

4

UN SERVICIO DE REPARACIÓN PREVENTIVA: EL ENTREMEZCLADO

Como ya se expuso en el capítulo 2, las distintas técnicas de recuperación *preventivas* se caracterizan por anticiparse a las pérdidas de paquetes o a la congestión en la red, al contrario que las técnicas de recuperación *reactivas*, las cuales se aplican una vez detectado el problema. Una de las técnicas preventivas más importantes es el entremezclado de paquetes [96], empleado en los servicios de *streaming* de audio y vídeo para mitigar la degradación causada por las ráfagas de pérdidas. Su aplicación es adecuada en situaciones de congestión, ya que no consume ancho de banda adicional.

Sin embargo, su empleo en transmisiones de voz interactivas está limitado debido al retardo incurrido al almacenar temporalmente los paquetes antes de reenviarlos. Dicho retardo limita su utilización a un número reducido de casos.

En este capítulo se propone un nuevo tipo de entremezcladores de bajo retardo que reordenan paquetes provenientes de distintos flujos de audio y de otras fuentes de datos. Como se demostrará, este esquema conserva la capacidad de los entremezcladores por bloque tradicionales de redistribuir las ráfagas de pérdidas en pérdidas aisladas, obteniendo un alto nivel de calidad perceptual incluso bajo condiciones severas de pérdidas, sin penalizar el retardo introducido.

Junto a la proposición de estos nuevos esquemas, se incluye un estudio analítico de las propiedades de los mismos, obteniendo las expresiones que permitan estimar *a priori* la calidad perceptual e inteligibilidad que proporcionan, dadas unas condiciones de red. Basado en este estudio, presentaremos y evaluaremos un algoritmo para seleccionar automáticamente el entremezclador más adecuado a las condiciones de red existentes en cada momento.

4.1 INTRODUCCIÓN

Los flujos de voz generados por aplicaciones de VoIP, al igual que la mayoría de los flujos interactivos multimedia, llevan impuestos rígidos requisitos de tiempo real, ya que los paquetes de voz han de llegar a su destino en un plazo de tiempo acotado. Para que una transmisión de voz pueda considerarse interactiva, la latencia o retardo extremo a extremo de los paquetes de voz debe ser inferior a $d^*_{max} =$ 300ms [96]. Otro de los obstáculos que los mecanismos de provisión de calidad de servicio para voz deben salvar es la pérdida de paquetes VoIP que, como es sabido, degradan la calidad percibida por los usuarios del flujo de voz recibido. El impacto de este problema es especialmente relevante cuando las pérdidas afectan a varios paquetes consecutivos ([98], [99]).

A diferencia de otras técnicas de reparación, el entremezclado de paquetes tiene como objetivo redistribuir las pérdidas de paquetes consecutivos, dispersándolos en ráfagas más pequeñas, en lugar de retransmitir información sobre los paquetes perdidos [96]. Por tanto, el entremezclado es una técnica adecuada para situaciones en los que las pérdidas de paquetes se originan por congestión en los *routers* intermedios, ya que en dichas situaciones cualquier retransmisión o adición de información redundante podría agravar aún más la congestión.

Por otro lado, la técnica de entremezclado tiene como principal desventaja la introducción de un retardo adicional incurrido al almacenar temporalmente los paquetes para su reordenación.

Tradicionalmente, estas técnicas han sido llevadas a cabo extremo a extremo ([96], [37]). Sin embargo, la aparición de tecnologías tales como las *redes activas* [26] y las *redes superpuestas* (Overlay Networks) [29], brindan nuevas posibilidades para los servicios de reparación. Por ejemplo, teniendo en cuenta que los nodos intermedios de la red pueden formar parte de rutas de distintos flujos multimedia, en [8] se proponen algoritmos de entremezclado que reordenan paquetes procedentes de más de un flujo para

reducir el retardo extremo a extremo.

En este capítulo se diseña y evalúa una familia de entremezcladores por bloque de bajo retardo basados en un esquema de agregación de tráfico de varios flujos multimedia. Dichos flujos multimedia se caracterizan por tener el mismo periodo entre paquetes, t_f. La denominación de "bajo retardo" se refiere a que la demora introducida es inferior a la introducida por los entremezcladores por bloque clásicos con la misma capacidad de dispersión.

Se ofrece finalmente un modelado analítico que permitirá realizar una evaluación cuantitativa del rendimiento de los entremezcladores en términos de retardos máximos y medios introducidos, y en función de las distribuciones de pérdidas resultantes. La caracterización realizada permitirá estimar de antemano el rendimiento de cada uno de los esquemas considerados para unas condiciones de red dadas.

El resto del capítulo se organiza como sigue: en primer lugar, en la sección 4.2, se revisa la teoría del entremezclado por bloque. Posteriormente, la sección 4.3 describirá los algoritmos de entremezclado que proponemos. El modelado del funcionamiento del entremezclador se presentará en 4.4, proporcionando algoritmos para estimar su rendimiento. Para evaluar los entremezcladores, en la sección 4.5 se detallarán los resultados de varios experimentos llevados a cabo mediante simulación. Finalmente, la sección 4.6 proporciona una discusión sobre aspectos prácticos de la técnicas presentadas, y en la sección 4.7 se resumirán las principales conclusiones obtenidas.

4.2 TEORÍA BÁSICA DEL ENTREMEZCLADO POR BLOQUE

Un entremezclador se define como un dispositivo cuya entrada es una secuencia de símbolos de un alfabeto dado, y cuya salida es una secuencia reordenada de los mismos símbolos de entrada. Más específicamente, si la secuencia de entrada se expresa como $\ldots, a_{-1}, a_0, a_1, a_2, a_3, \ldots$, y la secuencia de salida es $\ldots, b_{-1}, b_0, b_1, b_2, b_3, \ldots$ el

entremezclador define una permutación $\pi : \mathbb{Z} \mapsto \mathbb{Z}$ tal que $a_i = b_{\pi(i)}$. Esta permutación es una correspondencia uno a uno, por lo que asociado a π existe un entremezclador inverso π^{-1}.

En la práctica cada símbolo corresponde a un paquete de entrada, que será almacenado temporalmente en el entremezclador implementado en el nodo de entremezclado.

Un entremezclador por bloque (n_2, n_1) reordena la secuencia de símbolos de entrada de forma que ninguna secuencia de n_2 símbolos en la secuencia de salida contenga símbolos separados por menos de n_1 símbolos en la secuencia de entrada [100]. Esta condición puede formularse mediante la expresión 4.1

$$|\pi(i) - \pi(j)| \geqslant n_1,\ |i - j| \leqslant n_2 - 1 \tag{4.1}$$

Se dice que un entremezclador introduce una *dispersión* s si ninguna pareja de símbolos a la entrada en cualquier intervalo de longitud s, está separada por una distancia de menos de s símbolos en la salida. Por tanto, la dispersión s de un entremezclador determina el tamaño de ráfaga máximo que puede aislar en pérdidas individuales.

Se dice que un entremezclador es *periódico* si verifica la ec. 4.2

$$\pi(i + e) = \pi(i) + e, \forall i \tag{4.2}$$

donde e es el periodo del entremezclador.

Para reordenar la secuencia es necesario almacenar temporalmente los paquetes de entrada en el entremezclador. Esto implica que entre la entrada y la salida de cada símbolo, el entremezclado introduce un retardo. Dado que el retardo extremo a extremo de los paquetes de voz influyen en la calidad de las aplicaciones de VoIP, el diseño de un entremezclador de paquetes de audio debe considerar cuidadosamente el retardo introducido.

El entremezclador por bloque óptimo, es decir, aquel con una dispersión dada tenga el menor periodo, será el que introduzca un menor retardo. A este respecto, en [101] y [100] se demuestra que no existen entremezcladores por bloque con periodos inferiores a s^2, dada una dispersión

Algoritmo de entremezclado $I(s)$

```
1  siguiente = 0
2  filaOUT = s − 1 − (siguiente mod s)
3  columnaOUT = ⌊siguiente / s⌋
4  while se introduzca un paquete p_j
5     filaIN = ⌊j / s⌋
6     columnaIN = j mod s
7     matriz[filaIN][columnaIN] = p_j
8     // Si se introduce un paquete pendiente
9     while matriz[filaOUT][columnaOUT] es válido
10                extraer paquete de
    matriz[filaOUT][columnaOUT]
11       siguiente = siguiente + 1
12       filaOUT = s − 1 − (siguiente mod s)
13       columnaOUT = ⌊siguiente / s⌋
14    endwhile
15 endwhile
```

Figura 9: Algoritmo de entremezclado $I(s)$.

de $n_1 = s$. Esto implica que las dimensiones mínimas de la matriz de entremezclado asociado deben ser de s filas y s columnas ($s \times s$).

De esta manera, fijada una dispersión s para un entremezclador por bloque de retardo mínimo, el algoritmo de entremezclado (denominado $I(s)$ en el resto del capítulo) se describe en la figura 9;

De forma resumida, se trata de introducir los paquetes por filas, de izquierda a derecha, y de arriba a abajo. La figura 10 ilustra el algoritmo con un ejemplo sobre el entremezclador $I(4)$ (el entremezclador por bloque mínimo que aísla ráfagas de hasta tamaño 4). Cada vez que se completa una columna, se extraen los símbolos almacenados en ella en orden, desde la última fila hacia la primera.

Cada vez que llega un paquete al entremezclador, se lleva a cabo la introducción de un símbolo. De aquí en adelante, la introducción de un símbolo en el entremezclador lo denominaremos *etapa*. Pues bien, dado un entremezclador $I(s)$,

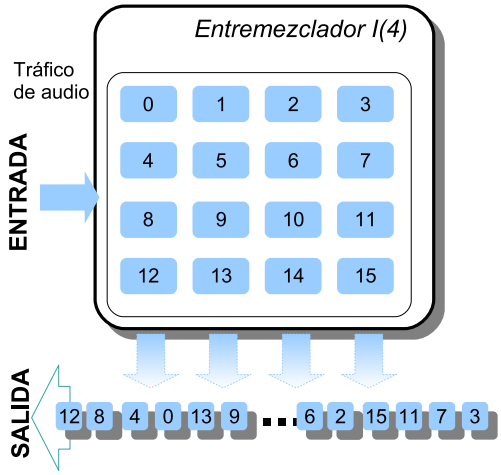


Figura 10: Ejemplo de entremezclador tradicional por bloque I(4).

si definimos D_{max} como el número de etapas máximo que un paquete permanecerá almacenado, se tiene que:

$$D_{max} = s \cdot (s - 1) \tag{4.3}$$

En el entremezclado por bloque, cada paquete puede experimentar un retardo diferente según su posición en la matriz de entremezclado y según la configuración del entremezclador. Por tanto, para ese caso, el retardo medio resultante por paquete puede ser inferior al máximo D_{max}. El número de etapas medio que un paquete permanece almacenado, D_{avg}, viene dado por la expresión 4.4:

$$D_{avg} = s \cdot (s - \frac{1+s}{2}) \tag{4.4}$$

Por tanto, un entremezclador de un solo flujo de entrada podrá aplicarse en transmisiones interactivas si cumple que $s \cdot (s-1) \cdot t_f < d^*_{max}$, donde $\frac{1}{t_f}$ corresponde a la tasa de generación de paquetes, y d^*_{max} es el retardo máximo extremo a extremo (en segundos) permitido para ese flujo. Por ejemplo, para valores típicos de $d^*_{max} = 300$ ms y $t_f = 20$ ms, no se puede generar un entremezclador por bloque que aísle pérdidas consecutivas mayores de 4 paquetes, aún en el mejor caso en el que el retardo de propagación extremo a extremo tendiera a 0 ms.

4.3 ESQUEMAS DE ENTREMEZCLADO MULTIFLUJO

El objetivo de entremezclar paquetes de distintos flujos es distribuir equitativamente entre ellos las ráfagas de pérdidas que se experimenten. De esta manera se reduce el tamaño de la ráfaga de pérdidas resultante por flujo. Para una longitud de ráfaga dada L_i, cuanto mayor sea el número de flujos n_f, menor será el tamaño de la ráfaga resultante por flujo L_o.

Los esquemas de entremezclado multiflujo de bajo retardo propuestos en esta tesis se basan en el funcionamiento del entremezclado por bloque descrito en la sección anterior. Los paquetes se introducen en las matrices por filas, y son extraídos por columnas. La principal diferencia de este esquema con respecto al entremezclador $I(s)$ es el uso de distintas matrices cuadradas, y el uso de paquetes de distintos flujos. Como se demostrará, comparado con el esquema $I(s)$, nuestra aproximación reduce el retardo introducido para una dispersión s dada.

Para ilustrar el funcionamiento de este tipo de entremezclado, en la sección 4.3.1 se muestra un caso particular denominado *entremezclador multiflujo elemental*. La descripción del entremezclador general se ofrece en la sección 4.3.2. En la sección 4.3.3 se describe además una mejora del entremezclador general mediante la inclusión en el entremezclado de tráfico adicional para reducir aún más el retardo introducido.

4.3.1 *Entremezclador multiflujo elemental* $II(n_f)$

El entremezclador $II(n_f)$ consiste en un planificador tipo *Round-Robin* para diferentes flujos. Como ejemplo, en la figura 11 se ilustra el funcionamiento del entremezclador $II(3)$ sobre tres flujos etiquetados como por f^1, f^2 y f^3.

El retardo introducido por este esquema es prácticamente cero. Concretamente, para el caso particular en el que todos los flujos exhiban el mismo t_f y estén sincronizados, es decir, que la llegada de paquetes de cada flujo se produzca en el mismo instante de tiempo, el retardo máximo obtenido

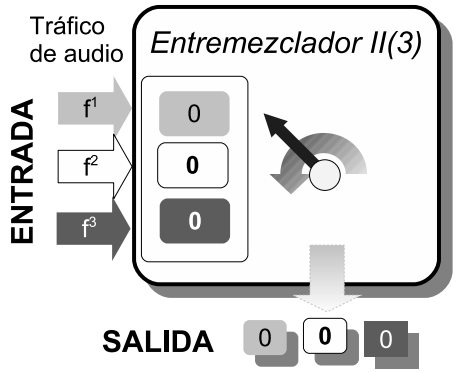


Figura 11: Esquema de entremezclado multiflujo elemental para $n_f = 3$.

con $II(s)$ sería $D_{max} = 0$.
Aún en caso de que los flujos no estuvieran sincronizados, el planificador podría configurarse para obtener un retardo por flujo acotado, $d_{max} \leqslant \frac{t_f}{2}$, ordenando los flujos adecuadamente. Básicamente se trataría de asignarle a cada paquete un número de secuencia para el entremezclado, y ordenar los flujos de forma que entre la llegada de los paquetes i-ésimos del flujo f^1 y el f^{n_f} no transcurra más de $\frac{t_f}{2}$ segundos.

La longitud de la ráfaga media resultante L_o para $II(n_f)$ está acotada por 4.5:

$$L_o \leqslant \frac{L_i}{n_f} \tag{4.5}$$

A pesar de su sencillez, este esquema no puede utilizarse en los casos en los que el número resultante de paquetes consecutivos perdidos por flujo sea mayor que cierto umbral, ya que la calidad percibida por el usuario y la inteligibilidad del flujo se degradarían rápidamente (ver capítulo 3). Por contra, su principal ventaja reside en que, independientemente del número de flujos disponibles y de la dispersión objetivo s, los paquetes experimentan un retardo insignificante.

Obsérvese que aunque los routers convencionales incluyen estrategias de encolado FIFO, por lo general la llegada de los paquetes de distintos flujos es un proceso estocástico, por lo que no se puede garantizar que se dé el

entremezclado elemental de los distintos flujos. Por tanto, para poder implementar este entremezclador, se requeriría utilizar en el *router* un clasificador, para distinguir cada flujo, y un planificador *round robin* para asegurar la salida multiplexada de los paquetes.

4.3.2 *Entremezclador multiflujo de bajo retardo* $\mathrm{II}(n_f, s)$

En el esquema elemental, los paquetes consecutivos de un flujo dado se separan a la salida una distancia de n_f símbolos gracias a la utilización de paquetes procedentes de otros flujos de idénticas características. Sin embargo, esta aproximación no es efectiva en los casos en los que el número de flujos de los que se dispone es bajo, como puede ocurrir por ejemplo en una red de acceso residencial, o en los que la longitud media de las ráfagas estimadas tenga como resultado una ráfaga media L_o por flujo excesivamente grande, ya que degradaría la calidad de los flujos resultantes de forma severa.

Los entremezcladores por bloque consiguen separar los símbolos consecutivos mediante su reordenación en matrices de dimensiones acordes a la dispersión objetivo s. Mientras que en el caso de los entremezcladores por bloque tradicionales $\mathrm{I}(s)$ todos los paquetes proceden del mismo flujo, en el entremezclador por bloque multiflujo II se utilizarán paquetes de distintos flujos, de forma que las matrices de entremezclado se completarán en un menor número de etapas, manteniendo la capacidad de dispersión.

Basados en esta idea, para dispersar ráfagas inferiores a $s + 1$ paquetes, proponemos el entremezclador $\mathrm{II}(n_f, s)$. En el nuevo esquema $\mathrm{II}(n_f, s)$, el proceso de extracción sigue siendo el mismo que en $\mathrm{I}(s)$, aunque el proceso de inserción de símbolos es diferente. A continuación se explica su funcionamiento.

De forma implícita, para el correcto funcionamiento de $\mathrm{II}(n_f, s)$, los flujos implicados deben tener el mismo periodo de generación de paquetes t_f y, por supuesto, deben compartir parte de la ruta que va desde el entremezclador a los respectivos destinos de cada flujo.

Sea n_m el menor número de matrices requerido para obtener un entremezclado periódico en el esquema $II(n_f, s)$. El valor de n_m vendrá dado por la expresión 4.6:

$$n_m = \begin{cases} 1 & \textbf{si } r = 0 \textbf{ ó } n_f > s \\ n_f & \textbf{en otro caso} \end{cases} \tag{4.6}$$

donde $r = s - \lfloor \frac{s}{n_f} \rfloor \cdot n_f$.

En el proceso de escritura, al rellenar las matrices del entremezclador, cada flujo mantendrá el orden relativo con respecto a los demás. Es decir, los paquetes del flujo $i+1$ se introducirán en filas posteriores a los del flujo i, nunca antes. Adicionalmente, para un flujo dado, cada fila será completada de izquierda a derecha de acuerdo al número de secuencia del paquete.

Para describir el procedimiento de escritura en la matriz, definimos R^i_j como el número de filas consecutivas que serán asignadas al flujo f^i en la matriz j, donde $i = 1, \ldots, n_f$ y $j = 1, \ldots, n_m$. Dado que el objetivo entremezclador es el de repartir las pérdidas que se experimenten entre los distintos flujos, en el procedimiento de escritura se intenta distribuir homogéneamente los paquetes de cada flujo entre las matrices de entremezclado. El procedimiento se basa en asignar el número de filas consecutivas para cada flujo y para cada matriz tal que el entremezclador resultante sea periódico y ningún flujo sea penalizado. En el caso particular en el que el número de filas de la matriz (s) es múltiplo del número de flujos (n_f), esta asignación es totalmente equitativa, y es igual al cociente entre filas y número de flujos. Sin embargo, en otro caso, es necesario asignar las filas restantes a los distintos flujos, de forma que el número total de filas asignadas por flujo sea el mismo.

Para simplificar dicho cálculo, la asignación se lleva a cabo en dos pasos: una asignación inicial equitativa del número de filas por flujo (ec. 4.7), y una actualización de R^i_j mediante la distribución de las filas restantes entre todos los flujos (ec. 4.8):

$$R^i_j = \lfloor \frac{s}{n_f} \rfloor, \ \forall i = 1 \ldots n_f, \ j = 1 \ldots n_m \tag{4.7}$$

$$R_j^{r_{i,j}} = R_j^i + 1,\ \forall i = 1 \dots n_f,\ j = 1 \dots n_m \tag{4.8}$$

donde el índice $r_{i,j}$ se define como:

$$r_{i,j} = \begin{cases} n_f - r + i & \textbf{si } i > j \\ n_f - r - j + 2 \cdot i & \textbf{si } i \leqslant j < n_f - r + i \\ i & \textbf{si } j \geqslant n_f \end{cases} \tag{4.9}$$

Por defecto, la dimensión de cada matriz es $s \times s$, al igual que en el entremezclador de retardo mínimo $I(s)$. Sólo existe un caso en el que su dimensión es inferior, dada por $(n_f \times 1)$, y se produce cuando el número de flujos es mayor que la ráfaga máxima esperada ($n_f \geqslant s$). Por tanto, el número de columnas de la matriz, n_c, se obtendrá mediante la expresión:

$$n_c = \begin{cases} 1 & \textbf{si } n_f \geqslant s \\ s & \textbf{en otro caso} \end{cases} \tag{4.10}$$

En cuanto a los requisitos de memoria, el entremezclador $II(n_f, s)$ necesita almacenar como máximo $e = n_m \cdot n_c \cdot s$ paquetes para ofrecer un entremezclado con un periodo e. Además, n_p, definido como el número de paquetes de cada flujo requerido para completar una matriz, viene dado por:

$$n_p = \frac{n_m \cdot n_c \cdot s}{n_f} \tag{4.11}$$

Para ilustrar gráficamente el procedimiento de escritura utilizaremos el siguiente ejemplo: se dispone de 3 flujos de audio diferentes que comparten una ruta en la que la máxima ráfaga estimada tiene una duración de 4 paquetes ($n_f = 3$ y $s = 4$). En este caso, según el algoritmo propuesto para la creación del entremezclador, serán necesarias $n_m = 3$ matrices de dimensión 4×4. Al rellenar la primera matriz, al primer y segundo flujo, f^1 y f^2 respectivamente, les corresponderá sólo una fila a cada uno, mientras que al tercer flujo, f^3 se le asignará 2 filas. La distribución de filas en las siguientes matrices, según los R_j^i calculados de acuerdo con la expresión 4.8, se muestra en la figura 12.

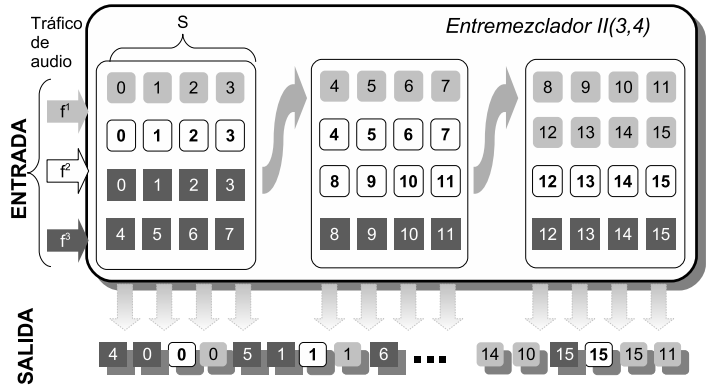


Figura 12: Esquema de funcionamiento del entremezclador $\mathrm{II}(3,4)$.

Por tanto, la salida del entremezclador $\mathrm{II}(3,4)$, tal y como se aprecia en la figura 12, vendrá dada por la secuencia: f^3_4, f^3_0, f^2_0, f^1_0, f^3_5, f^3_1, f^2_1, f^1_1, f^3_6, f^3_2, f^2_2, f^1_2, f^3_7, f^3_3, f^2_3, f^1_3, f^3_8, f^2_8, f^2_4, f^1_4, f^3_9, f^2_9, f^2_5, f^1_5, f^3_{10}, f^2_{10}, f^2_6, f^1_6, f^3_{11}, f^2_{11}, f^2_7, f^1_7, f^3_{12}, f^2_{12}, f^1_{12}, f^1_8 f^3_{13}, f^2_{13}, f^1_{13}, f^1_9 f^3_{14}, f^2_{14}, f^1_{14}, f^1_{10} f^3_{15}, f^2_{15}, f^1_{15}, f^1_{11}.

Con este esquema, ninguna ráfaga con un tamaño inferior o igual a s a la entrada, $s = 4$ en el ejemplo, tendrá como resultado dos o más paquetes perdidos consecutivos tras reordenar las secuencias originales de los flujos en su destino.

Obsérvese que el entremezclador $\mathrm{II}(n_f, s)$ es una generalización para varios flujos del entremezclador $\mathrm{I}(s)$. En concreto, el entremezclador $\mathrm{I}(s)$ coincide con el $\mathrm{II}(1, s)$.

4.3.3 *Entremezclador por bloque multiflujo no homogéneo* $\mathrm{II}(n_f, s, s')$

A pesar de que el entremezclador $\mathrm{II}(n_f, s)$ reduce en muchos casos el retardo introducido por el entremezclado, existen casos muy restrictivos en los cuales no puede aplicarse. Sin embargo, si en tales situaciones el entremezclador tiene en cuenta paquetes de otros flujos, es posible diseñar un entremezclador multiflujo que ofrezca un menor retardo.

Siguiendo con la filosofía de la familia de entremezcladores $\mathrm{II}(n_f, s)$, el entremezclador por bloque multiflujo no homogéneo $\mathrm{II}(n_f, s, s')$ tiene como objetivo reducir aún

más el retardo del entremezclador mediante la inclusión de paquetes procedentes de tráfico elástico, es decir, sin exigencias de retardo acotado.

De esta forma, si la tasa de llegada de tráfico adicional cumple las condiciones necesarias, como se mostrará más adelante, será posible obtener configuraciones de menor retardo de entremezclado con similar capacidad de dispersión. El entremezclador resultante se denominará entremezclador $II(n_f, s, s')$.

Este entremezclador consta de n_m matrices cuadradas, las cuales son construidas usando n_f flujos de audio. Los paquetes procedentes de flujos de tráfico adicional ocuparán las últimas filas de cada una de las n_m matrices, lo que permite emplear el algoritmo propuesto para el esquema $II(n_f, s)$ con ligeras modificaciones. La dispersión objetivo s sigue representando la longitud máxima de ráfaga que el entremezclador puede dispersar en pérdidas aisladas.

En este esquema se define s' como el número de filas en cada matriz del entremezclador destinada a almacenar el tráfico adicional. Dicho tráfico será etiquetado como flujo f^{n_f+1}.

El tráfico del flujo f^{n_f+1}, es decir, el tráfico adicional, rellenará sus correspondientes filas en las matrices desde abajo hacia arriba, y de izquierda a derecha. De esta manera, en el flujo entremezclado resultante se conservará el orden de los paquetes del tráfico adicional, propiedad deseable en toda estrategia de planificación.

El funcionamiento de este entremezclador se describe con la ayuda del ejemplo representado en la figura 13, en el que se muestra la instancia del entremezclador $II(2,4,1)$ ($n_f = 2$ flujos, ráfaga objetivo de $s = 4$ paquetes, verificándose que el tráfico adicional que permita usar $s' = 1$). En este ejemplo, la salida sería: f^3_0, f^2_4, f^2_0, f^1_0, f^3_1, f^2_5, f^2_1, f^1_1, f^3_2, f^2_6, f^2_2, f^1_2, f^3_3, f^2_7, f^2_3, f^1_3, f^3_4, f^2_8, f^1_8, f^1_4, f^3_5, f^2_9, f^1_9, f^1_5, f^3_6, f^2_{10}, f^1_{10}, f^1_6, f^3_7, f^2_{11}, f^1_{11}, f^1_7.

El funcionamiento de $II(n_f, s, s')$ requiere la redefinición de algunos de los parámetros definidos para el esquema $II(n_f, s)$. Sean $f^1, f^2, \ldots, f^{n_f}$ los n_f flujos de audio a entremezclar, y s la dispersión máxima esperada. En este

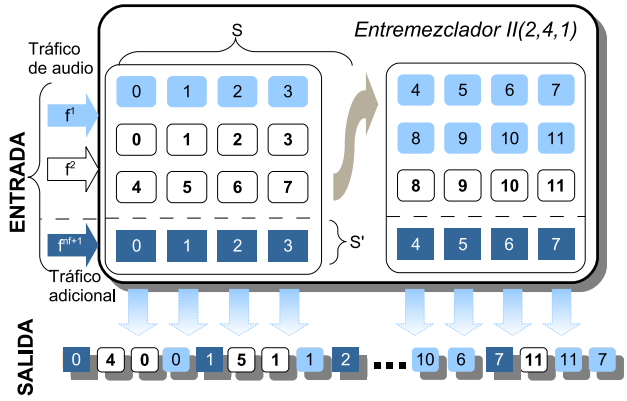


Figura 13: Esquema del entremezclador $II(2,4,1)$.

caso, de nuevo definimos R_j^i en dos pasos: una asignación inicial y su actualización:

$$R_j^i = \lfloor \frac{(s-s')}{n_f} \rfloor,\ i = 1 \dots n_f,\ j = 1 \dots n_m \tag{4.12}$$

$$R_j^{r_{i,j}} = R_j^i + 1,\ i = 1 \dots n_f,\ j = 1 \dots n_m \tag{4.13}$$

donde el índice de fila $r_{i,j}$ puede obtenerse igualmente mediante la expresión 4.14:

$$r_{i,j} = \begin{cases} n_f - r + i & \textbf{si } i > j \\ n_f - r - j + 2 \cdot i & \textbf{si } i \leqslant j < n_f - r + i \\ i & \textbf{si } j \geqslant n_f \end{cases} \tag{4.14}$$

en este caso el valor de r se define ahora como:

$$r = (s-s') - \lfloor \frac{s-s'}{n_f} \rfloor \cdot n_f \tag{4.15}$$

El número de columnas de la matriz n_c vendrá dado por:

$$n_c = \begin{cases} 1 & \textbf{si } n_f + s' \geqslant s \\ s & \textbf{en otro caso} \end{cases} \tag{4.16}$$

Además, para cada flujo, el número de paquetes entremezclados, n_p, viene dado por:

$$n_p = \frac{n_m \cdot n_c \cdot (s-s')}{n_f} \tag{4.17}$$

Para que el entremezclado siga manteniendo el mismo periodo e sin incrementar d_{max}, la tasa de paquetes entrantes de tráfico adicional, $\frac{1}{t_f^{n_f+1}}$, debe verificar la expresión 4.18:

$$\frac{1}{t_f^{n_f+1}} \geqslant \frac{e}{n_p \cdot t_f} = \frac{n_f}{t_f} \tag{4.18}$$

Esta condición garantiza que siempre que se completen las $s - s'$ posiciones de una columna con paquetes de los flujos multimedia, estarán disponibles al menos los s' paquetes de tráfico adicional necesarios para poder extraer los paquetes de dicha columna.

Por las definiciones hechas, se puede comprobar que el entremezclador $II(n_f, s)$ es el caso particular de $II(n_f, s, s')$ en el que $s' = 0$.

4.4 MODELADO DE LOS ENTREMEZCLADORES

El hecho de que los entremezcladores $II(n_f, s)$ y $II(n_f, s, s')$ reduzcan el retardo, con respecto a $I(s)$, amplía el abanico de posibles configuraciones para unas topología y condiciones de red dadas, especialmente en entornos con retardos más restrictivos.

Este hecho puede comprobarse en las tablas 1 y 2, en las que se muestran los retardos máximos d_{max} introducidos por los entremezcladores $II(n_f, s)$ y $II(n_f, s, s')$ respectivamente, para distintos valores de n_f y s. Cada fila corresponde a un valor de s, y cada columna a un valor de n_f. Los retardos, expresados en milisegundos, se han obtenido mediante la expresión analítica 4.26 que se discutirá en el apartado siguiente. Nótese que en el caso del entremezclador $II(n_f, s, s')$, existen varias posibles configuraciones dados los valores n_f y s, ya que se podrían combinar con distintos valores de s'.

En los resultados de la tabla 2 se muestran sólo los retardos del entremezclador que exhiben el menor d_{max} para unos n_f y s determinados. Nótese que se selecciona, de entre los posibles s' que cumplen que $d_{max} < 300ms$, el

	Número de flujos (n_f)												
$s_{\Downarrow}$	1	2	3	4	5	6	7	8	9	10	11	12	13
2	40	0	0	0	0	0	0	0	0	0	0	0	0
3	**120**	60	0	0	0	0	0	0	0	0	0	0	0
4	**240**	80	80	0	0	0	0	0	0	0	0	0	0
5	-	**200**	**200**	100	0	0	0	0	0	0	0	0	0
6	-	**240**	120	**240**	120	0	0	0	0	0	0	0	0
7	-	-	**280**	**280**	**280**	140	0	0	0	0	0	0	0
8	-	-	-	160	-	-	160	0	0	0	0	0	0
9	-	-	-	-	-	-	-	180	0	0	0	0	0
10	-	-	-	-	200	-	-	-	200	0	0	0	0
11	-	-	-	-	-	-	-	-	-	220	0	0	0
12	-	-	-	-	-	240	-	-	-	-	240	0	0
13	-	-	-	-	-	-	-	-	-	-	-	260	0
14	-	-	-	-	-	-	280	-	-	-	-	-	280

Tabla 1: Comparación de los retardos máximos introducidos por los entremezcladores $\Pi(n_f, s)$ para $s = 2 \ldots 14$ y $n_f = 1 \ldots 13$, expresados en milisegundos. Los valores en negrita son aquellos mejorados por el entremezclador $\Pi(n_f, s, s')$.

que obtiene menor retardo. Para resumir la información detallada en las tablas, estos resultados se presentan también gráficamente en la Fig. 14.

Tanto en las tablas 2 y 1 como en la Fig. 14, puede comprobarse cómo el nuevo entremezclador $\Pi(n_f, s, s')$ hace posible combinaciones adicionales con retardos inferiores a los del esquema $\Pi(n_f, s)$.

Las principales propiedades a evaluar para caracterizar un entremezclador para la de reparación de audio en tiempo real son dos: el retardo máximo extremo a extremo resultante que introduce para cualquier paquete (d_{max}), y la longitud media de ráfaga resultante (L_o). Otro parámetro característico usado como indicio de calidad es el retardo medio por paquete (d_{avg}).

	Número de flujos (n_f)												
$s_{\Downarrow}$	1	2	3	4	5	6	7	8	9	10	11	12	13
2	40	0	0	0	0	0	0	0	0	0	0	0	0
3	**60**	60	0	0	0	0	0	0	0	0	0	0	0
4	**80**	80	80	0	0	0	0	0	0	0	0	0	0
5	**100**	**100**	**100**	100	0	0	0	0	0	0	0	0	0
6	**120**	**120**	120	**120**	120	0	0	0	0	0	0	0	0
7	**140**	**140**	**140**	**140**	**140**	140	0	0	0	0	0	0	0
8	**160**	**160**	**160**	160	**160**	**160**	160	0	0	0	0	0	0
9	**180**	**180**	**180**	**180**	**180**	**180**	**180**	180	0	0	0	0	0
10	**200**	**200**	**200**	**200**	200	**200**	**200**	**200**	200	0	0	0	0
11	**220**	**220**	**220**	**220**	**220**	**220**	**220**	**220**	**220**	220	0	0	0
12	**240**	**240**	**240**	**240**	**240**	240	**240**	**240**	**240**	**240**	240	0	0
13	**260**	**260**	**260**	**260**	**260**	**260**	**260**	**260**	**260**	**260**	**260**	260	0
14	**280**	**280**	**280**	**280**	**280**	**280**	280	**280**	**280**	**280**	**280**	**280**	280

Tabla 2: Comparación de los retardos máximos introducidos por los entremezcladores $\mathrm{II}(n_f, s, s')$ para $s = 2 \ldots 14$ y $n_f = 1 \ldots 13$, expresados en milisegundos. Los valores en negrita son aquellos mejorados por el entremezclador $\mathrm{II}(n_f, s, s')$.

Una de las principales características del entremezclador multiflujo $\mathrm{II}(n_f, s, s')$ es que en algunas configuraciones introduce diferente retardo a los distintos flujos, dependiendo del orden de su aplicación en la matriz de entremezclado: cuanto mayor es el índice del flujo, mejor es el servicio recibido, ya que menor es el retardo experimentado.

4.4.1 *Análisis de los retardos de entremezclado*

Para dispersar una longitud de ráfaga fija s, el menor D_{max} que se puede obtener con un entremezclador $\mathrm{II}(n_f, s, s')$ se consigue cuando se cumplen las condiciones expresadas en 4.19 y 4.20. Dado que $\mathrm{II}(n_f, s)$ es la particularización

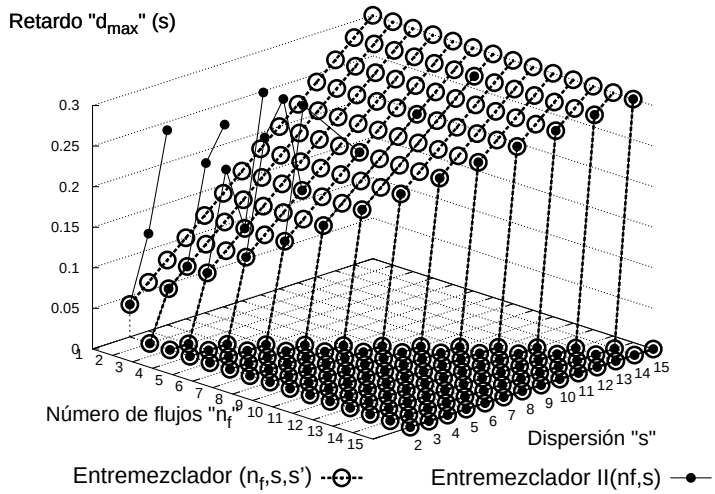


Figura 14: Comparación entre los retardos válidos introducidos por el esquema $II(n_f, s)$ y el esquema $II(nf, s, s')$ para VoIP.

de $II(n_f, s, s')$ cuando $s' = 0$, las siguientes expresiones pueden aplicarse igualmente a $II(n_f, s)$:

$$n_f = s - s' - 1 \tag{4.19}$$

$$\frac{(s - s')}{n_f} = 2 \tag{4.20}$$

La condición 4.19 corresponde al caso en el que en cualquier matriz del entremezclador sólo hay un flujo con dos filas asignadas. La condición 4.20 se refiere al caso en el que sólo es necesaria una matriz de entremezclado, y en la que todos los flujos tienen asignados dos filas. En ambos casos el retardo máximo introducido se debe a que, en el peor de los casos, un paquete estará almacenado como máximo s etapas, o lo que es lo mismo, el número de paquetes de una fila. Por tanto, en este caso $D_{max} = s$, y lo que es lo mismo, $d_{max} = s \cdot t_f$ segundos.

De lo anterior se puede extraer que el máximo valor de s que se puede asignar dado un flujo con un tiempo máximo de vida por paquete d^*_{max} y un periodo entre paquetes t_f, debe satisfacer que $s < \frac{d^*_{max}}{t_f}$. Como ejemplo numérico para el entremezclador $II(n_f, s, s')$, si $n_f < s$, $t_f = 20$ ms y

$d^*_{max} = 300$ ms, se obtiene que s debe ser menor que 15. Por contra, para el entremezclador extremo a extremo $I(s)$, bajo las mismas restricciones, la limitación impuesta por el retardo de entremezclado es mucho más restrictiva. En concreto, $s < 5$ para $I(s)$ (ver apartado 4.2).

Aunque el retardo máximo para todos los flujos entremezclados puede ser suficiente para decidir qué entremezclador utilizar, en algunos casos es más efectivo estimar el retardo por cada flujo. Así por ejemplo, en una topología en la cual el retardo extremo a extremo es diferente para cada flujo, si consideráramos sólo el peor caso (es decir, el correspondiente al camino más largo), el número de entremezcladores elegibles sería inferior que si consideramos una asignación por flujos ordenados, teniendo en cuenta los diferentes retardos que obtendría cada flujo individualmente. Por ejemplo, si consideramos sólo el retardo máximo global del entremezclador $II(3,5)$, de 200ms, sólo podríamos seleccionar este entremezclador si los 3 flujos estuvieran como máximo a 100 ms de su destino. Sin embargo, si se calcula el retardo máximo que introduce este entremezclador para cada flujo, se observa que los sólo uno de los flujos tiene como retardo máximo 200 ms, mientras que otros dos flujos sólo experimentan 100ms de retardo. Con esta información se podría seleccionar como viable el entremezclador $II(3,5)$ en una topología en la que dos de los nodos estén a 200ms de sus destinos, y en la que sólo uno de ellos se encuentre a 100ms.

Sea D_{max_i} el máximo retardo introducido por el entremezclador $II(n_f, s, s')$ para cualquier paquete del flujo f^i ($i = 1 \dots n_f$). Este valor coincidirá con la diferencia entre el mayor número de paquetes acumulados en cualquier matriz del entremezclador y el del número de paquetes que haya acumulado el flujo i en esa matriz. Este valor puede calcularse como:

$$D_{max_i} = U_r^{n_f} - P_r^i \tag{4.21}$$

donde el índice r se calcula mediante 4.15, y coincide con el índice de la matriz en la que el flujo f^{n_f} pasa de tener asignadas $R_j^i + 1$ filas por matriz a sólo R_j^i. Es, por tanto, el flujo que ha introducido más paquetes hasta la

matriz r, y además es el flujo que completa las columnas de las matrices de entremezclado. Es en esta matriz en donde se produce la mayor difererencia entre los paquetes de cualquier flujo y los del flujo f^{n_f}. Para efectuar el cálculo del retardo máximo, P_j^i representa el número de secuencia (módulo n_p) del primer paquete de las primeras columna y fila asignadas al flujo f^i en la matriz $j = r$. Además, se define U_j^i como el número de secuencia (módulo n_p) del paquete en la primera columna y última fila asignadas a f^i en la matriz $j = r$. Obsérvese que los rangos de valores para i y j son respectivamente $i = 1 \dots n_f$ y $j = 1 \dots n_m$. Los valores de P_j^i y U_j^i se pueden calcular mediante las expresiones 4.22 y 4.23 respectivamente:

$$P_j^i = \sum_{k=2}^{j} R_{(k-1)}^i \cdot s \tag{4.22}$$

$$U_j^i = P_j^i + (R_j^i - 1) \cdot s \tag{4.23}$$

Otro de los parámetros clave a considerar es el retardo medio por paquete para un flujo. En este caso, D_{avg_i}, definido como el retardo medio para el flujo f^i, se puede calcular como:

$$D_{avg_i} = \frac{\sum_{j=1}^{n_m} D_j^i}{n_p} \tag{4.24}$$

donde D_j^i viene dado por:

$$D_j^i = s \cdot \sum_{k=1}^{R_j^i} (U_j^{n_f} - P_j^i - (k-1) \cdot s) \tag{4.25}$$

Para concluir, el retardo máximo D_{max} que sufrirá un paquete en un entremezclador $II(n_f, s, s')$ dado, puede calcularse como:

$$D_{max} = \begin{cases} s \cdot (\lfloor \frac{s-s'}{n_f} \rfloor + r - 1) & \textbf{if } n_f \geqslant 2 \cdot r \\ s \cdot (\lfloor \frac{s-s'}{n_f} \rfloor + n_f - r) & \textbf{endif } n_f < 2 \cdot r \end{cases} \tag{4.26}$$

Dado que el mayor retardo introducido por el entremezclador lo experimenta el flujo con menor índice (es decir, f^1), el umbral máximo D_{max} coincide con D_{max_1}.

4.4.2 *Estimación de la longitud media de ráfaga resultante* L_o

Como así se ha establecido en [95], el valor de la longitud media de las ráfagas resultantes está correlacionada con el nivel de calidad perceptual experimentada por el usuario.

Una vez conocido el patrón de pérdidas que experimentará el flujo de salida del entremezclador, la estimación resultante de antemano para el entremezclador $\Pi(n_f, s, s')$ nos permitirá seleccionar la configuración más adecuada para cada estado de la red. Para ello será necesario estimar la distribución de pérdidas de paquetes entremezclados mediante la función de densidad de probabilidad de las longitudes de pérdidas $f_B(x)$, donde $f_B(i)$ representa la probabilidad de ocurrencia de ráfagas de tamaño i.

