TESIS DOCTORAL

"ESPECIFICACIÓN Y EVALUACIÓN DE UN ALGORITMO DE SINCRONIZACIÓN DE GRUPO DE FLUJOS MULTIMEDIA"

Fernando Boronat Seguí

DIRECTOR:

Dr. Juan Carlos Guerri Cebollada

Departamento de Comunicaciones

Diciembre 2003

Dedicada
a mi esposa Carmina
y a mis padres

Abstract

In this thesis an algorithm for Multimedia Group Synchronisation is proposed. It is based on feedback techniques and a global time reference.

The algorithm solves the multimedia group synchronization problem in non critical multimedia applications, such as remote learning and telesurveillance. It includes not only inter-stream synchronization but it also guarantees the inter-receiver synchronization, playing all the streams at the same time at all receivers. To cite an example, we can consider the remote learning application in which a teacher can send a film to his students and talk about it so they can work interconnected or answer him.

The proposed algorithm is based on RTP/RTCP utilization, defining new RTCP data structures or even modifying the existing ones. So the devices in a multimedia application can interchange control information useful for the algorithm. It also considers a global time provided by NTP.

The author presents several results from an objective and subjective evaluation. They show that the algorithm works well over point-multipoint and multipoint-multipoint communications, whit several sources and several receivers.

Resumen

En la presente Tesis se presenta un algoritmo de sincronización de grupo de flujos multimedia, basado en técnicas de realimentación y la utilización de una referencia de tiempo global común a todos los sistemas involucrados en una aplicación multimedia.

Dicho algoritmo trata de solucionar el problema de la sincronización de grupo de flujos en aplicaciones multimedia no críticas como, por ejemplo, las de aprendizaje a distancia o televigilancia. No sólo se ocupa de la sincronización local entre flujos en un mismo receptor, sino que garantiza también la reproducción simultánea (sincronizada), de todos los flujos en todos los receptores al mismo tiempo. Como ejemplo podríamos citar una aplicación de aprendizaje a distancia, en la que, por ejemplo, el profesor o tutor envíe un vídeo de una película (flujo de contenido almacenado) y, al mismo tiempo, en determinados instantes, pueda hacer comentarios (flujo de contenido en directo) sobre las imágenes que se están viendo de la película y los estudiantes discutir dichos comentarios o incluso contestar al profesor.

El algoritmo propuesto se basa en la utilización de los protocolos RTP y RTCP, definiendo nuevas estructuras de datos de éste último y modificando algunas ya existentes, para que los equipos que se comunican puedan intercambiar información esencial para dicho algoritmo. También hace uso del protocolo NTP como referencia de tiempo global.

Se presentan los resultados de una evaluación del mismo, tanto objetiva como subjetivamente, mostrando un comportamiento satisfactorio sobre configuraciones punto-multipunto y multipunto-multipunto, con varias fuentes y varios receptores de flujos multimedia.

Resum

Amb aquesta Tesi es presenta un algoritme de sincronització de fluxos d'informació de diferent naturalesa (multimedia), basat en tècniques de realimentació i en la utilització d'una referència de temps global comú a tots els sistemes involucrats a una aplicació multimedia

Tal algoritme tracta de solucionar el problema de la sincronització de grup de fluxes en aplicacions multimedia no critiques com, per exemple, les d'aprenentatge a distancia o televigilancia. No sols s'ocupa de sincronitzar localment en un mateix receptor els fluxes sinó també de garantir la representació simultània (sincronitzada) de tots els fluxes en tots el receptors al mateix temps. Com exemple, podríem anomenar una aplicació d'aprenentatge a distancia on el professor envia el vídeo d'una pel·lícula (fluxe de continguts guardats) i, al mateix temps, siga capaç de fer comentaris sobre les imatges que estiguen visualitzant i a més a més els estudiants puguen discutir eixos comentaris o inclòs contestar-li al professor.

L'algoritme proposat es basa en la utilització dels protocols RTP i RTCP, definint noves estructures de dades d'aquest últim arribant a definir-ne de noves o modificant les ja existents, per a que els equips intercanvien informació de control essencial per al funcionament de l'algoritme. A més a més fa ús del protocol NTP com a proveïdor d'una referència de temps global.

A la Tesi es presenten els resultats d'una avaluació del mateix, tat objectiva com subjectiva, mostrant un comportament satisfactori en configuracions punt multipunt o inclòs multipunt-multipunt, amb varies fonts de dades i múltiples receptors.