En esta sección proponemos un algoritmo para calcular la longitud media de las ráfagas resultantes L_o dado un entremezclador $\Pi(n_f, s, s')$ y un patrón de pérdidas caracterizado por $f_B(x)$. Se denominará B al mayor tamaño de ráfaga que aparezca en el patrón de pérdidas. El algoritmo consta de dos partes: la fase de inicialización y el cálculo de la estimación de L_o.

La fase de inicialización, resumida en el seudocódigo de la Fig. 15, consiste simplemente en el cálculo del patrón de salida de paquetes del entremezclador por algoritmo utilizado. Así pues, tras esta fase de inicialización, $out_0[i]$ contendrá el número de secuencia e identificador de flujo del paquete i-ésimo a la salida del entremezclador. Para ello, se alimenta el entremezclador con suficientes paquetes de cada flujo de voz (líneas $1-6$) y de paquetes adicionales (líneas $7-9$). Finalmente se extrae todos los paquetes reorganizados según el entremezclador elegido (líneas $10-13$).

El número de paquetes a introducir para la fase de inicialización debe cumplir dos condiciones: en primer lugar, el número total de paquetes debe corresponder a un número entero de periodos de entremezclado, para que puedan completarse todas las matrices de entremezclado, y así poder extraer todos los paquetes insertados. En segundo lugar, para poder calcular el efecto de la ráfaga de tamaño B en la salida del entremezclador, el número total de paquetes

Algoritmo de inicialización

1 **foreach** flujo i, donde $i = 1$ hasta n_f
2 establecer $nPaquetes = (\lfloor \frac{B}{n_m \cdot s^2} \rfloor + 1) \cdot s^2$
3 **endforeach**
4 **endfor** $j = 1$ hasta $nPaquetes$
5 introducir $paquete_j^i$ en el entremezclador
6 **endfor**
7 **for** k=1 hasta $nPaquetes$
8 introducir $paquete_k^{nf+1}$ en el entremezclador
9 **endforeach**
10 **while** el entremezclador no esté vacío
11 extraer paquete del entremezclador
12 introducir el paquete extraído en cola out_0
13 **endwhile**

Figura 15: Inicialización del algoritmo de estimación de L_o.

introducidos debe ser mayor a B. Ambas condiciones se satisfacen introduciendo s^2 paquetes por cada flujo (los necesarios por ciclo de entremezclado).

La Fig. 16 muestra el seudocódigo de la segunda parte del algoritmo para estimar el tamaño de ráfaga media resultante L_o. Durante esta fase se examina cada posible localización de las ráfagas de i paquetes en la salida de un ciclo completo, considerando como peso la frecuencia de las ráfagas de tamaño i. Para ello, en la línea 1 se itera sobre todas las posibles longitudes de ráfagas presentes en el patrón de pérdidas en al red, para comprobar el impacto medio que tiene esas ráfagas. A continuación, en la línea 5 se itera sobre cada una de las posibles posiciones de un periodo de entremezclado en las que podría incidir la ráfaga. En la línea 7 se obtiene la salida del flujo agregado, desentremezclándolo. Las líneas de la 8 a la 14 contabilizan las ráfagas resultantes para cada flujo. Por último, las líneas 16 a 18 calculan el tamaño medio de las ráfagas resultantes generadas por la ráfaga de longitud i, y la ponderan por la probabilidad de ocurrencia de dicha ráfaga, $f_B(i)$, en la línea 17.

Estimación de la ráfaga media resultante L_o

```
1  foreach longitud de ráfaga i existente en f_B(x)
2     inicializar secuencia out_tmp = out_0
3     se calcula el número de paquetes requeridos para un
   periodo:
4     e = n_m · n_c · s
5     foreach posición j = 1 hasta e
6        descartar i paquetes de out_tmp desde posición
   j
7        desentremezclar la secuencia out_tmp
8        foreach flujo k = 1 hasta n_f
9           foreach ráfaga raf de paquetes del flujo f^k
10             nRáfagas[raf] = nRáfagas[raf] + 1
11             si raf > 0
12               n = n + 1
13             is
14          endforeach
15       endforeach
16       foreach ráfaga m en nRáfagas
17          L_o=L_o + f_B(i) · m · nRafagas[m]/n
18       endforeach
19    endforeach
20 endforeach
```

Figura 16: Algoritmo de estimación de L_o resultante.

La dos fases del algoritmo nos permite predecir la longitud de ráfaga resultante tras reordenar los paquetes del flujo en el receptor. Su exactitud será verificada en la sección 4.5.2.

4.5 RESULTADOS EXPERIMENTALES

Para mostrar las capacidades de los distintos entremezcladores multiflujo propuestos se ejecuta una batería de simulaciones que considera distintas condiciones de red, evaluando su inteligibilidad y calidad perceptual resultante. Previamente se estudia la precisión del algoritmo propuesto

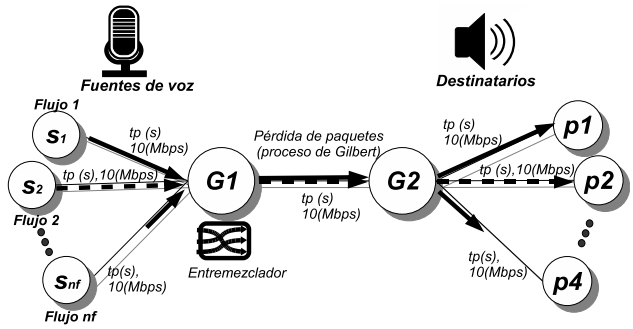


Figura 17: Configuración de la topología simulada.

para la estimación de retardos y las ráfagas medias resultantes. Para ambas pruebas se utiliza el conocido simulador de redes ns2 [102], configurado como se describe en el apartado 4.5.1.

Finalmente se realizarán comparaciones entre la calidad y capacidad de dispersión que ofrece $II(n_f)$ y $II(nf_s)$, así como entre $II(n_f, s)$ y $II(n_f, s, s')$.

4.5.1 *Entorno de simulación*

La topología establecida para las simulaciones se muestra en la figura 17. Consta de n_f fuentes de flujos de voz conectadas al primer nodo intermedio común en el que se localizará el entremezclador. Se establece también un enlace entre un nodo generador de tráfico adicional y el nodo de entremezclado.

La ruta compartida por todos los paquetes está formada por el enlace que une los dos nodos intermedios G1 y G2. Un modelo de Gilbert de dos estados [97] generará las pérdidas de paquetes en este enlace. La distribución de pérdidas generadas se caracterizan mediante una probabilidad de pérdidas p y la longitud de ráfaga media L_i. En las simulaciones, las fuentes envían sus paquetes de voz a los n_f destinatarios, como se puede apreciar en la misma figura. El tráfico de voz corresponde a un flujo de voz codificada con G.711 [15], obteniendo por tanto un flujo de 50 paquetes por segundo. Cada paquete encapsula una sola trama G.711 de 20ms de voz.

Tests	$\epsilon_{d_{max}}(s)$	$\sigma^2_{\epsilon_{d_{max}}}$	$\epsilon_{d_{avg}}(s)$	$\sigma^2_{\epsilon_{d_{avg}}}$	ϵ_{L_o}	$\sigma^2_{\epsilon_{L_o}}$
A	$2.61{\cdot}10^{-3}$	$2.58{\cdot}10^{-6}$	$3.04{\cdot}10^{-7}$	$6.64{\cdot}10^{-9}$	0.44	0.02
B	$1.78{\cdot}10^{-3}$	$1.33{\cdot}10^{-6}$	$3.38{\cdot}10^{-7}$	$2.99{\cdot}10^{-8}$	0.39	0.02

Tabla 3: Errores medios de los valores d_{max}, d_{avg} y L_o estimados para los conjuntos de simulaciones A y B.

Cada enlace tiene la misma capacidad (10Mbps) e igual tiempo de propagación t_p, cuyo valor concreto se especificará en cada simulación realizada.

4.5.2 *Verificación de la precisión de las estimaciones*

Para validar la expresión 4.26 en la que se calcula D_{max}, y el algoritmo de estimación de la longitud media de ráfaga de paquetes, descrito en la figura 16, en el entorno ns2 se simulan distintos escenarios que recorren el espacio de parámetros de configuración de los entremezcladores propuestos. De esta forma se pretende comparar los valores estimados con los obtenidos experimentalmente mediante simulación.

Para ello, la topología de cada simulación se configura tal como se detalló en la sección anterior, estableciendo en cada enlace un tiempo de propagación $t_p = 1ms$ para esta primera evaluación, dado que este parámetro no es relevante para la evaluación. Los resultados obtenidos se resumen en la tabla 3, donde se muestran la media y la varianza de los errores (ϵ) correspondientes a las estimaciones de d_{avg}, d_{max} y L_o para los distintos casos simulados. Cada ejecución simula un escenario con distintos parámetros de entremezclado, de distribución de pérdidas y de tráfico adicional presente. El tiempo de simulación en cada ejecución es de 5000 segundos, generándose 250000 paquetes/flujo en total.

Los resultados mostrados en la tabla 3 corresponden a los dos conjuntos de tests evaluados. El conjunto de tests A, está formado por las simulaciones realizadas sobre cada uno de los posibles entremezcladores $\Pi(n_f, s, s')$ cuyo retardo

máximo es inferior a $d^*_{max} = 300ms$, y distintos valores de ráfaga media de paquetes perdidos L_i. En concreto, $p = 0.2$, $L_i = \{5, 7, 10, 15, 20, 30\}$, $s' < s + n_f$, y el número de posibles entremezcladores que cumplen las restricciones es 146. En total, el número de combinaciones simuladas para el test A es 876.

Se puede apreciar en la tabla que el error medio de estimación del retardo máximo ($\epsilon_{d_{max}}$) es de 2.61 ms, y el error medio para la estimación del retardo promedio $\epsilon_{d_{avg}}$ es inferior al microsegundo. Con respecto al error cometido en la estimación del tamaño de ráfaga resultante, ϵ_{L_o}, se obtiene un error promedio de 0.44 paquetes, con una varianza de 0.02, lo que quiere decir que el algoritmo propuesto permite predecir L_o con un error inferior a medio paquete.

El segundo conjunto de pruebas, etiquetado como test B, tiene como objetivo mostrar la precisión de la estimación de los valores de L_o, d_{avg} y d_{max} al aplicar el entremezclador sobre flujos con un umbral de latencia d^*_{max} más permisivo que el impuesto para aplicaciones VoIP. Un ejemplo de este tipo de aplicaciones lo constituye el *streaming* de audio no interactivo. El test B, se compone de 955 simulaciones con entremezcladores que introducen un retardo máximo dentro del intervalo $300ms \leqslant d_{max} \leqslant 1500ms$. En este caso, los valores n_f, s, s' y L_i se seleccionan aleatoriamente. Para este conjunto de tests, el retardo máximo se puede calcular con un error inferior a $2ms$, como se aprecia en la segunda fila de la tabla 3, mientras que el error de estimación del retardo medio sigue siendo despreciable. A su vez, el error medio para la estimación de L_o disminuye de 0.44 a 0.39 paquetes.

Estos resultados indican que la estimación de los retardos que introduce un entremezclador, dados unos parámetros de configuración n_f, s, y s', se obtiene con una precisión del orden de milisegundos, para el retardo máximo, y de microsegundos para el cálculo del retardo medio.

En cuanto al error de predicción del tamaño medio de las rágafas resultantes L_o, las simulaciones han mostrado que es inferior a la unidad, lo que permite seleccionar con

suficiente fiabilidad el entremezclador que mejor aísle las ráfagas de pérdidas una vez conocidas las condiciones de red.

4.5.3 *Comparación de la longitud de ráfagas resultantes*

En este apartado se pretende mostrar el comportamiento de los entremezcladores multiflujo bajo condiciones distintas a las de diseño, es decir, cuando las ráfagas de pérdidas experimentadas son mayores que la dispersión objetivo s del entremezclador. Como se comentó en las primeras secciones, debido al límite de retardo impuesto por el tipo de flujo, el valor de dispersión s tiene un valor máximo. Sin embargo, aún en esos casos, el uso del entremezclador puede ser beneficioso, reduciendo el efecto de las ráfagas de pérdidas.

Para mostrar este hecho, en este apartado se evalúa mediante simulaciones la capacidad de dispersión del entremezclador, es decir, su capacidad de reducir L_o, sobre la topología presentada en la figura 17.

La longitud media de la ráfaga de perdidas sufridas por un flujo de voz está altamente correlacionada con la calidad percibida por el usuario final [103]. Es por ello que L_o es uno de los parámetros clave a evaluar para cada entremezclador.

Para ello, el modelo de Gilbert adoptado se configura en cada simulación para introducir pérdidas con un probabilidad $p = 0.2$, tomando L_i los valores 5, 10, 15, 20 y 30. Como resultado, en la figura 18 se muestran las longitudes de ráfagas medias resultantes frente a la dispersión s del entremezclador para estas condiciones tan adversas (nótese que en la gráfica no se especifica el número de flujos disponibles n_f, ya que este parámetro no afectó significativamente a dichos resultados).

Obsérvese que, incluso cuando la dispersión objetivo s del entremezclador es mayor que la ráfaga media experimentada L_i, (como en el caso de $L_i = 5$ y $s = 13$), el valor de L_o no llega a converger a la unidad como sería de esperar, es decir, a pérdidas aisladas. Esto se debe a que el modelo de Gilbert genera una distribución de pérdidas en la que

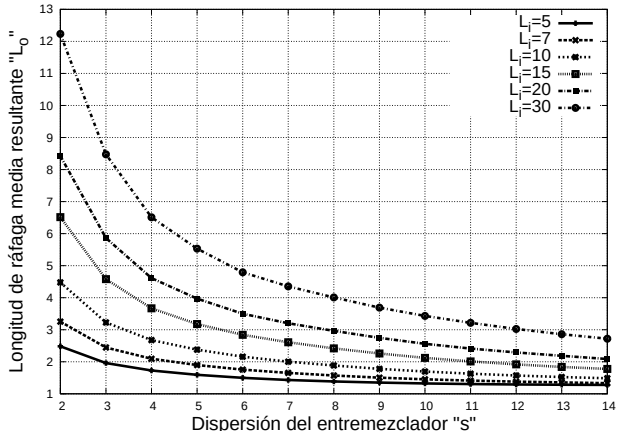


Figura 18: Ráfaga media resultante (L_o) frente a la dispersión objetivo del entremezclador (s), para los patrones de pérdidas $p = 0.2$ y $L_i = \{5, 10, 15, 20, 30\}$.

la probabilidad de ocurrencia de ráfagas mayores a la media L_i es mayor que cero. Concretamente, la probabilidad de obtener ráfagas de mayor tamaño viene dada por la expresión:

$$P(X > L_i) = (1 - p)^{L_i} \tag{4.27}$$

Dado que las ráfagas generadas en el experimento son más largas que la dispersión s para las que están diseñados los entremezcladores empleados, no todas las pérdidas de paquetes consecutivos pueden ser aislas. A pesar de ello, se comprueba que los entremezcladores sí que consiguen reducir la longitud de dichas ráfagas, aún cuando superan los parámetros de diseño de los entremezcladores.

4.5.4 *Evaluación de la inteligibilidad y de la calidad perceptual de los flujos entremezclados*

Adicionalmente a la evaluación del esquema propuesto mediante parámetros objetivos, en este apartado se realizará una evaluación mediante medidas subjetivas de calidad, más cercanas al rendimiento que percibe el usuario. Téngase en cuenta que al fin y al cabo se trata de un servicio de voz, el cual será percibido por un sujeto humano.

En este apartado se llevan a cabo dos estudios: la idoneidad del entremezclador básico $\Pi(n_f)$ en entornos donde L_i es mucho mayor que el número de flujos a entremezclar (n_f), y la proposición de criterios para seleccionar el entremezclador que provea mejor calidad experimentada por el usuario.

Para ello se obtendrán las puntuaciones de inteligibilidad W_{acc} y Corr, calculadas mediante el sistema de reconocimiento automático del habla descrito en el capítulo 3. Además, dado que la medida de inteligibilidad no puede medir otros parámetros de calidad tales como la interactividad del flujo, en las medidas se utilizará el valor R del *e-model* [78] y su equivalente en MOS_R para complementar la medida W_{acc}.

Estudio del rendimiento de $\Pi(n_f)$ *frente a* $\Pi(n_f, s)$

Como primer experimento, se llevan a cabo distintas simulaciones para obtener cuál es el máximo nivel de inteligibilidad que se lograría con los esquemas $\Pi(n_f)$ y $\Pi(n_f, s)$ dada una misma condición de red. Dado que el entremezclador $\Pi(n_f)$ no introduce ningún retardo, se podría argumentar que su utilización es recomendable para cualquier situación, ya que obtendrá un índice MOS_R en el *e-model* superior a la obtenida con cualquier entremezclador $\Pi(n_f, s, s')$. Sin embargo, como puede comprobarse en la figura 19, la capacidad de dispersión de $\Pi(n_f)$ es muy limitada para valores bajos de n_f, es decir, cuando hay pocos flujos disponibles.

En la figura 19 se indica adicionalmente la tasa de precisión de palabras correctas W_{acc} de los entremezcladores $\Pi(n_f)$ y $\Pi(n_f, s, s')$ frente al número disponible de flujos n_f, para unas condiciones de pérdidas con $L_i = 30$ y $p = 0.2$.

Examinando la figura se puede observar cómo según aumenta n_f, los resultados para ambos entremezcladores tienden a converger. Sin embargo, para valores bajos de n_f, la diferencia es notable (obteniendo ráfagas de hasta 5.5 paquetes menos, lo que se traduce en diferencias de 3.86 puntos W_{acc}, cuando $n_f = 2$). Esto reafirma que $\Pi(n_f)$ alcanza una alta capacidad de dispersión si existe un número suficientemente alto de flujos a entremezclar. Otra lectura

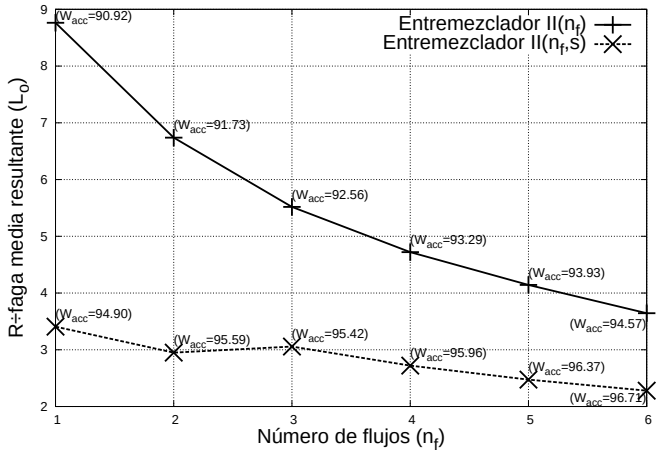


Figura 19: Comparación de los valores de L_o resultantes para $II(n_f)$ y $II(n_f, s)$ frente al número de flujos disponibles n_f.

de este hecho es que $II(n_f)$, aunque es sencillo y no introduce apenas retardo, no es apropiado para situaciones con ráfagas largas.

Criterios de selección de entremezcladores multiflujo

En este segundo experimento se simulan cuatro casos con dos flujos de voz y una distribución de pérdidas de $p = 0.1$ y $L_i = 30$. Sin embargo, el retardo de propagación entre fuentes y destinatarios son diferentes en cada simulación ($60ms$ para el primer escenario, $90ms$ para los escenarios 2 y 3, y 60.9ms para el escenario 4). Asimismo, la tasa de paquetes adicionales de tráfico adicional varía para cada uno de los escenarios (50 paquetes/s para los escenarios 1 y 2, 75.188 paquetes/s para el 3, y 100 paquetes/s para el escenario 4). Estas configuraciones permiten comprobar el rendimiento de los entremezcladores bajo distintas condiciones y niveles de restricción.

En cada situación se consideran todos los entremezcladores aplicables dadas las restricciones de retardo extremo a extremo de la topología y el tráfico de fondo de cada experimento. El retardo extremo a extremo propio de la topología limita el conjunto de entremezcladores viables. Como en simulaciones anteriores, cada paquete entremezclado co-

Entremezclador	L_o	d_{avg}	d_{max}	MOS_R	Corr	W_{acc}
Escenario 1: $t_p = 3 \times 20ms$, tráf. adicional: 50 paq/s						
II(2,6,2)	4.51	0.12 s	0.18 s	3.28	83.66	93.39
II(2,6)	3.57	0.18 s	0.30 s	3.20	85.70	94.21
II(2,5)	4.05	0.16 s	0.26 s	3.22	83.91	93.62
II(2)	10.20	0.06 s	0.06 s	3.36	77.03	90.42
Escenario 2: $t_p = 3 \times 30ms$, tráf. adicional: 50 paq/s						
II(2,6,2)	4.51	0.15 s	0.21 s	3.24	83.66	93.39
II(2,5)	4.05	0.19 s	0.29 s	3.09	83.91	93.62
II(2)	10.20	0.09 s	0.09 s	3.32	77.03	90.42
Escenario 3: $t_p = 3 \times 30ms$, tráf. adicional: 75.188 paq/s						
II(2,7,3)	4.08	0.16 s	0.23 s	3.21	84.37	93.97
II(2,5)	3.81	0.19 s	0.29 s	3.08	83.85	93.63
II(2)	8.97	0.09 s	0.09 s	3.31	77.57	91.05
Escenario 4: $t_p = 3 \times 20.1ms$, tráf. adicional: 100 paq/s						
II(2,8,4)	3.72	0.14 s	0.22 s	3.23	85.36	94.38
II(2,6)	5.99	0.12 s	0.30 s	1.66	53.98	78.02
II(2,5)	3.58	0.16 s	0.26 s	3.20	84.09	93.83
II(2)	7.93	0.06 s	0.06 s	3.34	77.63	91.21

Tabla 4: Medidas de calidad resultantes para los diferentes esquemas de entremezclado, con un modelo de pérdidas de Gilbert con $p = 0.1$, $L_i = 30$.

rresponde a una trama G.711.

Como resultado de las simulaciones, en la tabla 4 se muestran los valores L_o, d_{avg}, d_{max}, MOS_R, el porcentaje de frases correctas Corr, y el de precisión de palabra W_{acc} para cada escenario y entremezclador viable.

Es importante remarcar que el uso del *e-model* o el índice de reconocimiento automático empleados individualmente no proporcionan criterios suficientes para medir la calidad de las soluciones. De hecho, en los resultados mostrados puede observarse cómo existe relación entre el retardo in-

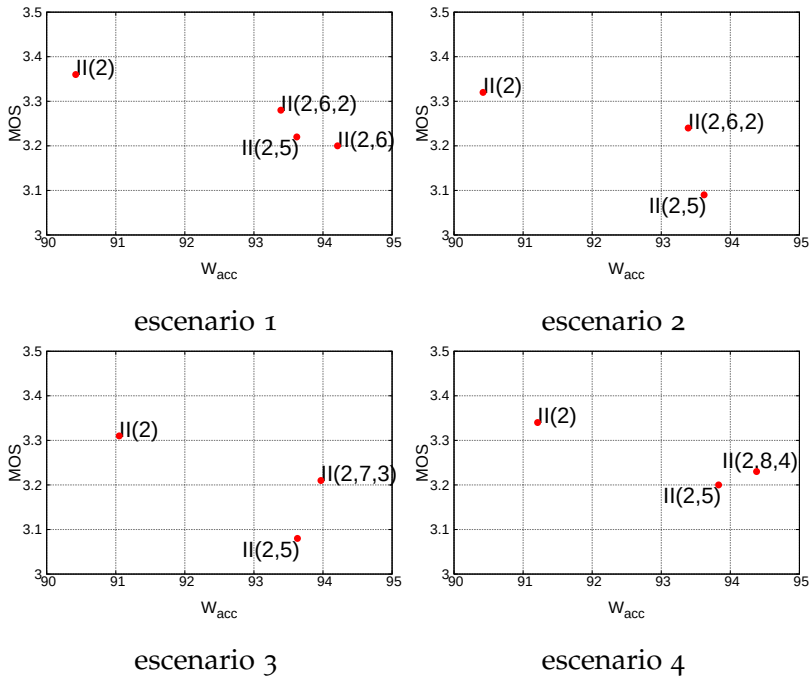


Figura 20: Espacio de las soluciones para cada escenario.

troducido y la capacidad de dispersión de los entremezcladores, parámetros contrapuestos. Además, la contribución de cada medida en la satisfacción del usuario dependerá de la aplicación específica. Por tanto, una combinación de ambas es necesaria, sin olvidar otros posibles criterios de selección. Por ejemplo, el retardo máximo de los paquetes puede determinar qué otros mecanismos de reparación pueden ser adoptados. Con esta filosofía, en la siguiente sección se propondrá un algoritmo para la selección automática del entremezclador mediante la combinación lineal ponderada de las dos medidas.

Analizando los resultados, en la figura 20 se ven representadas las distintas soluciones para cada escenario, mostrándolas como un punto en el plano de soluciones formados por el índice *MOS* y el índice W_{acc}. En estos experimentos, el criterio para seleccionar un entremezclador u otro es el de elegir el entremezclador que, dentro de la misma banda de calidad MOS que el mejor obtenido, tenga un buen índice W_{acc}. En este caso, se entiende como "buen índice" a los W_{acc} que estén el rango $W_{acc} \pm 1$ del mejor

obtenido. Dado que pueden existir distintas soluciones que satisfagan las condiciones impuestas, la selección del mejor entremezclador puede tener en cuenta además el tiempo de vida mínimo de los paquetes tras el entremezclado, es decir, la diferencia entre el retardo máximo impuesto y el tiempo consumido por el entremezclador y la propagación extremo a extremo ($d^*_{max} - d_{max}$).

Es reseñable el hecho de que las soluciones para los entremezcladores multiflujo se encuentren agrupados en un clúster de soluciones viables. Dado que las soluciones mostradas se localizan en la misma banda de calidad MOS_R, sólo se excluiría como candidato del escenario 4 a $II(2,6)$. Como segunda parte de la selección, se puede comprobar cómo en estos escenarios $II(n_f)$ se encuentra alejada del clúster en más de 1 punto W_{acc}, con lo que no sería contemplado como candidato, según los criterios especificados.

Cabe destacar que los resultados obtenidos en este experimento para $II(nf)$ y $II(n_f, s)$ son mejores que los esperados si no existiese tráfico adicional compartiendo el router. Esto se debe a que los paquetes del tráfico adicional se intercalan entre los paquetes de salida del entremezclador, dispersando los paquetes adicionalmente. Esto se puede comprobar examinando los resultados del entremezclador $II(2)$ en los escenarios 2 y 3, donde el único parámetro que varía es la tasa de tráfico adicional, y en el que el tamaño de ráfaga resultante L_o se reduce de 10.20 a 8.97 según se incrementa la tasa de paquetes de tráfico adicional de 50 a 75.188 paquetes/s.

Además, en esta evaluación, el sistema de reconocimiento automático proporciona los resultados para las mejores condiciones de codificación (uso del codificador sin pérdida y sin dependencia entre tramas G.711), bajo unas condiciones de pérdidas con probabilidad $p = 0.1$ y ráfaga media $L_i = 30$. Es por ello que incluso cuando existe una gran diferencia entre las ráfagas obtenidas para dos entremezcladores, sólo se observan pequeñas diferencias en la degradación de $Wacc$ tal y como se mostró en el modelo de inteligibilidad presentado en el capítulo 3. El efecto de las pérdidas de paquetes es más acusado si se utilizan codificadores con memoria, es decir, aquellos en los que la

decodificación de una trama depende de la anterior.

Analizando la tabla 4 se observa que el valor de W_{acc} obtenido en el escenario 1 con el entremezclador $II(2,6,2)$ se diferencia en 0.82 puntos con respecto al entremezclador $II(2,6)$, a cambio de obtener una mejora en el retardo medio de 60ms. Este tiempo suplementario haría posible, por ejemplo, el uso de técnicas de recuperación adicionales, incrementando aún más la calidad percibida. Incluso comparándolo con el entremezclador $II(2,5)$, que obtiene unos valores similares para L_o y W_{acc}, el entremezclador $II(2,6,2)$ propuesto consigue una mejora sustancial de 40ms. Observando el valor de d_{max}, se puede comprobar que el paquete que más tiempo ha permanecido en el entremezclador $II(2,6,2)$ tiene aún 120ms adicionales de vida útil que en el entremezclador $II(2,6)$.

En el escenario 2 se ha mantenido la configuración del primer escenario, exceptuando que el retardo extremo extremo se ha incrementado de 60 ms a 90 ms. Con este cambio en las condiciones del escenario, el entremezclador $II(2,6)$, candidato válido en el escenario 1, no puede ahora utilizarse. De esta manera el entremezclador $II(2,6,2)$ aparece como el mejor candidato, ya que, sin menoscabar los resultados de $II(2,5)$, introduce un menor retardo máximo (80 ms menos).

Se pueden hacer las mismas observaciones para el escenario 3. Aunque los entremezcladores $II(2,7,3)$ y $II(2,5)$ tienen un comportamiento similar en términos de W_{acc}, el primero reduce el retardo medio en 30ms, y el máximo en 60ms, lo que hace mejor candidato si se pretende combinar con otras técnicas de reparación. Esto implica, por ejemplo, que si se perdiera un paquete en el enlace G2-destinatario, sería viable utilizar algún protocolo de retransmisión automática (ARQ) entre G2 y el destino para volver a enviar el paquete. En este caso, el entremezclador II(2) obtiene un índice W_{acc} bastante inferior que el de $II(2,5)$ y $II(2,7,3)$.

El cuarto escenario está configurado para mostrar qué ocurre cuando se usa un entremezclador cuyo retardo de entremezclado sumado al de propagación es mayor que d^*_{max} en un escenario dado. En este escenario, el retardo

extremo a extremo entre fuente y destinatario es de 60.3ms (los tres enlaces tienen un tiempo de propagación $t_p = 20.1$ms). Esto implica que el retardo máximo permitido para que el entremezclador $II(2,6)$ sirva para aplicaciones de voz interactivas debe ser $d_{max} \leqslant 239.9$ms. Este es el caso de los entremezcladores $II(2,8,4)$ cuyo $d_{max} = 0.160$s, $II(2,5)$ con $d_{max} = 0.200$s, y $II(2)$ con $d_{max} \equiv 0$ms. El caso de $II(2,6)$ con $d_{max} = 0.240$s está en el límite, y los paquetes que sobrepasan el umbral serán descartados.

Nótese que aunque el entremezclador $II(2,8)$ no puede aplicarse en este escenario, ya que obtiene un $d_{max} = 0.480$s, muy superior al umbral de $d_{max} = 300$ms, el entremezclador $II(2,8,4)$ sí que se puede adoptar, dado que existe tráfico adicional que cumple las condiciones requeridas según la expresión 4.18 (en este caso, $\frac{1}{10\text{ms}} \geqslant \frac{2}{20\text{ms}}$).

Por último, el escenario 4 muestra otra situación en la que la solución de compromiso de $II(n_f, s, s')$ es preferible a la de $II(n_f)$ y $II(n_f, s)$. Si bien $II(2)$ obtiene el mejor valor de MOS_R, debido al retardo, y $II(2,5)$ obtiene el mejor valor de W_{acc}, el entremezclador $II(2,8,4)$ consigue como valoración la mejor relación entre ambos.

En este escenario, el entremezclador $II(n_f)$ aparece otra vez beneficiado, debido a que el tráfico adicional de 100 paquetes/s contribuye a aumentar su capacidad de dispersión de II(2), obteniendo unos resultados medios de L_o de 7.93 paquetes, frente a 10.20 paquetes que se obtienen cuando la tasa de tráfico adicional es sólo de 50 paquetes/s.

Como resultado de todo lo anterior, al no poder utilizar $II(2,6)$, el más adecuado es $II(2,8,4)$, que obtiene mejor resultado en ambos índices.

4.5.5 *Heurísticas para la selección de entremezcladores*

El objetivo de este apartado es proponer un procedimiento para seleccionar en línea el mejor entremezclador posible, dadas unas condiciones de red para tráfico de voz, una vez modelado el retardo introducido por un entremezclador multiflujo por bloque. Los criterios para realizar esta selec-

ción se basan en estimar la calidad perceptual y la inteligibilidad del tráfico resultante.

Dado que se han calculado las expresiones para predecir el retardo medio (D_{avg}) y máximo (D_{max}), así como el tamaño de ráfaga resultantes L_o que ofrece el entremezclador $II(n_f, s, s')$, podemos seleccionar, dado el patrón de pérdidas y el tiempo disponible para los paquetes de audio, el mejor entremezclador que se ajuste a las condiciones particulares de la red. Para ello, se estimará la calidad resultante de los entremezcladores que sean viables según las restricciones del escenario, seleccionando el que proporcione mejor experiencia de usuario.

Para empezar, el conjunto de entremezcladores que cumplen las restricciones de tiempo estará constituido por los entremezcladores $II(n_f, s, s')$ cuyo $d_{max} \leqslant 300$ms. Los valores de n_f, s, y s' para dichos entremezcladores verifican las siguientes condiciones :

$$\begin{aligned} n_f + s' &\geqslant s \\ s' = s - n_f - 1,\ para\ s &\leqslant \frac{d^*_{max}}{t_f} \end{aligned} \tag{4.28}$$

De este modo, dados un número de flujos n_f y una tasa de tráfico adicional disponible para entremezclado $\frac{1}{t_f n_f + 1}$, el conjunto de entremezcladores candidatos se determina calculando la longitud media de las ráfagas resultantes L_o y su retardo máximo d_{max}.

Dado que ni W_{acc} ni R son suficientes individualmente para valorar todas las dimensiones de la calidad perceptual de los flujos resultantes, se utilizará un único índice que contemple ambos resultados. Así, para cada entremezclador empleado se calcula el índice $Q_w(W_{acc}, R)$, computado como la combinación lineal de los valores W_{acc} y R. La salida de $Q_w(W_{acc}, R)$ es un valor en el intervalo $[0, 1]$, siendo la mejor puntuación $Q_w = 1$, y $Q_w = 0$ la más baja. Su expresión viene dada por 4.29:

$$Q_w(W_{acc}, R) = \frac{w|100 - W_{acc}| + (1 - w) \cdot |100 - R|}{100} \tag{4.29}$$

donde w es el peso del índice de inteligibilidad W_{acc} en el resultado de $Q_w(W_{acc}, R)$. El valor de w se asignará dependiendo del tipo de aplicación u objetivo del entremezclador. Para evaluar los resultados de este experimento se adopta el criterio de primar el nivel de inteligibilidad frente al nivel de retardo. A tal fin se estudia el comportamiento de Q_w con respecto al índice "w". Como se puede apreciar en las figura compuesta 21, donde se muestran los valores de Q_w para cada entremezclador y cada escenario según el peso w, el mejor valor de Q_w corresponde siempre al mismo en el rango $w = [0.41, 0.69]$ en cualquiera de los escenarios. La elección de cualquier valor en dicho rango mantendría estable la selección del entremezclador que tenga un mayor Q_w. Es por ello que se establece el peso $w = 0.6$.

Para determinar la mejor configuración del entremezclador para tráfico VoIP, dadas unas condiciones de red y de flujos de VoIP, se utilizará un estimador de Q_w, ($\hat{Q}_w$), calculado a partir de una valores $\hat{R}$ y $\hat{W}_{acc}$. Estos valores se estiman según el capítulo 3 a partir de los valores L_o, d_{avg} calculados mediante los algoritmos y expresiones descritos en la sección 4.4.

El algoritmo de selección en línea del entremezclador es el siguiente:

1. Generar el conjunto de entremezcladores viables (es decir, cuyos parámetros n_f, s y s' verifiquen 4.28), y $d_{max} \leqslant 300ms$.

2. Calcular para cada candidato su correspondiente índice $\hat{Q}_w$.

3. Seleccionar el candidato con mayor índice $\hat{Q}_w$.

Para validar esta aproximación, se ejecuta el algoritmo para los cuatro escenarios presentados en la sección anterior, calculando el índice $\hat{Q}_{0.6}$ con los valores de $\hat{R}$ y $\hat{W}_{acc}$ estimados. Como resultado, en la tabla 5 se muestra por cada escenario los valores estimados de $\hat{R}$, $\hat{W}_{acc}$ y $\hat{Q}_{0.6}$, así como el valor $Q_{0.6}$ computado sobre los valores obtenidos en los experimentos.

Puede comprobarse que la selección del candidato con mayor índice estimado $\hat{Q}_{0.6}$ coincide con el mejor candidato

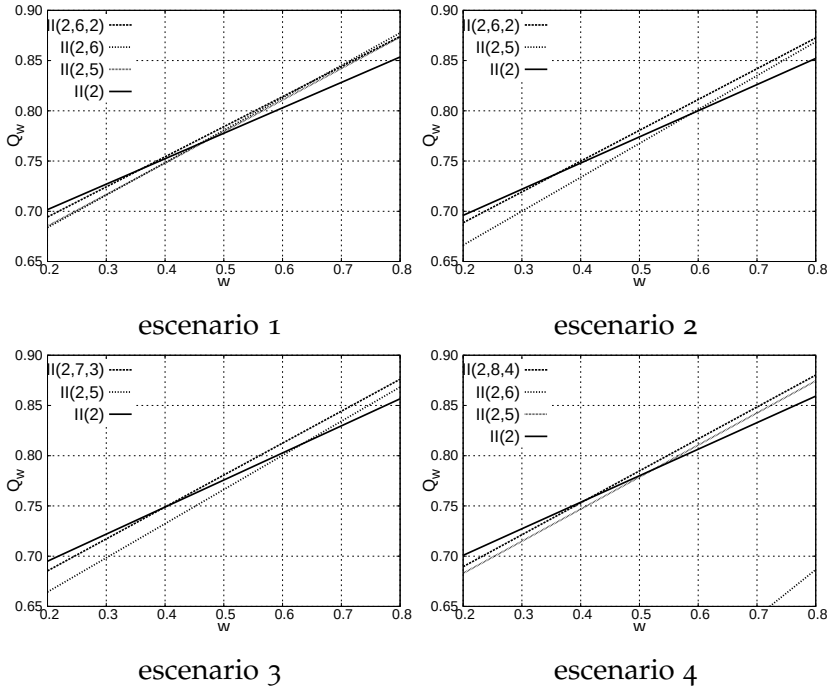


Figura 21: Dependiencia del índice Q_w con respecto a w, para los entremezcladores viables en los escenarios simulados.

evaluado mediante simulación. Examinando la tabla 5 se observa que la diferencia entre $Q_{0.6}$ y su estimación $\hat{Q}_{0.6}$ es menor de 0.004. Sólo en el caso en el que el entremezclador no cumple los requisitos para ser empleado, es decir, el entremezclador $\mathrm{II}(2,6)$ en el escenario 4, la estimación del índice de obtiene un mayor error.

No obstante, la precisión de los resultados permite garantizar que las soluciones escogidas mediante la heurística propuesta son las soluciones más adecuadas a cada situación, conocidas las condiciones de retardo extremo a extremo, la probabilidad de pérdidas de la ruta común de los flujos entremezclados, y la caracterización del tamaño medio de ráfagas de paquetes perdidos.

Nótese que las decisiones tomadas con esta heurística utilizando $w = 0.6$ no contradice los criterios expuestos en la sección 4.5.4. Por tanto, el modelado de los entremezcladores permite la ejecución de un algoritmo automático para seleccionar, dado un estado de red, el entremezclador

Entremezclador	$\hat{W}_{acc}$	$\hat{R}$	$Q_{0.6}$	$\hat{Q}_{0.6}$
Escenario 1: retardo total=60ms, tasa de f^{n_f+1}=50 paq/s				
II(2,6,2)	94.12	63.44	0.814	0.818
II(2,6)	94.44	61.92	0.813	0.814
II(2,5)	93.46	62.25	0.811	0.810
II(2,2)	90.18	65.11	0.803	0.801
Escenario 2: retardo total=90ms, tasa de f^{n_f+1}=50 paq/s				
II(2,6,2)	94.12	62.72	0.811	0.816
II(2,5)	93.46	59.88	0.801	0.802
II(2,2)	90.18	64.39	0.800	0.799
Escenario 3: retardo total=90ms, tasa de f^{n_f+1}=75.188 paq/s				
II(2,7,3)	94.71	62.18	0.813	0.817
II(2,5)	93.38	59.63	0.800	0.800
II(2,2)	90.04	64.11	0.803	0.797
Escenario 4: retardo total=60ms, tasa de f^{n_f+1}=100 paq/s				
II(2,8,4)	95.22	62.59	0.817	0.822
II(2,6)	79.31	31.21	0.593	0.601
II(2,5)	93.35	61.91	0.811	0.808
II(2,2)	90.02	64.80	0.806	0.799

Tabla 5: Resultado de Q_w, e índices estimados $\tilde{R}$, $\tilde{W}_{acc}$ y $\tilde{Q}_w$ resultantes para los diferentes escenarios de la sección anterior.

que ofrezca el mejor compromiso entre los parámetros de calidad perceptual de inteligibilidad e interactividad.

4.6 DISCUSIÓN

Si bien el entremezclado de paquetes se ha adoptado con éxito para transmisiones de vídeo ([104], [105]), hasta donde el conocimiento del doctorando alcanza, su aplicación sobre flujos interactivos de voz ha sido muy limitada. Su uso para este tipo de medios se ha centrado en las transmisiones

de flujos continuos de paquetes de audio sin requisitos estrictos de tiempo ([96], [106]).

La aproximación multiflujo que se ha presentado en este capítulo amplía el ámbito de aplicación de la técnica de entremezclado a nuevas condiciones de tiempo real. Sin embargo, el esquema propuesto presenta varios requisitos para ser aplicado correctamente.

Como primera condición a satisfacer, la red debe ser capaz de ejecutar el entremezclador en un nodo intermedio de la ruta que han de seguir los flujos. Esto se puede llevar a cabo en los paradigmas de redes activas y redes superpuestas, si bien es necesario definir un protocolo de señalización cuyas funciones comprenden la ubicación del entremezclador, identificación de los flujos a entremezclar, etc. Actualmente, el entremezclado multiflujo se podría implementar como *proxy* de reparación al que se suscriben los distintos flujos de voz de los clientes VoIP.

Para que la técnica de entremezclador propuesta funcione correctamente, es necesario disponer de la descripción actualizada del estado de la red (la distribución de pérdidas experimentadas), así como la información de las características de los distintos flujos (tasa de paquetes por segundo, número de flujos y latencia extremo a extremo). Esta monitorización puede llevarse a cabo mediante agentes especializados [107], o analizando la información proporcionada por los paquetes RTCP asociados a los flujos de audio [108].

Como se detalló en la descripción del esquema, otro de los requisitos del entremezclador consiste en que los flujos deben estar sincronizados. En caso contrario, el retardo del entremezclado puede incrementarse, ya que la salida de los paquetes de un flujo depende de la llegada de los paquetes de los otros flujos.

Para aplicar el entremezclador sobre flujos con distintas características, es decir, con diferentes periodos de generación de paquetes, se podría configurar un entremezclador por grupo de tráfico multimedia con iguales características. Las salidas de estos entremezcladores podrían ser a su vez objeto de entremezclado, realizando así un entremezclado compuesto de varias etapas.

Por último, a la hora de generar los posibles entremezcladores candidatos es necesario considerar la memoria reservada para efectuar la reordenación de los paquetes a su recepción. Este parámetro, y los recursos de memoria y capacidad de procesamiento disponibles en el nodo entremezclador son dos condiciones adicionales que deben satisfacer los entremezcladores candidatos.