Agradecimientos

Es difícil para mí expresar agradecimiento a todas las personas que me han apoyado en la realización de la presente Tesis. Sin embargo me gustaría intentarlo.

En primer lugar, de forma especial, me gustaría agradecer a Juan Carlos Guerri todo el trabajo y esfuerzo que ha dedicado en la dirección y supervisión de esta Tesis Doctoral, así como sus siempre acertados consejos y aportaciones que han contribuido a mejorarla día a día. Su calidad humana y su excelencia docente e investigadora convierten en un privilegio el trabajo junto a él.

Quiero agradecer, a continuación, a Manuel Esteve su confianza por haber confiado en mí en un primer momento para realizar la Tesis y participar en el grupo de Investigación que dirige, *Sistemas de Tiempo Real Distribuidos*, a pesar de la distancia.

Además, también quiero mostrar mi agradecimiento a Gunnar Hellstrom por su ayuda y colaboración prestada durante el desarrollo de la herramienta para transmisión de texto utilizando RTP, siguiendo la RFC escrita por él.

Por supuesto, también me gustaría agradecer a mis compañeros de Valencia y Gandia su apoyo prestado durante estos años animándome a terminar la Tesis, en especial, a Juan Luís, por sus sabios consejos.

También me gustaría mostrar mi agradecimiento especial a Carlos Turró que me ha ayudado en la preparación de las pruebas de Campus, desde Valencia, prestándome su tiempo y conocimientos en todo momento, sin recibir nada a cambio. También agradezco a todos los evaluadores de las pruebas subjetivas su dedicación y apoyo, en especial, a José María, Edgardo y Salva, que me ayudaron en las mismas.

También quiero agradecer el apoyo constante a mi familia que siempre está y estará animándome a seguir mejorando. Por supuesto, para Carmina, que siempre está a mi lado apoyándome en todo lo que hago, sin quejarse nunca, no tengo suficientes palabras de agradecimiento que expresen todo lo que siento.

Índice

CAPÍTULO 1. INTRODUCCIÓN Y OBJETIVOS

CAPÍTULO 2. ESTADO DEL ARTE

CAPÍTULO 3. SINCRONIZACIÓN, PROTOCOLOS Y APLICACIONES UTILIZADAS EN LA TESIS

CAPÍTULO 5. TRANSMISIÓN DE DATOS SOBRE RTP/RTCP

CAPÍTULO 6. EVALUACIÓN OBJETIVA Y RESULTADOS

CAPÍTULO 7. EVALUACIÓN SUBJETIVA Y RESULTADOS

CAPÍTULO 8. CONCLUSIONES, APORTACIONES Y LÍNEAS FUTURAS DE INVESTIGACIÓN

Capítulo 1

Introducción y Objetivos de la Tesis

1.1. Introducción

Toda Tesis debe tener como objetivo aportar algún conocimiento o solución a los problemas o necesidades que actualmente existen en el área de conocimiento en la que se centra la realización de la misma.

En el mundo de las telecomunicaciones y, más exactamente, en el de la ingeniería telemática, se está produciendo una *evolución* constante. Gracias a los avances técnicos, cada vez son mayores las posibilidades que se ofrecen a los usuarios finales. Están apareciendo aplicaciones con nuevas exigencias en entornos cada vez más heterogéneos, como, por ejemplo, aplicaciones de teletrabajo, de vídeo bajo demanda, de aprendizaje a distancia, de televigilancia, de telemedicina, etc. Dichas exigencias plantean nuevos retos que es necesario acometer, y resolver, para continuar la evolución y desarrollo al ritmo que se ha impuesto en los últimos años.

Durante las pasadas décadas, las aplicaciones telemáticas, básicamente las constituían el correo electrónico, la transferencia de ficheros y las emulaciones de terminal remoto. Dichas aplicaciones únicamente requerían una comunicación orientada a la conexión libre de errores que pudieran producirse por pérdidas de paquetes. Sin embargo, en ningún momento era necesario cumplir requerimientos temporales, eximiendo al sistema de comunicaciones de poseer características apropiadas para permitir la ejecución de aplicaciones de tiempo real.

Actualmente, las nuevas *aplicaciones multimedia distribuidas* se caracterizan por tener requerimientos de tiempo real. Por *multimedia* se quiere expresar la integración de diferentes tipos de información como, por ejemplo, texto, imágenes, vídeo, audio, etc. Las nuevas aplicaciones multimedia se pueden considerar como aplicaciones de tiempo real, con un elevado grado de exigencia en cuanto a los requerimientos de tiempo de transferencia y, en ellas, el concepto de *tiempo* adquiere gran importancia.

La existencia de aplicaciones multimedia en un entorno *local* o *monopuesto*, como, por ejemplo, en una única estación de trabajo, no plantea problemas de comunicaciones, ya que todas las comunicaciones se realizan entre procesos propios del sistema local.

Sin embargo, como los datos que se utilizan en dichas aplicaciones, como el audio y el vídeo, ocupan una gran cantidad de espacio de almacenamiento, la tendencia es, por una parte, mantener los datos distribuidos en diferentes servidores (servidor de audio, servidor de vídeo, servidor de imágenes, etc.) y, simultáneamente, realizar una compresión eficiente de los datos antes de realizar el almacenamiento o la transferencia. La existencia de *sistemas distribuidos* de información hace factible la aparición de nuevas aplicaciones multimedia como, por ejemplo, el vídeo bajo demanda o las aplicaciones de aprendizaje a distancia, siempre que se añadan los requisitos de tiempo real al sistema de comunicaciones que se utilice como soporte de las mismas y mecanismos que garanticen el mantenimiento de las relaciones temporales en el momento de la reproducción.

Las aplicaciones tradicionales desarrolladas para las redes de datos, siempre implican la comunicación entre dos sistemas finales. En las redes multiservicio en las que se utilizan aplicaciones como las de videoconferencia, procesamiento colaborativo o difusión masiva a toda la red, se requieren comunicaciones simultáneas entre grupos de sistemas finales. Este proceso, conocido como comunicación *multipunto*, está contrapuesto a las comunicaciones punto a punto (o *unicast*) a las que estábamos acostumbrados.

Dentro de las comunicaciones multipunto se pueden destacar las comunicaciones *multicast*, en las que se optimiza la utilización de recursos para el envío de información a un grupo de usuarios receptores. Se envía copia de cada

paquete sólo a los sistemas finales que explícitamente han indicado su intención de formar parte del grupo. Es un concepto basado en el receptor, que es quien se une a un grupo *multicast* y la información es enviada desde la fuente a todos los miembros del grupo.

Como se ha comentado, el término multimedia engloba básicamente voz, vídeo y datos. Estos tipos distintos de tráfico con los que nos encontramos tienen unos *requisitos* distintos en lo que a su transmisión a través de la red se refiere. El más sencillo de transportar, y de ahí su presencia dominante hasta ahora en las redes, son los datos. Sin embargo, la tecnología ha evolucionado en los últimos años de manera que los otros tipos de tráfico, voz y vídeo, se pueden transmitir por el mismo tipo de infraestructuras por los que hasta la fecha sólo se transportaban datos, constituyendo lo que se conoce como *Integración de Servicios*.

En cuanto a los *tipos de redes* existentes, se deberá considerar cuáles son adecuadas para las aplicaciones multimedia distribuidas. Como se ha comentado anteriormente, las aplicaciones tradicionales de transmisión de datos, imágenes, texto y/o ficheros (correo electrónico, transferencia de ficheros, web, etc.) eran tolerantes a retardos y *jitter* de las redes pero no eran tolerantes a errores o pérdidas. Además, requerían poco ancho de banda y podían hacer uso de redes de datos tradicionales por conmutación de paquetes, basándose en la utilización de protocolos 'seguros'.

Por otra parte, las aplicaciones de tiempo real (voz, vídeo en tiempo real, videoconferencias, etc.), como es lógico, no son tolerantes a los retardos y, por ello, han utilizado las líneas telefónicas de conmutación de circuitos (RTB o RDSI-BE), las redes de TV por cable, radioenlaces, etc., en las cuales prácticamente no existen variaciones en el retardo sino que se tiene un retardo o tiempo de propagación 'constante'. En ellas, la asignación de recursos (*slots* de tiempo, circuitos, frecuencia) se realiza en exclusividad a las conexiones, ofreciendo un servicio orientado a la conexión y garantizado (retardo fijo, ancho de banda garantizado, etc.).

La *solución para las actuales aplicaciones multimedia* pasa por diferentes posibilidades que pueden llegar desde utilizar CD-ROMs locales, lo cual no es una solución en red y tiene el problema de la actualización de la información, o bien utilizar circuitos o líneas dedicadas (de elevado coste), o circuitos virtuales dedicados (circuitos lógicos dedicados sobre conmutadores digitales y asignación, por tanto, de recursos para evitar la congestión, lo cual tiene un alto coste económico), hasta llegar a la utilización de redes de conmutación de paquetes (IP). Sobre éstas últimas, en general, se ha mantenido la opinión de que no pueden utilizarse para tráfico en tiempo real ya que no garantizan Calidad de Servicio ante restricciones de ancho de banda y retardo. Sin embargo, mediante el uso de determinados mecanismos adaptativos, técnicas de compresión, técnicas de

transmisión multicast, protocolos de reserva de recursos y mecanismos de prioridades, sí pueden ofrecer servicios de tiempo real aprovechando las características del tráfico multimedia.