Pese a todas estas restricciones, si se contempla la adopción de redes superpuestas, o incluso de *proxy* de reparación, es de esperar que las referidas limitaciones de recursos no sean críticas, ya que tanto receptores como nodos entremezcladores se ejecutarían en máquinas de propósito general, cuya capacidad supera con creces los recursos demandados por la técnica propuesta.

Por último comentar que, además del algoritmo heurístico descrito, en este trabajo se formulan otras propuestas para la selección en línea del entremezclador más adecuado en el capítulo 6.

4.7 CONCLUSIONES

En este capítulo se ha propuesto varios entremezcladores de paquetes de bajo retardo, concebidos para su utilización sobre aplicaciones de transmisión de voz sobre IP con restricciones de tiempo real. Dichos entremezcladores aprovechan la presencia de paquetes de distintas flujos para conseguir la dispersión especificada, reduciendo así el retardo producido por el almacenamiento temporal de cada paquete requerido para realizar la reordenación.

Asismismo se han obtenido las expresiones que permiten predecir el retardo máximo y medio por flujo y por entremezclador, así como un algoritmo que permite estimar el tamaño de las ráfagas resultantes una vez recuperado el orden original de los paquetes de voz. Dichas expresiones han sido validadas por medio de simulación, mostrando que el error de las estimaciones de retardos es inferior a varios milisegundos, y el error cometido en la estimación de las longitudes de las ráfagas resultates son inferiores a medio paquete.

Por último, hemos presentado un algoritmo heurístico basado en criterios de calidad e inteligibilidad para seleccionar el mejor adaptado de entre los posibles entremezcladores aplicables bajo unas condiciones concretas. Los resultados obtenidos han mostrado que el algoritmo realiza la misma selección basándose en los índices de calidad estimados que un algoritmo que dispusiera de los índices de calidad generados a posteriori experimentados.

Cabe destacar que el ámbito de aplicación de esta técnica preventiva de bajo retardo no se circunscribe a las transmisiones de voz interactivas, sino que puede ser empleado para otro tipo de transmisiones multimedia con restricciones de tiempo. Especial mención su posible aplicación en sistemas de reconocimiento remoto, cuyos beneficios han sido demostrados en otros trabajos sobre el tema [3].

5

UN SERVICIO DE DETECCIÓN TEMPRANA PARA LA REPARACIÓN REACTIVA DE PÉRDIDAS

Al contrario que los mecanismos de reparación preventivos, los mecanismos de reparación reactivos llevan a cabo su cometido en respuesta a la ocurrencia de algún problema que pueda afectar a la calidad del flujo de voz implicado. En el presente estudio, dicho problema consiste en la pérdida de paquetes de voz. Para ejecutar una acción de reparación en un entorno con exigencias de tiempo real, es necesario identificar el evento problemático tan pronto como sea posible, para así conceder suficiente tiempo al procedimiento de reparación.

Para flujos continuos de VoIP, se puede explotar el carácter periódico de la generación de paquetes para detectar las pérdidas, dado que se conoce su patrón de generación.

En este capítulo se presentan varios mecanismos de detección temprana para flujos de audio en tiempo real, que pretenden identificar las pérdidas de paquetes en el menor tiempo que sea posible. Dicho mecanismo se combina con la técnica de retransmisión automática, ofreciendo así la posibilidad de usar este procedimento para mejorar la calidad de flujos de VoIP.

Además de diseñar y evaluar varias alternativas para la detección temprana, se propone un algoritmo de localización del servicio de detección, que intenta maximizar la calidad percibida por los usuarios mediante medidas de evaluación perceptual.

5.1 INTRODUCCIÓN

La solicitud de retransmisión automática (ARQ, *Automatic Retransmission reQuest*) es uno de los mecanimos reactivos básicos empleados para recuperar paquetes perdidos [96].

El funcionamiento de este esquema es sencillo: tras detectar que no se ha recibido un paquete, el receptor solicita al emisor que lo retransmita.

A pesar de su amplia utilización en protocolos de transporte fiable de datos, tales como SRM [109], esta técnica no es adecuada para el tráfico con requisitos de tiempo real, ya que el tiempo total requerido para completar el reenvío puede ser inaceptable.

Como en cualquier mecanismo de reparación reactivo, para poder disparar la estrategia de reparación, retransmisión en este caso, es necesario detectar previamente la pérdida del paquete. Para utilizar este tipo de reparaciones sobre flujos de paquetes de voz, es necesario que el diseño del mecanismo de recuperación contemple en su funcionamiento los retardos tolerados por la aplicación para que el paquete recuperado pueda utilizarse.

Sin embargo, en la mayoría de los esquemas de reparación reactiva estudiados en la bibliografía se emplea como detector un esquema que simplemente comprueba la continuidad de los números de secuencia de los paquetes. Como se demostrará en posteriores secciones, este esquema de detección no es el más adecuado para la transmisión de flujos VoIP.

Si bien es deseable que la identificación del paquete perdido se produzca en el menor tiempo posible, existe otro parámetro a tener en cuenta a la hora de diseñar o adoptar un detector de pérdidas: las falsas alarmas. Una falsa alarma (FA) consiste en una detección errónea, es decir, la notificación de la pérdida de un paquete cuando ésta no se ha producido. Este evento conlleva la ejecución innecesaria del mecanismo de recuperación que se adopte. En el caso de ARQ se traduce en la solicitud y retransmisión innecesaria del paquete identificado como perdido, consumiendo así ancho de banda, malgastando los recursos de red, o incluso agravando la congestión.

Para reducir los retardos de retransmisión se han propuesto estrategias de retransmisión local ([109],[42]). En estos esquemas, cuando se solicita la retransmisión de un paquete, es un nodo más cercano que el emisor el que retransmite una copia. La adaptación de este esquema sobre

redes activas ha permitido que la retransmisión se lleve a cabo en nodos intermedios [110]. En concreto, en la sección 5.2 se analiza dicha técnica empleada para otro tipo de flujos de paquetes, analizando su idoneidad para flujos de paquetes con requisitos de tiempo real.

Para contemplar los requisitos del tráfico VoIP, en este capítulo proponemos varias alternativas de detección que pretenden reducir el retardo de detección. El retardo de detección (t_d) se define aquí como el tiempo transcurrido entre que el paquete debería haber llegado al nodo del detector y el instante en el que se notifica su pérdida. Su presentación y análisis se exponen en las secciones 5.3, 5.4 y 5.5, en las que se describe, respectivamente, el fundamento de la detección de pérdida de paquetes, la técnica básica de detección basado en secuencia, y nuestras propuestas basadas en predicción de llegada. En la sección 5.6 se propone un detector híbrido que aprovecha el funcionamiento de los dos tipos de detectores.

Por último, en la sección 5.8 se propone un algoritmo de localización del mejor nodo, tal que la reparación mejore la calidad percibida.

Las anteriores propuestas se evaluarán mediante simulación en las secciones 5.7 y 5.9, dando paso a una discusión sobre la idoneidad de este servicio en 5.10, y a las conclusiones y el trabajo futuro en el apartado 5.11.

5.2 RETRANSMISIÓN LOCAL ACTIVA

El uso de Redes Activas para la recuperación de pérdidas de paquetes dentro de la red ha sido ampliamente estudiado en [26], [34], [111]. Para reducir el retardo de la retransmisión, se lleva a cabo dentro de la red algunas de las operaciones de reparación, en lugar de extremo a extremo.

Un mecanismo ampliamente utilizado, principalmente en transmisiones en multidifusión, es la técnica de retransmisión local de paquetes ([33], [112], [110], [42]). Este esquema dispone un agente para el almacenamiento temporal en uno o varios nodos de la red, el cual retransmite los paquetes perdidos solicitados por los receptores que han

detectado una pérdida. Con el objetivo de reducir aún más el retardo se puede considerar otro mecanismo adicional. Este mecanismo consiste en la detección de la pérdida del paquete en cualquier nodo intermedio, el cual solicitará la retransmisión al agente. Este esquema se denominará RA (Retransmisión Activa), y su funcionamiento básico se ilustra en la figura 22.

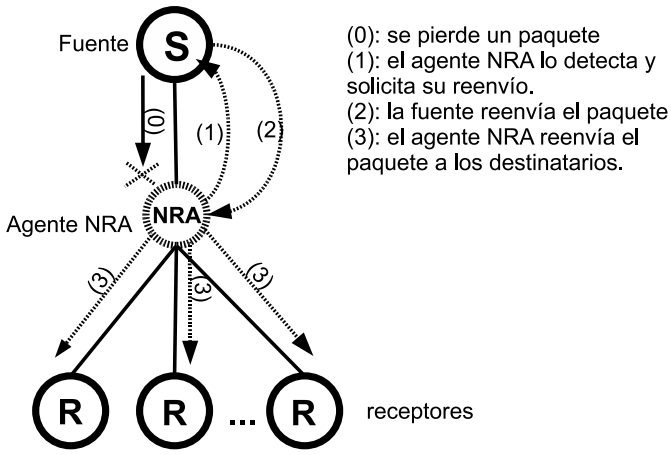


Figura 22: Diagrama de funcionamiento de la retransmisión local activa.

La ventaja de utilizar una *cache* intermedia está clara, ya que reduce sensiblemente el tiempo necesario para atender el reenvío del paquete perdido y almacenado temporalmente [110]. De la misma manera, la estrategia de efectuar la detección y la solicitud de retransmisión de forma proactiva aumenta la eficacia del sistema.

Sin embargo, en términos generales, las soluciones propuestas, tales como [30] y [110] llevan a cabo la detección de los paquetes perdidos verificando simplemente el orden de los números de secuencia de los paquetes. Este tipo de mecanismos de detección de pérdidas introducen un retardo de detección que depende del número de paquetes consecutivos perdidos, como se detallará en el análisis de la sección 5.4. Esta es la principal razón por la que dichos esquemas no son los más apropiados para VoIP en condiciones de pérdidas a ráfagas.

5.3 MECANISMOS DE DETECCIÓN TEMPRANA

En este apartado se estudian esquemas que identifiquen los paquetes perdidos para poder iniciar el proceso de reparación reactiva. La principal diferencia entre estas propuestas y las de otros trabajos similares reside en que nuestra propuesta identifica exactamente el paquete perdido, mientras que los restantes estiman la probabilidad de que se produzcan pérdidas según una estimación del estado de la red, pero sin identifican los paquetes concretos. Por ello, con los otros esquemas sólo se pueden poner en marcha medidas preventivas para tráfico posterior. De entre estos mecanismos destacan [113] o [114], en los que se predice, según un modelo probabilístico de pérdidas o de retardos respectivamente, la probabilidad estimada de ocurrencia de errores.

El objetivo de los detectores desarrollados en este trabajo es efectuar la detección tan pronto como sea posible, generando el menor número posible de falsas alarmas. Así se pretende maximizar la calidad percibida por el usuario, al reducir el retardo medio de los paquetes recuperados y el porcentaje del paquetes perdidos.

5.4 DETECCIÓN POR ALTERACIÓN DE SECUENCIA (DAS)

El esquema de *detección por alteración de secuencia* (DAS), se basa en la premisa de que las unidades de datos enviados por la fuente se numeran secuencialmente, de forma que la discontinuidad en la numeración de los paquetes recibidos se identifica como pérdida de paquetes. Es decir, se notifica que ha ocurrido una pérdida cuando el número de secuencia de un paquete recién llegado es mayor que el número de secuencia más uno del anterior paquete recibido correctamente.

Aunque las rutas en Internet no sean objeto de altas incidencias de reordenación de paquetes (debido a cambios en las tablas de encaminamiento, ejecución de algoritmos de reparación en algunos enlaces, etc, [20]), para confiar en este esquema, es necesario asumir que los paquetes

Algoritmo DAS
Cuando se reciba el paquete i*:*

```
1  if (lastReceived + 1 < i) then
2     for j = lastReceived + 1 : i − 1 : 1
3        notify(j)
4     endfor
5  endif
```

Figura 23: Algoritmo de detección por alteración de secuencia (DAS). La función $notify(j)$ alerta al mecanismo de reparación de la pérdida del paquete j.

del flujo llegarán siempre en orden, o en caso contrario se producirán falsas alarmas.

Nótese que este tipo de detectores exhibe una gran dependencia con respecto al tamaño de la ráfaga de paquetes consecutivos perdidos. Cuanto mayor es la ráfaga de paquetes perdidos, más tardía será la identificación de las pérdidas.

Por otro lado, DAS ofrece la ventaja de ser computacionalmente muy simple, ya que por cada flujo sólo es necesario comprobar el número de secuencia de cada paquete, y mantener el número de secuencia del último paquete. Además, DAS reacciona rápidamente cuando se producen pérdidas aisladas. Un ejemplo de esta aproximación puede encontrarse en [112], en la que se utiliza *cache* y retransmisión locales.

En el esquema DSA, el retardo de detección del primer paquete de una ráfaga de tamaño L_k, viene dado por la siguiente expresión:

$$d_{DAS}(L_k) = L_k \cdot t_f + \mu \tag{5.1}$$

donde t_f es el periodo de generación de paquetes del flujo monitorizado, $L_k > 0$ representa el número de paquetes consecutivos perdidos, y μ representa el *jitter* experimentado por el primer paquete del flujo que llega correctamente tras la ráfaga. Dado que el retardo de detección en DAS

depende del número de paquetes perdidos de forma consecutiva, cuanto mayor sea la probabilidad de pérdidas en ráfagas, menor es la probabilidad de recuperar dichos paquetes perdidos en un plazo de tiempo admisible. Concretamente, si por ejemplo $t_f = 30ms$, cuando el número de paquetes consecutivos perdidos sea $L_i > 10$, según 5.1 suponiendo que no hay *jitter*, el retardo de detección para alguno de los paquetes perdidos es mayor que el umbral permitido, $d_{max} = 300ms$.

Este hecho puede visualizarse fácilmente en la figura 24, donde se representa las distribuciones acumuladas de retardos de detección de paquetes, según la expresión 5.1, para distintos patrones de pérdidas, considerando $t_f = 20ms$. Las pérdidas se generan a partir del un modelo de Gilbert [97]. Recuérdese que el modelo de Gilbert se puede caracterizar mediante una ráfaga media L_i y una probabilidad de pérdidas p. Por ejemplo, se puede apreciar que para $L_i = 2$, el 87.5% de las pérdidas de paquetes se detectan antes de $60ms$, con lo que sólo un 12.5% de paquetes superan este retardo. Sin embargo, es notable que, cuando $L_i = 5$, aproximadamente el 33% de las pérdidas tardarán más de $100ms$ antes de que se pueda notificar el problema. Por último, se puede apreciar en la curva correspondiente al caso de $L_i = 20$ de la figura 24 que el 86% de las pérdidas tardarán más de $60ms$ en ser detectados, y el 46.4% de las detecciones serán inútiles, ya que habrán pasado más de $300ms$ desde su generación.

Analíticamente, el retardo de detección medio de DAS (d_{DAS}) en el mejor caso, es decir, sin contemplar retardos variables en la llegada de los paquetes, viene dada por la expresión 5.2:

$$d_{DAS} = \sum_{i=1}^{n} p^i \cdot (1-p) \cdot i \cdot t_f \tag{5.2}$$

donde p es la probabilidad de pérdidas del camino entre el emisor y el detector, y n es el tamaño máximo de ráfaga de pérdidas considerado en el cálculo.

Sin embargo, si asumimos un modelo de error más complejo, como el de Gilbert, que contempla los tamaños medios

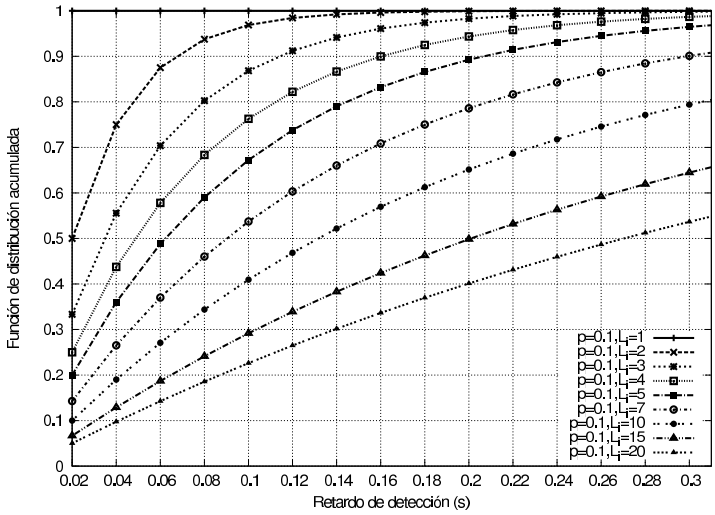


Figura 24: Distribuciones acumuladas de retardos de detecciones en DAS para distintos escenarios con distribución de pérdidas modelados con Gilbert para distintos L_i.

de ráfagas de pérdidas, el retardo medio de detección se puede obtener mediante la expresión 5.3:

$$d_{DAS} = \sum_{j=1}^{n} (\frac{1}{L_i}) \cdot (1 - \frac{1}{L_i})^{j-1} \cdot j \cdot t_f \tag{5.3}$$

Nótese que los n primeros paquetes pertenecientes a ráfagas de pérdidas de tamaño $n + m$, con $m \cdot t_f < d_{max}$, serán detectados demasiado tarde, sin posibilidad de ser retransmitidos.

5.5 DETECCIÓN POR PREDICCIÓN DE LLEGADAS

Para aliviar las deficiencias del esquema anterior, proponemos una aproximación basada en la predicción del instante de llegada de cualquier paquete para determinar si se ha perdido.

En condiciones ideales de red en los que no se añade un retardo variable a los paquetes, el receptor es capaz de estimar el instante de llegada de cada paquete. Esta estimación es posible ya que el patrón de generación de paquetes de un flujo continuo puede modelarse generalmente mediante

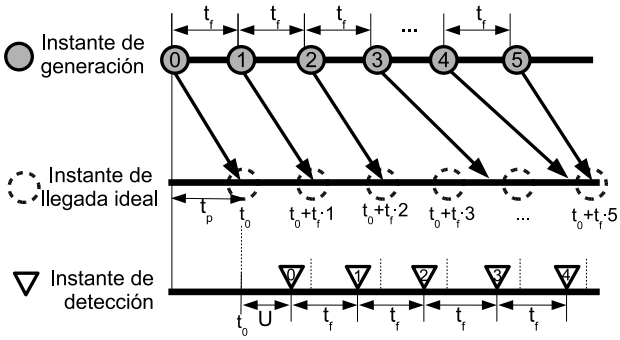


Figura 25: Efecto del *jitter* sobre los retardos de los paquetes de un flujo.

una función del periodo de generación de paquetes (t_f). Sin embargo, en una red real cada paquete puede experimentar distintos retardos aunque pertenezca al mismo flujo. Este escenario se ilustra en la figura 25, donde se aprecia cómo cada paquete, aunque generado con una diferencia de t_f unidades de tiempo, llega en un instante diferente al tiempo de llegada ideal, debido a los retardos variables introducidos en el camino.

Para considerar el *jitter* en la estimación del retardo, se define un umbral de retardo permitido U para cada paquete, que permite amortiguar las variaciones introducidas por el *jitter*. De esta manera, una vez que han llegado los primeros paquetes del flujo, se puede establecer el instante t_i máximo en que llegará el paquete i mediante la expresión 5.4:

$$t_i = t_0 + i \cdot t_f + U \tag{5.4}$$

donde t_i es el instante de llegada del paquete i—ésimo, y t_0 es el instante de llegada del paquete 0.

Mediante este esquema se puede controlar el retardo máximo de detección de un paquete, que era uno de los objetivos en la detección en flujos con retardos máximos acotados. Además, con esta aproximación, el retardo de detección no se ve afectado por el tamaño de las ráfagas de pérdidas.

Debido a que minimizar el retardo de detección es un factor crítico para los flujos de tiempo real, tanto la selección

Ajuste del valor de referencia t_0
Cuando se reciba el paquete i*:*

```
1  if (currentTime < (t0 + i · tf)) then
2      t0 = currentTime − i · tf
3  endif
```

Figura 26: Algoritmo de corrección de t_0.

del valor adecuado de U, lo más bajo posible, como la precisión del instante t_i estimado, son determinantes para el éxito de esta técnica. Veamos cómo estimar dichos valores.

Para empezar, dado que la estimación del instante de llegada depende de t_0, es fundamental que este valor se estime correctamente. Como es posible que la llegada del primer paquete, que es el que determina el valor de t_0, se haya visto influenciada por el *jitter* de la red, se hace necesario el ajuste de este valor durante el funcionamiento del detector. Para ello se puede utilizar un sencillo algoritmo, descrito en la figura 26.

En el citado algoritmo se asume que si un paquete llega antes de lo que se estimaba mediante la expresión 5.4 para el instante de llegada ideal, es porque la estimación de t_0 fue errónea. Nótese que se asume igualmente que el retardo mínimo de la ruta entre el emisor y el receptor se mantiene constante durante la transmisión del flujo multimedia (retardo t_p en la figura 25).

En el ejemplo de la figura 27 se puede apreciar cómo, mediante el algoritmo descrito, el umbral de detección disminuye durante la recepción de los primeros 258 paquetes (instante etiquetado en la figura). Esto se debe a que los primeros paquetes, a partir de los cuales se calcula el valor t_0 de referencia, llegan con retardos adicionales.

Por otra parte, es fundamental estimar adecuadamente el umbral U. El valor de U acota el retardo de detección. Si el valor de U es demasiado bajo, generará demasiadas falsas alarmas. Pero si el valor de U es demasiado alto, el retardo de detección y, en consecuencia, el retardo de reparación, puede sobrepasar el límite permitido por la aplicación. Esto

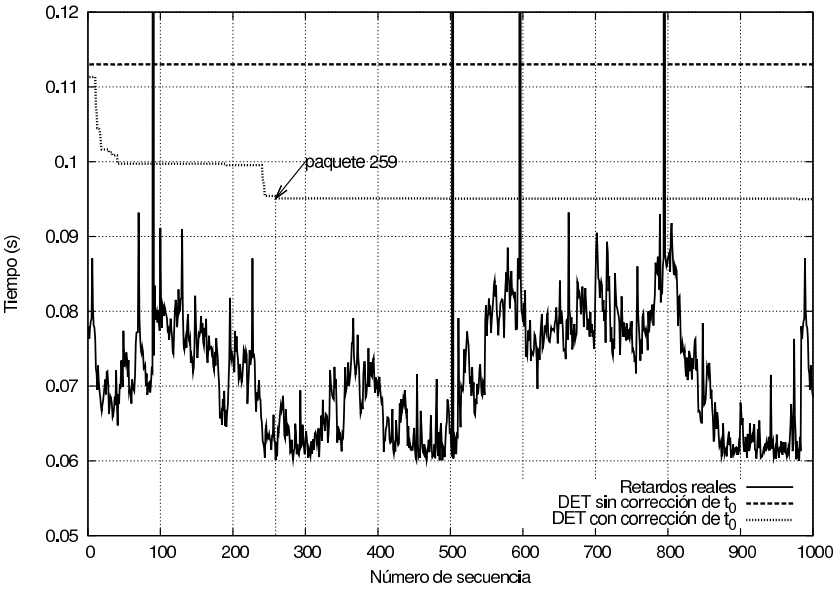


Figura 27: Cronograma donde se muestra el retardo de detección para el esquema DET con y sin corrección de la estimación del tiempo de referencia t_0.

puede traducirse incluso en la recuperación de paquetes inútiles, degradando así la calidad percibida por el usuario.

En los siguientes apartados se proponen varias estrategias para calcular el valor U (detección por umbral fijo y detección por umbral adaptativo junto a sus variantes).

5.5.1 *Detección por umbral fijo (DET)*

La técnica de *detección por umbral fijo (DET)* se basa en la utilización de un temporizador con un periodo de expiración constante. Se asume que el nodo donde se realiza la detección conoce el periodo de generación de paquetes del flujo t_f, obtenido por ejemplo mediante la información de los paquetes RTP [115], o proporcionadas por el protocolo SDP [116]. Éste es el caso más simple de detector por estimación de instante de llegada, ya que el valor de U se fija al inicio, y no se actualiza durante su aplicación.

Su funcionamiento es sencillo, y se resume en la figura 28: cuando llega el primer paquete del flujo, se establece

Algoritmo DET
Cuando se recibe un paquete:
function *receiveHandler(sequenceNumber)*

```
1  if (firstPacketReceived) then
2      t0 = currentTime
3      call timeoutHandler(1) at t0 + U + tf
4  endif
5  insert sequenceNumber into arrivedPackets
```

cada vez que expira el temporizador:
function *timeoutHandler(sequenceNumber)*

```
1  if sequenceNumber ∉ arrivedPackets then
2      call notify(sequenceNumber)
3  endif
4  call timeoutHandler(sequenceNumber+1) at
   currentTime + tf
```

Figura 28: Algoritmo de detección por estimación de tiempo de llegada (DET).

el instante de referencia t_0. Se inicializa un temporizador de detección para que expire en el instante $t_0 + t_f + U$. Con esto, se asume que el siguiente paquete llegará entre los instantes $(t_0 + t_f)$ y $(t_0 + t_f + U)$. Ante la llegada de cada paquete, el temporizador se reinicia, calculando el instante esperado del siguiente paquete mediante la expresión 5.4.

Dado que el retardo máximo tolerado para cada paquete de voz está limitado por d_{max}, el valor máximo de U está también acotado superiormente, debiendo cumplir $U < 0.300s$.

Es más, en el cálculo del valor máximo de U hay que considerar que el retardo de detección, sumado al retardo de reparación que introduzca la técnica elegida debe ser inferior a d_{max}. En este sentido, la asignación del valor U debe contemplar asimismo el retardo mínimo de propagación t_p para que el mecanismo de detección sea útil.

Para asignar un valor a U, es recomendable establecer una fase de análisis del flujo al que se va a aplicar. Por ejemplo, construyendo un histograma de retardos de paquetes, como se propone en el algoritmo de reproducción

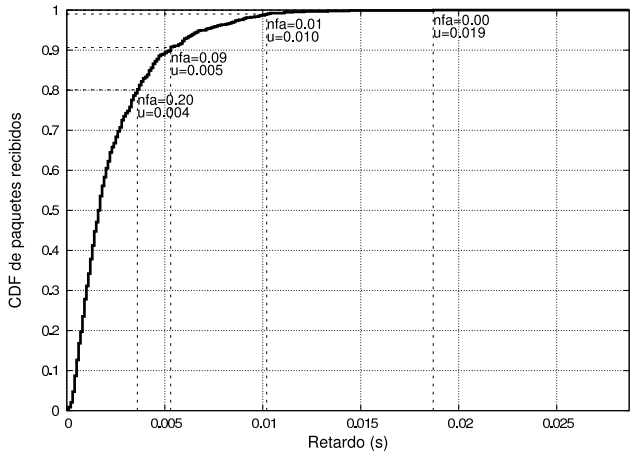
continua *Concord* ([19]), es posible asignar el menor valor a U que cumpla que se van a producir menos de nfa falsas alarmas. Para ilustrar este procedimiento, en la figura 29b se muestra el *jitter* para los primeros 750 paquetes de un flujo cuya traza ha sido obtenida según se describe en la sección 5.7.2. Además de los retardos experimentados, se trazan 4 rectas horizontales que representan distintos umbrales fijos de tiempo calculados para que se produzca como mucho un 1%, 5%, 10% y 20% de falsas alarmas. Dichos umbrales han sido calculados a partir del histograma de retardos de este mismo flujo mostrado en la figura 29a. Como se puede observar en la figura, dado un porcentaje de falsas alarmas (expresado en tanto por uno en la figura), se puede encontrar fácilmente el retardo por debajo del cual el $100 \cdot (1 - nfa)\%$ de los paquetes llegarán al detector.

Por último, se puede observar que a un valor de nfa menor le corresponde un valor de U más alto, ya que es menos restrictivo. Por otro lado, a pesar de la sencillez del esquema, su efectividad depende de la correcta estimación del umbral, que a su vez depende del cálculo del histograma, el cual debe ser actualizado periódicamente.

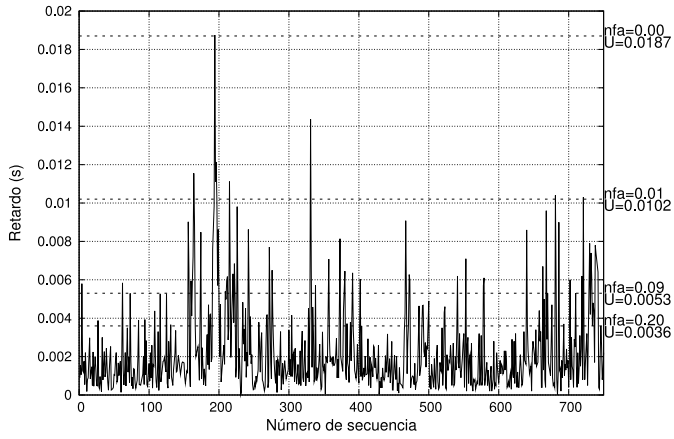
5.5.2 *Detección por umbral adaptativo (DEAT)*

Justificado por las deficiencias de DET anteriormente mencionadas, en este apartado se presentan técnicas que adaptan dinámicamente su umbral U_a, guiadas por la estimación en los nodos de detección de las condiciones actuales de la red. Estos esquemas se denotarán mediante $DEAT - X$ donde X se sustituye por el identificador del estimador utilizado. La estimación correcta de los valores de U_a implica que el mecanismo de detección disparará el procedimiento de recuperación de pérdidas en el momento adecuado.

A pesar del *jitter* de la red, dado que en este tipo de flujos continuos el periodo entre generación de paquetes consecutivos (t_f) es muy bajo, los retardos de los paquetes consecutivos exhiben una alta correlación [117]. Este hecho puede ser explotado para diseñar un predictor basado en retardos previamente observados.



a)



b)

Figura 29: (a) Selección del umbral U a partir del histograma de retardos. (b) Serie temporal de retardos y umbrales.

Dependiendo de la expresión utilizada para obtener la predicción, se pueden elaborar numerosas variantes de DEAT. Nótese, sin embargo, que este tipo de estimadores de retardos deben ofrecer estimaciones conservadoras para no generar excesivo número de falsas alarmas. Por otro lado, aunque no se especifique en cada esquema para facilitar su comprensión, los umbrales se acotarán para que su valor no sea superior al del umbral máximo U_{max}, valor máximo calculado para que no expire el tiempo de vida de un paquete.

En este trabajo estudiamos dos esquemas de predicción. El primero, utilizado como referencia, es un predictor basado en el algoritmo de estimación de los temporizadores de retransmisión en TCP, propuesto inicialmente por Van Jacobson [118], y utilizado en [16] y [119] para su uso en mecanismos de ajuste de reproducción dinámico. El segundo esquema que proponemos es un nuevo procedimiento de detección de pérdidas basado en un predictor lineal suavizado capaz de tener en cuenta las tendencias de los retardos, atajando de esta manera las deficiencias del predictor de Jacobson. Este nuevo detector se inspira en el predictor de Holt para predecir series temporales [120]. Estos dos procedimientos se detallan en las subsecciones siguientes, y se evalúan en la sección 5.7.

DEAT con predictor de Jacobson (DEAT-JAC)

Se ha seleccionado el algoritmo de Jacobson como predictor de referencia debido a su extendida utilización para estimar retardos de paquetes en TCP [121]. Nótese que en el caso de TCP, el predictor debe estimar el retardo de llegada de la confirmación en el emisor, mientras que en el presente esquema, el retardo que se estima es el de llegada de un paquete, en el receptor o un nodo intermedio.

Para aplicar el algoritmo de Jacobson [118] en el esquema de detección, se tiene en cuenta el retardo del último paquete recibido. Para estimar U_a, $DEAT-JAC$ utiliza una media ponderada del retardo de los paquetes pasados, y le añade una cantidad proporcional a la desviación observada. Concretamente, en el nodo donde se efectúa la detección, para el paquete i, el umbral U_a se calcula como:

$$\begin{aligned} \hat{d}(i) &= (1-\alpha)\cdot\hat{d}(i-1) + \alpha\cdot d(i-1) \\ e(i) &= d(i-1) - \hat{d}(i-1) \\ \delta(i) &= (1-\beta)\cdot\delta(i-1) + \beta\cdot|e(i)| \\ U_a(i) &= \hat{d}(i) + \gamma\cdot\delta(i) \end{aligned} \tag{5.5}$$

donde $d(i-1)$ denota el retardo de llegada del anterior paquete $i-1$, y $\alpha = \frac{1}{8}$, $\beta = \frac{1}{4}$, $\gamma = 4$, según la recomendación de Jacobson [118]. En el caso en que el anterior

paquete se hubiese perdido, para asegurar el correcto funcionamiento del algoritmo, se adopta la estrategia de receso exponencial (*exponential backoff*), basada en el algoritmo de Karn [122]:

$$U_a(i) = U_a(i-1) \cdot 2 \tag{5.6}$$

Esta medida pretende que DEAT-JAC reaccione adecuadamente ante falsas alarmas, recuperándose rápidamente ante una predicción errónea.

DEAT con el predictor de Holt (DEAT-HOLT)

En esta subsección se propone un nuevo predictor de retardos para mejorar la eficacia de la detección, de acuerdo a la dinámica de retardos de red esperado. Como se ha mencionado anteriormente, los retardos de paquetes consecutivos están altamente correlacionados. Asumimos además en este esquema que el retardo de cada paquete puede modelarse como una combinación lineal de una tendencia y un componente de ruido. La tendencia de creciente o decreciente de los retardos puede venir producida por las situaciones previas a la congestión, en la que generalmente se experimenta un crecimiento gradual de los retardos. La componente de ruido trata de modelar el retardo variable que se experimenta en las colas de los routers.

Asumiendo el modelado del retardo mediante ambas componentes, proponemos un suavizado lineal exponencial basado en el predictor de Holt [120].

Más específicamente, para un paquete con número de secuencia i, el tiempo de llegada estimado $D(i)$ se calcula mediante la expresión 5.7:

$$\begin{aligned} L(i) &= \alpha \cdot d(i-1) + (1-\alpha) \cdot (L(i-1) + s(i-1)) \\ s(i) &= \beta \cdot (L(i) - L(i-1)) + (1-\beta) \cdot s(i-1) \\ D(i) &= L(i) + s(i) \end{aligned} \tag{5.7}$$

donde $L(i)$ representa el nivel, y $s(i)$ indica la pendiente estimada. Dado que puede esperarse una componente ruidosa, la tendencia calculada se modificará por medio de

$(L(i) - L(i-1))$ y β. Este término se añade a la estimación previa del retardo $s(i-1)$, ponderada por $(1-\beta)$.

Para obtener una cota superior del retardo real, el umbral $U_a(i)$ se calcula tal como se indica en la expresión 5.8:

$$U_a(i) = D(i) + C \tag{5.8}$$

donde C es una componente de compensación introducida para reducir el número de falsas alarmas debidas a las fluctuaciones aleatorias de los retardos. Cuando el paquete i llega al nodo del agente de detección, C se actualiza mediante la expresión 5.9:

$$\begin{aligned} e(i) &= D(i) - d(i) \\ \delta(i) &= (1-\gamma)\cdot\delta(i-1) + \gamma\cdot|e(i)| \\ C &= \delta(i)\cdot\phi \end{aligned} \tag{5.9}$$

Los valores para α, β y γ se definen en el intervalo $[0, 1]$. Estos valores indican el peso de las muestras pasadas en las expresiones 5.7 y 5.9. Los valores concretos para cada escenario se deben obtener empíricamente.

Como nota final, cuando el paquete previo al de la detección se ha perdido, y por tanto la información de su retardo no está disponible, las expresiones 5.7, 5.8 y 5.9 no pueden calcularse. En este caso, se utilizan los últimos valores de C y D disponibles:

$$U_a(i) = D(i-1) + C \tag{5.10}$$

5.6 DETECCIÓN HÍBRIDA POR ALTERACIÓN DE SECUENCIA Y TIEMPO DE LLEGADA (DEAT-DAS)

En las anteriores secciones se ha explicó que el esquema DAS no es adecuado para escenarios en los que se produzcan pérdidas consecutivas, ya que el retardo de reparación depende de este factor.

Por otro lado, la detección basada en un umbral U de tiempo fijo acota el tiempo de detección a un valor tan alto que, para evitar falsas alarmas, puede no ser el óptimo. La

propuesta de detectores que dinámicamente ajustan el umbral para reducir los tiempos de detección alivia el problema pero, al utilizar esquemas de adaptación conservativos en pos de reducir una vez más el número de falsas alarmas, pueden no reducir al máximo el retardo de detección.

Proponemos un nuevo esquema de detección que combina la independencia de los detectores DEAT con respecto a la longitud de las ráfagas de pérdidas de paquetes con la rapidez de detección de DAS para pérdidas aisladas. Este esquema, etiquetado como DEAT-DAS, incorpora el procedimiento de detección de pérdidas (por número de secuencia a las capacidades del esquema adaptable DEAT). Por tanto, se espera que DEAT-DAS explotará las ventajas de los dos esquemas que lo componen. Suponiendo que los escenarios más favorables para el esquema DEAT-DAS será la unión de los escenarios más favorables para sus esquemas constituyentes (*jitter* bajo y pérdidas aisladas).

5.7 RESULTADOS EXPERIMENTALES I: EVALUACIÓN DE LOS DETECTORES

Tras describir los mecanismos de detección propuestos, en este apartado nuestro objetivo es evaluar y comparar los procedimientos de detección de pérdidas en términos de tiempo de detección (figura de mérito en en aplicaciones de tiempo real), y tasa de falsas alarmas generado por cada detector de pérdidas dado.

5.7.1 *Entorno de simulación*

La evaluación se ha llevado a cabo mediante simulación. En la implementación del simulador utilizado se han incorporado módulos que permiten utilizar los agentes de detección y reparación propuestos en nodos intermedios de la red.

En este paquete de experimentos se utiliza la topología mostrada en la figura 31. En este escenario la fuente genera paquetes de audio con un periodo $t_f = 20ms$. Cada paquete

Algoritmo DEAT-DAS
Cuando se recibe un paquete:
function *receiveHandler(sequenceNumber)*

```
1  if (firstPacketReceived) then
2      t_0 = currentTime
3      call timeoutHandler(1) at t_0 + U + t_f
4  endif
5  insert sequenceNumber into arrivedPackets
6  if (lastReceived + 1 < sequenceNumber) then
7      for j = lastReceived + 1 to sequenceNumber −
   1
8          call notify(j)
9      endfor
```

cada vez que expira el temporizador:
function *timeoutHandler(sequenceNumber)*

```
1  if sequenceNumber ∉ arrivedPackets then
2      call notify(sequenceNumber)
3  endif
4  call timeoutHandler(sequenceNumber+1) at
   currentTime + t_f
```

Figura 30: Algoritmo de detección por estimación de tiempo de llegada ($DEAT - DAS$).

contiene sólo una trama de audio PCM de 160 muestras de 8 bits, muestreadas a 8KHz. Como medida básica, los paquetes de audio se numeran secuencialmente.

Para calcular el tiempo de detección t_d, se asume que el nodo emisor y el nodo activo mantienen sus relojes sincronizados, y que en cada paquete se añade una marca de tiempo durante su generación. Si bien la marca de tiempo la suelen añadir los protocolos tales como RTP [108], el requisito de sincronizar todos los relojes se impone sólo para obtener las estadísticas de la simulación, y no es obligatorio en la implementación real. No obstante, dicha sincronización se puede llevar acabo mediante el protocolo NTP (*Network Time Protocol*,[123]).

Ya que la aplicación de audio interactivo impone una restricción de tiempo muy estricta, en nuestras simulaciones se

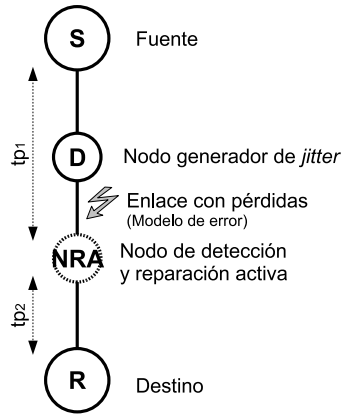


Figura 31: Topología simulada.

establecerá $d_{max} = 300ms$ como retardo máximo tolerado para un paquete de audio. Por lo tanto, si un paquete no llega a su destino en menos de $300ms$, la trama de audio ya no es válida y se descarta.

5.7.2 *Modelos de pérdidas y retardos*

En el presente estudio evaluaremos los procedimientos diseñados simulando la dinámica de la red mediante dos modelos diferenciados. No obstante, como es sabido, el modelado del tráfico en Internet es un difícil problema, debido al tremendo número de escenarios heterogéneos posibles. No existe una sola representación que pueda modelar toda la casuística de dinámicas de red [20]. Por ello, para simular distintas condiciones de pérdidas en la red, introduciremos artificialmente pérdidas de paquetes considerando un modelo de Gilbert [97], recorriendo los valores típicos de sus parámetros.

Alternativamente, para considerar distribuciones de *jitter* y retardos realistas, evaluaremos nuestras técnicas de detección aplicándolas sobre trazas obtenidas de forma experimental.

Sin la intención de ser exhaustivos, sino más bien ilustrar los comportamientos de las técnicas en distintos escenarios, se escogen cuatro trazas, que exhiben distintos patrones

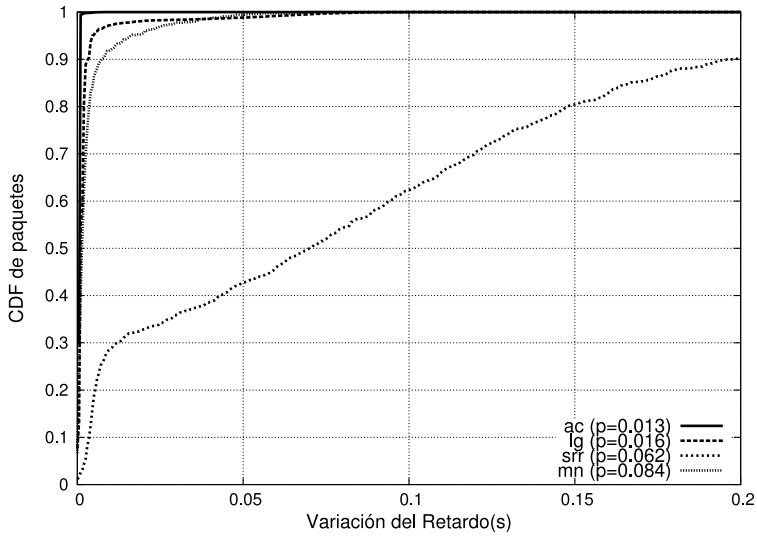


Figura 32: Histograma del *jitter*, y probabilidad de pérdidas p de las trazas ac,lg,mn y srr.

de retardo y pérdidas. Los sitios de destino se eligieron aleatoriamente de entre servidores web de diferentes localizaciones geográficas dispersas por todo el mundo.

Para obtener dichas trazas enviamos para cada destino 20000 paquetes ICMP del tipo *echo-request*, con un periodo de generación entre paquetes de 20ms, recogiendo las respuestas ICMP *echo-reply* consecuentemente [124].Finalmente, con la ayuda de tcpdump [125], y asumiendo que los tiempos de propagación eran simétricos, se calcularon las estadísticas mostradas en la tabla 6. En la tabla, las etiquetas Q1, Q2 y Q3 denotan los tres primeros cuartiles, y C_{95} se refiere al percentil 95. En la Fig. 32 se muestra más detalladamente el histograma de retardos de cada traza.

Para caracterizar las ráfagas de cada traza, la tabla 7 detalla la distribución de las longitudes de las ráfagas de pérdidas de las trazas capturadas.

5.7.3 *Resultados experimentales*

Utilizando la topología de la figura 31, se llevan a cabo una serie de simulaciones que recogen distintas condiciones del estado de la red. En dicha figura se etiqueta con tp_1 al

Traza	p	d_{avg}(ms)	Q_1(ms)	Q_2(ms)	Q_3(ms)	C_{95}(ms)
ac	0.013	0.7	0.0	0.5	1.0	1.0
lg	0.016	2.8	0.5	1.0	2.0	4.5
mn	0.084	4.1	0.5	1.5	3.0	16.0
srr	0.048	83.5	7.0	70.0	131.5	228.5

Tabla 6: Resumen de los retardos de las trazas seleccionadas.

	Longitudes de ráfagas						
Traza	1	2	3	4	5	6	7
ac	0.977	0.020	0.000	0.004	0.000	0.000	0.000
lf	0.535	0.233	0.105	0.093	0.017	0.012	0.006
mn	0.849	0.136	0.015	0.000	0.000	0.000	0.000
srr	0.719	0.219	0.000	0.031	0.031	0.000	0.000

Tabla 7: Resumen de las distribuciones de pérdidas de las trazas seleccionadas.

retardo desde el emisor de los paquetes hasta el detector NRA. En las simulaciones se elige un valor pequeño de $tp_1 = 10ms$, por lo que el retardo de los paquetes cuando llegan a NRA lo determina principalmente el *jitter* de las trazas. Para evaluar el rendimiento de los mecanismos, comprobando un número significativo de escenarios, se diseñan dos conjuntos de experimentos. En el primer conjunto de experimentos no se introduce ningún *jitter*, de forma que se estudia sólo la respuesta de los detectores ante escenarios de pérdidas, y en el segundo se aplican las distribuciones de retardos mostrados en el apartado anterior. Además, se incorpora un modelo de generación de errores que barre distintas distribuciones de pérdidas.

En ambos conjuntos de experimentos, las pérdidas sólo se producen en el sentido de la fuente a los receptores, por lo que los mensajes de solicitud de retransmisión no se ven afectados. Además, se establecen los siguientes parámetro del estimador de Holt, obtenidos mediante experimentación: $\alpha = 0.7$ $\beta = 0.7$, $\gamma = 0.7$ y $\phi = 4$.

CONJUNTO DE EXPERIMENTOS I En primer lugar se utiliza un escenario en el que el único retardo se debe al retardo de propagación tp_1, por lo que sólo se consideran las pérdidas de paquetes. Se aplica un modelo de Gilbert para generar las pérdidas entre el nodo D y el nodo NRA. En este último se encuentra el agente de detección activa. Se establece $U = 150ms$ aunque en este escenario, libre de retardos variables, sería viable asignar a U un valor cercano a 0, sin que se produjeran falsas alarmas. Se ha elegido este valor para ilustrar cómo los mecanismos adaptativos aprovechan su capacidad para reducir el retardo con respecto a la estrategia de umbral fijo.

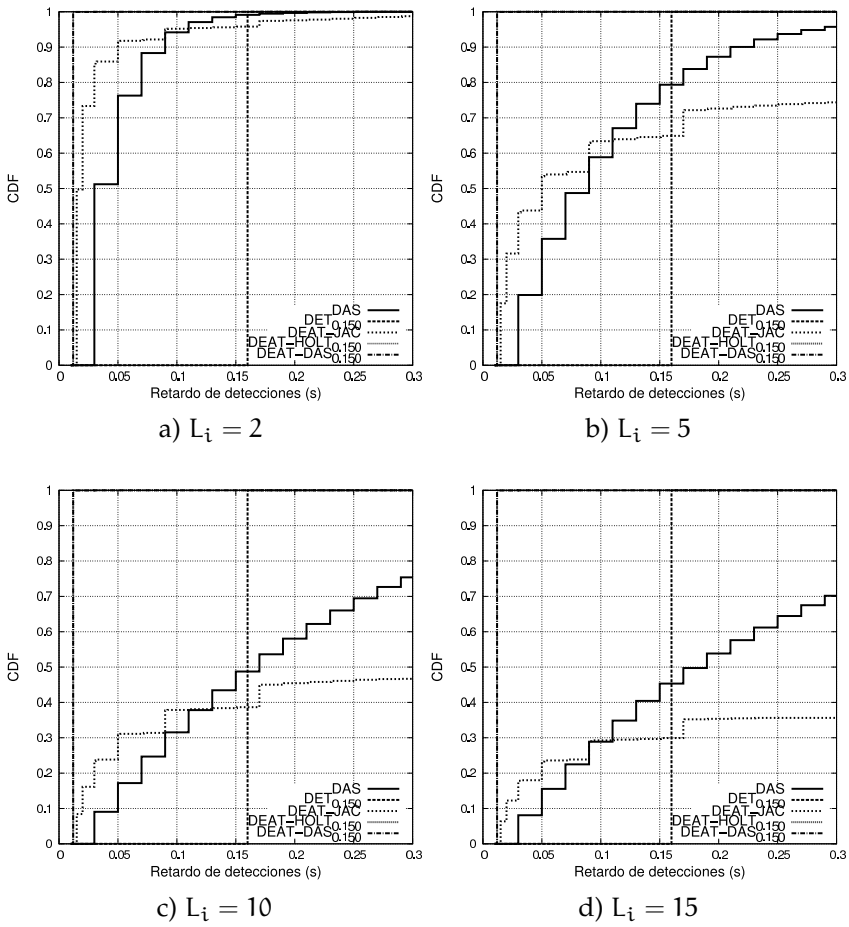
Para obtener resultados estadísticamente significativos, en cada ejecución se simula 5000s, lo que se traduce en el envío de 250000 paquetes, dado que el periodo de generación $t_f = 0.02s$.

En las subfiguras 33a, 33b, 33c y 33d se muestran los histogramas acumulados del retardo de detección para los esquemas DAS, DET, DEAT-JAC, DEAT-HOLT y DEAT-DAS. El subíndice de las etiquetas de cada método se refiere al valor del umbral U que le ha sido asignado. En la figura, para un retardo dado, cada punto de la gráfica representa el porcentaje acumulado de paquetes perdidos, expresado en tanto por uno, con un tiempo de detección menor o igual que el retardo especificado en el eje de abcisas.

Mediante esta representación de los histogramas acumulados, los distintos mecanismos de detección pueden ser evaluados y comparados, interpretando como mejor resultado aquella curva que alcance la máxima probabilidad acumulada con menor retardo, y que obtenga menos pérdidas de paquetes debido a paquetes obsoletos.

A modo de ejemplo, en la figura 33c se puede observar que el 37.8% de las detecciones se efectuaron antes de $110ms$ tanto en DAS como en DEAT-JAC. De la misma manera se observa que aproximadamente el 70% de las detecciones de DAS se llevaron a cabo antes de $250ms$, mientras que DEAT-JAC sólo detecta en torno al 45% para ese mismo retardo.

En las figuras 33c y 33d se ponen de manifiesto las deficiencias de DAS ante la presencia de ráfagas de pérdidas



a) $L_i = 2$ b) $L_i = 5$

c) $L_i = 10$ d) $L_i = 15$

Figura 33: Retardos de detección para distintas L_i, fijando $U = 150ms$ y $p = 0.1$.

largas. Nótese que incluso en ausencia de retardos variables o *jitter*, DAS no es capaz de detectar a tiempo más que el 75.4% de las pérdidas de paquetes (es decir, detectará tarde el 24.6% de las pérdidas) en la subfigura 33c ($p = 0.1$ y $L_i = 10$), mientras que para la subfigura 33b ($p = 0.1$ y $L_i = 5$), las pérdidas para DAS debidas a paquetes obsoletos era sólo del 4.3%.

En las figuras 33a, 33b, 33c y 33d, la curva de DET (que en la práctica se reduce a un histograma escalón) indica que todas las pérdidas identificadas tienen el mismo retardo de detección. Para DET, el umbral U determina la posición de la transición del 0% al 100% del gráfico. En general

se puede observar que los esquemas que consideran el retardo en su funcionamiento superan el rendimiento de DAS. Para el esquema DET esto es cierto siempre que el umbral especificado U sea menor que el retardo mínimo en el que DAS puede detectar cualquier pérdida aislada, lo que en ausencia de retardos adicionales es igual a t_f (ver la expresión 5.1).

En este caso, en el que el *jitter* es insignificante, no hay diferencia entre DEAT-HOLT y DEAT-DAS, ya que ambos logran un retardo de $t_d \approx 0$ para el 100% de las detecciones. Es por esta razón que en la figura aparecen solapadas las curvas para ambos detectores en el instante 0.010s, que corresponde al t_p de los experimentos. Es decir, el mecanismo de detección no ha introducido ningún retardo.

Adicionalmente, como se muestra en la figura 33b y 33c, DEAT-JAC se ve claramente superado por DEAT-HOLT, ya que la estrategia de receso exponencial de DEAT-JAC para reducir la tasa de FA se pone en marcha cada vez que se producen pérdidas de paquetes. A causa de ello, incluso DAS supera el rendimiento de DEAT-JAC en este escenario cuando se evalúan la tasa de detecciones efectuadas con un retardo de entre aproximadamente 110ms y 300ms.

Dado que en este escenario no hay *jitter* ni se reordenan los paquetes artificialmente, no se detecta, para ningún detector, ninguna falsa alarma en los resultados de las simulaciones.

De los resultados mostrados se extrae que DEAT-JAC, el esquema de referencia, exhibe importantes deficiencias a la hora de enfrentarse a escenarios con alta tasa de pérdidas, especialmente si son consecutivas. Este hecho es debido al receso exponencial propio de este esquema. Por tanto se hace patente la necesidad de definir otros mecanismos menos conservadores de corrección del predictor.

Como apunte final, se ve cómo los mecanismos de detección basados en predicción de llegada propuestos pueden configurarse para que no superen un cierto retardo máximo, siendo independiente de la distribución de pérdidas experimentadas.

Para considerar un escenario más realista, en el que exista una fluctuación de retardos que pueda provocar que los detectores generen FA, se simula el segundo conjunto de experimentos.

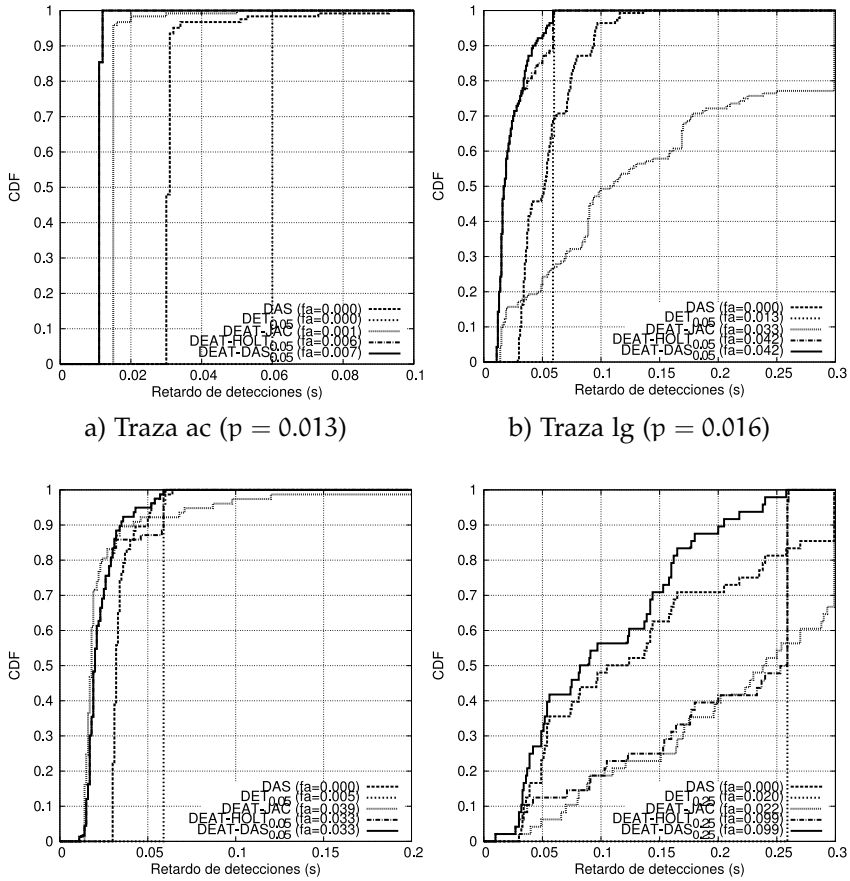
CONJUNTO DE EXPERIMENTOS II: En este caso se simulan, adoptando el contexto detallado en la figura 31, flujos de paquetes a los que se les añade en el nodo D de la topología un retardo variable provinientes de trazas reales (ver la sección 5.7.2). De esta manera se consigue evaluar el rendimiento de los detectores en escenarios con pérdidas de paquetes controlados por un modelo ajustable, y retardos que siguen un modelo más realista.

En la figura 34 se representan los retardos de detección resultantes de utilizar los distintos esquemas sobre las trazas etiquetadas como ac,lg,mn y srr, presentadas en la sección 5.7.2. En este caso, las únicas pérdidas experimentadas entre la ruta de la fuente de paquetes y el detector son las registradas en las trazas reales, descritas en el apartado 5.7.2. Las falsas alarmas en que incurre cada mecanismo están especificadas entre parántesis, en tantos por uno, en las leyendas de las subfiguras.

En este conjunto de experimentos, se puede observar la dependencia del rendimiento de los detectores, con la distribución de retardos de la traza en particular.

Para empezar con el análisis de los resultados, se puede ver que tanto DEAT-HOLT como DEAT-DAS reducen el retardo de detección con respecto a los demás detectores sin generar significativamente el número de falsas alarmas que produce DEAT-JAC, exceptuando el caso de la traza srr, en el que aumentan sensiblemente la tasa de falsas alarmas. Nótese no obstante que dicha tasa no sobrepasa del 10%, obteniendo además substanciales reducciones en los retardos de detección.

Igualmente se observa que DET, a pesar de ser una técnica estática que no es capaz de detectar pérdidas por debajo del umbral U definido, se puede usar para acotar los retardos máximos de detección, independientemente del *jitter* que se experimente. Para que DET no produzca una tasa de falsas alarmas mayor a un 2%, en 34a, 34b, 34c $U = 50ms$,



a) Traza ac (p = 0.013) b) Traza lg (p = 0.016)

c) Traza mn (p = 0.084) d) Traza srr(p = 0.062)

Figura 34: Histograma de retardos y probabilidad de pérdidas de las trazas capturadas para el experimento.

mientras que en la subfigura 34d se utiliza $U = 250ms$, debido a la dispersa distribución del *jitter* de srr (ver figura 32).

Analizando los resultados de DAS, en todas las trazas se observa que el mínimo retardo incurrido en la detección de paquetes perdidos corresponde a $t_d = 20ms$. Evidentemente esto ocurre porque para detectar la pérdida de un solo paquete en DAS es necesario esperar a que llegue el siguiente, por lo que transcurren al menos $1 \cdot t_f$ unidades de tiempo.

Las subfiguras 34b y 34d demuestran que, a pesar de las deficiencias de DAS, la técnica de detección mediante

número de secuencia pueden ser competititvas en escenarios en donde las ráfagas de pérdidas no sean de gran tamaño, incluso cuando la tasa de error es muy alta, especialmente si se compara con el rendimiento ofrecido por DEAT-JAC en ambos escenarios.

En particular, DEAT-JAC no ofrece buenos resultados en escenarios en los que la variación de los retardos es muy acusada, o la frecuencia de aparición de errores es apreciable, como ocurre en 34b y 34d. Por el contrario, en escenarios como 34c y 34d, DEAT-JAC revela un comportamiento similar a DEAT-HOLT, si bien el valor de nfa puede ser inferior gracias a las conservativas medidas de corrección adoptadas en el esquema DEAT-JAC.

En los casos en los que sí existe retardo adicional, los esquemas DEAT-HOLT y DEAT-DAS sí que ofrecen algunas diferencias. Esto se debe a que las detecciones realizadas por DAS en el esquema híbrido son consideradas en la detección por el predictor de Holt de DEAT-DAS, actualizando rápidamente el conocimiento sobre las pérdidas y los retardos experimentados hasta ese momento.

Nótese que el experimento sobre la traza srr pone de manifiesto cómo la hibridación de DAS y DEAT-HOLT dan como resultado una técnica que mejora los retardos de detección de ambas por separado. De hecho, DEAT-DAS efectúa un 40% más detecciones con un retardo igual o menor que $100ms$ con respecto a DEAT-HOLT, y casi un 10% más que DAS. Este hecho se hace más evidente cuando se examina la tasa acumulada para retardos de detección menor o igual que $200ms$, donde DEAT-DAS efectúa un 50% más de detecciones que DEAT-HOLT, y casi un 20% más que DAS.

Para esta traza, DEAT-JAC consigue una reducción de los retardos de detección comparable a la de DEAT-HOLT, alcanzando una tasa de falsas alarmas mucho menor.

Llegados a este punto, con el objeto de refinar aún más los casos de estudio de la evaluación, se añade un ingrediente adicional al escenario. Así, para controlar la distribución de pérdidas que se experimentan en las simulaciones, se habilita el modelo de Gilbert mencionado en la sección 5.7.2. En las figuras 35a, 35b, 35c, 35d, 35e y 35f se muestran varios

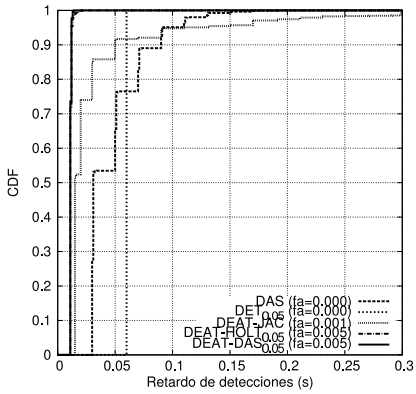
casos significativos, que ilustran el comportamiento de los detectores ante distintos tamaños de ráfagas de pérdidas.

En concreto se puede observar que, con sólo modificar el tamaño medio de las ráfagas de pérdidas experimentadas, las distribuciones de retardos difieren. Este fenómeno es más reseñable en el caso de DEAT-JAC y DAS, como puede apreciarse en las subfiguras correspondientes a una misma traza, pero con distinta distribución de errores (p,L_i). Este hecho refuerza la idea de que, tanto DAS como DEAT-JAC funcionan mejor en entornos con ráfagas de pérdidas muy cortas.

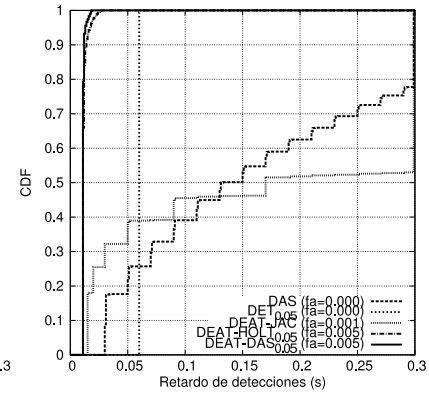
Este efecto es menos acusado para los mecanismos DEAT-DAS y DEAT-HOLT. Para ilustrar esto, véanse las figuras 35c y 35d, en las que para la misma traza mn se obtienen distintos histogramas debido a que la distribución de errores difiere. Mientras que en DEAT-JAC se pasa de detectar el 80% de las pérdidas en menos de $60ms$ cuando $p = 0.1$ y $L_i = 2$, a casi la mitad cuando L_i sube hasta 10, DEAT-HOLT sólo lo hace del 90% al 75%. Un comportamiento similar puede encontrarse para las trazas ac y lg expuestas en la figura 35.

Para resumir las peculiaridades de los mecanismos de detección mencionadas, las figuras 36a y 36b ilustran varios casos de detección, mostrando cómo el comportamiento de los distintos mecanismos. Nótese que no se muestra el umbral adaptativo de DEAT-HOLTT por claridad de la figura, ya que en este ejemplo se solaparía con DEAT-DAS. En este caso, se muestra el tiempo transcurrido entre el instante de generación de cada paquete y la estimación del momento de llegada al nodo D (o la detección de su pérdida, en su caso) para cada detector basado en predicción de llegada.

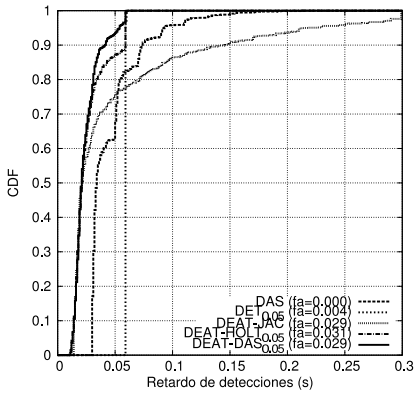
Como nota general, destacar que en DEAT-DAS la detección mediante la comprobación del número de secuencia reduce el retardo de detección de pérdidas aisladas (etiquetas c). Sin embargo, dichas detecciones no obtienen siempre el mínimo retardo. Véase por ejemplo en la figura 36b el paquete 476, que, aunque se trata de una pérdida aislada, es detectado al expirar el temporizador ajustado por el estimado de Holt.



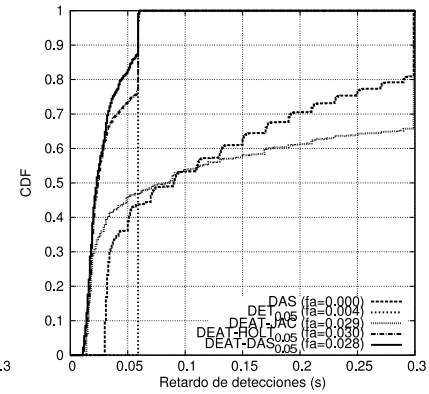
a) traza ac, $p = 0.1$, $L_i = 2$



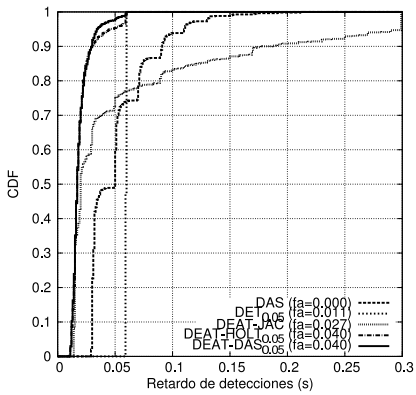
b) traza ac, $p = 0.1$, $L_i = 10$



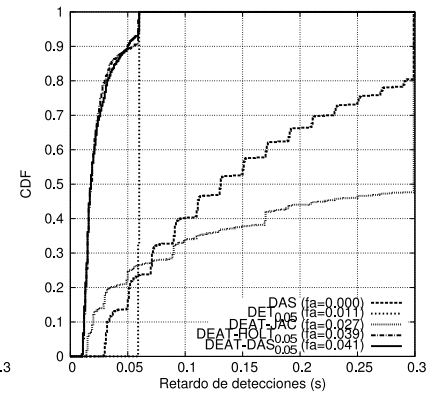
c) traza mn, $p = 0.1$, $L_i = 2$



d) traza mn, $p = 0.1$, $L_i = 10$

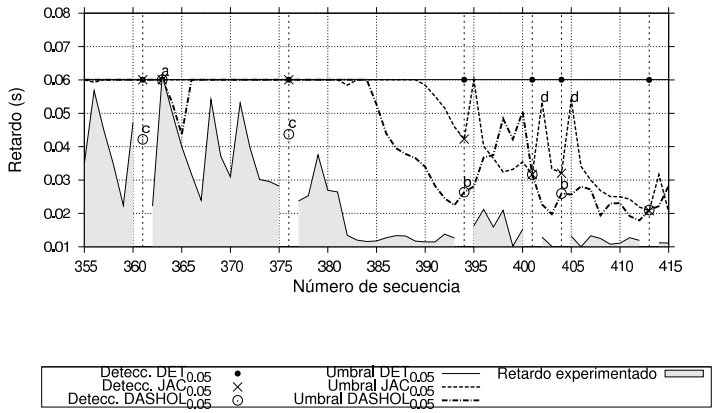


e) traza lg, $p = 0.1$, $L_i = 2$

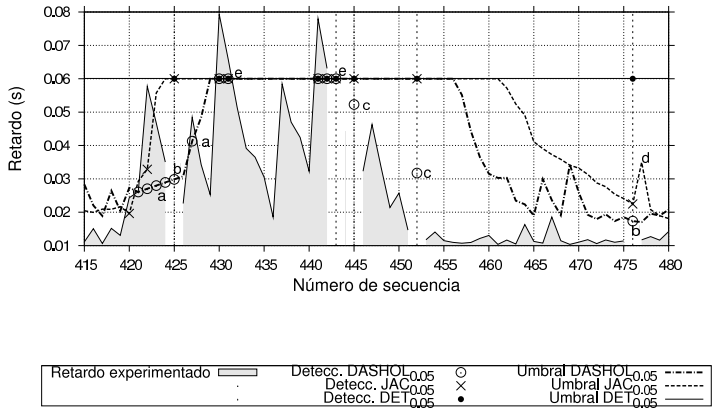


f) traza lg, $p = 0.1$, $L_i = 10$

Figura 35: Histogramas de retardo de detección para distintas trazas, con pérdidas generadas con $p = 0.1$ y $L_i = 2$ (a,c,e), y caso extremo $p = 0.1$ y $L_i = 10$ (b,d,f).



a)



b)

Figura 36: Segmentos del cronograma para traza mn. Las etiquetas a corresponden a falsas alarmas de DEAT-HOLT; las etiquetas b señalan las detecciones tempranas de DEAT-HOLT; las c etiquetan las detecciones debidas a la detección por alteración de secuencia; d señala el efecto del receso exponencial de DEAT-JAC.

Nótese asimismo en las subfiguras 36a y 36b que, cada vez que ocurre una pérdida (o una falsa alarma, no hace distinciones), DEAT-JAC agrava el retardo de estimación de llegada. Como efecto, en caso de ráfagas de pérdidas DEAT-JAC no mantendrá homogéneo el tiempo de detección para todos los paquetes.

5.8 LOCALIZACIÓN DE SERVICIOS DE DETECCIÓN TEMPRANA EN ARQ LOCAL

Para que el servicio de detección de pérdidas ofrezca la mejor calidad, y para minimizar la cantidad de recursos de la red, es necesario proporcionar la localización óptima del servicio, de forma que la calidad percibida por los usuarios sea mejorada. Este problema ha sido estudiado con éxito en [110], donde se propone minimizar la probabilidad de pérdidas tras el proceso de recuperación. Sin embargo, ése no es el único ni el más significativo parámetro que puede medir la calidad de la señal percibida en el lado receptor.

Una vez propuestos los mecanismos de detección adecuados al tráfico VoIP, en esta sección proponemos un algoritmo para la localización eficiente de servicios de detección y retransmisión local de paquetes perdidos en entornos de VoIP [110], [30]. Nuestro objetivo es el de incrementar la calidad percibida por el oyente final. Dada una ruta entre el emisor y el receptor con varios nodos candidatos a albergar el módulo de reparación, el algoritmo determinará la mejor localización del detector de paquetes perdidos considerando una medida subjetiva de calidad de la voz.

Más precisamente, usando una versión simplificada del e-model [88], el algoritmo estima la localización del agente detector de pérdidas que maximiza la calidad subjetiva esperada extremo a extremo. Para lograrlo, junto con el algoritmo propuesto se proporcionan las expresiones analíticas para la predicción de la probabilidad de pérdida y el retardo medio esperados.

El problema a resolver es el de seleccionar de forma óptima el agente de detección y su ubicación en la topología presentada en la figura 37 en la que varios nodos intermedios con el agente de ARQ activo estén disponibles, de forma que la puntuación MOS resultante será maximizada.

En la práctica, el algoritmo consiste en calcular la puntuación MOS esperada para cada tipo de detector en cada posible localización de los nodos candidatos (NRA_1, NRA_2, ..., NRA_n). Para conseguir esto, el retardo de propagación del emisor al nodo intermedio considerado (tp_1) y del nodo intermedio NRA_i al receptor (tp_2), así como las probabili-

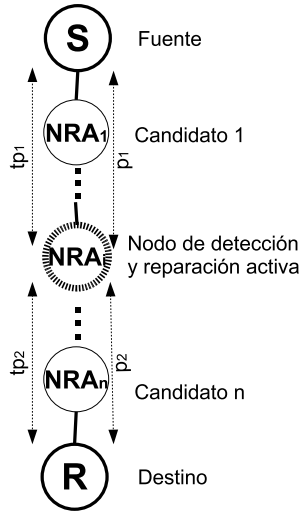


Figura 37: Topología de referencia para el análisis teórico de los detectores.

dades de pérdidas de cada parte de la ruta (p_1 y p_2 respectivamente), deben ser conocidas.

Cuando se ha estimado el valor MOS que se obtendría en cada nodo candidato, se construye una lista ordenada. Si definimos MOS_{max} como el mayor valor de puntuación MOS obtenido de entre todos los candidatos, y U_{MOS} como el nivel de degradación de la calidad tolerado, los nodos con una puntuación MOS mayor que $MOS_{max} - U_{MOS}$ son nodos elegibles. En este estudio se adopta un valor de $U_{MOS} \leqslant 0.5$, dado que mayores valores pueden degradar la calidad de la solución de forma excesiva, comparada con la solución óptima.

La selección de entre las posibles soluciones se llevará a cabo teniendo en cuenta la disponibilidad en dicho nodo, así como el coste de migrar del actual nodo activo al nuevo nodo seleccionado. De forma adicional se podrían contemplar otros criterios para seleccionar un nodo de la sublista de nodos válidos. El algoritmo debe ejecutarse cada vez que p_i ó tp_i sufra cambios significativos.

En el escenario simple mostrada en la Fig. 37, el nodo S envía los paquetes de voz al nodo R a través de varios nodos que puedan ejecutar un agente AR. El nodo seleccionado

es el etiquetado como NRA_i, donde el procedimiento de recuperación se lleva a cabo. p_1 y p_2 denotan las probabilidades de pérdidas entre S y NRA, y NRA-R, respectivamente, mientras que t_{p_1} y t_{p_2} se refieren a los retardos de propagación para los mismos tramos.

5.8.1 *Análisis de las pérdidas de paquetes y del retardo en el nodo NRA*

Dado que el algoritmo de localización propuesto se basa en la puntuación MOS estimada por el modelo *e-model*, es necesario calcular para cada posición los parámetros requeridos para el *e-model*, es decir, el retardo medio por paquete d y la probabilidad de pérdida e.

Para el escenario presentado en la figura 37 asumiremos que el máximo retardo extremo a extremo permitido a cualquier paquete (d_{max}) es $300ms$, el periodo de generación entre paquetes t_f constante, el canal de control no sufre errores, y la fluctuación del retardo es despreciable. Como se mencionó anteriormente, el enlace de S a NRA experimentará una probabilidad de pérdidas igual a p_1 y un retardo de propagación tp_1, mientras que el enlace de NRA a R mostrará una probabilidad p_2 y un retardo de propagación tp_2. En este análisis, los mecanismos de recuperación se denominarán *DAS-NRA* cuando se utilice el agente de detección *DAS*, y *DET-NRA* cuando se emplee el agente de detección *DET*.

Para el análisis se comparará la retransmisión con detección por alteración de secuencia DAS y con el detector DET. Hay que tener en cuenta que DET ofrece una cota superior para los resultados que ofrecería DEAT con el mismo umbral U. Por tanto, DET-NRA proporcionará el peor caso de cualquier variante de DEAT.

Retardo y probabilidad de pérdidas para DAS-NRA

Definamos A_i como el evento correspondiente a que se produzcan i pérdidas consecutivas en el tramo $S - NRA$, B_j como el evento correspondiente a que un paquete sea retransmitido j veces de S a NRA, y C_k, cuando un pa-

quete tenga que ser retransmitido k veces de NRA a R. La probabilidad $p(A_i \cap B_j \cap C_k)$ se puede calcular mediante la expresión (5.13):

$$p(A_i \cap B_j) = \begin{cases} 1 - p_1 & \textbf{si } i = 0 \\ p_1^{i+j} \cdot (1 - p_1)^2 & \textbf{si } i > 0 \end{cases} \tag{5.11}$$

$$p(C_k) = (1 - p_2) \cdot p_2^k \tag{5.12}$$

$$p(A_i \cap B_j \cap C_k) = p(A_i \cap B_j) \cdot p(C_k) \tag{5.13}$$

A cada ocurrencia del evento $A_i \cap B_j \cap C_k$ hay asociado un retardo $r(A_i, B_j, C_k)$, calculado mediante la expresión 5.16:

$$r(A_i, B_j) = \begin{cases} i \cdot t_f + tp_1 \cdot (3 + 2 \cdot j) & \textbf{si } i > 0 \\ tp_1 & \textbf{si } i = 0 \end{cases} \tag{5.14}$$

$$r(C_k) = tp_2 \cdot (2 \cdot k + 1) \tag{5.15}$$

$$r(A_i, B_j, C_k) = r(A_i, B_j) + r(C_k) \tag{5.16}$$

(Nótese que cuando $i = 0$, j sólo puede ser 0).

En este caso, la probabilidad extremo a extremo de que un paquete deba ser recibido antes de d_{max} cuando se usa ARQ, se expresa en 5.17:

$$p_{success} = \sum_{i=1}^{i=i_{max}} \sum_{j=0}^{j=j_{max}} \sum_{k=0}^{k=k_{max}} p(A_i \cap B_j \cap C_k) + \sum_{k=0}^{k=k_{max}} p(A_0 \cap B_0 \cap C_k) \tag{5.17}$$

para los valores de i, j y k que cumplan $r(A_i, B_j, C_k) < d_{max}$. De esta forma, la probabilidad de pérdida de $e_{NRA-DAS}$ puede expresarse según se muestra en 5.18:

$$e_{NRA-DAS} = 1 - p_{success} \tag{5.18}$$

De la expresión 5.16, los valores máximo i_{max}, j_{max} y k_{max} pueden obtenerse evaluando $r(A_i, B_j, C_k) < d_{max}$.

Con esos valores, el número de operaciones necesarios para calcular la probabilidad está acotado. Por tanto, el retardo medio $d_{NRA-DAS}$ puede calcularse como aparece en 5.19:

$$d_{DAS-NRA} = \sum_{i=0}^{i=i_{max}} \sum_{j=0}^{j=j_{max}} \sum_{k=0}^{k=k_{max}} r(A_i, B_j, C_k) \cdot \cdot p(A_i \cap B_j \cap C_k) \quad (5.19)$$

Retardo y probabilidad de pérdidas de DET-NRA

Para analizar la calificación de calidad esperada en un nodo con DET-NRA, definimos B_j como el evento correspondiente a que un paquete sea retransmitido j veces desde S a NRA, y C_k el evento correspondiente a que un paquete tenga que ser retransmitido k veces desde NRA a R para DET-NRA. La probabilidad $p(B_j \cap C_k)$ puede calcularse en nuestro caso mediante la expresión 5.22:

$$p(B_j) = (1 - p_1) \cdot p_1^j \quad (5.20)$$
$$p(C_k) = (1 - p_2) \cdot p_2^k \quad (5.21)$$
$$p(B_j \cap C_k) = (1 - p_1) \cdot p_1^j \cdot (1 - p_2) \cdot p_2^k \quad (5.22)$$

El retardo asociado $r(B_j, C_k)$ puede calcularse mediante la expresión 5.25:

$$r(B_j) = p_1 \cdot (2 \cdot j + 1) \quad (5.23)$$
$$r(C_k) = p_2 \cdot (2 \cdot k + 1) \quad (5.24)$$
$$r(B_j, C_k) = r(B_j) + U + r(C_k) \quad (5.25)$$

En nuestro caso, expresamos en 5.26 la probabilidad de que un paquete sea recibido antes de d_{max} cuando se empleen los mecanismos de recuperación:

$$p_{success} = \sum_{j=0}^{j=j_{max}} \sum_{k=0}^{k=k_{max}} p(B_j \cap C_k)$$

para valores de j y k tales que $r(B_j, C_k) < d_{max}$. Los valores máximos j_{max} y k_{max} pueden calcularse a partir de 5.25.

Finalmente, la probabilidad $e_{DET-NRA}$ se muestra en la expresión (5.26):

$$e_{DET-NRA} = 1 - p_{success} \tag{5.26}$$

El retardo medio $d_{DET-NRA}$ puede calcularse mediante la expresión 5.27:

$$d_{DET-NRA} = \sum_{j=0}^{j=j_{max}} \sum_{k=0}^{k=k_{max}} r(B_j, C_k) \cdot \cdot p(B_j \cap C_k) \tag{5.27}$$

Una vez conocidos $e_{DET-NRA}$, $e_{DAS-NRA}$, $d_{DET-NRA}$ y $d_{DAS-NRA}$ para cada nodo candidato, la selección del nodo donde situar el agente es directa, tal como se explicó en la sección 5.8. En la sección 5.9 se proporcionan resultados experimentales para validar el algoritmo presentado.

5.9 RESULTADOS EXPERIMENTALES II: LOCALIZACIÓN ÓPTIMA DE LOS SERVICIOS DE DETECCIÓN

La evaluación del rendimiento del algoritmo de localización propuesto ha sido obtenido tras ejecutar un nutrido conjunto de simulaciones. En la topología propuesta, consideraremos varios nodos programables entre el emisor y el receptor que dispongan de los mecanismos de recuperación presentados en la sección 5.2.

Se generan distintos escenarios especificando distintos valores para la probabilidad de pérdida p y el retardo de propagación t_p extremo a extremo. Además, también se selecciona aleatoriamente el número de nodos programables intermedios. De esta forma se generan distintas combinaciones de retardos parciales y probabilidades de pérdidas por enlace que concuerden con los valores t_p y p extremo a extremo asignados a cada escenario simulado, o lo que es lo mismo, $\prod_{i=1}^{i=n}(1-p_i) = 1-p$ y $\sum_{i=1}^{i=n} tp_i = t_p$.

Una vez que el escenario ha sido construido, se procede a estimar la puntuación MOS que se obtendría en cada nodo para las dos técnicas de detección de pérdida de paquetes *DET* y *DAS*, de acuerdo a las expresiones descritas en las secciones 2.6.2 y 5.8.1.

Para cada escenario diferente, y por cada combinación de localización del agente de reparación y tipo de algoritmo de detección, se lleva a cabo una simulación para resolver empíricamente el nodo y el algoritmo óptimos para la topología dada. En cada escenario comparamos las selecciones efectuadas por nuestro algoritmo y los resultados de las simulaciones.

La efectividad del algoritmo se verifica calculando la diferencia entre la máxima puntuación MOS obtenida en todas las simulaciones y la puntuación MOS obtenida en la simulación seleccionando el nodo seleccionado por el algoritmo propuesto.

5.9.1 *Parámetros de simulación*

Se generan un total de 487 escenarios donde el retardo extremo a extremo $t_p \in \{20ms, 50ms, 80ms, 100ms, 200ms\}$ y donde p toma valores desde 0.1 hasta 0.8 con incrementos de 0.1. Por cada escenario, se llevan a cabo distintas simulaciones con diferente número de nodos intermedios (de 4 a 6) distribuidos de forma aleatoria.

Para medir la relevancia de un escenario, la diferencia media entre el valor MOS esperado para cada nodo y el máximo valor MOS esperado será considerado. Cuanto mayor sea este valor, mayor será el impacto de seleccionar el nodo correcto en la calidad percibida. En concreto, el 30% de las simulaciones realizadas exhiben una distancia media mayor que 0.5, que representa una diferencia significativa en la escala MOS.

En la figura 38a, se muestra por cada escenario la función de distribución acumulada (CDF) de las distancias medias de MOS. Como se puede observar, el 50% de los escenarios presentan una distancia media de 0.25 puntos MOS,

y alrededor del 10% muestran un valor mayor que 1 en la escala MOS.

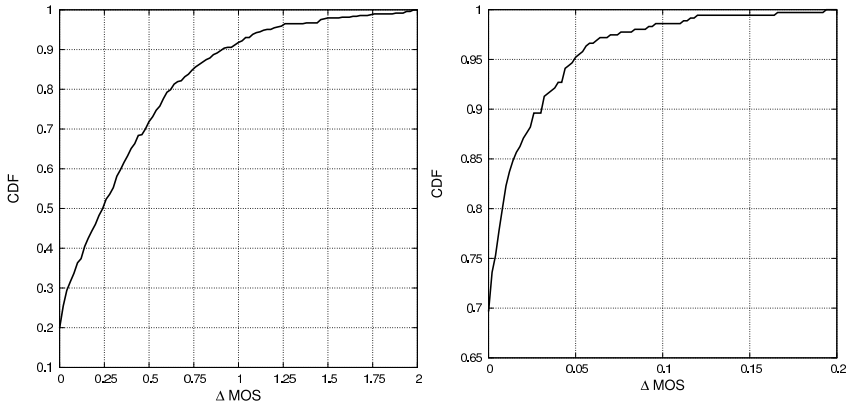
5.9.2 *Resultados de las simulaciones*

Para caracterizar el rendimiento del algoritmo de selección calculamos la distancia entre el mejor valor MOS obtenido para dicho escenario y el valor MOS obtenido en el nodo seleccionado. Consecuentemente, la diferencia entre ambas puntuaciones MOS deben ser inferiores a U_{MOS} para que el algoritmo de selección tenga algún valor, y dar un resultado cercano al óptimo.

Como resultado, en la figura 38 se muestra la función de distribución acumulada de las distancias de los valores MOS obtenidas en los escenarios anteriores. El porcentaje de diferencias inferiores o iguales a un error dado se muestra en la figura, en la que el eje vertical representa la probabilidad acumulada, y el eje horizontal representa la diferencia de puntuación MOS entre las soluciones óptima y del algoritmo.

Puede verse que el 95% de las selecciones del algoritmo obtiene una diferencia inferior a 0.05 de la escala MOS, y menos de un 1.5% de las selecciones generan un error de entre 0.1 y 0.2 puntos.

Si bien se sabe que, además de la tasa de pérdidas, los retardos influyen en la calidad percibida por el usuario, según el modelo *e*, no es menos cierto que la distribución de pérdidas resultantes impacta significativamente en la inteligibilidad del flujo de voz (ver capítulo 3). Como muestra de cómo puede impactar la selección correcta de un nodo y el mecanismo de reparación óptimo en la inteligibilidad de un flujo de voz, la figura 39 ilustra el resultado de uno de los experimentos realizados, medidos mediante el procedimiento automático presentado en el capítulo 3. Es reseñable comprobar cómo la correcta elección del mecanismo de detección y de la localización del nodo logran, para el escenario mostrado, una diferencia del 10.6%. Perceptualmente la mejora es más que destacable, ya que se pasa de $W_{acc} = 87.75\%$ a $W_{acc} = 97.75\%$.



a) b)

Figura 38: a) CDF de las distancias MOS medias de las distintas posiciones en cada escenario, y b) CDF del error entre el MOS resultante tras aplicar el algoritmo y la mejor solución obtenible.

5.10 DISCUSIÓN

Si bien los mecanismos de detección basados en la predicción de llegada sobrepasa en la mayoría de los escenarios al rendimiento de la detección por alteración de secuencia, también es cierto que introduce cierta complejidad al nodo de detección. La necesidad de disponer de temporizadores para activar alarmas que verifiquen si ha llegado el paquete correspondiente demanda recursos adicionales de los que no se requerían con DAS. Sin embargo, este tipo de requisitos está sobradamente justificado con las mejoras perceptuales que se obtienen.

Uno de los puntos controvertidos de la detección activa de pérdidas en nodos intermedios es el de la escalabilidad. Si hay que mantener información de estado por cada flujo de paquetes, parece que el procedimiento no es apropiado para nodos de redes troncales en las que un número importante de flujos recorren dichos nodos. Si bien se podría discutir la viabilidad de su implementación en dicho escenario, parece claro que su utilización en nodos pasarela de redes de acceso residencial, redes p2p de telefonía [10], o *set-top-boxes* de servicios de *triple-play* parecen escenarios

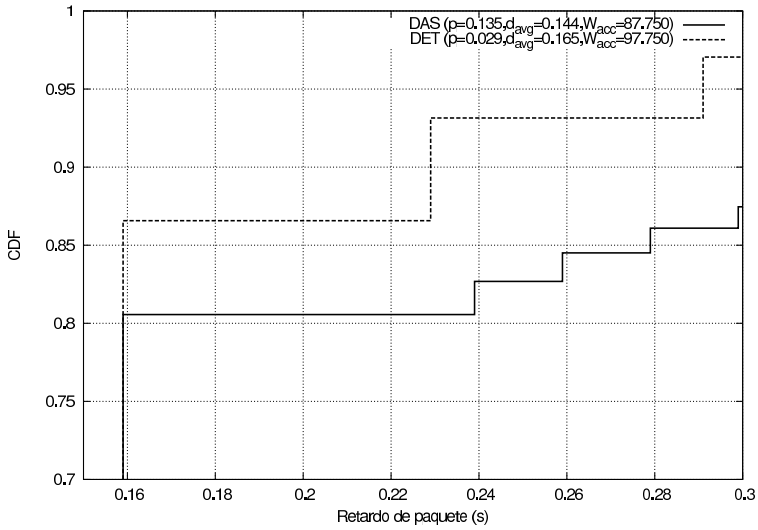


Figura 39: Distribución acumulada de retardos de paquetes, para los NRA con detectores DET y DAS.

idóneos para su implantación, dado que el número de posibles conversaciones de voz es mucho más reducido, y que su localización intermedia lo hace un sitio lógico donde ejecutar los mecanismos.

Aunque se podría pensar que los mecanismos de detección propuestos en este capítulo son similares a las técnicas de reproducción continua adaptativa de *streams* ([126], [18]), nótese que existe una diferencia fundamental entre ambos tipos de procedimientos. Mientras que la reproducción adaptativa pretende ajustar el búfer de reproducción para no dejar de reproducir paquetes debidos a su retardo variable, ajustando siempre el tiempo de reproducción por encima del retardo de los paquetes, en el caso de las detecciones se pretende detectar con el mínimo retardo los paquetes perdidos, por lo que el temporizador utilizado debería estar lo más ajustado a los retardos reales, para así identificar pérdidas con la mínima demora. El ajuste del temporizador al retardo real de los paquetes es necesario para no generar falsas alarmas, pero no para el correcto funcionamiento del detector.

Dado que las falsas alarmas dependen del escenario concreto y del mecanismo de detección elegido, en cada situación es necesario valorar el tráfico supérfluo u *over-*

head generado por las falsas alarmas de cada técnica con respecto a la calidad del servicio recibido por el usuario para poder elegir su implementación.

5.11 CONCLUSIONES

En este capítulo se han presentado varios mecanismos de detección de pérdidas basados en predecir el instante de llegada de cada paquete que, combinados con la técnica de retransmisión local, ofrecen un retardo medio de detección inferior al de los mecanismos clásicos de detección, basados en la observación de la continuidad de los números de secuencia. Aunque con estas técnicas es posible que el número de detecciones erróneas sea mayor, el aumento no es significativo si se compara con otras técnicas de adaptación de temporizadores utilizados.

Además se ha propuesto un algoritmo para seleccionar en qué nodo intermedio del camino entre el emisor y el/los receptor/es se debe activar un mecanismo de detección concreto para maximizar el valor esperado de calidad perceptual que se obtendría con su uso. En este estudio, hemos propuesto un algoritmo para ubicar y seleccionar un detector de pérdidas de paquetes basado en una medida subjetiva de calidad de voz. El algoritmo propuesto estima el retardo de recuperación del paquete considerando las condiciones de red actuales expresadas en términos de probabilidades de pérdida y retardos de propagación. En la decisión sobre la ubicación del servicio, nuestro objetivo es maximizar la puntuación perceptual que el *e-model* provee.

Tras las ejecuciones preliminares de las simulaciones utilizando el algoritmo propuesto, se muestra que la degradación introducida por los errores de estimación es menor que 0.05 puntos MOS para el 95% de los casos simulados. Por tanto se puede inferir que el algoritmo selecciona el mecanismo de selección apropiado, así como su mejor localización bajo unas condiciones de red dadas, basándose en un criterio de calidad de voz subjetiva.

Basados en los resultados experimentales y teóricos llevados a cabo, se concluye que DAS es adecuado para es-

cenarios en los que predominen pérdidas aisladas y en los que el *jitter* no sea mayor que $n \cdot t_f$, con $n > 1$, mientras que no debe utilizarse en aquellos escenarios en los que predominen las ráfagas de pérdidas. En el mejor caso, en el que las pérdidas se produzcan aisladamente, el retardo de detección mínimo que se obtiene es t_f, según la expresión 5.1

El uso de DET es el más indicado en escenarios donde el histograma del *jitter* sea conocido, cumpliendo que el $100 \cdot (1 - nfa)\%$ de los retardos sean inferiores que el U escogido. En el caso de pérdidas aisladas, si $U > t_f$, es más indicado el uso de DAS. En el caso de pérdidas consecutivas, DET supera el rendimiento de DAS, al no ser dependiente del tamaño de la ráfaga de pérdidas.

Los detectores adaptativos permiten reducir el retardo incurrido con DET, sobre todo en escenarios donde el histograma de retardos exhibe una larga cola, que exigiría para DET un valor demasiado alto para efectuar las detecciones. Ante casos de ráfagas de pérdidas, el detector DEAT-DASHOL es el más indicado, superando en mucho el rendimiento de DEAT-JAC.

Casi invariablemente, el detector DEAT-DAS iguala o supera el rendimiento de DEAT-HOLT.

El predictor presentado estima la tendencia del retardo, y obtiene una estimación robusta a pesar de los retardos aleatorios introducidos por la red por los encaminadores previos. Por tanto, con las suposiciones adoptadas, esta aproximación permite estimar el retardo máximo esperado por cada paquete puede estimarse en el nodo NRA sin desperdiciar tiempo espúreo y sin incrementar el número de falsas alarmas.

6

MECANISMOS AUTOSELECTIVOS DE SELECCIÓN Y ADAPTACIÓN DE SERVICIOS DE RECUPERACIÓN EN LÍNEA

Actualmente, los protocolos de red se diseñan para obedecer estrictas interfaces y reglas de comportamiento, ajustándose de esta manera a las pilas de protocolos bien definidas. Para los diseñadores de protocolos de red es difícil ofrecer un amplio espectro de protocolos alternativos adecuados a diversas situaciones, y conseguir con ello que la pila de protocolos evolucione para ajustarse a las nuevas necesidades.

A este respecto, la tendencia tradicional es la de diseñar protocolos que se puedan adaptar al mayor rango de casos de uso. Sin embargo, incluso los protocolos más adaptables no pueden hacer frente a todas las situaciones posibles. Cuando se alcanzan los límites de la adaptabilidad, la capacidad para conmutar a otros protocolos aparece como una alternativa deseable.

En este capítulo se propone LAIN (*Lightweight Autonomous resIlient Networks*), una plataforma que explota un conjunto de protocolos alternativos, y muestra el beneficio que supone para las aplicaciones la elección en línea del protocolo mejor adaptado. La plataforma ejecuta experimentos continuos sobre protocolos alternativos para seleccionar automáticamente aquellos que mejor se ajustan a las necesidades de la aplicación correspondiente, en las condiciones de red vigentes. En este capítulo se ofrecen resultados preliminares que evidencian la viabilidad de esta aproximación, y se justifica la necesidad de sistemas de este tipo para posibilitar en el futuro la evolución de la red y de los protocolos. En particular se muestra cómo estos mecanismos pueden ser aplicados en escenarios de reparación con requisitos de tiempo real.

6.1 INTRODUCCIÓN

Por lo general se considera que un protocolo está más evolucionado que su predecesor si se puede aplicar en más situaciones y con mejores prestaciones. En nuestra opinión, esto impone una limitación en el diseño de protocolos y servicios, llevando al funcionamiento subóptimo de la red, ya que crea una rígida barrera artificial ante mejores soluciones, en algunos casos muy especializadas. Además, la actual concepción puede suponer un lastre para la innovación, imponiendo la necesidad de decidir si diseñar contemplando el peor caso o el caso medio [127]. En ambos casos quedarán muchas situaciones sin cubrir adecuadamente, dejándose de explotar muchos protocolos útiles.

En este estudio se concibe un entorno de red donde multitud de protocolos podrán coexistir y competir entre ellos para proporcionar el mejor servicio. Algunos protocolos con un bajo rendimiento, restrictivos o incluso defectuosos en el entorno general, pueden ser eventualmente explotados, dado que pueden ser útiles en algún otro contexto. En este capítulo se presenta un entorno que permite la coexistencia de instancias de protocolos alternativos que se ejecutan en paralelo o en serie, y que son seleccionados automáticamente en tiempo de ejecución.

La plataforma propuesta está, por tanto, basada en la premisa de hacer experimentos continuos sobre protocolos que se estén ejecutando, monitorizándolos continuamente, midiendo su rendimiento, y seleccionando el protocolo más adecuado en concordancia.

6.1.1 *Selección de protocolos en línea*

La aproximación de conmutación de protocolos, de servicios o de servidores no es nueva, y su aplicación en Internet está cada vez más extendida.

Desde las capas más bajas de la pila de protocolos aparecen soluciones de respaldo en módems dependiendo de las capacidades del otro extremo y de las condiciones de la línea. A nivel de red y transporte, protocolos tales como

STUN [128], ensayan distintas posibilidades hasta que la conectividad, así como el rendimiento, concuerde con los umbrales deseados. En cuanto al transporte, a modo de ejemplo, la compresión en línea de las cabeceras puede activarse dependiendo de si la cola de TCP se encuentra lo suficientemente llena [129]. A nivel de aplicación, muchos mecanismos tolerantes a fallos se apoyan en instancias alternativas de entre las que pueden conmutar, como es el caso de servidores de nombre secundarios, o el *microrreinicio* de servidores [130].

El conmutado de protocolos puede aplicarse a pilas de protocolos enteras, como ocurre en el cambio de celda en redes inalámbricas, donde tanto el funcionamiento en paralelo de diferentes pilas, o la conmutación de pilas completas es parte del funcionamiento normal del sistema.

Sin embargo, todas esas actividades de conmutación han sido propuestas como soluciones específicas de un contexto.

En este estudio se propone una lógica de conmutación que pueda tomar decisiones a partir de la información suministrada por el entorno, utilizando técnicas aplicables a un amplio espectro de situaciones.

Para cumplir este objetivo de forma robusta, el sistema necesita monitorizar constantemente el funcionamiento de cada protocolo y compararlo con el funcionamiento objetivo deseado. De esta forma, protocolos alternativos que implementen la misma funcionalidad de manera diferente pueden competir entre ellos. El sistema seleccionará el más adecuado (o el subconjunto de ellos) para cada situación. Esta presión selectiva, combinada con la evaluación del rendimiento, tiene como resultado un sistema que siempre busca proporcionar el servicio esperado, adoptando un mejor protocolo cuando el activo no cumpla sus expectativas.

6.1.2 *Experimentos en línea y lógica de conmutación*

Una de las alternativas para seleccionar protocolos de forma dinámica consiste en ejecutar una versión en producción que proporcione el servicio requerido, o la mejor obtenida

hasta el momento, junto a una versión experimental, potencialmente mejor, y comparar continuamente el rendimiento de cada una.

La versión en producción proporciona el servicio real, y la conmutación hacia la versión experimental sólo ocurre cuando el rendimiento de ésta última pasa a ser claramente superior al del primero. Esta aproximación consume obviamente recursos extra, ya que es necesario mantener canales alternativos de prueba, lo que puede considerarse como un coste extra por obtener la robustez que este esquema provee. Sin embargo, replicar los recursos es simplemente una primera aproximación al problema. Se pueden diseñar mejores esquemas de reserva en los que el canal experimental provea una parte del servicio. La forma exacta de reservar los recursos, cómo verificar que se cumplen las expectativas, cuándo conmutar, etc., son cuestiones discutidas en este trabajo, estableciendo un marco para futura investigación y desarrollo.

La contribución de este estudio es triple: en primer lugar se describen los elementos de una red autoselectiva en la sección 6.2. A continuación, en las secciones 6.3 y 6.4 se muestran mediante simulación el funcionamiento y los beneficios que se obtienen mediante la conmutación de protocolos en dos contextos diferentes, uno a nivel de transporte, y otro en transmisiones de voz. Finalmente, en la sección 6.5 se discute la aproximación de la conmutación comparada con sistemas existentes y a la luz de futuras redes que soporten la evolución automática de protocolos.

6.2 EXPERIMENTACIÓN DE PROTOCOLOS AUTOSELECTIVOS EN LÍNEA

En una red ideal, la elección entre varios servicios equivalentes podría basarse en el conocimiento completo de los factores que determinan la calidad de los servicios para una tarea y situación dados. Este conocimiento podría obtenerse tanto por cálculos analíticos o mediante información obtenida en experiencias pasadas. Las hipótesis asumidas en este trabajo se basan en que modelar todas las condi-

ciones de la red y todos los comportamientos posibles de los protocolos será cada vez menos viable. Se asume igualmente que no se podrá recopilar todo el estado de la red.

Los experimentos de protocolos en línea son un complemento necesario a los métodos de verificación existentes, y pueden incluso utilizarse en casos donde no haya un criterio definido para seleccionar el protocolo que mejor se adapte a las condiciones de red. En esta sección se estructura el problema de la conmutación automática de protocolos, y se discute el problema de realizar evaluaciones continuas, tanto del servicio en producción como del experimental.

6.2.1 *Plataforma LAIN*

La figura 40 muestra los elementos necesarios para implementar una plataforma que realice la selección y evaluación de servicios en línea. En la parte superior de la figura se muestra una aplicación de ejemplo. A cada aplicación que utilice la plataforma se le asigna una instancia del módulo de selección de protocolos propia. Por tanto, cada instancia puede ser adaptada a los requisitos específicos de cada aplicación.

Los elementos de la plataforma LAIN son:

- *Catálogo de protocolos:* Repositorio de protocolos candidatos que pueden ser elegidos en el sistema. Dependiendo de la granularidad con la que se analice, puede consistir en pilas de protocolos completas, protocolos individuales, funciones de componentes de protocolos tales como entremezcladores o separadores de flujos, así como instancias de acceso a otros servicios.

  Para llevar a cabo la selección de componentes, los protocolos se agrupan en servicios, entendidos como un conjunto perfectamente determinado de tareas con una interfaz bien conocida. La implementación de cada protocolo candidato debe cumplir la correspondiente especificación de servicio para poder ser eligible como candidato en la ronda de ensayos. El perfil

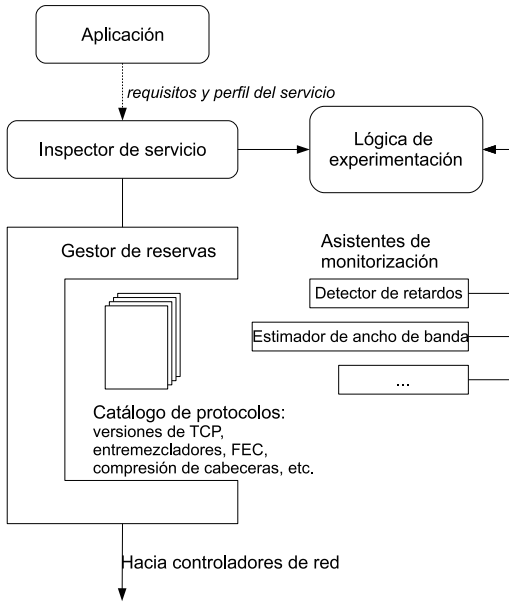


Figura 40: Arquitectura de la plataforma para la evalución en línea de protocolos y su conmutado.

del servicio incluye la definición de una interfaz de entrada, de salida y de monitorización. Mientras que las interfaces de entrada y salida establecen la interacción con la aplicación y los servicios subyacentes, la interfaz de monitorización ofrece al inspector del servicio parámetros de rendimiento (tales como el número de paquetes correctamente recibidos). El perfil del servicio establece asimismo los requisitos funcionales para el servicio, es decir, qué servicios necesita. Por ejemplo, un servicio fiable de transporte puede requerir del sistema un servicio para calcular un código de detección de errores.

- *Asistentes de monitorización:* Miden parámetros de la red y de las instancias de ejecución de los protocolos, y suministran dicha información al experimen-

tador (o módulo de lógica de experimentación). Los asistentes de monitorización son entidades dinámicas que pueden ser activadas o desactivadas según lo requiera el experimentador. Más aún, se puede introducir nuevos monitores en el sistema dinámicamente, siempre que proporcionen una descripción adecuada de su interfaz.

- *Inspector del servicio:* Verifica si un servicio dado cumple con la demanda de la aplicación, notificando al experimentador cualquier desviación. Para ello, el inspector utiliza una función de idoneidad, o función de fitness, que mide el rendimiento de cada función de red o protocolo. Dicha función puede consistir en una combinación de componentes objetivos (objetivos y parámetros de rendimiento observables) así como subjetivos (calidad de servicio percibida). Dado que los criterios de la evaluación del servicio están determinados por la aplicación particular que requiere el servicio, así como de las preferencias del usuario, la función de idoneidad la debe proporcionar el solicitante del servicio. La función de idoneidad puede hacer uso de cualquier parámetro monitorizado para calcular su valor, así como funciones de utilidad que estimen la calidad de experiencia (QoE) que el usuario percibiría (por ejemplo, el modelo E para servicios VoIP [78]). El inspector del servicio puede ser pasivo (simplemente observa los datos del flujo) o activo, (colaborando con entidades externas y/o añadiendo sus propios campos de protocolo a los datos de la aplicación).

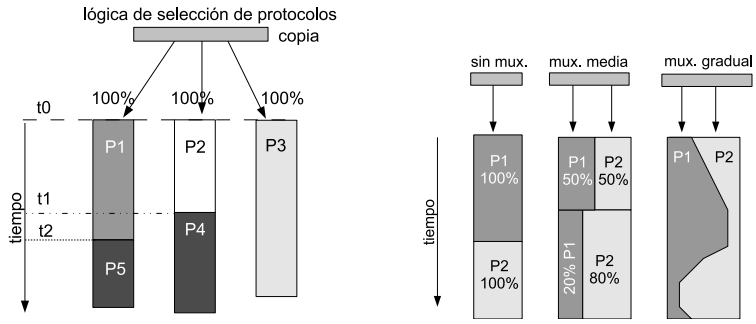
- *Lógica experimentación:* Evalúa los protocolos activos de acuerdo a la realimentación proporcionada por el inspector del servicio y los asistentes de monitorización, y toma las decisiones de cómo configurar los experimentos activos consecuentemente. Otras posibles acciones incluyen activar, desactivar, reiniciar o reconfigurar protocolos con diferentes parámetros. La aplicación puede definir, en el perfil del servicio, el porcentaje de tiempo máximo del servicio que se

puede emplear para los ensayos. De esta forma, la lógica de experimentación puede detener un test que incumpla las restricciones impuestas.

- *Gestor de recursos:* Aplica las reservas de recursos que cada protocolo requiere, observando las políticas de reserva que se definan, tales como las cuotas de recursos máximas permitidas por protocolo. Los recursos gestionados pueden ser propios del nodo en el que se ejecuta, como por ejemplo cantidad de memoria, o incluso recursos de la red, tales como las tasas de envío permitidas. Para llevar a cabo las reservas cuando sea necesario, cada protocolo puede declarar qué recursos necesita.

6.2.2 *Inspección y decisión en tiempo de ejecución*

En el núcleo de la plataforma se encuentra la lógica de experimentación, la cual modifica la configuración de los experimentos basándose en la realimentación proporcionada por los asistentes de monitorización y el inspector del servicio. Las aplicaciones demandan ciertos servicios, bien sea seleccionando el inspector apropiado de entre las categorías predefinidas, o especificando un perfil de servicio deseado (retardo, tasa de pérdidas, tiempo de descarga, ancho de banda, u otros parámetros ponderados según su influencia sobre el resultado objetivo). Esta especificación recuerda a otras aproximaciones de especificación de QoS existentes [131]. La diferencia más destacable reside en que el experimentador hace lo que está en su mano para alcanzar los objetivos deseados, probando múltiples alternativas, y sin emplear un sistema de control de admisión o similar. El experimentador comienza el servicio con una selección tentativa de al menos un protocolo para el servicio de producción y, opcionalmente, uno o varios experimentales. El protocolo en producción es típicamente el más maduro de entre las opciones existentes, o el que se estime más adecuado para una aplicación, según el historial previo. Durante el funcionamiento del protocolo seleccionado, el experimentador monitoriza la *meteorología de la red* [132] y



(a) experimentos en paralelo (b) experimentos en serie

Figura 41: Diferentes casos de experimentos en paralelo (a) y en serie (b).

conmuta proactivamente entre protocolos, evaluando periódicamente si el rendimiento es competitivo y, en tal caso, conmutando al competidor. Es posible también usar dos o más variantes en paralelo, y ajustar la tasa de cada variante paulatinamente.

Se conciben dos modos de operación: en serie y en paralelo. En la figura 41 se muestra cómo los experimentos en paralelo y en serie se pueden realizar. La figura 41(a) muestra un típico experimento en paralelo: en el instante t_0 el experimento comienza con tres protocolos en paralelo: P_1, P_2, y P_3. Cada protocolo recibe una copia completa del tráfico, obteniendo una redundancia que permite ofrecer cierta tolerancia a fallos. Si uno de los protocolos funciona incorrectamente, alguno de los restantes puede proporcionar aún el servicio. En el instante t_1 el experimentador decide que es mejor reemplazar P_2 por P_4, y en el instante t_2 el protocolo P_1 es reemplazado por P_5.

En la figura 41(b) se muestran tres alternativas para los experimentos en serie: el primero (a la izquierda) es el caso más simple, donde sólo un protocolo se usa en cada momento, y el experimentador conmuta entre protocolos, en respuesta a las observaciones de los monitores y el inspector de servicio. En el segundo caso (subfigura del centro) se lleva a cabo una multiplexación de protocolos sencilla, en la que sen envía parte del tráfico mediante un protocolo, y el resto con el otro. Las particiones de los distintos proto-

colos varían bruscamente. El tercer caso (el de la derecha) muestra un esquema de multiplexación gradual, en donde el porcentaje de tráfico que se asigna a cada protocolo se ajusta dinámicamente.

La elección de los periodos de observación y de conmutación son cruciales: observaciones demasiado cortas no proporcionan suficiente información estadística, mientras que observaciones demasiado largas proporcionan una retroalimentación demasiado tardía como para tomar decisiones de conmutación.

Más aún, un periodo de test demasiado corto no deja suficiente tiempo a los nuevos protocolos para inicializarse y adaptarse al nuevo entorno de ejecución, no aprovecha el tiempo de transición requerido, y podría llevar a fluctuaciones de rendimiento indeseados. Por contra, un periodo de conmutación demasiado largo podría traducirse en una adaptación lenta.

La duración de un test individual puede acotarse a partir del perfil de requisitos de la aplicación, en la que el usuario puede establecer el umbral máximo de degradación del servicio durante la fase de test. Esta cota puede variar dependiendo de la importancia y prioridad del servicio requerido. Por supuesto, la duración mínima del test debe ser superior al retardo de inicialización del protocolo. La duración máxima del test la acotará el usuario. Para determinar la duración óptima de la ejecución individual de un test es necesario medir el tiempo que un protocolo necesita para estabilizar su rendimiento, para poder así obtener una caracterización fiable. La estabilización del protocolo podría detectarse mediante análisis estadísticos. Por ejemplo, el test de Kolmogorov-Smirnov [133] podría emplearse para verificar si muestras pasadas y actuales pertenecen a la misma distribución estadística, estableciendo así que el impacto de la ejecución del protocolo se mantiene estable. De forma similar, este tipo de herramientas estadísticas podrían utilizarse para determinar si las condiciones de la red han cambiado o no.

6.3 CASO DE ESTUDIO I: CONMUTACIÓN ENTRE PROTOCOLOS DE TRANSPORTE PARA UNA APLICACIÓN DE TRANSFERENCIA DE FICHEROS

En esta sección se lleva a cabo una serie de experimentos mediante simulación que demuestran la utilidad de la conmutación dinámica entre protocolos. En este caso, el estudio se ha centrado en una aplicación de transferencia de ficheros. El objetivo es seleccionar el protocolo de transporte que mejor se adapte a las condiciones de la red, tal y como se describe en la sección 6.3.1. Para comenzar, se mostrará un escenario simple en el que la conmutación entre dos protocolos de transporte consigue un mejor rendimiento global. Para ello, en la sección 6.3.2 se muestran dos escenarios de referencia. En cada escenario, sólo uno de los dos protocolos consigue su mejor resultado, mientras que el competidor obtiene un resultado deficiente. En el apartado 6.3.3 se presentará un escenario mixto en el que se supone que se dispone del total conocimiento de las condiciones de la red, y donde la aplicación del esquema de conmutación obtiene un mejor rendimiento.

Finalmente, para mostrar la viabilidad de un efectuar la conmutación de forma autónoma, en la sección 6.3.4 se implementa un diseño simple de la plataforma LAIN, donde los ensayos en línea son iniciados mediante un procedimiento automático.

6.3.1 *Configuración del entorno de experimentación*

El siguiente experimento se implementa en el simulador *ns2* [102]. El experimento consiste en la realización de varias simulaciones en diferentes escenarios. La aplicación de transferencia de ficheros adoptada es capaz de seleccionar como protocolo de transporte tanto TCP [121] como BUDP, una alternativa basada en UDP [14] que incluye un mecanismo de retransmisión automática, y que hemos denominado transferencia *UDP en bloque* (Bulk UDP). En nuestro caso, tanto TCP como BUDP ofrecen el mismo servicio de entrega fiable de los datos.

En estos experimentos se emplea la conmutación simple en serie, donde la lógica de selección escoge sólo un protocolo cada vez (véase la Fig. 41(a)). Nótese que cada vez que se efectúa una conmutación es necesario reiniciar el protocolo de transporte. Esto significa que, cuando se conmuta desde TCP, se ha de cerrar la conexión, con lo que es necesario realizar una reconexión tras la conmutación a este protocolo. Este procedimiento de reinicio vacía la memoria intermedia de envío de TCP, y libera los recursos que mantenía. Por tanto, cada conmutación conlleva una nueva reserva de recursos, la configuración de parámetros y la liberación de recursos. Nótese que se trabaja con protocolos existentes tales como TCP. Por tanto, no es necesario reimplementarlos o cambiar su comportamiento para usarlos en LAIN. Con esta aproximación se define un marco versátil y aplicable sin necesidad de modificar los protocolos existentes o futuros: el procedimiento para la conmutación puede estar desacoplado del funcionamiento de los protocolos, es decir, tratarlos como cajas negras, utilizando como indicadores la realimentación del cliente del servicio, o su impacto en la red.

El protocolo BUDP propuesto, basado en UDP para transportar los datos, no contempla mecanismos de control de congestión, por lo que envía a una tasa de transmisión fija. Básicamente envía en bloques los datos que aún no se han recibido en el destino. El protocolo y la interacción entre cliente y servidor pueden describirse como sigue:

1. El cliente envía una solicitud especificando el grupo de octetos que necesita para completar el fichero (nb_i octetos en el ciclo de solicitudes número i).

2. El cliente estima entonces el tiempo que requiere para obtener todos los paquetes (t), basándose en el retardo de propagación extremo a extremo (t_p), la tasa de transmisión del emisor (r), y el número de paquetes requeridos (n).

3. El servidor envía los bytes solicitados en bloque, a una tasa r, encapsulados en paquetes de m bytes.

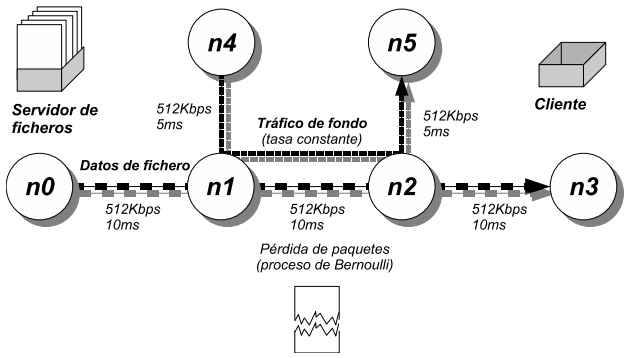


Figura 42: Escenario simulado.

4. Si el fichero ha sido completamente recibido (es decir, la lista de bytes pendientes está vacía), la etapa de transmisión del fichero finaliza, y el protocolo termina.

5. Cuando el tiempo de transmisión esperado t expira, el cliente *reinicia* el ciclo y vuelve al primer paso.

Es importante resaltar que se han escogido TCP y BUDP como un simple ejemplo para ilustrar el concepto de conmutación autoselectiva de protocolos. Ya que ambos protocolos se comportan de forma radicalmente diferente, los efectos de conmutar entre ellos puede ser claramente observado. El propósito del presente estudio no es el de analizar el rendimiento de TCP, ni de proponer el uso general de un protocolo tan simple como BUDP como alternativa a TCP para determinados escenarios: el objetivo de este estudio es demostrar que un entorno de selección dinámica de protocolos puede ofrecer ventajas con respecto a las pilas de protocolos estáticas.

6.3.2 *Rendimiento individual de los protocolos*

En esta sección se estudia el rendimiento de los protocolos TCP y BUDP por separado, en los mejores y peores escenarios para cada protocolo. Mientras que TCP es el más adecuado para las situaciones más frecuentes en Internet, no está diseñado para adaptarse a todas las situaciones. Por ejemplo, es un hecho bien conocido el que TCP exhibe un

comportamiento deficiente en determinadas circunstancias, como es el caso de una alta probabilidad de pérdida de paquetes, y de retardos extremo a extremo altos. En casos en los que no se produce congestión y se dispone de un ancho de banda constante, el uso de un protocolo sin control de congestión ni retransmisión automática de paquetes puede llevar a un mejor resultado en cuanto a la demora de los paquetes.

Este caso puede ilustrarse mediante un escenario en el que se produzcan pérdidas de paquetes que no se deban a la congestión, caso que se muestra en el subapartado de *Escenario con pérdidas periódicas de paquetes*. Por otro lado, en el posterior subapartado *Escenario con periodos de congestión* se mostrará cómo una red congestionada no es el mejor contexto para utilizar un protocolo que no se adapte al ancho de banda disponible, como es el caso de BUDP.

Escenario con pérdidas periódicas de paquetes

Este experimento se puede describir mediante el escenario mostrado en la figura 42. En el nodo $n0$, un servidor de ficheros envía al cliente (en el nodo $n3$) fragmentos de 500 bytes de datos, a una tasa de envío de 512Kbps. En las simulaciones, no se considera la sobrecarga producida por las cabeceras introducidas por el protocolo de transporte. El tamaño total del fichero a enviar es de $3 \cdot 10^6$ bytes. Cada enlace de la topología tiene una capacidad de 512 kbps. El retardo de propagación para cada enlace es de 10ms. En el enlace intermedio $n1 - n2$, se produce la pérdida de paquetes con una probabilidad de pérdidas constante p durante cada simulación. En este escenario no existe tráfico adicional entre los nodos $n4$ y $n5$.

Se llevan a cabo 50 simulaciones utilizando TCP, y otras 50 simulaciones utilizando BUDP como agentes de transporte. Como resultado, la figura 43 muestra el tiempo medio para transmitir los $3 \cdot 10^6$ bytes del fichero en función de la probabilidad de pérdidas del enlace, tanto para las ejecuciones de los protocolos TCP como las de BUDP.

Puede verse en la figura 43 cómo el tiempo necesario para entregar el fichero completo en el caso de la transfe-

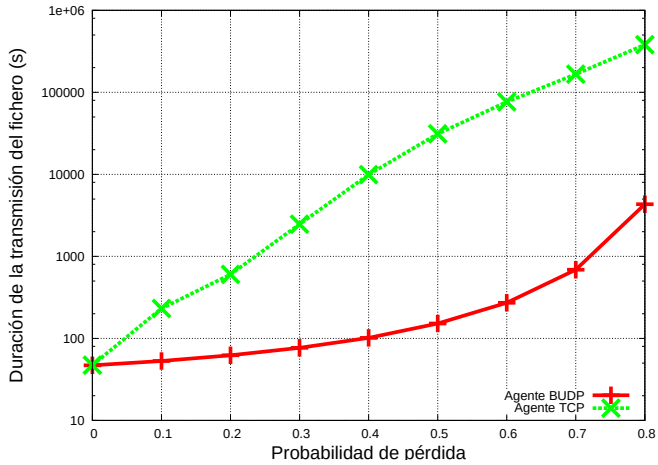


Figura 43: Duración de la transmisión frente la probabilidad de pérdida.

rencia TCP crece aproximadamente de forma exponencial con la probabilidad de pérdida de paquetes. Esto se debe principalmente al esquema de control de congestión de TCP. Cuando expira el temporizador de retransmisión, la ventana de congestión se reduce típicamente a la mitad. La ampliación de la ventana no se produce hasta recibir la confirmación de los segmentos correctamente enviados, y por tanto, si se da una alta probabilidad de pérdidas, la ventana de congestión mantiene su tamaño mínimo durante casi toda la simulación.

Como resultados destacables de este experimento, en la Fig. 43 se puede ver cómo la duración de la transferencia del fichero con el protocolo BUDP es muy inferior al de TCP. Mientras que TCP excede los 150 segundos cuando $p = 0.1$, BUDP no llega a los 152 segundos hasta que $p = 0.5$. Incluso para los casos más extremos de $p = 0.7$, BUDP transfiere el fichero en menos de 690 segundos, mientras que TCP necesita más de 240 veces este valor.

El rendimiento de BUDP en este escenario se debe principalmente a su modo de operación: el emisor no necesita esperar confirmaciones positivas para enviar el siguiente bloque de paquetes. Por otra parte se obtiene que, para un mismo valor de p, tanto TCP como BUDP retransmiten un número similar de paquetes.

Por tanto, este escenario es el más adecuado para el protocolo BUDP. En la siguiente sección se mostrará qué ocurre en un escenario sin errores en el enlace común, en el que sí se genera tráfico adicional.

Escenario con periodos de congestión

En este escenario, las únicas pérdidas de paquetes se producen por la congestión del encaminador $n1$ (ver Fig. 42). En el esquema de la figura 42 se puede apreciar que el tráfico adicional definido entre $n4$ y $n5$ será el causante de la congestión.

Durante cada la simulación se genera tráfico de fondo entre $n4$ y $n5$ con una tasa de bits constante para consumir un determinado porcentaje del ancho de banda, especificado en cada experimento. El tráfico adicional generado atraviesa el enlace $n1 - n2$, y provoca que el encaminador $n1$ no sea capaz de procesar todo el tráfico entrante (paquetes de datos del fichero y tráfico de fondo), llegando a descartar paquetes de ambos tipos de tráfico.

Por cada combinación de protocolo de transporte (BUDP o TCP) y tasa de tráfico de carga, se ejecutan 50 simulaciones. Las tasas de carga de tráfico adicional se configuran para consumir un porcentaje de ancho de banda del enlace compartido, que va desde el 0% al 100%. Como resultado de esta batería de pruebas, en la figura 44 se puede ver para cada protocolo la duración de la transmisión frente al porcentaje de ancho de banda del canal disponible. Nótese que el ancho de banda disponible estimado, notado como B_a de aquí en adelante, se calcula como la diferencia entre la capacidad del enlace y la tasa de bits del tráfico de carga y, aunque el ancho de banda disponible calculado sea cero, algunos paquetes de la transferencia de ficheros pueden ser entregados correctamente, provocando el descarte de paquetes del tráfico de carga. Al contrario que en las pruebas del apartado anterior, en este escenario no se obtienen diferencias significativas en cuanto al tiempo medio de transferencia del fichero con cada protocolo.

Sin embargo, como puede observarse en la figura 44, el rendimiento del protocolo BUDP cae cuando examinamos

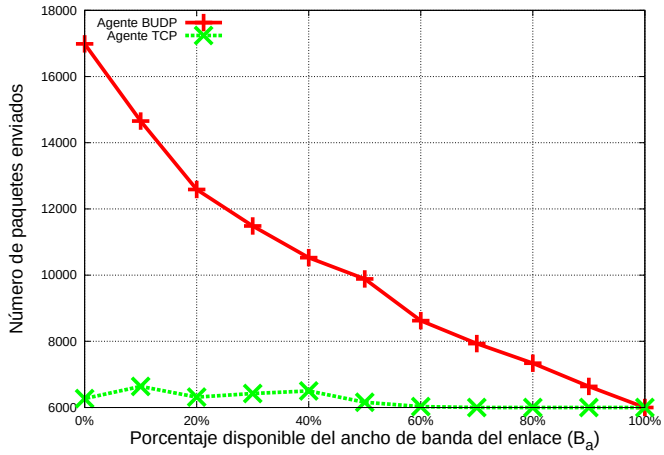


Figura 44: Número de paquetes enviados frente al porcentaje de ancho de banda disponible en la ruta (B_a).

el número de paquetes requeridos para completar la transmisión del fichero. Como se ha mencionado anteriormente, la falta de una estrategia adaptativa de envío o un procedimiento de control de congestión hace que BUDP sea inadecuado para los casos de congestión. En esta situación, BUDP agravaría la congestión, provocando así el descarte de un mayor porcentaje de paquetes.

En la figura 44 también puede observarse que en este escenario, TCP incurre en una sobrecarga inapreciable: mientras que BUDP envía más del doble del tamaño del fichero cuando el tráfico de carga utiliza el 80% de la capacidad del enlace, TCP sólo requiere retransmitir 314 paquetes adicionales. De hecho, en las simulaciones realizadas, el número máximo de retransmisiones de TCP es de 637 paquetes.

6.3.3 *Rendimiento de la conmutación de protocolos*

Los anteriores experimentos han mostrado que cada uno de los protocolos propuestos es preferible al otro según las condiciones de la red. Además, se ve en este caso cómo el rendimiento de cada uno de los dos protocolos es muy

deficiente en el escenario en el que su oponente obtiene mejores resultados.

Esta sección estudia la máxima mejora que puede obtenerse mediante la conmutación en condiciones ideales, es decir, suponiendo que se conoce instantáneamente cuándo se da un cambio en las red, comparado con el rendimiento que ofrece cada protocolo individual. De esta manera, evaluamos la viabilidad del conmutado autoselectivo, sin tener en cuenta la influencia de los algoritmos de monitorización, evaluación y decisión de conmutar.

Con este fin, el conmutado se efectuará adoptando unas hipótesis de funcionamiento ideales, es decir, el sistema tiene un conocimiento global de las condiciones de la red y del comportamiento de cada uno de los protocolos candidatos, e instantáneamente conmuta al protocolo más adecuado cuando esas condiciones cambian.

En este caso, el tamaño del fichero es de $15 \cdot 10^6$ bytes (30000 paquetes de 500 bytes cada uno), y se establece la tasa de envío a 512kbps. En este escenario, detallado en la figura 42, tanto el tráfico de carga como las pérdidas en el enlace $n1 - n2$ serán simuladas.

Durante los segundos 100 a 200, y 300 a 400, el tráfico de carga se inyectará a una tasa de $(1 - B_a) \cdot 512 = 460.8$kbps, así que sólo 51.2kbps estarán disponibles durante dicho periodo ($B_a = 0.1$). Durante el resto de la simulación, no se generará ningún tráfico adicional, aunque en el enlace $n1 - n2$ se inducirán pérdidas con una probabilidad $p = 0.3$ (tal como se indica en las figuras 45 y 46).

En las figuras 45 y 46 se muestra cómo la conmutación de protocolos puede producir una solución de compromiso para las condiciones cambiantes simuladas. En este caso en particular, la duración de la transmisión, al igual que el número de paquetes enviados para completar la transferencia, son cercanos a los mejores por cualquiera de los dos protocolos considerados.

Durante los periodos de congestión (desde el segundo 100 al segundo 200, y desde el segundo 300 al 400) se selecciona el protocolo TCP. Durante los periodos de pérdida de paquetes sin congestión, se activa la aproximación BUDP en su lugar. Como se puede observar en la Fig. 45, mien-

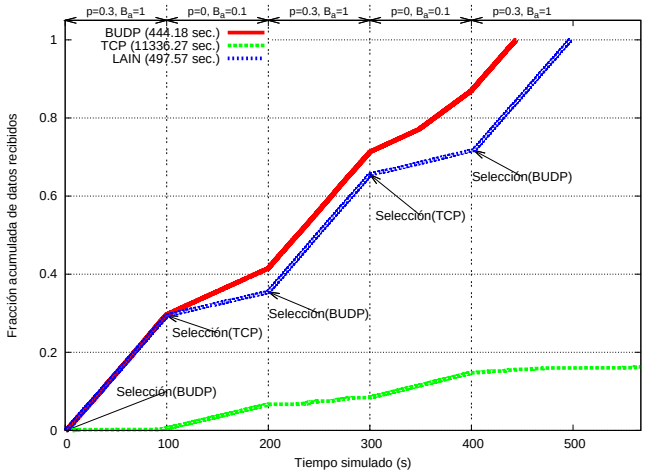


Figura 45: Evolución de la fracción del fichero recibido.

tras que la transmisión con TCP puro necesita 11336.27 segundos para entregar el fichero completo, el esquema LAIN finaliza a los 497.6 segundos. Nótese que, si bien el protocolo BUDP puro tarda 53.4 segundos menos que LAIN, BUDP envía 56681 paquetes para conseguir entregar todo el fichero completo, con una sobrecarga resultante del 88.9%. Sin embargo, la aproximación de conmutación de protocolos envía 42050 paquetes (es decir, 14631 paquetes menos), obteniendo así una sobrecarga de 40.2% paquetes adicionales. En este caso, TCP necesitó enviar sólo un 42.9% de paquetes adicionales, debido principalemtne a las retransmisiones requeridas durante los periodos de pérdidas de paquetes con $p = 0.3$.

Como resultado, LAIN obtiene un rendimiento global que mejora los obtenidos por los protocolos TCP y BUDP puros, reduciendo por un lado la sobrecarga de los paquetes enviados y obteniendo un retardo final cercano al de la aproximación BUDP.

6.3.4 *Rendimiento del mecanismo de test en serie*

En un entorno ideal, el sistema podría conocer el protocolo más adecuado para cada estado de la red, tal como se muestra en los ejemplos de la sección previa.

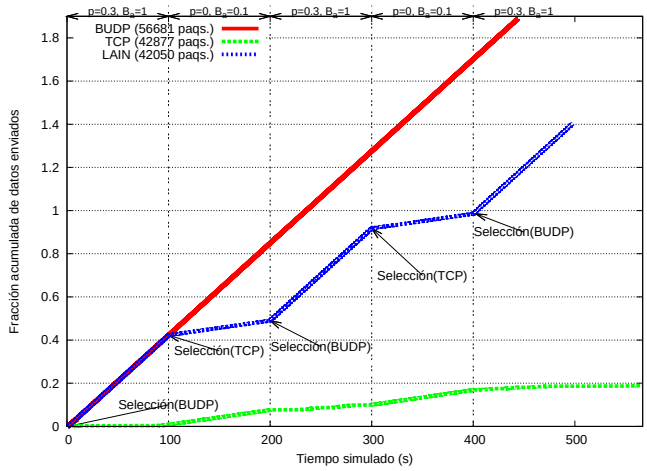


Figura 46: Fracción de fichero enviado con respecto al tiempo de duración

Sin embargo, en la práctica no es posible tener un conocimiento completo sobre el estado de la red y la adecuación de cada nuevo protocolo: se requeriría efectuar exhaustivas comparaciones entre los protocolos existentes, y tendrían que ser identificados los parámetros de red implicados, objetivos que no se pueden satisfacer fácilmente.

En un entorno autonómo, es decir, que pueda adaptarse por sí solo a las nuevas condiciones ambientales, ante la introducción de nuevos protocolos, o incluso ante la presencia de eventos desconocidos, el sistema debería ser capaz de adataptarse o evolucionar [134] para ajustarse al nuevo estado.

Distintas cuestiones surgen con este esquema: ¿cuántos protocolos candidatos deberían componer el test? ¿en qué orden? ¿cuál debe ser la duración del test? ¿qué eventos deberían iniciar los tests?

Los siguientes experimentos mostrarán los beneficios de utilizar este test en línea usando un mecanismo automático para activar la fase de test y de conmutación. Para ilustrar la plataforma LAIN, detallaremos los elementos previamente descritos en la sección 6.2 que han sido instanciados en esta prueba.

INSPECTOR DEL SERVICIO: El servicio implementado en estos experimentos tiene como objetivo entregar el

fichero completo tan pronto como sea posible, consumiendo el mínimo ancho de banda. Por tanto, el inspector del servicio debería tener en cuenta la duración de la transferencia, y la sobrecarga producida por el reenvío de paquetes para evaluar la bondad del rendimiento del protocolo. La función simple de idoneidad Φ utilizada en el experimento se detalla en la expresión 6.1:

$$\Phi(t) = n_r(t) - n_l(t) \tag{6.1}$$

donde $n_r(t)$ es el número de paquetes recibidos correctamente hasta el instante t (no se consideran los paquetes replicados), y $n_l(t)$ es el número de pérdidas de paquetes detectados hasta el instante t, incluyendo las pérdidas de paquetes retransmitidos. Obsérvese que $\Phi(t)$ podría tener un valor negativo en los casos en los que una selección incurriera en un alto porcentaje de pérdidas, como por ejemplo cuando se utiliza BUDP en un escenario con una congestión de tráfico severa, o con una tasa de pérdidas p más alta de 50%, escenarios contraproducentes que deben ser penalizados.

ASISTENTES DE MONITORIZACIÓN: para detectar si cambian las condiciones de la red, la parte servidor del monitor envía a la parte cliente paquetes de sondeo, con un consumo de ancho de banda inapreciable (10 paquetes de sondeo por segundo, que consume 0.4 kbps). Cuando el cliente detecta un cambio, le notifica al servidor el nuevo estado de la red.

El test se ejecutará cada vez que el monitor de red advierta un cambio. Para ello, los estados de red actual y previo se caracterizan mediante la media y desviación típica de los retardos de los paquetes. Para obtener dichos valores, se aplica un suavizado exponencial. Concretamente, se utiliza la expresión del cálculo del RTO propuesto por Jacobson para TCP [118]. El monitor de red escogido establece que las condiciones de la

red (el retardo de paquetes en este caso) son estables cuando se verifica la condición 6.2:

$$\mu_{t_{i-1}} - 2 \cdot \sigma_{t_{i-1}} < \mu_{t_i} < \mu_{t_{i-1}} + 2 \cdot \sigma_{t_{i-1}} \quad (6.2)$$

donde $\mu_{t_{i-1}}$ y $\sigma_{t_{i-1}}$ representan la media suavizada y desviación típica del estado previo de la red, respectivamente, y μ_{t_i} caracteriza el retardo medio más reciente. Estos valores se actualizan con cada paquete de prueba, mediante las expresiones 6.3 y 6.4:

$$\sigma_{t_i} = (1-\beta) \cdot \sigma_{t_{i-1}} + \beta \cdot |\mu_{t_{i-1}} - (t_{now} - t_{sent})| \quad (6.3)$$

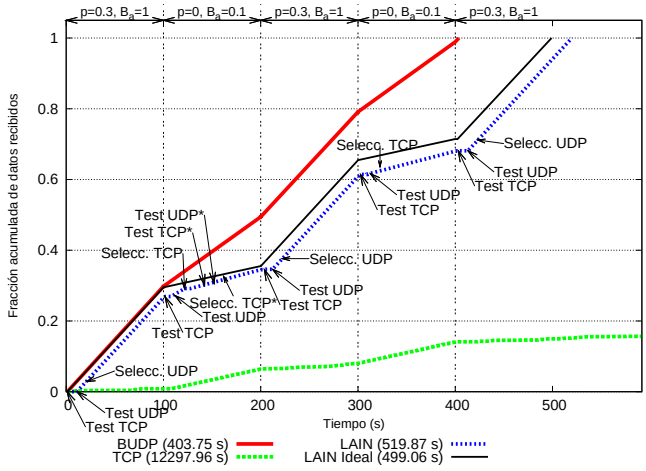
$$\mu_{t_i} = (1-\alpha) \cdot \mu_{t_{i-1}} + \alpha \cdot (t_{now} - t_{sent}) \quad (6.4)$$

donde t_{now} es el momento de llegada del paquete, y t_{sent} es la marca de tiempo del envío del paquete. Por otro lado, $\alpha = \frac{1}{4}$ y $\beta = \frac{1}{8}$, tal y como se recomienda en [118].

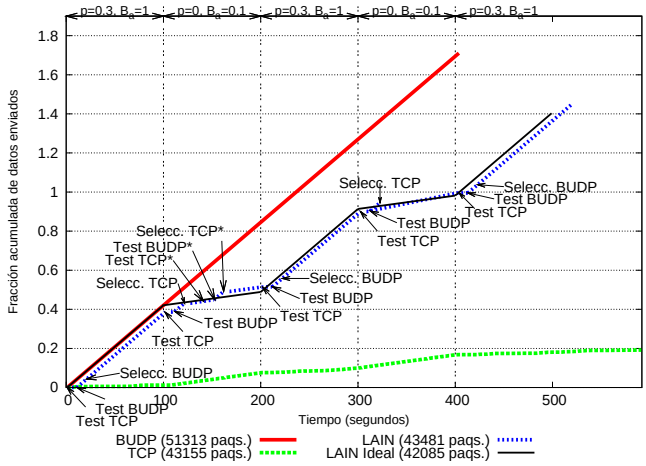
EXPERIMENTADOR: En caso de que se detecte un cambio, es decir, que no se verifique 6.2, se inicia la fase de tests.

Como resultado de las simulaciones se obtienen las Fig. 47a y Fig. 47b, para BUDP y TCP. Como en los escenarios previos, el tiempo empleado para la transmisión del fichero utilizando BUDP es 23.66 veces menor que para TCP. Sin embargo, TCP envía 7832 paquetes menos que el protocolo BUDP para completar la transferencia.

Observando la Fig. 47b puede verse que el uso del monitor de red propuesto para iniciar los tests produce activaciones innecesarias. En concreto, el segundo 141.1 de la simulación mostrada en la figura se inicia una ronda de tests (los test y las conmutaciones innecesarios se han destacado con el símbolo "*"). Esto evidencia que el monitor de red y la lógica de experimentación particulares deben ser diseñados robustos para evitar iniciar tests innecesarios.



a)



b)

Figura 47: a) Fracciones del fichero recibidas, y b) enviadas con respecto al tiempo.

6.4 CASO DE ESTUDIO II: CONMUTACIÓN ENTRE ENTREMEZCLADORES PARA UNA APLICACIÓN DE STREAMING DE AUDIO EN TIEMPO REAL

Aunque el marco propuesto tiene como propósito principal su aplicación en sistemas autónomos en los que la conmutación de protocolos se realice tras largos periodos de tests y monitorización, en esta sección se presenta un escenario en el que se muestra cómo la conmutación de protocolos puede ser utilizada en entornos en los que exista

la necesidad de adaptarse a las condiciones de red en cortos periodos de tiempo.

La transmisión de voz sobre IP (VoIP) constituye un ejemplo de aplicación con los mencionados requisitos. El tráfico VoIP se caracteriza principalmente por su fuertes restricciones temporales (el retardo extremo a extremo de los paquetes debe ser inferior a un umbral máximo $d^*_{max} = 300ms$, y a su vez es relativamente tolerante a tasas bajas de pérdidas de paquetes [96], aunque si las pérdidas son consecutivas puede resultar en una degradación de la calidad percibida [103]).

Como se ha estudiado en el capítulo 4, el entremezclado puede ser de aplicación para mitigar los efectos no deseados de pérdidas consecutivas.

El entremezclador multiflujo por bloque $\Pi(n_f, s)$ estudiado en el capítulo 4 que toma paquetes de distintos flujos de VoIP y genera un flujo agregado con capacidad de aislar pérdidas de hasta longitud s, introduciendo para ello el mínimo retardo.

El entremezclador $\Pi(n_f, s)$ requiere como parámetros el número de flujos disponibles para entremezclar (n_f) y el tamaño de ráfaga que debe ser capaz de aislar s. Para cada par (n_f, s), el dispositivo introduce un retardo por paquete acotado por d_{max} (ver expresión 4.26 del capítulo 4). Este retardo no es monótono decreciente con respecto a s. En su lugar, el retardo introducido por el entremezclador, (d_{max}), sigue una tendencia no monótona.

En los experimentos presentados en esta sección se emplea un entremezclador que debe conmutar el parámetro s para ajustarse a las ráfagas de pérdidas experimentadas en cada momento, y conseguir así unas distribuciones de pérdida de paquetes y de retardos por paquete óptimas. Dicha selección se ha de realizar respetando el máximo retardo tolerado d^*_{max}.

6.4.1 *Elementos de la plataforma*

Los distintos elementos de la plataforma se instancian para este escenario de la siguiente manera:

Asistente de monitorización

El monitor implementado para este conjunto de experimentos comprueba si el patrón de pérdidas varía o si se mantiene estable. Para ello, la parte del monitor ubicada en el receptor mantiene un registro de la distribución de las pérdidas experimentadas mediante un histograma.

Periódicamente, el monitor de pérdidas envía al experimentador las estadísticas actualizadas de los patrones de pérdidas y retardo medio de los paquetes. Más concretamente, el lado del monitor en el receptor devuelve a la fuente el valor del percentil 80 de la función de distribución acumulada (CDF) del histograma de longitudes de pérdidas. La interpretación de dicho valor es que el 80% de las ráfagas experimentas son iguales o inferiores a este valor (notado como $b_{80^{th}}$ a partir de este punto).

Obsérvese que el histograma se recalcula cada vez que se lleva a cabo una conmutación o se inicia una ronda de tests.

Inspector del servicio

El inspector del servicio instanciado en este experimento define una función de idoneidad Φ que considera la longitud media de ráfaga y el retardo extremo a extremo experimentado, definida mediante la expresión 6.5:

$$\Phi = \frac{1}{b_{80^{th}}} \cdot \frac{d^*_{max} - d}{d^*_{max}} \tag{6.5}$$

donde d representa el retardo medio por paquete, d^*_{max} es el máximo retardo tolerado (300ms para estas simulaciones), y $b_{80^{th}}$ representa el percentil 80 de la CDF de longitudes de ráfagas obtenido.

Se establece que un candidato tiene un mejor rendimiento que otro si su función de idoneidad es mayor que la del segundo. La función de idoneidad se diseña, por tanto, para que de un mayor valor a los resultados con una longitud media de ráfagas y un retardo medio inferiores.

Lógica de experimentación

Periódicamente, cada 75 segundos se ejecutan los tests. Se establece este periodo de activación de los tests porque se

conoce a priori que el periodo mínimo de cambios en las condiciones de red en este escenario será mucho mayor. Además, cada vez que la distribución de pérdidas varía, se inicia una ronda de pruebas. Al comienzo de cada ensayo se genera un conjunto de candidatos, que en este caso no es más que diferentes entremezcladores posibles. Dichos candidatos deben satisfacer las condiciones impuestas, es decir, introducir un retardo inferior a d^*_{max}. Los parámetros utilizados para estos candidatos son $n_f = 4$, y s con valores cercanos al $b_{80^{th}}$ proporcionado por el asistente de monitorización.

En las simulaciones presentadas se genera una lista de hasta 21 candidatos por torneo, que son todos los posibles entremezcladores que verifican las restricciones impuestas, ordenados según su cercanía al valor $b_{80^{th}}$. Durante la ronda de torneos, sólo los tres primeros candidatos compiten entre ellos. Cada uno se activa durante 5 segundos, registrando el correspondiente valor experimentado de $\Phi_{d,b_{80^{th}}}$. Tras la ronda de tests, se selecciona el candidato que obtuviera la major puntuación de idoneidad, sustituyendo así al entremezclador en producción.

Las rondas de tests se disparan cuando se detecta un cambio del patrón de pérdidas. Un cambio en la estabilidad de las condiciones de la red se detecta si se verifican las condiciones siguientes:

$$\lfloor b_{80^{th},i-1} - 1.5 \rfloor > b_{80^{th},i} > \lceil b_{80^{th},i-1} + 1.5 \rceil \qquad (6.6)$$

donde $b_{80^{th},i}$ es el valor más reciente, y $b_{80^{th},i-1}$ representa al valor de $b_{80^{th}}$ del último periodo estable. El actual valor de $b_{80^{th},i}$ se recalcula tras cada actualización de cada paquete de monitorización mediante la media suavizada siguiente:

$$b_{80^{th},i-1} = 0.9 \cdot b_{80^{th},i-1} + 0.1 \cdot b_{80^{th},i} \qquad (6.7)$$

Es importante puntualizar que el número de candidatos, su orden de selección, la duración de cada experimento, y la frecuencia de los torneos son parámetros críticos que deben ser cuidadosamente escogidos. Dada la naturaleza exploratoria de este trabajo, dichos valores han sido seleccionados de forma simple, tras algunos ensayos previos con diferentes valores.

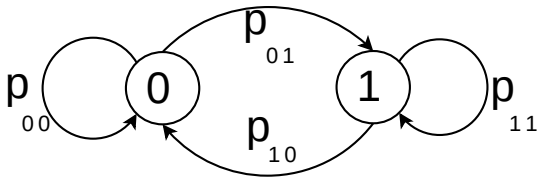


Figura 48: Modelo de Gilbert para la generación de pérdidas.

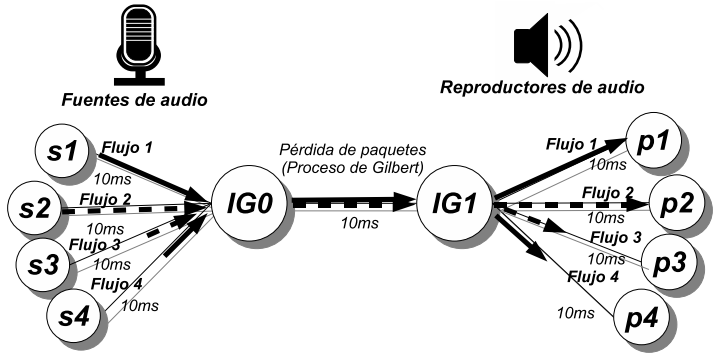


Figura 49: Topología del escenario de entremezclado.

6.4.2 *Configuración experimental*

El escenario simulado (Fig. 49) se compone de 4 flujos de audio, S_1,S_2,S_3 y S_4, que comparten un enlace común en su ruta hacia los diferentes destinos considerados (P_1,P_2,P_3 y P_4). El camino compartido experimenta un patrón de pérdidas cambiante que sigue un *modelo de Gilbert* [97]. Dicho modelo de error captura el comportamiento de pérdidas consecutivas de la red.En las simulaciones, el módulo de pérdidas se ubica en el enlace $IG0 - IG1$ (ver Fig. 49).

La probabilidad de transición desde el estado libre de pérdidas al estado de error (p_{01}), y la del estado de error al estado libre de pérdidas (p_{10}) caracterizan el modelo (ver Fig. 61).

Los flujos de audio tienen una tasa de 50 paquetes por segundo. Cada paquete de audio mantiene un número de secuencia, para que puedan ser reproducidos y sincronizados en el orden correcto en el receptor.

El entremezclador $II(4, s)$ se implementa en el nodo IG0. Para evaluar la calidad perceptual de de las transmisiones

de voz, se ha seguido la recomendación G.107 de la ITU-T (*modelo E*, [78]). Concretamente, se ha empleado la versión simplificada del modelo E para VoIP propuesta en [88]. Los parámetros de entrada de esta simplificación del modelo E son la probabilidad de pérdidas resultantes y el retardo medio extremo a extremo, a partir de los cuales se estima la calificación MOS. Una definición más completa del modelo simplificado utilizado se puede consultar en el capítulo 8.

Una de las deficiencias del modelo E es que, aunque estima la calidad de voz, no modela completamente todos los factores perceptuales involucrados. De hecho, el modelo E no captura el comportamiento de las pérdidas consecutivas (sólo tiene en cuenta la probabilidad global de pérdidas), y por tanto no es adecuado para evaluar el rendimiento del entremezclador.

Para proporcionar una valoración completa de la calidad que contemple todos los parámetros que afectan a la calidad perceptual, en las evaluaciones se proveerá también la longitud de ráfaga resultante L_o y el retardo medio por paquete d_{avg}, parámetros clave para evaluar la calidad de un flujo VoIP.

6.4.3 *Rendimiento de los tests en serie.*

La evaluación se realiza en un escenario donde los patrones de error varían con el tiempo. El procedimiento de conmutación tiene como objetivo adaptar el entremezclador a las condiciones existentes, para obtener mejores resultados (y calidad perceptual) que en una aproximación clásica (estática).

La evaluación se realizará con la puntuación MOS obtenida por la aplicación del modelo E y la media de retardos por paquete y longitud de ráfagas resultantes.

En el experimento con la aproximación clásica (no adaptable) el entremezclador $\Pi(4, s)$ fija al principio de las simulaciones la longitud de ráfaga objetivo s.

Para comparar las estrategia estática con la aproximación adaptativa, se llevan a cabo varias simulaciones con todos los posibles valores de s que cumplen las restricciones de

Esquema	s	MOS	L_o	d_{avg} (s)
LAIN	$[4-13]$	**2.33**	**2.87**	**0.099**
Estático	5	2.35	3.31	0.080
	6	**2.31**	**2.85**	**0.120**
	7	2.29	2.58	0.135
	8	2.32	2.38	0.110
	9	2.01	2.24	0.211
	10	1.59	2.11	0.280
	12	1.65	1.94	0.270

Tabla 8: Resultados para el escenario de condiciones de red cambiantes.

tiempo que VoIP impone. De esta forma se averigua cuál sería la mejor solución empleando sólo la estrategia estática, que constituye el modelo de referencia.

Las distribuciones de las pérdidas de paquetes de red durante la sesión de voz sobre IP varían. Concretamente, desde el segundo 0.001 al 500, el modelo de Gilbert se configura con una probabilidad de pérdida $p = 0.3$ y una longitud de ráfaga media $L_i = 12$ ($p_{01} = 0.03571$ y $p_{10} = 0.08334$). Desde el segundo 500 al segundo 1000 se configura $p = 0.2$ y $L_i = 20$ ($p_{01} = 0.0125$, y $p_{10} = 0.05$).

La tabla 8 presenta los resultados para estas simulaciones. Para todos los posibles valores válidos de s, se muestra la puntuación MOS obtenida por el modelo E, la media de L_o y el retardo medio por paquete d_{avg}.

Se puede observar cómo para obtener la misma puntuación MOS y valor L_o que en la aproximación LAIN, el esquema fijo requiere que $s = 6$. Sin embargo, aún en este caso, el retardo medio introducido d_{avg} es 0.019 segundos mayor que la solución de LAIN. Para reducir el retardo, es posible elegir $s = 5$, aunque la longitud de ráfagas media b_{Lo} aumenta hasta 3.31, la cual excede en 0.44 al valor obtenido mediante LAIN. Nótese que cuando existe una alta probabilidad de pérdidas (como es el valor $p = 0.3$ de

este caso), incluso pequeñas diferencias en B_{Lo} tienen un gran impacto en la calidad percibida.

6.5 DISCUSIÓN Y TRABAJO RELACIONADO

En las secciones previas se ha demostrado la viabilidad de aplicar la conmutación dinámica de protocolos a nivel de transporte tanto para tráfico elástico como inelástico. En ambos casos se ha observado cómo la conmutación de protocolos de forma dinámica puede ser beneficiosa en muchas situaciones, aún adoptando mecanismos de monitorización y estrategias de selección muy simples. El tipo de monitorización y la lógica de conmutación más apropiados son dos áreas clave que requieren futura investigación. En esta sección, discutimos cuestiones que requieren investigación adicional, así como trabajos relacionados, y se presenta una perspectiva del futuro con la evolución automática de protocolos.

6.5.1 *Variedad de protocolos*

La plataforma presentada asume que existe la posibilidad de realizar la selección entre distintos protocolos disponibles. Diferentes aproximaciones han proporcionado soluciones esta cuestión, como por ejemplo x-kernel, ComScript, las redes activas [26], kits de herramientas como Click, y *middleware* activo. Como ejemplo reciente, en [135] se presenta un sistema de flujo de datos en el que los protocolos pueden ensamblar dinámicamente bloques elementales reusables para redes superpuestas peer-to-peer. Más cercano a TCP, el trabajo (Self-Spreading Transport Protocols, [136]) propone una extensión XTP y más tarde un sistema de actualización de protocolo, el sistema STP, donde muchos protocolos pueden ser potenciales candidatos a ejecutarse, y por tanto, se requiere una selección entre ellos.

STP incluye un gestor de políticas que rechaza o acepta protocolos basándose en su origen (p.e., un fabricante de software confiable) o la historia de su comportamiento. Sin embargo, no se ha presentado ningún esquema sobre cómo

obtener dicha historia. Se establece que el diseño de las políticas es un elemento importante, pero se deja como trabajo futuro.

Creemos que no hay carencia de mecanismos para la composición y despliegue de protocolos, pero sí que escasean políticas y algoritmos para automatizar la selección de los elementos a componer e implementar, y evaluar dicha selección en su uso en producción.

En [137] se propone un algoritmo que efectúa tal elección en el contexto de la computación autonómica, para seleccionar automáticamente entre componentes competitivos basándose en sus características de carga de trabajo. En [138] los autores proponen un algoritmo para selecionar el servicio web más adecuado de entre un conjunto de servicios candidatos. En ambos casos se requiere disponer de una caracterización del rendimiento muy precisa para seleccionar correctamente y llevar a cabo la conmutación de los servicios.

El marco de trabajo que constituye LAIN se diferencia fundamentalmente de otros trabajos en que no asume el conocimiento completo del funcionamiento y rendimiento de los protocolos, y que tampoco se dispone de la completa caracterización de la red. Es por ello que no puede asumirse que se disponga del modelado de cada protocolo para cada posible escenario. Es, por tanto, un esquema más robusto frente a errores de funcionamiento del protocolo. Por otro lado, nuestro marco requiere conmutar entre protocolos de forma oportuna considerando una información de monitorización fiable.

Por último, el estudio de un algoritmo que realice la conmutación óptima es uno de nuestros objetivos de trabajo futuro claves.

6.5.2 *Interfaces*

Dado que LAIN prevé que protocolos totalmente nuevos sean introducidos dinámicamente en el sistema, es necesario definir una interfaz de programación de aplicación (API) extensible y flexible, para permitir que los protocolos

nuevos y existentes puedan interoperar, y permitir a las aplicaciones que especifiquen sus necesidades y preferencias de forma flexible e independiente del protocolo.

Existe una creciente tendencia hoy en día en diseñar interfaces basadas en ontologías, en las cuales las interfaces de los componentes se describen mediante lenguajes estandarizados tales como IDL [139], WSDL [140] u OWL [141]. En este contexto, existen propuestas de herramientas de razonamiento para encontrar la mejor correspondencia entre los requisitos de las aplicaciones y los potenciales servicios que puedan satisfacerlos.

Como primera aproximación, interfaces de descripción de servicios podrían usarse en LAIN para identificar protocolos candidatos antes de probarlos. Sólo aquellos que satisfagan las especificaciones de la aplicación serán considerados. Sin embargo, dichas descripciones no eximen a la plataforma de llevar a cabo regularmente tests intensivos para evaluar el comportamiento real del protocolo, que en muchas situaciones puede incumplir la descripción.

Sin embargo creemos que a largo plazo dichas descripciones predefinidas no ofrecen una solución ideal. Como posible aproximación, en programación genética se estudia cómo distintos módulos y sus interfaces pueden emerger mediante evolución [142].

Extendiendo esta idea en el futuro de los sistemas de software distribuidos, cabe imaginar como escenario plausible, para dar respuesta a las necesidades de las aplicaciones y servicios cambiantes, la cooperación entre entidades independientes, bien definidas tendiendo a unidades atómicas más pequeñas que interactúan entre ellas que forman agrupaciones de poca duración para resolver un problema dado, o proporcionar un servicio determinado, y que finalmente se disuelven cuando ya no son necesarios. Un nuevo tipo de interfaces serán necesarios para esos sistemas, con todas las implicaciones que conlleva, en términos de cómo controlar el sistema y especificar un comportamiento deseado sin preprogramarlo.

6.6 CONCLUSIONES

En este capítulo se presenta una plataforma para la evaluación y selección de protocolos de forma dinámica, mediante la proposición de experimentos que evidencian la viabilidad de la conmutación línea entre protocolos. Para ello se han instanciado dos casos de estudio que cubren dos categorías de aplicaciones muy diferentes: una aplicación elástica de transferencia de ficheros, y una aplicación de tráfico inelástico de audio. Los beneficios y compromisos referentes a la conmutación de protocolos han sido expuestos en cada caso.

Como en cualquier campo emergente, aún quedan por resolver muchos asuntos. El diseño de un comprobador de servicio para protocolos puede ser un problema complejo. Aunque se han propuesto esquemas para diferenciar comportamientos normales y anormales de protocolos, [143], la verificación de la funcionalidad de un servicio a partir de una descripción a alto nivel se encuentra, en gran parte, sin resolver.

La evaluación de la adecuación de los protocolos a las condiciones de red requiere tests continuos: su duración y los recursos que se asignan deben ser evaluados con detenimiento. Más aún, los tests pueden afectar al comportamiento de la red, dificultando la comparación entre protocolos. La lógica de monitorización y conmutación demanda más investigación para poder detectar la correlación entre acciones y eventos, un problema clásico en sistemas de realimentación requeridos para la evaluación continua. La coordinación (señalización) es también un asunto a considerar: en un sistema distribuido, el lado remoto debe informar de la idoneidad de forma precisa y fiable; el proceso de conmutación de protocolos debe ocurrir de una forma coordinada para que todos los puntos finales usen protocolos compatibles.

Otro de los asuntos a tener en cuenta es cómo asegurar equidad entre sesiones competitivas cuando distintos protocolos pueden ser seleccionados en cualquier momento. La solución clásica consiste en imponer un buen comportamiento, como por ejemplo un esquema amigable con TCP,

o un esquema justo de reserva de recursos. Una solución diferente podrá consistir en permitir a la red realizar a su vez experimentos sobre los usuarios para asegurar la convergencia hacia una reserva ecuánime.

Aunque desafiante, esta parece ser una prometedora línea de investigación, que conduce en última instancia a las redes del futuro en las que los protocolos pueden sintetizarse automáticamente, los cuales evolucionan para ajustarse a los nuevos requisitos, y que son capaces de recuperarse o sanar ante fallos.

7

CONCLUSIONES

Este capítulo tiene como objeto presentar las conclusiones fruto de la realización de la tesis, dando una visión integral del trabajo realizado.

En el presente trabajo se ha abordado el reto de mejorar la calidad de experiencia (QoE) de flujos de Voz sobre redes IP (*best effort*), contribuyendo finalmente en varios aspectos clave. Para ello, se ha optado por aplicar mecanismos de reparación adaptables guiados por funciones de utilidad. Las funciones de utilidad utilizadas modelan el efecto de las condiciones de red sobre la calidad que perciben los usuarios finales. Una de los aspectos que caracterizan a las soluciones aportadas es que los mecanismos de reparación se localizan en algunos nodos intermedios de la red, pudiéndose desplegar en redes superpuestas, o implementándolas como pasarelas de reparación individuales. Para conseguir estos objetivos, se han diseñado, evaluado y analizado varias alternativas.

Como se ha expuesto en el capítulo 8, el reconocimiento automático de voz (ASR) es una efectiva medida de calidad perceptual de flujos de VoIP. Desafortunadamente su utilización depende del empleo de un reconocedor que no puede ser ejecutado en tiempo real. Por tanto no es posible su utilización para tomar decisiones de ajuste de mecanismos de reparación en línea. Por esta razón se ha propuesto una expresión analítica que modela el comportamiento de un reconocedor de referencia. El reconocedor adoptado utiliza una base de datos de palabras y una tarea de reconocimiento estandarizadas. La tarea de reconocimiento consiste en identificar números de varios dígitos conectados. Como para reconocer cada palabra no se puede extraer información por el contexto en el que aparecen, que sería el caso en el que un oyente tendría ventaja sobre la máquina,

se espera que haya una gran correlación entre los porcentajes de reconocimiento automático y la inteligibilidad que experimentaría un usuario real.

El modelado se ha realizado evaluando 3957 trazas mediante el ASR de referencia. Cada traza fue generada mediante modelos de Gilbert con parámetros que representaban las condiciones de red más habituales. Como resultado, la función de estimación obtiene un error de ajuste medio inferior a 0.35 puntos porcentuales.

Para validar el uso del estimador en escenarios donde la distribución de pérdidas no corresponde al ideal, es decir, no ha sido generado por un modelo de Gilbert, se han comparado los resultados del ASR con la estimación ofrecida sobre un conjunto de trazas reales y trazas obtenidas mediante simulación. La comparación ha mostrado que el error medio de predicción es inferior a 1.05 puntos en la escala de 1 a 100.

Como resultado de lo anterior, se ha propuesto una función de utilidad, que estima el nivel de inteligibilidad, medido en términos de *precisión de palabras* (W_{acc}), de un flujo de voz codificado con un esquema PCM, que puede ejecutarse en línea. De esta forma, conociendo la probabilidad de pérdidas p, y la longitud media de ráfagas L_i experimentadas por un flujo de voz, proporciona una estimación del W_{acc} correspondiente.

En el capítulo 4, se ha procedido al diseño de mecanismos de reparación preventivos ubicados en nodos intermedios de la red. En concreto, se han desarrollado varios entremezcladores de paquetes de bajo retardo, denominados $II(n_f, s)$ e $II(n_f, s, s')$. Estos han sido diseñados para su utilización sobre aplicaciones de transmisión de voz sobre IP con restricciones de tiempo real. Dichos entremezcladores aprovechan la concurrencia de paquetes de distintas flujos para dispersar las ráfagas de pérdidas, reduciendo así el retardo introducido por el procedimiento de entremezclado.

Asimismo se han obtenido analíticamente las expresiones que permiten predecir el retardo máximo y medio por flujo y por entremezclador. Se ha proporcionado además un algoritmo que permite estimar el tamaño de las ráfagas

resultantes una vez reconstruido el orden original de los paquetes de voz. Dichas expresiones han sido validadas por medio de simulación, mostrando que el error de las estimaciones de retardos es del orden de unos pocos milisegundos, y el error cometido en la estimación de las longitudes de las ráfagas resultantes son inferiores a 0.5 paquetes.

Una vez estimados los parámetros de probabilidad de pérdidas y ráfagas medias que se obtendrían con el empleo de esta técnica, hemos presentado un algoritmo heurístico para seleccionar de entre las configuraciones viables, el que mejor calidad perceptual e inteligibilidad estimadas ofrezca bajo unas condiciones de red dadas. Los resultados obtenidos han mostrado que el algoritmo heurístico realiza la misma selección utilizando las estimaciones de calidad calculadas que la versión que conociera los índices de calidad experimentados a posteriori.

El ámbito de aplicación de esta técnica de entremezclado de bajo retardo no se circunscribe a las transmisiones de voz interactivas, sino que puede ser empleado para otro tipo de transmisiones multimedia con restricciones de tiempo.

Para ofrecer complementariamente mecanismos de reparación reactivos ubicados en nodos intermedios, se han presentado varios mecanismos de detección de pérdidas basados en la estimación del instante de llegada de cada paquete, tanto de forma estática (DET), como adaptativa (DEAT-JAC, DEAT-HOLT). Estos mecanismos, combinados con la técnica de retransmisión local, ofrecen un retardo medio de detección inferior al de los mecanismos clásicos de detección (DAS), que se basan en verificar la continuidad de los números de secuencia. Aunque con la técnica propuesta aumenta ligeramente la tasa de detecciones erróneas, el incremento no es significativo, comparado con otras técnicas de adaptación dinámica de temporizadores.

Tras el análisis de las técnicas de detección propuestas, se ha diseñado un algoritmo que selecciona el nodo intermedio que sea la mejor ubicación del mecanismo de detección para maximizar el valor esperado de calidad perceptual, mediante el empleo del *emodel*. Para ello, el algoritmo propuesto estima el retardo de recuperación del paquete

considerando las condiciones de red actuales expresadas en términos de probabilidades de pérdida y retardos de propagación.

Tras las ejecuciones de las simulaciones utilizando el algoritmo propuesto, se muestra que la degradación de la calidad introducida por los errores de estimación es inferior a 0.05 puntos MOS para el 95% de los casos simulados. Por tanto, se puede inferir que el algoritmo selecciona el mecanismo de selección apropiado, así como su mejor localización bajo unas condiciones de red dadas, basándose en el criterio de calidad de voz subjetiva.

Basados en los resultados experimentales y teóricos llevados a cabo, se concluye que DAS es adecuado para escenarios en los que predominan las pérdidas aisladas y en los que el jitter sea inferior al periodo de generación de cada paquete (t_f) DAS no es adecuado para aquellos escenarios en los que predominen las ráfagas de pérdidas. El uso del detector DET es el más indicado en escenarios donde el histograma del jitter sea conocido, cumpliendo que el porcentaje de los retardos inferiores al umbral U escogido sea mayor que la tasa de falsas alarmas tolerado. DET supera el rendimento de DAS en escenarios con pérdidas consecutivas, al no ser dependiente del tamaño de la ráfaga de pérdidas. Los detectores adaptativos permiten reducir el retardo incurrido por la versión no adaptable (DET), sobre todo en escenarios donde el histograma de retardos exhibe una larga cola: en ese caso, se exigiría para DET un umbral de detección demasiado conservador. Ante casos de ráfagas de pérdidas, el detector DEAT-DAS es el más indicado, superando en mucho el rendimiento del detector dinámico de referencia, DEAT-JAC. Se puede comprobar que, casi invariablemente, el detector híbrido DEAT-DAS iguala o supera el rendimiento de DEAT-HOLT.

Para concluir, en el capítulo 6 de la tesis se ha abordado otro de los aspectos fundamentales a considerar a la hora de proporcionar mecanismos adaptables de reparación de flujos de VoIP. Se trata de la selección automática, y en tiempo de ejecución, de la técnica que mejor calidad de experiencia ofrezca al usuario final, bajo las condiciones de red experimentadas en cada momento. Si bien, junto a las

técnicas de reparación se han ofrecido heurísticas para seleccionar sus mejores parámetros (ubicación, configuración de los entremezcladores, etc.), se hace igualmente necesario tener un mecanismo para poder seleccionar, de entre las técnicas disponibles en un nodo de reparación, la que mejor calidad ofrezca en cada momento.

Se presenta para ello una plataforma, LAIN, para la evaluación y selección de protocolos de forma dinámica, mediante la proposición de experimentos que evidencian la viabilidad de la conmutación en línea entre protocolos. Para ello se han instanciado dos casos de estudio que cubren dos categorías de aplicaciones muy diferentes: una aplicación elástica de transferencia de ficheros, y una aplicación de tráfico inelástico de transmisión de voz. Los beneficios y compromisos referentes a la conmutación de protocolos han sido expuestos en cada caso.

Como en cualquier campo emergente, aún quedan por resolver muchos aspectos sobre la selección automática. De entre ellos destacan el diseño de un comprobador de servicio para protocolos, la duración óptima de los tests de evaluación del rendimiento de los protocolos candidatos, la coordinación del proceso de conmutación de protocolos entre los puntos finales para que usen protocolos compatibles, y la provisión de equidad entre sesiones competitivas cuando distintos protocolos pueden ser seleccionados en cualquier momento.

Aunque desafiante, ésta parece ser una prometedora línea de investigación, que conduce en última instancia a las redes del futuro en las que los protocolos pueden sintetizarse automáticamente, los cuales evolucionan para ajustarse a los nuevos requisitos, y que sean capaces de recuperarse ante fallos.

Como apunte final, las distintas propuestas realizadas en este trabajo son piezas fundamentales de una futura plataforma que ofrezca mejoras de calidad de servicio para transmisiones de voz sobre IP, aplicando técnicas de reparación seleccionadas según varias métricas de calidad perceptual, que pueden seleccionarse y ajustarse en tiempo real.

8

CONCLUSIONS

This chapter aims to summarize the main conclusions drawn as a result of the work presented in the different chapters.

In the present thesis, we have addresed the challenge of improving the Quality of Experience to VoIP flows over the best effort IP network. To reach this goal, we have adopted two design decisions. On the one hand, we assume the use of adaptative recuperation mechanisms driven by utility functions that model the impact of the technique on the end user quality of experience. On the other hand, we have adapted these recovery mechanisms to use them at intermediate nodes within an overlay network. As a result, we have contributed to this objective in several ways.

As we have exposed in Chapter 8, the automatic speech recognition (ASR) is an effective measure of the quality of a VoIP flow. However, the use of this type of measure entails the use of recognition machines which can not perform the evaluation in real time. Unfortunately, this leads to the impossibility of adopting the automatic speech recognition to drive online decisions.

For this reason we propose an analytic expression that models the behaviour of a baseline speech recognition machine. This reference recognition framework uses standardized recognition tasks and word databases. The ASR task is the identification of numbers with several conected digits, pronounced in English. This type of task improves the correlation between the inteligibility obtained by the machine and the inteligibility perceived by a user, since the digits can not be guessed by a human subject from the context of the previous or subsequent digits.

The modelling is carried out by using the baseline recognition machine on 3957 different error patterns. Each pattern is generated by a Gilbert model configured with different

parameters. The average estimation error is lower than 0.35 points over the 100 points of the $W_a$cc scale.

To provide further results, the accuracy of the model is evaluated on traces that were not generated by a Gilbert Model. The result is that the average prediction error is lower than 1.05 points over 100.

As a contribution, we have proposed a utility function that estimates the intelligibility level (W_{acc}) of a voice flow coded with a PCM scheme, which can be used online, with the probability of loss p and the average burst length L_i as inputs.

In chapter 4, we have provided the design of a preventive recovery mechanism that can be hosted at an intermediate node within the network. More precisely, we have developed several low delay packet interleavers, referred to as $\Pi(n_f, s)$ and $\Pi(n_f, s, s')$. They have been designed to be applied on real-time VoIP flows. These interleavers scramble packets from different flows to scatter the losses of consecutive packets. Doing this, the interleaving delay is reduced.

Additionally we have provided the analytical expressions of the maximum and average interleaving delay per flow and interleaver. We have proposed as well a heuristic algorithm that estimates the resulting average burst length after the reconstruction of the interleaved voice flow.

The aforementioned expressions have been validated by means of simulation, showing that the error prediction of the estimated delay is around a few milliseconds. Furthermore, the error in the burst length prediction is less than 0.5 packets.

After validating these expressions, we have proposed a heuristic algorithm for selecting among the feasible interleavers given a network condition, by using intelligibility and subjective quality criteria. The results have shown that this algorithm chooses the interleaver that exhibits the best performance by simulation.

Complementary, in chapter 5 we have also proposed several reactive recuperation mechanisms, based on the de-

tection of lost packet by estimating the arrival time of each packet. A static version called DET, and several adaptive alternatives (DEAT-JAC, DEAT-HOLT, DEAT-DAS), have been provided. These detectors, incorporated into local retransmission procedures, result in lower recuperation delays than the delays obtained by using DAS instead.

After the characterization of the proposed detection procedures, we have proposed an algorithm that selects the location of the intermediate node and the selection procedure which maximize the estimated subjective quality, calculated by means of the *e-model*. This can be done because the characterization of the detection procedures includes the estimation of the delays and probability of loss obtained after the use of the recovery procedure.

After simulating the selection algorithm, it is shown that the difference between the results of the best location and detector and the results of the location and detector selected by the algorithm, differ in average in 0.05 points in the MOS scale. Therefore, the proposed algorithm do select the best location and detection alternative, by means of maximizing a subjective quality criterion.

From the theoretical and experimental results we can conclude that the use of DAS is adequated in scenarios in which isolated packet losses prevail, and in which the *jitter* delays are lower than the period of packet generation (t_f). On the other hand, DET has proven to be best suited for scenarios in which the *jitter* histogram is known, and the percentaje of packet delays under the detection threshold U is greater than the percentage of tolerated late packet losses.

As it has been shown, the adaptive DEAT detectors reduce the delay incurred by the static DET version, specially when the delay histogram exhibits a long tail. It has been shown that the DEAT-DAS algorithm performs better than the rest for bursty error scenarios.

To conclude, in chapter 6 we have addressed the online automatic selection and switching of protocols which best fit the existing network conditions. Althought in each chapter we have provide heuristic algorithms for selecting the

best parameters for each reparation mechanism, we also need a procedure to select the best technique which maximizes the subjective quality adopted to measure the service. To this end, the LAIN framework has been proposed. To assess the feasibility of this approach, we have conducted several experiments for serveral cases of study: one for elastic applications, and the other for VoIP flows. For each case, the respective benefits and tradeoffs are discussed.

As in every emergent research field, further study is required to solve many additional issues about the automatic protocol switching. Among them, the design of the service checker, the optimal duration of the online tests, the coordination of the endpoints to assure the switching between compatible protocols, and the fair coexistence of several competitive protocols.

Althought challenging, this seems to constitute a promising research area which will drive ultimately to future networks in which the protocols can synthetize themselves, evolving to cope with new requisites, and enabled to heal themselves in case of a faulty operation.

As a final remark, the different proposals provided in this dissertation constitute fundamental pieces of a future framework which will provide Quality of Service enhancements for VoIP services, by means of using packet loss reparation mechanisms that can be selected and ajusted in real time, by maximizing quality of user's experience metrics.

9

SUMMARY

In this chapter, the main contributions of this thesis are summarized in English.

9.1 INTRODUCTION AND PROBLEM STATEMENT

The convergence of voice and data is empowering the development of new applications and services, reducing infraestructure and maintenance costs. The trend is to integrate every communication service over the IP network.

Voice over IP applications (VoIP) transmit the voice signal, previously coded and packetized, over an IP packet switched network. However, the IP protocol provides a best effot service, which means that it neither can guarantee the effective delivery, nor the delay that a packet will incur to reach its destination. In particular, it can not guarantee the delivery within the tight deadlines that a real-time application would demand.

To alleviate the drawbacks that the best effort service imposes to the real-time multimedia services, a number of end to end error control techniques have been proposed. These techniques aim to diminish the impact that the end to end delay, the packet delay fluctuation and the loss of packets have on the quality of the final perceived signal [96].

Loosely speaking, the recuperation schemes can be classified in *reactive* and *preventive* measures. While reactive measures are taken after a packet loss is detected, preventive techniques try to prepare the multimedia flow for this event.

Interactive multimedia streams must reach their destination hosts before a tight time limit (e.g. 300 ms for voice streams). In this context, the use of end-to-end recuperation techniques to face packet losses is highly restricted. That is, recovering a lost packet potentially can introduce a delay that in many cases is not affordable. Furthermore, it is

well established that in streaming applications packet losses are more harmful as they are consecutive, given that the subjective quality degradation increases as the burst length increases [144].

With the advent of new paradigms such as the *Active Networks* ([26],[36]) or the *Overlay Networks* ([29]), alternative recuperation techniques have been designed ([145], [33], [32]). This type of techniques tries to improve the quality of the multimedia flow by executing part of the recuperation algorithm at an intermediate network node.

9.1.1 *Dissertation's goals*

The overall objective of this thesis is to provide new adaptive reparation mechanisms for VoIP flows, whith the aim of maximizing the quality of experience (QoE) that the final users perceive. The specific objectives can be broken down into the following targets:

- The development of new reparation mechanisms which can benefit from its deployment at intermediate nodes.
- The characterization of the developed mechanisms, providing a mean to estimate the resulting pattern of packet losses and average delay.
- The provision of measures that objectively quantifies the quality of experience perceived by the end users.
- The design of algorithms which select the best parameters for a recuperation technique, given a network condition. This selection must be driven by the estimation of the perceived QoE.
- The proposal of a general framework which performs automatic protocol switching. This framework should select the protocol (or recuperation technique) which best fits the corresponding network condition.

9.2 I-MODEL: AN INTELLIGIBILITY MEASURE OF VOIP FLOWS

Several measure tools are being used for the speech quality evaluation. Generally, measure tools such as the PESQ (ITU-T recommendation P.862 [75]) and the *e-model* [78], are widely adopted. Alternatively, another QoE measure is presented in [94], where the resulting percentage of automatic recognition and the PESQ MOS score are related. Furthermore, a mapping function from the human intelligibility test to an automatic recognition system is proposed in [84].

Therefore, we can use this score to measure the speech quality. We will use an automatic speech recognition system (ASR) in order to provide an intelligibility index of the resulting speech signals because its lower cost compared with the MOS tests with a number of listeners.

Speech recognition performance is measured in terms of the *Word Error-Rate* (WER), defined as:

$$WER = \frac{n_i + n_s + n_d}{n_t} \tag{9.1}$$

where n_s is the number of substituted words, n_i is the number of spurious words inserted, n_d is the number of deleted words and n_t is the overall number of words.

In Fig. 50, the dependency between the word accuracy (calculated as $(1 - WER) \cdot 100$) on the error rate and the burst length is shown. The points in the represented surface correspond to the obtained word accuracy percentage after the execution of the automatic recognition process. The test voice set was generated applying a pattern of losses with a probability of loss $p = \{0.1, 0.2, 0.3, 0.4, 0.5\}$, and a burst length $L_i = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. It can be seen in the figure that the number of consecutive losses strongly influences the degradation of the recognition rate.

For ASR evaluation we use the connected digit Project Aurora 2 database [146]. An simple error concealment is used to recover the lost frames. It consists in repeating the last frames in order to substitute the lost one. Each frame contains 20ms from the original speech encoded with a

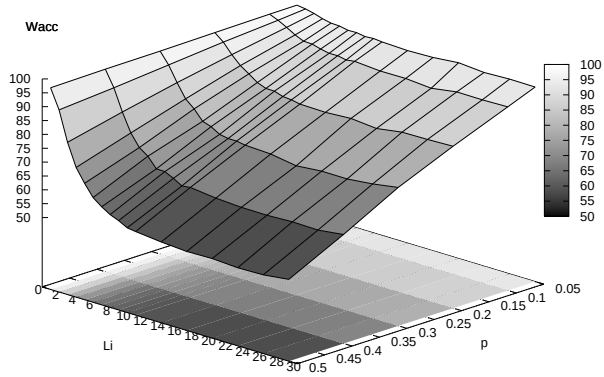


Figure 50: Preliminary intelligibility results for PCM encoded speech, for several error patterns.

16-bit PCM scheme. The error models are applied on such frames. The recognition task for the experiments is based on the the Aurora 2 database. This database consists of connected digit sequences for American English talkers. A feature extractor segments the received speech signal in overlapped frames of 25 ms every 10 ms. Each speech frame is represented by a feature vector containing 13 *Mel Frequency Cepstrum Coefficients* and the *log-Energy*. Finally, the feature vectors are extended with their first and second derivatives. The speech recognizer is based on *Hidden Markov Models* (HMM). We use eleven 16-state continuous HMM word models, (plus silence and pause, that have 3 and 1 states, respectively), with 3 gaussians per state (except silence, with 6 gaussians per state). The HMM models are trained from a set of 8440 noise-free sentences, while the out-of-train-test set comprises 4004 noise-free sentences.

In order to model the ASR behaviour, a total of 2836 loss patterns, generated by a Gilbert model with parameters in the range $p = [0.01, 0.20]$ (with steps of 0.01), and $L_i \in [1, 10]$ (in steps of 1), were processed by the ASR. For each pair (p, L_i), an average of 11.04 distinct patterns were generated and evaluated. The results can be observed in Fig. 51, where the average W_{acc} value for each tuple, as well as the 95% interval of confidence obtained are plotted.

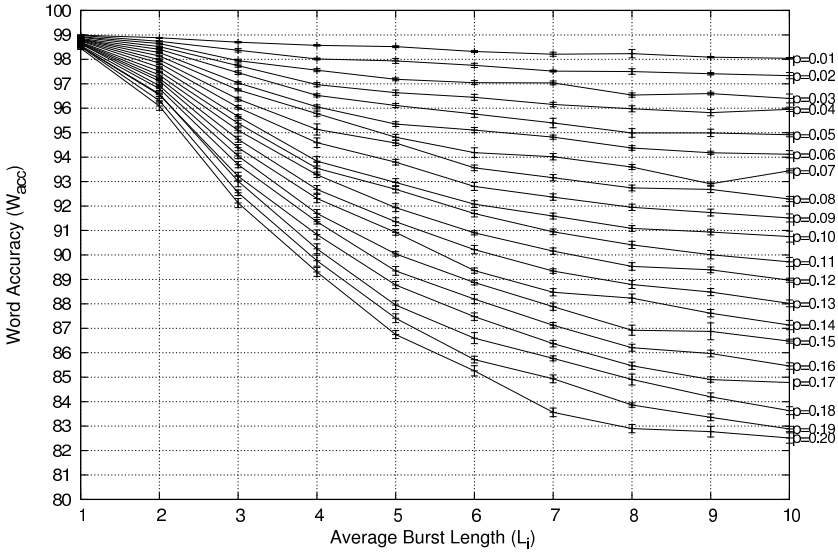


Figure 51: Average W_{acc} result for $p = 0.01$ to $p = 0.20$, in 0.01 steps, versus the average loss burst length L_i=1,2,3,4,5,6,7,8,9,10. For each value, the 95% confidence interval is shown .

9.2.1 *The i-model Utility Function*

By means of curve fitting, the *i-model* utility function shown in expression 9.2 was defined. The average error of the $imodel_{p,L_i}$ prediction over the experimental data is 0.3420 points, with a standard deviation equal to 0.0079. Please note that the W_{acc} index ranges from 0 to 100.

$$imodel_{p,L_i} = 99.036 - p \cdot \left(\frac{217.908}{1 + e^{-\frac{L_i}{132.775}}} - 2.429 \right) \quad (9.2)$$

As it can be seen, the p and L_i parameters which characterizes the Gilbert Model that adjust the distribution of burst lengths the evaluated flow, is mapped into the W_{acc} index.

To assess the validity of the proposed model, a number of error patterns which do not adhere to the Gilbert model are evaluated. A total of 98 error patterns that were generated by network simulations, and 20 error patterns that were obtained from real network traces are evaluated by both

the reference ASR and the *i-model* function. To do this, the histogram of each of the pattern of losses is fitted by a Gilbert model, obtaining thus the corresponding p and L_i parameters that the *i-model* needs as input. Finally, the W_{acc} index obtained by the actual ASR is compared with the W_{acc} obtained by the *i-model* function. As an outcome, the average prediction error of the *i-model* for the patterns generated by simulation is 0.8704 with a standard deviation of 0.0022, while the average prediction error for the real traces is 0.3384, with a standard deviation of 0.0267. The overall average error considering both sets is 0.7847 with a standard deviation of 0.0059.

This results show that the *i-model* can be used reasonably well on flows with an arbitrary histogram, by means of using the Gilbert Model parameters of p and L_i which best fits its histogram.

9.3 LOW DELAY INTERLEAVERS FOR VOIP

Since we focus our interest on the burstiness of the packet losses problem in this work, we take up again the packet interleaving approach but now by considering the new processing capabilities of the network elements. Because of intermediate network nodes can process different multimedia flows, we propose an interleaver algorithm that take advantage of that fact. Furthermore, given the router processing capabilities, the interleaver can be adapted to the network condition dynamics. We claim that our algorithm efficiently combat the bursty-error-prone nature of the Internet.

The interleaving is a preventive technique which rearranges packets from an ordered flow in such a way that, given a target packet loss burst length (s), the consecutive losses are scattered into isolated losses.

9.3.1 *Basic interleaver*

The basic operation of a traditional packet block interleaver (Type $I(s)$ interleaver), can be seen in Fig. 52.

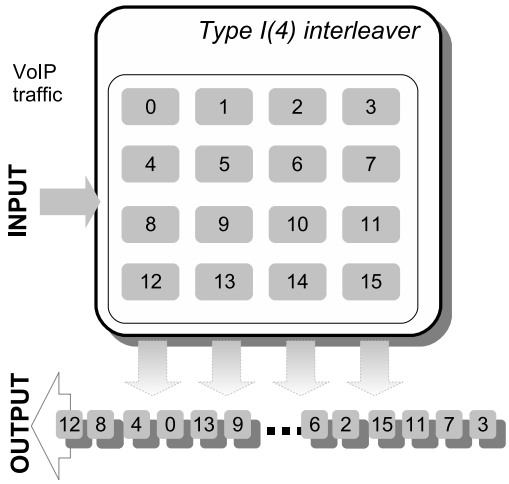


Figure 52: Example of Type I(4) Block Interleaver with $s = 4$.

As it can be seen, the interleaver rearranges the sequence of packets in such a way that s consecutives losses become isolated eliminations. To do this, it needs to store some packets for some time. For end-to-end single flow block interleavings, this technique can be only applied to scenarios in which the number of consecutive losses is $s \leqslant 4$. Otherwise, as the theoretical maximum delay exposes [100], some interleaved packets would arrived late (i.e., delay larger than 300ms) to their destination.

9.3.2 *Multiflow Packet Interleaver Type* $\mathrm{II}(n_f, s, s')$

If we take advantage of using packets from different flows to interleave them, the delay caused by the interleaving can be decreased. From that idea we have developed a new low delay Multiflow Packet Interleaver Type $\mathrm{II}(n_f, s, s')$. In particular, the Type $\mathrm{II}(n_f, s, s')$ interleaver takes packets from n_f VoIP flows with the same interpacket period t_f, scrambles them to cope with a target spread s, reserving s' rows of the internal interleaver matrix for additional background traffic.

To illustrate the Type $\mathrm{II}(n_f, s, s')$ interleaver operation, Fig. 53 shows the scheme of $\mathrm{II}(2, 4, 1)$ ($n_f = 2$ flows, maxi-

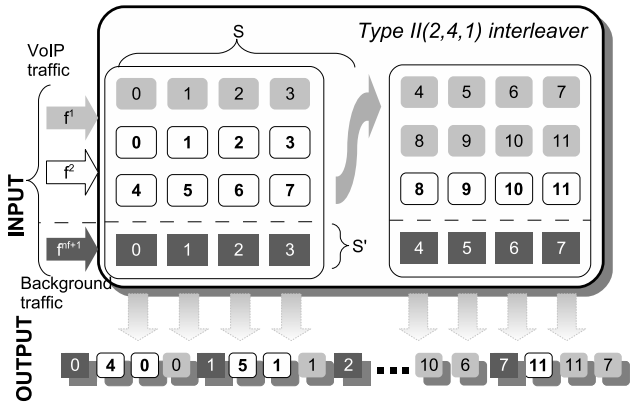


Figure 53: Multiflow interleaver $II(2,4,1)$.

mum burst length to isolate $s = 4$). As a result, the interleaved sequence is that follows:

f^3_0, f^2_4, f^2_0, f^1_0, f^3_1, f^2_5, f^2_1, f^1_1, f^3_2, f^2_6, f^2_2, f^1_2, f^3_3, f^2_7, f^2_3, f^1_3, f^3_4, f^2_8, f^1_8, f^1_4, f^3_5, f^2_9, f^1_9, f^1_5, f^3_6, f^2_{10}, f^1_{10}, f^1_6, f^3_7, f^2_{11}, f^1_{11}, f^1_7.

To describe formally the operation of the Type $II(n_f, s, s')$ interleaver, we have to provide some definitions. First, let $f^1, f^2, \ldots, f^{n_f}$ be n_f flows to interleave, and s the objective spread. Let us define several new expressions for indicating the position of every new packet inside the interleaver.

As it can be seen in Fig. 53, the basic idea of the block interleaver is to write packets into the interleaver matrix by rows (from left to right), and to read them by columns (from bottom to top, and from left to right).

For Type $II(n_f, s, s')$, the reading procedure is the same. However, for the writing procedure, the rows that can occupy each flow are established. More precisely, R^i_j is defined as the number of consecutive rows that will be assigned to flow f^i in matrix j ($i = 1, \ldots, n_f$, and $j = 1, \ldots, n_m$). The calculation of R^i_j is a two step procedure:

$$R^i_j = \lfloor \frac{(s - s')}{n_f} \rfloor,\ i = 1 \ldots n_f,\ j = 1 \ldots n_m \tag{9.3}$$

$$R^{r_{i,j}}_j = R^i_j + 1,\ i = 1 \ldots n_f,\ j = 1 \ldots n_m \tag{9.4}$$

where the $r_{i,j}$ row index can be calculated by 9.5:

$$r_{i,j} = \begin{cases} n_f - r + i & \textbf{if } i > j \\ n_f - r - j + 2 \cdot i & \textbf{if } i \leqslant j < n_f - r + i \\ i & \textbf{if } j \geqslant n_f \end{cases} \tag{9.5}$$

The r value can be obtained by 9.6

$$r = (s - s') - \lfloor \frac{s - s'}{n_f} \rfloor \cdot n_f \tag{9.6}$$

Under some circunstances, the dimension of the interleaver matrix is not $s \times s$. The number of columns needed to build the internal interleaver matrices (n_c) is:

$$n_c = \begin{cases} 1 & \textbf{if } n_f + s' \geqslant s \\ s & \textbf{otherwise} \end{cases} \tag{9.7}$$

We can also calculate the number of internal matrices n_m, and number of packets n_p needed to fullfill the interleaver (equations 9.8 and 9.9 respectively):

$$n_m = \begin{cases} 1 & \textbf{if } r = 0 \textbf{ or } n_f > s \\ n_f & \textbf{otherwise} \end{cases} \tag{9.8}$$

$$n_p = \frac{n_m \cdot n_c \cdot (s - s')}{n_f} \tag{9.9}$$

In addition to VoIP flows, further packets can be used by interleavers which shares a common path. The incoming rate of this additional traffic ($\frac{1}{t_f^{n_f+1}}$) have to obey the expression 9.10:

$$\frac{1}{t_f^{n_f+1}} \geqslant \frac{e}{n_p \cdot t_f} = \frac{n_f}{t_f} \tag{9.10}$$

9.3.3 *Maximum and average packet delay*

A critical parameter of the interleavers is the maximum delay that can suffer any packet of the flow. Since we have to provide QoS, the maximum delay d_{max} must be under $300ms$. To calculate it, we have developed the following expressions.

Let D_{max_i} be the maximum delay for any packet of flow f^i $(i = 1 \dots n_f)$ that a Type $II(n_f, s, s')$ will introduce. This value can be calculated with the following expressions:

$$D_{max_i} = U_r^{n_f} - P_r^i \tag{9.11}$$

where

$$P_j^i = \sum_{k=2}^{j} R_{(k-1)}^i \cdot s \tag{9.12}$$

and

$$U_j^i = P_j^i + (R_j^i - 1) \cdot s \tag{9.13}$$

Other figure of merit of the interleaver is the average packet delay per flow f^i, D_{avg_i}.

$$D_{avg_i} = \frac{\sum_{j=1}^{n_m} D_j^i}{n_p} \tag{9.14}$$

where

$$D_j^i = s \cdot \sum_{k=1}^{R_j^i} (U_j^{n_f} - P_j^i - (k-1) \cdot s) \tag{9.15}$$

To conclude, the maximum delay for any interleaved packet, D_{max} is:

$$D_{max} = \begin{cases} s \cdot (\lfloor \frac{s-s'}{n_f} \rfloor + r - 1) & \textbf{if } n_f \geqslant 2 \cdot r \\ s \cdot (\lfloor \frac{s-s'}{n_f} \rfloor + n_f - r) & \textbf{if } n_f < 2 \cdot r \end{cases} \tag{9.16}$$

As a remark, the maximum delay introduced to f^1 (D_{max_1}) coincides with D_{max} .

Resulting average burst length L_o

```
1  foreach burst length i in f_B(x)
2     initiate sequence out_tmp = out_0
3     // The number of packets to fill the matrices (e):
4     e = n_m · n_c · s
5     foreach index j = 1 to e
6        discard i packets from out_tmp from index j
7        deinterleave out_tmp
8        foreach flow k = 1 to n_f
9           foreach burst lenght raf of packets from f^k
10             nRáfagas[raf] = nRáfagas[raf] + 1
11             if raf > 0
12               n = n + 1
13             endif
14          endforeach
15       endforeach
16       foreach burst length m in nRáfagas
17          L_o=L_o + f_B(i) · m · nRafagas[m]/n
18       endforeach
19    endforeach
20 endforeach
```

Figure 54: Resulting L_o algorithm.

9.3.4 *Estimated average resulting burst length*

To characterize the perceptual performance of the interleaver, the estimated resulting burst length L_o has to be provided. This value is calculated by means of the algorithm exposed in Fig. 54, from the histogram of the experienced burst lengths, $f_B(x)$:

9.3.5 *Experimental validation*

To validate the proposed algorithms to estimate the interleaver Type II(n_f, s, s') characterization, providing their performance results, we obtain, by simulations, the experimental results of the *e-model* and the ASR. In addition, we

estimate the average burst length and average delay to obtain an estimation of the resulting *e-model* and W_{acc} values. To use both *e-model* R value and W_{acc} as a quality measure, we define an overall quality index, expressed in 9.17. The overall quality index, $Q_w(W_{acc}, R)$, is expressed as a lineal combination of W_{acc} y R. The $Q_w(W_{acc}, R)$ values range from $Q_w = 0$ (worst quality) to $Q_w = 1$ (best joint quality).

$$Q_w(W_{acc}, R) = \frac{w|100 - W_{acc}| + (1-w) \cdot |100 - R|}{100} \quad (9.17)$$

where w corresponds to the W_{acc} weight on $Q_w(W_{acc}, R)$. The value of w is selected according to the target QoE dimension that application demands (intelligibility vs. fidelity).

In the table 9 , the experimental value of $Q_w(W_{acc}, R)$ is calculated for every feasible interleaver on 4 scenarios. The estimated value of $\tilde{R}$ and $\tilde{Q}_{0.6}$ are also provided.

In the figure there are also two interleavers which are particular cases of the Type $\Pi(n_f, s, s')$ interleaver. These interleavers are the Type $\Pi(n_f, s)$ (equivalent to $\Pi(n_f, s, 0)$), and the Type $\Pi(n_f)$ interleaver (equivalent to the Type $\Pi(n_f, n_f, 0)$ interleaver).

As final result, please note how the difference between the experimented result $Q_{0.6}$ and the estimated $\tilde{Q}_{0.6}$ is lower than 0.005 over 1.

9.4 EARLY PACKET LOSS DETECTION AND RETRANSMISSION

To carry out automatic retransmission for delay-sensitive data such VoIP, we propose an algorithm for efficient packet loss detection services allocation in VoIP environments. We aim to increase the final user voice perceived quality. Given a path between the sender and the receiver, the algorithm will determine the optimal location of the packet loss detector by considering a subjective voice quality measure. More precisely, by using a simplified version of the *e-model*, the algorithm estimates the loss detection agent location which maximizes the expected end to end subjective quality. To

Interleaver	$\tilde{W}_{acc}$	$\tilde{R}$	Experimental $Q_{0.6}$	estimated $\tilde{Q}_{0.6}$
Scenario 1: total delay=60ms, f^{n_f+1}=50 rate (packt/s)				
II(2,6,2)	94.12	63.44	0.814	0.818
II(2,6)	94.44	61.92	0.813	0.814
II(2,5)	93.46	62.25	0.811	0.810
II(2,2)	90.18	65.11	0.803	0.801
Scenario 2: total delay =90ms, f^{n_f+1}=50 rate (packt/s)				
II(2,6,2)	94.12	62.72	0.811	0.816
II(2,5)	93.46	59.88	0.801	0.802
II(2,2)	90.18	64.39	0.800	0.799
Scenario 3: total delay =90ms, f^{n_f+1}=75.188 rate (packt/s)				
II(2,7,3)	94.71	62.18	0.813	0.817
II(2,5)	93.38	59.63	0.800	0.800
II(2,2)	90.04	64.11	0.803	0.797
Scenario 4: total delay=60ms, rate f^{n_f+1}=100 (packet/s)				
II(2,8,4)	95.22	62.59	0.817	0.822
II(2,6)	79.31	31.21	0.593	0.601
II(2,5)	93.35	61.91	0.811	0.808
II(2,2)	90.02	64.80	0.806	0.799

Table 9: Results for Q_w, in addtion to $\tilde{R}$, $\tilde{W}_{acc}$ and $\tilde{Q}_w$, in 4 scenarios.

achieve this, beside of the proposed algorithm, analytical expressions for the prediction of the loss probability and the average delay are provided. The proposed procedure is evaluated and validated by means of simulation.

The use of the active networks for packet loss recovering inside the network has been studied in ([26]). To reduce the incurred delay in the recovering procedure, some actions are achieved within the network. A very well established mechanism, mainly for multicasting transmissions, is the local packet retransmission approach ([110],[33]). These schemes use an intermediate memory (cache) of packets in one (or more) intermediate network node, which retrans-

mits the missing packets requested by the receivers which have detected any loss. In order to further reduce the delay required in the recuperation process, another mechanism can be considered. It consists in detecting the packet loss at any additional intermediate node, which will request the retransmission to the nearest cache agent.

However, in general terms, the proposed solutions (as [110], and [30]) base the packet loss detection in a simple packet sequence number checking procedure. A packet loss is detected (and the retransmission request is triggered as response) when a sequence number gap is identified, and consequently, it is assumed that the intermediate packets have been lost.

Note that this kind of detectors, referred hereafter to as DAS, exhibits a strong dependency on the burst length of consecutive losses. The longer the loss burst is, the later the detection will be identified. To reduce this potential weakness, some approaches based on arrival time prediction for each packet have been also proposed. In these schemes the time instant in which the packet should arrive is estimated. If the packet does not arrive before that time instant, a packet loss is notified. Several detectors based on the packets delays are proposed in [5]. Additionally, in this section we propose a new detection procedure based on packet delay estimations.

The recuperation agent that will be used in this work performs a packet cache, local retransmission and early packet loss detection. Two different detection modules will be considered in our study. The recuperation agent located inside the network will be hereafter referred to as NRA (highlighted in Fig. 55 with a thick dashed circle). In the simple scenario of Fig. 55, node S sends the voice packets to the node R through the programmable node NRA, where the recuperation procedure is performed. p_1 and p_2 respectively denote the packet loss probabilities, while t_{p_1} and t_{p_2} are the propagation delays.

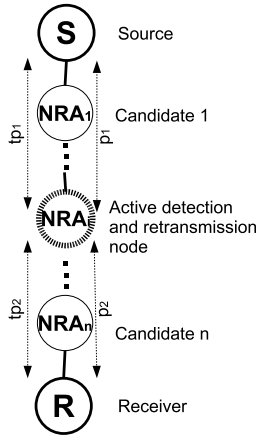


Figure 55: Scenario and topology scheme.

Packet loss detection based on an exponential smoothing delay forecasting

In this section we propose a new procedure to detect packet losses for multimedia flows. This detection technique will be referred to as *DEAT-HOLT*. Briefly, under this approach the arrival time to the detection agent node is estimated for each packet. If the packet does not arrive before the calculated deadline, the packet will be considered as a loss. Since for a given VoIP flow, the packets can suffer variable delay within the network (i.e. *jitter*), this estimation needs to be dynamically adapted the network behavior.

This approach is adopted because of the high correlation that consecutive packet delays can exhibit. In fact, we can represent the packet delay as the contribution of a long-term trend plus a short-term noise component. Therefore, if we assume this model, an exponential smoothing forecasting can be used, such as Holt's predictor [120]. More specifically, for packet number i, the estimated arrival time $D(i)$ is calculated by using the expression (9.18):

$$\begin{aligned} L(i) &= \alpha \cdot d(i-1) + (1-\alpha) \cdot (L(i-1) + s(i-1)) \\ s(i) &= \beta \cdot (L(i) - L(i-1)) + (1-\beta) \cdot s(i-1) \\ D(i) &= L(i) + s(i) \end{aligned} \tag{9.18}$$

Where $L(i)$ represents the level and $s(i)$ indicates the estimated slope. Since a noise component can be expected,

the calculated trend is modified by means of $(L(i) - L(i-1))$ and β. This term is added to the previous delay estimation $s(i-1)$, weighted by $(1-\beta)$.

In order to obtain an upper bound for the actual delay, the adaptable threshold $U_a(i)$ is calculated as shown in equation (9.19):

$$U_a(i) = D(i) + C \tag{9.19}$$

Where C is an offset component introduced to reduce the number of false alarms (premature detections of losses) due to the random delay fluctuations. When packet i arrives the detection agent node, C is updated in equation (9.20):

$$\begin{aligned} e(i) &= D(i) - d(i) \\ \delta(i) &= (1-\gamma)\cdot\delta(i-1) + \gamma\cdot|e(i)| \\ C &= \delta(i)\cdot\phi \end{aligned} \tag{9.20}$$

Values for α, β, γ and ϕ are in the range $[0,1]$. These values indicate the weight of the past values in the expressions (9.18) and (9.20).

As a final note, when previous packet is lost, and thereby its delay information is not available, equations (9.18), (9.19) and (9.20) can not be calculated. In this case, the last available C and D values are used.

$$U_a(i) = D(i-1) + C \tag{9.21}$$

The explained predictor estimates the trend of the delay, and it obtains a robust estimation in spite of the random delays introduced in the network by previous routers. Therefore, given the current network conditions, with this approach the maximum expected delay for each packet can be estimated at the NRA node without wasting spurious time and without increasing the number of false alarms.

Packet loss and delay analysis at the NRA

Since the proposed location algorithm is based on the estimated MOS score given by the *e-model*, it is necessary to calculate for each location the required *e-model* input parameters (that is to say, the average packet delay d and the probability of loss e).

For the simple VoIP scenario presented in Fig.55 we will assume that the maximum end-to-end delay for a packet (d_{max}) is 300ms; the interpacket generation period (t_f) is 20ms, the control channel does not exhibits errors, and the experienced *jitter* is negligible. As mentioned before, the link from S to NRA will have a loss probability equal to p_1 and a propagation delay tp_1, whereas the link from NRA to R will have a loss probability p_2 and a propagation delay tp_2.

In this analysis, the recuperation mechanisms will be referred to as *DAS-NRA* when the *DAS* detection agent is used, and *DEAT-HOLT-NRA* when the *DEAT-HOLT* detection agent is used.

DAS-NRA delays and loss probability

Let us define A_i as the event of i consecutive packet losses, B_j the event of a packet to be retransmitted j times from S to NRA, and C_k the event of a packet to be retransmitted k times from NRA to R. The $p(A_i \cap B_j \cap C_k)$ probability can be calculated with the expression (9.24):

$$p(A_i \cap B_j) = \begin{cases} 1 - p_1 & \text{if } i = 0 \\ p_1^{i+j} \cdot (1 - p_1)^2 & \text{if } i > 0 \end{cases} \tag{9.22}$$

$$p(C_k) = (1 - p_2) \cdot p_2^k \tag{9.23}$$

$$p(A_i \cap B_j \cap C_k) = p(A_i, B_j) \cdot p(C_k) \tag{9.24}$$

For each occurrence of the event $A_i \cap B_j \cap C_k$, there is an associated delay ($r(A_i, B_j, C_k)$), calculated in the expression (9.27):

$$r(A_i, B_j) = \begin{cases} i \cdot t_f + tp_1 \cdot (3 + 2 \cdot j) & \text{if } i > 0 \\ tp_1 & \text{if } i = 0 \end{cases} \tag{9.25}$$

$$r(C_k) = tp_2 \cdot (2 \cdot k + 1) \tag{9.26}$$

$$r(A_i, B_j, C_k) = r(A_i, B_j) + r(C_k) \tag{9.27}$$

Note that when $i = 0$, j can be only 0.

In our case, the end-to-end probability of a packet to be received before d_{max} when using the recuperation procedures is expressed in (9.28):

$$p_{success} = \sum_{i=1}^{i=i_{max}} \sum_{j=0}^{j=j_{max}} \sum_{k=0}^{k=k_{max}} p(A_i \cap B_j \cap C_k) + \sum_{k=0}^{k=k_{max}} p(A_0 \cap B_0 \cap C_k) \quad (9.28)$$

For values of i, j and k such as $r(A_i, B_j, C_k) < d_{max}$. Therefore, the loss probability of $e_{NRA-DAS}$ can be expressed as it is shown in (9.29):

$$e_{NRA-DAS} = 1 - p_{success} \quad (9.29)$$

From the equation (9.27), the maximum i_{max}, j_{max} and k_{max} values can be obtained such as $r(A_i, B_j, C_k) < d_{max}$. With those values, the number of operations needed to calculate the probability is bounded. Therefore, the average delay $d_{NRA-DAS}$ can be calculated as it is expressed in (9.30):

$$d_{NRA-DAS} = \sum_{i=0}^{i=i_{max}} \sum_{j=0}^{j=j_{max}} \sum_{k=0}^{k=k_{max}} r(A_i, B_j, C_k) \cdot \cdot p(A_i \cap B_j \cap C_k) \quad (9.30)$$

DEAT-HOLT-NRA delays and probability of loss

Let us define B_j as the event of a packet to be retransmitted j times from S to NRA, and C_k the event of a packet to be retransmitted k times from NRA to R. The $p(B_j \cap C_k)$ probability can be calculated in our case with the expression (9.33):

$$p(B_j) = (1 - p_1) \cdot p_1^j \quad (9.31)$$

$$p(C_k) = (1 - p_2) \cdot p_2^k \quad (9.32)$$

$$p(B_j \cap C_k) = (1 - p_1) \cdot p_1^j \cdot (1 - p_2) \cdot p_2^k \quad (9.33)$$

The associated delay $r(B_j, C_k)$ can be calculated with the expression (9.36):

$$r(B_j) = p_1 \cdot (2 \cdot j + 1) \quad (9.34)$$

$$r(C_k) = p_2 \cdot (2 \cdot k + 1) \quad (9.35)$$

$$r(B_j, C_k) = r(B_j) + r(C_k) \quad (9.36)$$

In our case, we express in (9.37) the end-to-end probability of a packet to be received before d_{max} when the recuperation mechanisms are used:

$$p_{success} = \sum_{j=0}^{j=j_{max}} \sum_{k=0}^{k=k_{max}} p(B_j \cap C_k)$$

For values of j and k such as $r(B_j, C_k) < d_{max}$. From (9.36), the maximum values j_{max} and k_{max} can be obtained.

Finally, the probability $e_{DEAT-HOLT-NRA}$ is shown in equation (9.37):

$$e_{DEAT-HOLT-NRA} = 1 - p_{success} \tag{9.37}$$

The average delay $d_{DEAT-HOLT-NRA}$ can be calculated by means of the expression (9.38):

$$d_{DEAT-HOLT-NRA} = \sum_{j=0}^{j=j_{max}} \sum_{k=0}^{k=k_{max}} r(B_j, C_k) \cdot \cdot p(B_j \cap C_k) \tag{9.38}$$

9.4.1 *Allocation and agent selection algorithm*

The problem to be solved is to optimally select the detection agent and its location in the topology presented in the Fig.55 such as the resulting MOS score will be maximized, when several intermediate nodes with the NRA agent are available.

Therefore, the algorithm simply consists on calculating the expected MOS score for each detector type in each available intermediate active node location. To achieve this, the propagation delay from the sender to the intermediate node (tp_1) and from the intermediate node NRA to the receiver (tp_2), as well as the packet loss probabilities for each link (p_1 and p_2 respectively), must be known.

When all the nodes have calculated their MOS value, an ordered list is built. The nodes with a MOS score higher than $MOS_{max} - U_{MOS}$ can be selected. MOS_{max} is the maximum MOS score obtained, and U_{MOS} indicates the tolerated quality degradation related to the maximum. We suggest

that $U_{MOS} \leqslant 0.5$, since greater values could degrade the solution excessively, compared with the optimum solution.

The selection from the possible solutions will be made according the resources availability at that node, as well as the cost of the migration from current active node to the new selected active node. Other criteria to select a node from the sublist of valid nodes can be considered.

The algorithm should be performed every time that p_i or tp_i suffer significant changes.

9.4.2 *Experimental results*

The performance evaluation of the proposed algorithm has been obtained after running some simulations. In the adopted topology, we will consider a number of programmable nodes between the sender and the receiver that will be able to perform the recuperation mechanism.

Several scenarios are generated specifying different values for the end-to-end loss rate (p) and end-to-end propagation delay t_p. In addition, the number of intermediate programmable nodes is also provided. In so doing, a number of propagation delays between intermediate nodes distributions, and a number of per link packets loss rates are generated for each simulated scenario.

Once the scenario has been built, the expected MOS score is estimated for each node according to the expressions described in section 9.4 for the two proposed packet loss detection techniques (*DEAT-HOLT* and *DAS*).

For each different scenario, and for each combination of programmable node and detection algorithm, a simulation is conducted to obtain the actual optimum node and detection procedure for that scenario.

In each scenario we compare the selection made by the algorithm with the simulations results. The algorithm performance is therefore checked by calculating the difference between the maximum MOS score obtained by simulation and the MOS score obtained in simulation at the node selected by the proposed algorithm.

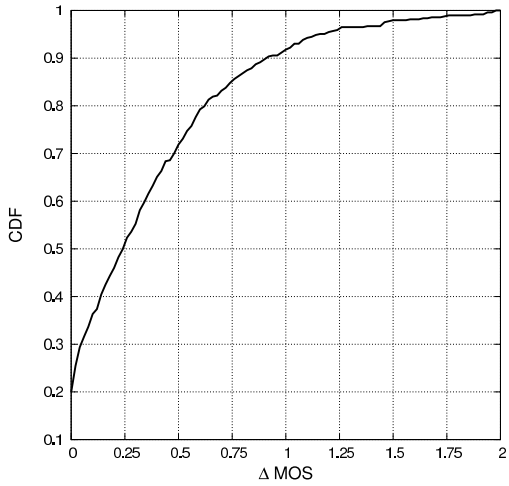


Figure 56: Average distance to the maximum MOS CDF for the simulated scenarios.

9.4.3 *Simulation parameters*

A total of 487 scenarios were generated, with $t_p = \{20ms, 50ms, 80ms, 100ms, 200ms\}$ and p with values ranging from 0.1 to 0.8 with a step of 0.1. In each scenario, several simulations were conducted with different number of intermediate nodes (from 4 to 6) randomly distributed.

To measure the relevance of one scenario, the average difference between the expected MOS for each node and the maximum expected MOS is taken into account. The higher this value is, the higher the impact of a proper selection will be in the perceived quality. The 30% of the conducted simulations exhibit an average distance greater than 0.5, which is a subjective significant difference in the MOS scale.

In Fig.56, the cumulative distribution function (CDF) for the MOS average distance is shown for every scenario. As we can see, the 50% of the scenarios present an average distance equal to 0.25, and almost the 10% exhibit a value greater than 1 in the MOS score scale.

9.4.4 *Simulations results*

In order to characterize the selection algorithm performance, the distance between the best MOS value obtained for the simulated scenario and the MOS value obtained in the selected node is computed. Therefore, the difference between both MOS scores must be lower than U_{MOS} for the selection algorithm to be valuable and to result in near optimal solutions.

As a result, the cumulative distribution of the distances of the MOS values obtained in previous scenarios is depicted in the Fig.57. The percentage of differences less or equal than a given error is depicted in the figure, where the vertical axis represents the cumulative probability, and the horizontal axis represents the MOS score differences between the optimal and the algorithm solutions.

It can be seen that the 95% of the algorithm selections achieves a difference lower than 0.05 in the MOS score scale, and less than the 1.5% of the selections generate an error between 0.1 and 0.2 points in the MOS score.

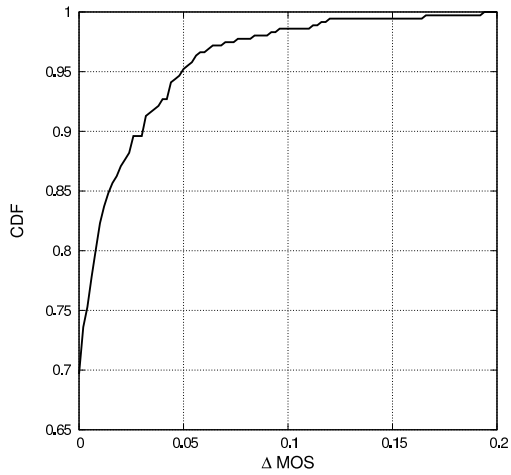


Figure 57: Resulting MOS distances CDF for the algorithm.

9.5 A FRAMEWORK FOR PROTOCOL SELECTION

Currently, a protocol tends to be considered as more evolved than its predecessor if it is applicable in more contexts and situations. In our opinion, this leads to a wrong filter in protocol and service design and suboptimal network operations because it establishes an artificially high barrier against better, although perhaps specialized, solutions.

Resilience to protocol failures or misbehavior is of paramount importance in this context, as a full model comprising the behaviors of all possible protocols would be unrealistic and cannot be counted upon. The framework is thus based on the premise of doing continuous experiments on running protocols, by constantly monitoring their behavior, measuring their performance, and selecting the most suitable protocols accordingly.

The approach of *protocol, service and server switching* is not new and is increasingly applied in the Internet.

One possibility to select protocols automatically in a dynamic way is to run a productive or current best practice version in parallel with an experimental, potentially better version and continuously compare their performance. The productive version actually delivers the service, and switching to the experimental one only occurs when the latter becomes clearly superior to the former. This approach obviously consumes extra resources, since parallel, alternative trial channels are maintained; but it could be regarded as a price to pay for robustness.

In an ideal networking world, choosing among two potential services could be based on full knowledge of the factors that determine the services' fitness for a given task and situation. This knowledge can be gained through analytic work and/or through past experiences. Our hypothesis is that a complete modeling will less and less be possible, that network state can not always be sufficiently gathered, and that not all implementations will adhere to the supposed model anyway. Online protocol experiments are a necessary addition to existing assessment methods and can even be used in cases where no model is available.

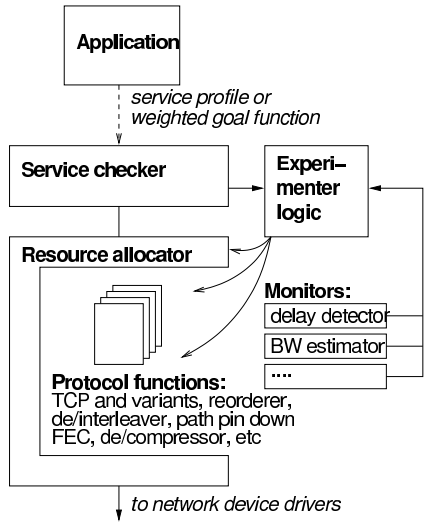


Figure 58: Architectural framework for online protocol evaluation and switching.

9.5.1 *Framework*

Fig. 58 shows the elements needed for implementing a framework for online service evaluation and selection. A sample application is displayed on top. Every application using the framework has its own instance of the protocol selection framework. Therefore each instance can be adapted to the specific requirements of an application.

The elements of the framework are:

- *Protocol functions:* Pool of protocol candidates from which our system can choose. Depending on the granularity we are looking at, these are complete stacks, single protocols, sub-protocol functions like stream splitters, as well as access to remote service instances.

  In order to perform the components selection, the protocol functions are grouped into services. A service will be defined as a well-known set of tasks, with a well-known interface. Each alternative protocol implementation must adhere to the proper service specification, in order to be eligible as a candidate in the test round. Service profiles include an input, output, and monitoring interface. While input and output interfaces rule the interaction with the application and

the underlaying services, monitoring interfaces export performance parameters (such as correct packets received) to the service checker. This profile also states the functional requirements for the service (it is to say, which support functions are needed). For instance, a reliable transport service may require a system function to calculate an error correction code.

- *Monitor helpers:* Measure parameters from the network and from running protocol instances and provide this information to the experimenter. The monitors helpers are dynamic entities that can be activated or deactivated on demand. Moreover, new monitors can be added to the framework, as long as they provide a proper interface description.

- *Service checker:* Verifies whether a given service complies to the application request, signaling any deviations to the experimenter. To this end, it uses a fitness function which measures the performance of each protocol function. It may be a combination of objective (goals and observable performance parameters) and subjective (perceived quality of service) components. Since the criteria of the service evaluation are determined by the particular requesting application and user's preferences, the fitness function should be provided by the service petitioner. The fitness function can use any monitored parameter to perform its calculations, as well as utility functions which estimate the quality of experience that the user would perceive (e.g. the *e-model* for VoIP services [78]). The checker can be either passive (just assessing by looking at the flow of data) or active by collaborating with external entities and/or adding its own protocol fields to the application data.

- *Experimenter logic:* Evaluates the running protocols according to the feedback provided by the service checker and monitors, and makes decisions on how to reconfigure the running experiments properly. Possible actions include launch, stop, restart, or reconfigure

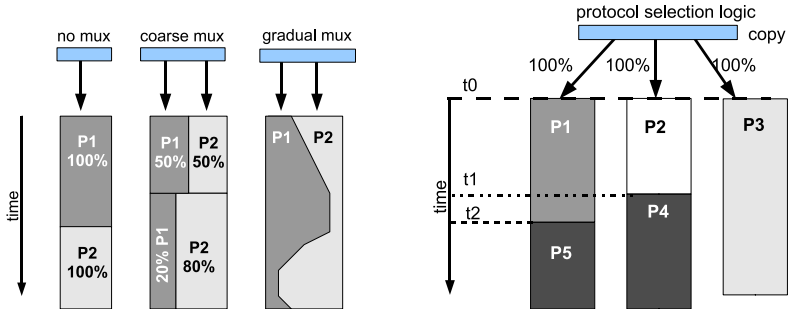
protocols with different parameters. The application may define the percentage time allowed to perform the experiments. Consequently, the experimenter logic can stop a test which breaks the imposed constraints.

- *Resource allocator:* Executes resource policies such as splitting downstack data streams to two transport paths or imposing rate limitations. To carry out reservations when needed, each protocol function may state its resource requirements.

9.5.2 *Runtime inspection and decision*

At the core of the framework lies the experimenter decision logic which modifies the configuration of the experiments based on the feedback provided by the monitors and checkers. Applications request services either by choosing the appropriate service checker among predefined categories, or by specifying a target service profile (delay, loss, downtime, bandwidth, each parameter potentially associated with a weight). This reminds us of existing QoS approaches. An important difference is that here the experimenter makes its best to achieve the desired goals by probing multiple alternatives, without an admission control system or similar.

The experimenter starts the service with a first rough choice of at least one production protocol, and optionally one or more experimental ones. The production protocol is typically the most mature among the existing options, or the one judged most suitable to the application, by static knowledge. The experimenter then monitors the "network weather" [132] and can proactively switch protocols, otherwise it periodically evaluates the competition performance and potentially switches to the competitor. The change of a performance variable may be also the trigger of a switch. It is also possible to use two or more variants in parallel and shift traffic ratios smoothly. Fig. 59 shows how parallel and serial experiments can be performed. Fig. 59(a) shows a typical parallel experiment. At time t_0 the experiment starts with three protocols in parallel: P_1, P_2, and P_3. Each



(a) parallel experiments (b) serial experiments

Figure 59: Several cases of parallel (a) and serial (b) experiments

protocol receives a full copy of the traffic, leading to redundancy which is used for fault tolerance: if one of the protocols misbehaves, the service is still delivered by one of the others. At t_1 the experimenter decides that it is better to replace P_2 by P_4, and at t_2 P_1 is replaced by P_5. Fig. 59(b) shows three possibilities for serial experiments: the first one (left side) is the simplest case, where only one protocol is used at a given time, and the experimenter switches protocols in response to observations from the monitors and the checker. In the second case (middle) a coarse multiplexing of protocols is performed, with part of the traffic sent to one protocol and part to the other. The shares of the different protocols vary in coarse chunks. The third case (right side) shows a fine-grain, gradual multiplexing scheme in which the percentage of traffic that goes to each protocol is adjusted dynamically.

The choice of the observation and switching period is crucial: too short observations do not provide sufficient statistical information, and too long observations provide slow feedback to switching decisions. Furthermore, a too short switching period does not leave sufficient time for new protocols to initialize and adapt to the new environment, does not capitalize on the transition time spent, and could lead to undesirable performance oscillations. In contrast, a too long switching period would result in slow adaptation.

The duration of an individual test can be bound from the application requirements profile, in which the user can specify a maximum threshold of the service degradation

during the test stage. This quota can vary depending on the importance and priority of the requested service. The minimum duration of the test must be greater than the protocol initialization delay. The maximum duration will be bound by the total grant for the test stage. To determine the optimal duration for an individual protocol test, it is necessary to measure the time that a protocol needs to stabilize its performance, so that it can be characterized. The stabilization of the protocol operation could be detected by statistical analysis. For instance, the Kolmogorov-Smirnov test could be used to verify whether current and past performance samples belong to the same statistical distribution, assessing that the protocol operation impact remains stable. Similarly, this kind of statistic tools could be employed to determine whether the network conditions change.

9.5.3 *Case Study: Switching Interleavers in Audio Streams*

Although the proposed framework is mainly targeted to autonomous systems in which protocol switching is a long term target, in this section we present a scenario which shows how the switching procedure can be applied to applications which need to adapt in a short period of time.

The so called *multiflow block interleaver* type $\Pi(n_f, s)$ [8] is a special kind of block interleaver which takes packets from several VoIP sessions and outputs an aggregate stream with the desired isolation capabilities, introducing the minimum theorical delay.

In addition to the number of available flows (n_f) to perform the interleaving, the device needs as input the target burst length (s), that is to say, the maximum burst length that it will isolate. For a given pair of n_f and s, the resulting device provides the minimum delay interleaving with a maximum of $d_{n_f,s}$ seconds of end-to-end delay. Such delay is not monotonically decreasing with the target burst length.

Instead, the $d_{n_f,s}$ delay introduced by the interleaving follows a non monotonic trend. The calculation of that delay is shown in detail in [8]. In our experiments we will activate an interleaver that has to switch the s target parameter to

the actual burst lengths in order to optimize the resulting packet loss distributions and the incurred delay.

Such selection has to be made regarding that the maximum delay must be lower that D_{max}.

9.5.4 *Framework elements*

Several LAIN framework elements had to be instantiated for our simulations.

Monitor helpers

The implemented monitor checks whether the packet loss pattern changes or keeps stable. To this end, the monitor at the receiver side logs the burst distribution of the transmission path by means of maintaining a histogram of the experienced packet losses.

Periodically, the loss monitor sends the updated statistics of the link loss pattern and the average packet delay (d) to the experimenter. More precisely, the receiver side of the monitor sends back to the source the 80^{th} percentile of the receiver cumulative distribution function (CDF) of the burst length. The meaning of this value is that the 80% of the experimented bursts of losses are equal or lower than this score, noted as $b_{80^{th}}$ hereafter.

Finally, the histogram is reset each time that a switch is performed or a test round is triggered.

Service Checker

The service checker for this experiment uses a fitness function $\Phi_{d,b_{80^{th}}}$ which considers the average burst length and the experimented end-to-end delay, (eq. 9.39):

$$\Phi_{d,b_{80^{th}}} = \frac{1}{b_{80^{th}}} \cdot \frac{D_{max} - d}{D_{max}} \tag{9.39}$$

Where d is the average packet delay, D_{max} is the maximum tolerated end-to-end delay (300 ms in our case), and $b_{80^{th}}$ is the 80^{th} percentile of the experimented burst length CDF.

A candidate performs better than another if its fitness score is greater than the other candidate. The fitness score

is thus designed to highlight those results with a final short average burst length and shorter delays.

Experimenter logic

Each time the network loss distribution changes, the test tournament is triggered. Also, periodic tests tournaments are held each 75 seconds. At the beginning of the test, a number of interleaver candidates are generated. Such candidates must verify that their maximum delay is lower than the maximum threshold D_{max}. Their input parameters are $n_f = 4$ and s (which is chosen to be close to the $b_{80^{th}}$ value reported by the monitor helper).

In our simulations, only the 3 candidates (out of 21) with the shortest distance to the $b_{80^{th}}$ value are tested. They are activated for a period of 5 seconds, registering their fitness scores. After the test round, the one with the best fitness value is selected, and the production interleaver is substituted by switching.

A test round is also triggered when a pattern change is detected. A change of network's conditions stability is detected when one of the conditions expressed in Eq. 9.40 below is verified:

$$\lceil b_{80^{th},i-1} + 1.5 \rceil < b_{80^{th},i} < \lfloor b_{80^{th},i-1} - 1.5 \rfloor \qquad (9.40)$$

where $b_{80^{th},i}$ is the most recently reported value, and $b_{80^{th},i-1}$ is the last stable period value. The current $b_{80^{th},i}$ is recalculated after every update report by the smoothing average detailed in the equation 9.41:

$$b_{80^{th},i-1} = 0.9 \cdot b_{80^{th},i-1} + 0.1 \cdot b_{80^{th},i} \qquad (9.41)$$

It is worth noting that the number of candidates, their selection order, the period of the per protocol tests and the frequency of the tournament test are key parameters which have to be carefully selected. Due to the exploratory nature of our experiments, those parameters have been selected to be simple, and therefore not necessarily optimized.

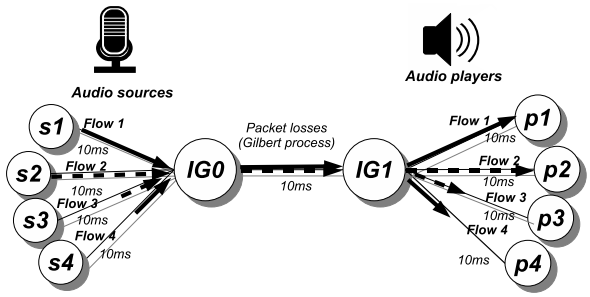


Figure 60: Interleaving scenario topology

9.5.5 *Experimental setup*

This scenario consists of 4 audio sources which send packets through a common path. The path is subject to a changing loss pattern generated by a *Gilbert model* [97].

Fig. 60 shows the simulated topology. It consists of 4 source nodes which originate the audio streams, 4 sink nodes which represent the listeners, and 2 common nodes, labeled IG0 and IG1, which are connected to all the streams via the common link where errors occur. The source nodes produce audio packets with a sending rate of 50 packets per second. The so called *Type* $\mathrm{II}(4, s)$ *Multiflow block Interleaver* is located at node IG0.

In order to assess the perceived quality of voice transmissions, the *e-model* has been proposed by the ITU-T [78]. The *e-model* provides an estimation of the Mean Opinion Score (MOS) of the voice quality that real users would assess in a MOS test. The MOS scale ranges from 1 ("bad quality") to 5 ("excellent quality") [67]. A drawback of the *e-model* is that, although it estimates the voice quality, it does not completely model all the involved perceptual factors. In fact, the *e-model* does not capture the bursty behavior of losses (it only uses the overall probability of loss), and therefore it is not sufficient to assess the performance of the interleaving scheme. To provide a complete quality score which takes into account all the impairments which affect the perceived quality, the average resulting burst length L_o and the average packet delay μ_d, key parameters to assess a VoIP stream, will also be provided in our evaluations.

9.5.6 *Serial on-line testing performance*

The evaluation is held in a scenario where the error patterns change in time. The switching procedure has to result in the adaptation of the interleaver to the given conditions, in order to obtain better performance (and perceived quality) than in a classical approach.

The evaluation will be provided with the MOS score obtained by the application of the *e-model* and the average per packet delays and resulting burst lengths.

In the experiment with the fixed $TypeII(4, s)$ interleaver, the target burst length (s) is set at the beginning of the simulations. In order to compare the static strategy with the adaptive approach, several simulations with different values for s from 5 through 13, which are the possible values within the time restrictions, are conducted to find the best result with a static strategy.

The network loss properties change during the voice over IP session. More precisely, from seconds 0.001 to 500 the Gilbert model is characterized by $p_{01} = 0.03571$ and $p_{10} = 0.08334$, resulting in an overall probability of loss $p = 0.3$ and an average burst length $L_i = 12$. From seconds 500 to 1000, $p_{01} = 0.0125$, and $p_{10} = 0.05$, resulting in $p = 0.2$ and $L_i = 20$.

Table 10 presents the results for these simulations. The MOS score obtained by the *e-model*, the average burst length L_o and the average per packet delay μ_d are shown for the different possible s values verifying the D_{max} deadline constraint. To obtain the same MOS score and L_o value as the LAIN approach with the fixed scheme, at least $s = 6$ must be chosen. However, the average introduced delay μ_d is 0.019 seconds greater. To obtain a lower delay, $s = 5$ could be chosen for the fixed approach, but the average burst length $L_o = 3.31$ exceeds in 0.44 the value obtained by our LAIN approach. Note that for a high probability of loss (as in this case, $p = 0.3$), even small differences in the average burst length have a great impact on the perceived quality.

Scheme	**s**	**MOS**	**L_o**	μ_d (sec.)
LAIN	[4 – 13]	**2.33**	**2.87**	**0.099**
Fixed	5	2.35	3.31	0.080
	6	**2.31**	**2.85**	**0.120**
	7	2.29	2.58	0.135
	8	2.32	2.38	0.110
	9	2.01	2.24	0.211
	10	1.59	2.11	0.280
	12	1.65	1.94	0.270

Table 10: Results for changing network conditions

9.6 CONCLUSIONS

Please refer to chapter 8, to read the main conclusions extracted from the work of the thesis.

A

CARACTERIZACIÓN DEL MODELO DE GILBERT

El modelo de Gilbert [] ha sido ampliamente empleado para representar las pérdidas de paquetes en ráfagas en distintos estudios [] [] []. El modelo de Gilbert se representa mediante una cadena de Markov de dos estados, representando uno de ellos el estado libre de pérdidas, y el otro el estado de pérdidas a ráfagas. Del estado libre de errores (etiquetado con 0) ser pasa al estado que representa el periodo de pérdidas (etiquetado con 1) con una probabilidad p_{01}. La transición inversa se da con una probabilidad p_{10}. Con probabilidad $p_{11} = 1 - p_{10}$ se mantiene en el estado de pérdidas, y con probabilidad $p_{00} = 1 - p_{01}$ en el estado libre de pérdidas.

En la figura 61 se muestra el modelo de Gilbert de dos estados.

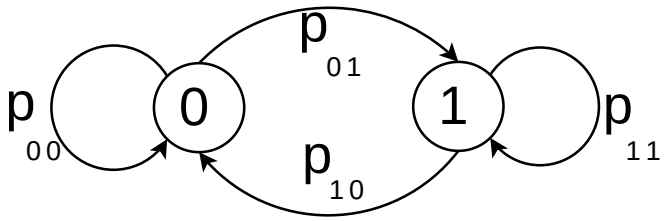


Figure 61: Gilbert model

Distribución de pérdidas de un modelo de Gilbert

La función de distribución de ráfagas de pérdidas $h(i)$ para un modelo de Gilbert con parámetros p_{10} y $p_0 1$, es la siguiente:

$$h(i) = (1 - p_{10})^{i-1} \cdot p_{10} \tag{A.1}$$

Port tanto, la distribución acumulada (CDF) de ráfagas de pérdidas $H(i)$ puede calcularse como:

$$H(i) = \sum_{j=0}^{i-1} (1-p_{10})^j \cdot p_{10} \tag{A.2}$$

$$= p_{10} \cdot \sum_{j=0}^{i-1} (1-p10)^j \tag{A.3}$$

$$= p_{10} \cdot \frac{(1-p_{10})^i - 1}{-p_{10}} \tag{A.4}$$

$$= 1-(1-p_{10})^i \tag{A.5}$$

Ajuste de una PDF mediante el modelo de Gilbert

Mediante el ajuste por mínimos cuadrados podemos obtener los parámetros a y b para la PDF de que mejor ajsute a una distribución dada.

Para ello es necesario realizar una transformación de la expresión original. Dada la ecuación de la recta:

$$y = x \cdot M + C \tag{A.6}$$

Aplicando el logaritmo sobre la expresión A.1, se obtiene la expresión mostrada en A.7:

$$\begin{aligned} ln[h(i)] &= ln[(1-p_{10})^{i-1} \cdot p_{10}] \\ &= (i-1) \cdot ln[(1-p_{10})] + ln[(p_{10})] \end{aligned} \tag{A.7}$$

Realizando la siguiente asignación:

$$x_i = i \tag{A.8}$$

$$y_i = ln(h(i)) \tag{A.9}$$

$$M = ln[(1-p_{10})] \tag{A.10}$$

$$C = ln[p_{10}] - ln[1-p_{10}] \tag{A.11}$$

Donde x_i se refiere al tamaño de ráfaga ($i = 1, 2, 3, \ldots$), e y_i representará la probabilidad de ocurrencia de ráfagas de i paquetes.

Los valores óptimos de C y M para el anterior ajuste se calculan de la siguiente manera:

$$M = \frac{(\sum_{i=1}^{i=n} X_i \cdot Y_i) - n \cdot \overline{X} \cdot \overline{Y}}{\sum_{i=1}^{i=n} {X_i}^2 - n \cdot \overline{X}^2} \tag{A.12}$$

$$C = \frac{\overline{Y} \cdot (\sum_{i=1}^{i=n} {X_i}^2) - \overline{X} \cdot (\sum_{i=1}^{i=n} X_i \cdot Y_i)}{\sum_{i=1}^{i=n} {X_i}^2 - n \cdot \overline{X}^2} \tag{A.13}$$

Por último, el valor p_{10} del modelo de Gilbert se extrae mediante la expresión A.14:

$$p_{10} = e^{C+M} \tag{A.14}$$

El otro valor requerido para construir el modelo de Gilbert, p_{01}, se puede calcular mediante la ecuación A.15, y el valor p de la traza modelada:

$$p_{01} = \frac{p \cdot p_{10}}{1 - p} \tag{A.15}$$

BIBLIOGRAFÍA

[1] John Wroclawski et al. Integrated services (intserv) charter, 2000. (Cited on page 1.)

[2] Brian Carpenter et al. Differentiated services (diffserv) charter, 2003. (Cited on page 1.)

[3] Antonio M. Peinado Victoria Sánchez Angel M. Gómez, Juan J. Ramos-Muñoz. Multi-flow block interleaving applied to distributed speech recognition over ip networks. In *ICSLP 2006*, 2006. (Cited on pages 4 and 100.)

[4] Juna Manuel López Soler Juan José Ramos Muñoz, Angel Manuel Gómez García. Intelligibility evaluation of a voip multi-flow block interleaver. In *IN PROCEEDINGS OF THE SEVENTH ANNUAL INTERNATIONAL WORKING CONFERENCE ON ACTIVE AND PROGRAMMABLE NETWORKS (IWAN 2005)*, 2005. (Cited on pages 4 and 46.)

[5] Juan Manuel López Soler Juan José amos Muñoz, Juan Francisco Núñez Negrillo. Detección activa de pérdidas de paquetes en flujos de audio en tiempo real. In *IV Jornadas de Ingeniería Telemá tica (JITEL'03)*. (Cited on pages 4 and 202.)

[6] Juan J. Ramos-Munoz and Juan M. Lopez-Soler. Efficient allocation for voip packet losses detection services in programmable and active networks. In *ICAS-ICNS '05: Proceedings of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services*, page 52, Washington, DC, USA, 2005. IEEE Computer Society. (Cited on page 4.)

[7] JUAN MANUEL LÓPEZ SOLER JUAN JOSÉ RAMOS MUÑOZ. UbicaciÓn eficiente de servicios de

detecciÓn de pÉrdidas para voip en redes programables y de recubrimiento. In *LIBRO DE PONENCIAS: V JORNADAS DE INGENIERÍA TELEMÁTICA (Jitel 2005)*, 2005. (Cited on page 4.)

[8] J.J. Ramos Munoz and J.M. López Soler. Low delay multiflow block interleavers for real-time audio. *Lecture Notes in Computer Science*, 3420/2005, 2005. (Cited on pages 4, 60, and 216.)

[9] Juan J. Ramos-Munoz, Lidia Yamamoto, and Christian Tschudin. Serial experiments online. *SIGCOMM Comput. Commun. Rev.*, 38(2):31–42, 2008. (Cited on page 4.)

[10] Skype. (Cited on pages 8 and 140.)

[11] G. Carle and E. Biersack. Survey of error recovery techniques for ip-based audio-visual multicast applications, 1997. (Cited on pages 9, 14, and 20.)

[12] Rec. g. g.114 - one-way transmission time. (Cited on pages 9 and 11.)

[13] C. Perkins, O. Hodson, and V. Hardman. A survey of packet-loss recovery techniques for streaming audio. *IEEE Network, September/October*, pages 40–48, 1998. (Cited on pages 9, 20, 22, and 23.)

[14] User datagram protocol (rfc 768), August 1980. (Cited on pages 9 and 155.)

[15] Itu-t recommendation g.711. pulse code modulation (pcm) of voice frequencies, November 1988. (Cited on pages 10 and 82.)

[16] Ramachandran Ramjee, James F. Kurose, Donald F. Towsley, and Henning Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *INFOCOM (2)*, pages 680–688, Toronto, Canada, June 1994. (Cited on pages 13 and 115.)

[17] Jonathan Rosenberg, Lili Qiu, and Henning Schulzrinne. Integrating packet FEC into adaptive voice playout buffer algorithms on the internet. In *INFOCOM (3)*, pages 1705–1714, 2000. (Cited on page 13.)

[18] S. B. Moon, J. Kurose, and D. Towsley. Packet audio playout delay adjustment: Performance bounds and algorithms. *ACM/Springer Multimedia Systems*, January 1998. (Cited on pages 13 and 141.)

[19] C.J. Sreenan, Jyh-Cheng Chen, P. Agrawal, and B. Narendran. Delay reduction techniques for playout buffering. *Multimedia, IEEE Transactions on*, 2(2):88–100, Jun 2000. (Cited on pages 13 and 113.)

[20] V. Paxson. End-to-end internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277–292, 1999. (Cited on pages 13, 14, 105, and 120.)

[21] S. Casner R. Frederick V. Jacobson Audio-Video Transport Working Group, H. Schulzrinne. Rfc 1889. rtp: A transport protocol for real-time applications. (Cited on pages 13 and 14.)

[22] Henning Schulzrinne Wenyu Jiang. Modeling of packet loss and delay and their efect on real-time multimedia service quality. (Cited on page 14.)

[23] Alan Clark. Modeling the effects of burst packet loss and recency on subjective voice quality. In *Proceedings of IP Telephony Workshop*, 2001. (Cited on page 14.)

[24] Mark Handley Anna Watson Vicky Hardman, Martina Angela Sasse. Reliable audio for use over internet. In *Proc. INET'95*, 1995. (Cited on pages 14, 23, 29, and 32.)

[25] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance v.1, 1993. (Cited on page 14.)

[26] David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall, and Gary J. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, 1997. (Cited on pages 15, 16, 17, 60, 103, 174, 190, and 201.)

[27] David Wetherall. Experience with a capsule-based active network. (Cited on page 15.)

[28] K. Psounis. Active networks: Applications, security, safety, and architectures. *IEEE Communications Surveys*, 2(1):2–16, 1999. (Cited on page 15.)

[29] D. Doval and D. O'Mahony. Overlay networks : A scalable alternative for p2p. In *IEEE Internet Computing*, volume 7, pages 2–5, July 2003. (Cited on pages 15, 60, and 190.)

[30] C. D. Pham M. Maimour. A loss detection service for active reliable multicast protocols. In *Proceedings of the International Network Conference (INC'2002)*, 2002. (Cited on pages 15, 104, 132, and 202.)

[31] Long Le, Henning Sanneck, Georg Carle, and Tohru Hoshi. Active concealment for internet speech transmission. In *Second International Working Conference on Active Networks IWAN*, pages 239–248, 2000. (Cited on pages 15 and 24.)

[32] Georg Carle, Henning Sanneck, Sebastian Zander, and Long Le. Deploying an active voice application on a three-level active network node architecture. *Lecture Notes in Computer Science*, 2207:65–83, 2001. (Cited on pages 15, 25, and 190.)

[33] W.; Tschudin C.; Turau V Banchs, A.; Effelsberg. Multicasting multimedia streams with active networks. In *Proceedings of the 23rd Annual Conference on Local Computer Networks (LCN)*, pages 150–159, Lowell, MA, USA, October 1998. (Cited on pages 15, 17, 31, 103, 190, and 201.)

[34] Andrew T. Campbell, Herman G. De Meer, Michael E. Kounavis, Kazuho Miki, John B. Vicente, , and Daniel Villela. A survey of programmable networks. *ACM SIGCOMM Computer Communications Review*, 29(2):7–23, April 1999. (Cited on pages 16 and 103.)

[35] Danny Raz and Yuval Shavitt. New models and algorithms for programmable networks. *Computer Networks (Amsterdam, Netherlands: 1999)*, 38(3):311–326, 2002. (Cited on page 16.)

[36] D. Wetherall, J. Guttag, and D. Tennenhouse. Ants: A toolkit for building and dynamically deploying network protocols, 1998. (Cited on pages 16 and 190.)

[37] D. Towsley, J. Kurose, and S. Pingali. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. *IEEE Journal on Selected Areas in Communications*, 15(3):398–406, 1997. (Cited on pages 19 and 60.)

[38] M. Maimour and C.D. Pham. Dynamic replier active reliable multicast. In *Proceedings ISCC 2002. Seventh International Symposium on Computers and Communications*, pages 275–282. IEEE Computer Society, 2002. (Cited on pages 19 and 20.)

[39] M.; Sedano Ruiz M.; Azcorra Salona, A.; Calderon Pastor. A strategy for comparing reliable multicast protocols applied to RMNP and CTES. In *IEEE Conference on Protocols for Multimedia Systems - Multimedia Networking.*, pages 46–55, Santiago de Chile, Chile, 1997. (Cited on page 19.)

[40] M. Schwartz S. Pejhan and D. Anastassiou. Error control using retransmission schemes in multicast transport protocols for real-time media. *IEEE/ACM Transactions on Networking*, 4(3):413–427, June 1996. (Cited on page 20.)

[41] Neda Nikaein and Christian Bonnet. QoS-based adaptive error control for wireless networks. *WIRELESS*

COMMUNICATIONS AND MOBILE COMPUTING, in press, 2002. (Cited on page 20.)

[42] X. Xu, A. Myers, H. Zhang, and R. Yavatkar. Resilient multicast support for continuous-media applications. St. Louis, Missouri, USA, May 1997. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV). (Cited on pages 20, 22, 102, and 103.)

[43] A.C. Weaver W.T. Strayer, B.J. Dempsey. *XTP, THE XPRESS TRANSFER PROTOCOL*. Addison-Wesley Publishing Company, 1992. (Cited on page 21.)

[44] John C. Lin and Sanjoy Paul. RMTP: A reliable multicast transport protocol. In *INFOCOM*, pages 1414–1424, San Francisco, CA, March 1996. (Cited on page 21.)

[45] C. Liu S. McCanne L. Zhang S. Floyd, V. Jacobson. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 1997. (Cited on pages 21 and 22.)

[46] Dianne Kiwior el al. Protocol on active networks for adaptive multicast applications. web site. http://www.tascnets.com/panama. (Cited on page 22.)

[47] Ming-Syan Chen Wen-Tsai Liao, Jeng-Chun Chen. Adaptive recovery techniques for real-time audio streams. In *Proc. of IEEE INFOCOM 2001*, pages 815–823, 2001. (Cited on page 24.)

[48] Nguyen Tuong Long Le. Development of a loss-resilient internet speech transmission method. Master's thesis, Department of Electrical Engineering at the Technical University of Berlin, 1999. (Cited on page 24.)

[49] B. Wah and D.Lin. Transformation-based reconstruction for real-time voice transmissions over the interne.

IEEE Transactions on Multimedia, 1:342–351, December 1999. (Cited on page 25.)

[50] K. Bern Younes H. Sanneck, A. Stenger and B. Girod. A new technique for audio packet loss concealment. In *IEEE Global Internet*, pages 48–52, 1996. (Cited on page 25.)

[51] Y. L. Chen and B. S. Chen. Model-based multirate representation of speech signals and its aplication to recovery of missing speech packets. *IEEE Transactions on Speech and Audio Processing*, 15(3):220–231, May 1997. (Cited on page 26.)

[52] Srivatsan Varadarajan Hung Q. Ngo and Jaideep Srivastava. Error spreading: Reducing bursty errors in continuous media streaming. In *Proceedings of IEEE Multimedia Systems '99 (ICMCS)*, pages 314–319, 1999. (Cited on page 27.)

[53] Jaideep Srivastava Srivatsan Varadarajan, Hung Q. Ngo. Error spreading: A perception-driven approach to handling error in continuous media streaming. *IEEE/ACM Transactions on Networking*, 10(1):139–152, 2002. (Cited on page 27.)

[54] E. Steinbach Y. Liang and B. Girod. Real-time voice communication over the internet using packet path diversity. In *Proc. ACM Multimedia '01*, 2001. citeseer.nj.nec.com/liang01realtime.html. (Cited on page 28.)

[55] Y. Liang, E. Steinbach, and B. Girod. Multistream voice over ip using packet path diversity. In *In Proceedings of IEEE 4th Workshop on Multimedia Signal Processing*, 2001. citeseer.nj.nec.com/liang01multistream.html. (Cited on page 28.)

[56] Thinh Nguyen and Avideh Zakhor. Path diversity with forward error correction (pdf) system for packet switched networks. (Cited on page 28.)

[57] Andrés Vega-García Jean-Chrysostome Bolot. The case for fec-based error control for packet audio in the internet. *to appear in ACM Multimedia Systems*, 1997. (Cited on page 29.)

[58] Matthew D. Walker Andrew G. Davis, Rory S. Turnbull. Robust audio streaming over ip. In *INET'99 proceedings*, 1999. (Cited on page 29.)

[59] Michael G. Luby, Michael Mitzenmacher, M. Amin Shokrollahi, Daniel A. Spielman, and Volker Stemann. Practical loss-resilient codes. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 150–159, New York, NY, USA, 1997. ACM. (Cited on page 30.)

[60] Jing Xu. Turbo coded hybrid type ii arq system. (Cited on page 30.)

[61] Luca Trevisan David P. Willianson Madhu Sudan, Sanjeev Khanna. Coding theory: Tutorial survey. *SIAM Journal on Computing*, 30(6):1863–1920, 2001. (Cited on page 30.)

[62] Vincent Roca. A state of the art of reliable multicast protocols. (Cited on page 30.)

[63] Luigi Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM Computer Communication Review*, 27(2):24–36, April 1997. (Cited on page 30.)

[64] Carl-Erik W. Sundberg Emin Martinian. Low delay burst erasure correction codes. In *International Conference on Communications*, 2002. (Cited on page 30.)

[65] Emin Martinian and Carl-Erik Sundberg. Low delay burst erasure correcting codes for packet transmission. In *International Symposium on Information Theory*, 2002. (Cited on page 30.)

[66] Don Towsley Jim Kurose Dan Rubenstein, Sneha Kasera. Improving reliable multicast using active parity encoding services (APES). *IEEE INFOCOM 1999*, 1999. (Cited on page 31.)

[67] ITU-T Recommendation P.800, Methods for Subjective Determination of Transmission Quality, August 1996. (Cited on pages 33, 43, and 219.)

[68] A. Takahashi, H. Yoshino, and N. Kitawaki. Perceptual qos assessment technologies for voip. 42(7):28–34, July 2004. (Cited on pages 33 and 34.)

[69] Lingfen Sun and E. C. Ifeachor. Voice quality prediction models and their application in voip networks. 8(4):809–820, Aug. 2006. (Cited on page 34.)

[70] Peppino Fazio Salvatore Marano Floriano De Rango, Mauro Tropea. Overview on voip: Subjective and objective measurement methods. *IJCSNS International Journal of Computer Science and Network Security*, 6:140–153, 2006. (Cited on page 34.)

[71] Etsi final draft eg 201 377-3 v1.1.1, "non-intrusive objective measurement methods applicable to networks and links with classes of services", stq specification and measurement of speech transmission quality, April 2003. (Cited on page 34.)

[72] Itu-t recommendation g.861. objective quality measurement of telephone-band (300-3400 hz) speech codecs, August 1996. (Cited on page 35.)

[73] A.W. Rix and M.P. Hollier. The perceptual analysis measurement system for robust end-to-end speech quality assessment. *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*, 3:1515–1518 vol.3, 2000. (Cited on page 35.)

[74] W. Yang. *Enhanced Modified Bark Spectral Distorsion (EMBSD): An Objective Speech Quality Measure Based*

on Audible Distorsion an Cognition Model. PhD thesis, Temple University, 1999. (Cited on page 35.)

[75] ITU-T Recommendation P.862, Perceptual evaluation of speech quality (PESQ), an objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs, February 2001. (Cited on pages 35, 36, 43, 44, and 191.)

[76] Analysis and interpretation of inmd voice-service measurements, May 2004. (Cited on page 35.)

[77] P. Coakley S. Broom and P. Sheppard. Getting the message loud and clear - quantifying call clarity. *British Communication Engineering*, 17, april 1998. (Cited on page 35.)

[78] Itu-t recommendation g.107. the emodel, a computational model for use in transmission planning, july 2002. (Cited on pages 35, 37, 43, 44, 87, 151, 172, 191, 213, and 219.)

[79] ITU-T Recommendation P.563, Single-ended method for objective speech quality assessment in narrow-band telephony applications, May 2004. (Cited on page 35.)

[80] S. Broom. High level description of psytechnics itu-t p.vtq candidate. ITU-T, Delayed Contribution, COM12-D175, 2003. (Cited on page 35.)

[81] Telchemy Inc. Description of vqmon algorithm. ITU-T, Delayed Contribution 105, jan 2001. (Cited on page 36.)

[82] ITU-T Recommendation P.564, Conformance testing for narrowband voice over IP transmission quality assessment models, July 2006. (Cited on page 36.)

[83] W. M. Liu, K. A. Jellyman, J. S. D. Mason, and N. W. D. Evans. Assessment of objective quality measures for speech intelligibility estimation. In *Proc. IEEE International Conference on Acoustics, Speech and Signal*

Processing ICASSP 2006, volume 1, pages I–I, 14–19 May 2006. (Cited on pages 36 and 46.)

[84] Wenyu Jiang and H. Schulzrinne. Speech recognition performance as an effective perceived quality predictor. In *Proc. Tenth IEEE International Workshop on Quality of Service*, pages 269–275, 15–17 May 2002. (Cited on pages 36, 44, 45, and 191.)

[85] A. E. Conway. Output-based method of applying pesq to measure the perceptual quality of framed speech signals. In *Proc. WCNC Wireless Communications and Networking Conference 2004 IEEE*, volume 4, pages 2521–2526, 21–25 March 2004. (Cited on page 36.)

[86] S. Pennock. Accuracy of the perceptual evaluation of speech quality (pesq) algorithm. In *MESAQIN*, 2002. (Cited on pages 36 and 44.)

[87] Antonio Estepa, Rafael Estepa, and Juan M. Vozmediano. On the suitability of the e-model to voip networks. In *ISCC '02: Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)*, page 511, Washington, DC, USA, 2002. IEEE Computer Society. (Cited on page 37.)

[88] R. G. Cole and J. H. Rosenbluth. Voice over ip performance monitoring. *SIGCOMM Comput. Commun. Rev.*, 31(2):9–24, 2001. (Cited on pages 37, 44, 132, and 172.)

[89] Itu-t recommendation g.107 (08/08) : The e-model: a computational model for use in transmission planning, aug 2008. (Cited on page 39.)

[90] Joonbum Byun Peter Flynn Choon Shim Hongbing Zhang, Liehua Xie. Packet loss burstiness and enhancement to the e-model. In *Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS INternational Workshop on Self-Assembling Wireless Networks (SNPD/SAWN'05)*, 2005. (Cited on page 39.)

[91] L. Ding and R. A. Goubran. Speech quality prediction in voip using the extended e-model. In *Proc. IEEE Global Telecommunications Conference GLOBECOM '03*, volume 7, pages 3974–3978, 1–5 Dec. 2003. (Cited on page 39.)

[92] L. Ding and R. A. Goubran. Assessment of effects of packet loss on speech quality in voip. In *Proc. 2nd IEEE Internatioal Workshop on Haptic, Audio and Visual Environments and Their Applications HAVE 2003*, pages 49–54, 20–21 Sept. 2003. (Cited on page 39.)

[93] Lingfen Sun and E. C. Ifeachor. Prediction of perceived conversational speech quality and effects of playout buffer algorithms. In *Proc. IEEE International Conference on Communications ICC '03*, volume 1, pages 1–6, 11–15 May 2003. (Cited on page 39.)

[94] Lingfen Sun and E. Ifeachor. New models for perceived voice quality prediction and their applications in playout buffer optimization for voip networks. In *Proc. IEEE International Conference on Communications*, volume 3, pages 1478–1483, 20–24 June 2004. (Cited on pages 45 and 191.)

[95] Wenyu Jiang and Henning Schulzrinne. Comparison and optimization of packet loss repair methods on voip perceived quality under bursty loss. In *NOSSDAV 2002*, May 2002. (Cited on pages 45 and 79.)

[96] C. Perkins, O. Hodson, and V. Hardman. A survey of packet loss recovery techniques for streaming audio. *IEEE Network*, 12:40–48, Sep/Oct 1998. (Cited on pages 46, 59, 60, 98, 101, 168, and 189.)

[97] E.N. Gilbert. Capacity of a burst-noise channel. Technical Report 39, Bell Syst., 1960. (Cited on pages 48, 82, 107, 120, 171, and 219.)

[98] Clark A. Modeling the effects of burst packet loss and recency on subjective voice quality. In *IPtel 2001 Workshop*, 2001. (Cited on page 60.)

[99] Diot C. Boutremans C., Iannaccone G. Impact of link failures on voip performance. Technical Report IC/2002/015, Sprint Labs, 2002. (Cited on page 60.)

[100] J. Ramsey. Realization of optimum interleavers. *Information Theory, IEEE Transactions on*, 16:338–345, 1970. (Cited on pages 62 and 195.)

[101] Kenneth Andrews, Chris Heegard, and Dexter Kozen. A theory of interleavers. Recent Results Session, ISIT'1997, Ulm, Germany, July 1997. (Cited on page 62.)

[102] S. McCanne and S. Floyd. ns2 network simulator. (Cited on pages 82 and 155.)

[103] Wenyu Jiang and Henning Schulzrinne. Comparison and optimization of packet loss repair methods on voip perceived quality under bursty loss. In *NOSSDAV '02: Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, pages 73–81, New York, NY, USA, 2002. ACM Press. (Cited on pages 85 and 168.)

[104] J.G.; Girod B. Liang, Y.J.; Apostolopoulos. Model-based delay-distortion optimization for video streaming using packet interleaving. *Signals, Systems and Computers, 2002. Conference Record of the Thirty-Sixth Asilomar Conference on*, 2:1315–1319 vol.2, 3-6 Nov. 2002. (Cited on page 97.)

[105] Rtp payload format for transport of mpeg-4 elementary streams (rfc 3640), November 2003. (Cited on page 97.)

[106] Options for repair of streaming media (rfc2354), June 1998. (Cited on page 98.)

[107] N. Miller and P. Steenkiste. Collecting network status information for network-aware applications. In *INFOCOM (2)*, pages 641–650, 2000. (Cited on page 98.)

[108] Rtp: A transport protocol for real-time applications (rfc 1889), January 1996. (Cited on pages 98 and 119.)

[109] J.; Towsley D. Kasera, S.K.; Kurose. A comparison of server-based and receiver-based local recovery approaches for scalable reliable multicast. volume 3, pages 988 –995. IEEE Computer and Communications Society, 1998. (Cited on page 102.)

[110] Diane Kiwior and Stephen Zabele. Active resource allocation in active networks. *IEEE Journal on Selected Areas in Communications*, 19(3):452–459, March 2001. (Cited on pages 103, 104, 132, 201, and 202.)

[111] D. Wetherall. Service introduction in an active network, 1999. (Cited on page 103.)

[112] M.; Azcorra A.; Alonso C.; Calderon, M.; Sedano. Active network support for multicast applications. *IEEE Network*, 12(3):46–52, May/June 1998. (Cited on pages 103 and 106.)

[113] Ali C. Begen and Yucel Altunbasak. Timely inference of late/lost packets in real-time streaming applications. In *Picture Coding Sym. (PCS)*, December 2004. (Cited on page 105.)

[114] Jim Kurose Maya Yajnik, Sue Moon and Don Towsley. Measurement and modelling of the temporal dependence in packet loss. In *Proc. INFOCOM'99*, 1999. (Cited on page 105.)

[115] R. Frederick V. Jacobson H. Schulzrinne, S. Casner. Rfc3550 - rtp: A transport protocol for real-time applications, July 2003. (Cited on page 111.)

[116] C. Perkins M. Handley, V. Jacobson. Rfc4566: Sdp: Session description protocol, July 2006. (Cited on page 111.)

[117] Paul Skelly Don Towsley Sue B. Moon, Jim Kurose. Correlation of packet delay and loss in the internet. Technical report, Department of Computer Science, University of Massachusetts, 1998. (Cited on page 113.)

[118] Van Jacobson. Congestion avoidance and control. In *ACM SIGCOMM '88*, pages 314–329, Stanford, CA, August 1988. (Cited on pages 115, 165, and 166.)

[119] Henning Schulzrinne. Voice communication across the internet: a network voice terminal. Technical report, Dept. of Computer Science, U. Massachusetts, Amherst MA, July 1992. (Cited on page 115.)

[120] C. Holt. Forecasting seasonal and trends by exponentially weighted moving averages. Research Memorandum 52, Office of Naval Research, 1957. (Cited on pages 115, 116, and 203.)

[121] Transmission control protocol (rfc 793), September 1981. (Cited on pages 115 and 155.)

[122] P. Karn and C. Partridge. Improving round-trip time estimates in reliable transport protocols. *ACM Transactions on Computer Systems*, 9:364–373, 1991. (Cited on page 116.)

[123] David L. Mills. Rfc1305 - network time protocol (version 3) specification, implementation and analysis, March 1992. (Cited on page 119.)

[124] J. Postel. Rfc792 - internet control message protocol, September 1981. (Cited on page 121.)

[125] tcpdump/libpcap. (Cited on page 121.)

[126] N. Laoutaris and I. Stavrakakis. Intrastream synchronization for continuous media streams: a survey of playout schedulers. *Network, IEEE*, 16(3):30–40, May/Jun 2002. (Cited on page 141.)

[127] Nandita Dukkipati, Yashar Ganjali, and Rui Zhang-Shen. Typical versus Worst Case Design in Networking. In *Fourth Workshop on Hot Topics in Networks (HotNets-IV)*, College Park, Maryland, USA, November 2005. (Cited on page 146.)

[128] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. RFC 3489 - STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs), March 2003. (Cited on page 147.)

[129] C.D. Knutson, H.R. Duffin, J.M. Brown, S.B. Barnes, and R.W. Woodings. Dynamic autonomous transport selection in heterogeneous wireless environments. *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, 2:689–694 Vol.2, 21-25 March 2004. (Cited on page 147.)

[130] George Candea, Shinichi Kawamoto, Yuichi Fujiki, Greg Friedman, and Armando Fox. Microreboot - A Technique for Cheap Recovery. In *Proc. 6th Symposium on Operating Systems Design and Implementation (OSDI)*, San Francisco, CA, USA, December 2004. (Cited on page 147.)

[131] J. Wroclawski, 1997. (Cited on page 152.)

[132] Rich Wolski. Dynamically Forecasting Network Performance Using the Network Weather Service. *Journal of Cluster Computing*, 1:119–132, January 1998. (Cited on pages 152 and 214.)

[133] F. J. Jr. Massey. The kolmogorov-smirnov test of goodness of fit. *Journal of the American Statistical Association*, 46, 1951. (Cited on page 154.)

[134] Christian Tschudin and Lidia Yamamoto. Self-evolving network software. *Praxis der Informationsverarbeitung und Kommunikation*, pages 206–210, December 2005. (Cited on page 164.)

[135] Tyson Condie, Joseph M. Hellerstein, Petros Maniatis, Sean Rhea, and Timothy Roscoe. Finally, a Use for Componentized Transport Protocols. In *Fourth Workshop on Hot Topics in Networks (HotNets-IV)*, College Park, Maryland, USA, November 2005. (Cited on page 174.)

[136] Parveen Patel, Andrew Whitaker, David Wetherall, Jay Lepreau, and Tim Stack. Upgrading Transport Protocols using Untrusted Mobile Code. In *Proc. 19th ACM Symposium on Operating System Principles*, October 2003. (Cited on page 174.)

[137] D. M. Yellin. Competitive algorithms for the dynamic selection of component implementations. *IBM Systems Journal*, 42(1):85 – 97, January 2003. (Cited on page 175.)

[138] Ada Diaconescu and John Murphy. Automating the performance management of component-based enterprise systems through the use of redundancy. In *ASE '05: Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, pages 44–53, New York, NY, USA, 2005. ACM. (Cited on page 175.)

[139] D. A. Lamb. Idl: sharing intermediate representations. *ACM Trans. Program. Lang. Syst.*, 9(3):297–318, 1987. (Cited on page 176.)

[140] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (WSDL) 1.1. ., 2001. (Cited on page 176.)

[141] M. Dean and G. Schreiber. OWL web ontology language reference. W3C recommendation, W3C, February 2004. (Cited on page 176.)

[142] Tina Yu. Hierarchical processing for evolving recursive and modular programs using higher-order functions and lambda abstraction. *Genetic Programming and Evolvable Machines*, 2(4):345–380, 2001. (Cited on page 176.)

[143] Juan M. Estévez-Tapiador, Pedro García-Teodoro, and Jesús E. Díaz-Verdejo. Measuring normality in http traffic for anomaly-based intrusion detection. *Computer Networks*, 45(2):175–193, 2004. (Cited on page 177.)

[144] Henning Schulzrinne Wenyu Jiang. Perceived quality of packet audio under bursty losses. In *IEEE INFOCOM 2002*. (Cited on page 190.)

[145] Lidia Yamamoto and Guy Leduc. An active layered multicast adaption protocol. In H. Yasuda: Lecture Notes on Computer Sciences., editor, *Second International Working Conference on Active Networks (IWAN 2000). Tokyo, Japan*, pages 180–194. Springer-Verlag Berlin Heidelberg, October 2000. (Cited on page 190.)

[146] D. Pearce H.G. Hirsch. The aurora experimental framework for the performance evaluations of speech recognition systems under noisy condition. In *ISCA ITRW ASR 2000*, 2000. (Cited on page 191.)