Tal y como se aprecia, la integración de servicios no es sencilla y habrá, por tanto, que adecuar las redes de comunicaciones, tal cual se conocen en la actualidad, de manera que sean, además, capaces de albergar los nuevos tipos de tráfico como, por ejemplo, los que originan el vídeo y el audio.

Además, en general, interesan, por motivos de costes, más que soluciones de cambios del hardware del usuario, soluciones software que sean de bajo coste para éste, y que sean independientes del tipo de subredes que tengan que atravesar los paquetes. Obviamente, dichas aplicaciones deberán desarrollarse basándose, cómo no, en estándares como, por ejemplo, los que se detallan a continuación:

- IETF: protocolos como IP (*Internet Protocol*), RTP/RTCP (*Real-Time Transport Protocol/ Real-Time Transport Control Protocol*), SIP (*Session Initiation Protocol*), SDP (*Session Description Protocol*), RTSP (*Real-Time Streaming Protocol*), RSVP(*Resource Reservation Protocol*), etc.

- ITU-T: Técnicas de compresión como H.261, H.263, H.323 etc.

- Lenguajes de alto nivel: C/C++, Java, etc.

En definitiva, las redes deberán proporcionar una mínima *Calidad de Servicio* para poder facilitar esta integración de servicios.

Entre los parámetros que afectan a la calidad de la transmisión de flujos multimedia, se encuentran los siguientes:

- *Requerimientos de ancho de banda (AB)*: velocidad de transmisión de la infraestructura de red.

- *Latencia o retardo:* tiempo que tarda la información en atravesar la red desde la fuente al receptor.

- *Jitter*: variación de los tiempos de llegada entre los paquetes. Para minimizar el efecto de este factor, los paquetes entrantes han de ser almacenados en un *buffer*, del que se extraerán para ser consumidos con la tasa correspondiente.

- *Tasa de pérdidas de paquetes:* cuando un paquete del flujo multimedia se pierde en la red, puede ser necesario disponer de algún tipo de

recuperación o compensación de la información para minimizar el efecto causado en el extremo receptor.

El servicio se deberá adaptar, por tanto, a la falta o escasez de recursos para que estos parámetros afecten lo menos posible a la calidad del mismo.

Actualmente, Internet proporciona una única calidad de servicio que viene definida por el *servicio best-effort*, en el que los paquetes son transmitidos sin garantizar ningún ancho de banda o retraso mínimo entre los extremos, los *routers* utilizan una disciplina FIFO, todos los paquetes tienen la misma prioridad y no es posible solicitar una *reserva de recursos* para una determinada conexión. En este contexto, la calidad de las aplicaciones de tiempo real depende únicamente del ancho de banda disponible en cada momento.

La tendencia es la inclusión de los denominados *Servicios Integrados* y *Servicios Diferenciados*. Con los *Servicios Integrados* se consigue, manteniendo el servicio best-effort, mejorar el modelo de red IP para soportar transmisiones en tiempo real y garantizar una cierta calidad de servicio para cada flujo de transmisión, utilizando un protocolo de señalización adicional y mediante la reserva de recursos (RSVP). Para ello, los *routers* deberán disponer de ciertas funciones de control de tráfico. Con los *Servicios Diferenciados*, el tráfico se puede dividir en grupos con diferentes parámetros de Calidad de Servicio (sin señalización específica) con el objetivo de reducir la carga de la gestión de recursos en comparación con el modelo de Servicios Integrados.

Ante la situación descrita, con la aparición de nuevas aplicaciones multimedia en entornos heterogéneos y las nuevas exigencias que éstos conllevan, se están produciendo constantes avances en diferentes campos relacionados con el entorno de las aplicaciones de tiempo real distribuidas.

Podemos destacar diferentes campos, estrechamente relacionados con el tema central de la Tesis:

- *Técnicas de Sincronización*. En este campo se incluyen los algoritmos y protocolos que tienen como objetivo proporcionar un servicio de sincronización entre flujos de información en las aplicaciones multimedia. Entre dichas técnicas se encuentran otros aspectos relacionados con los algoritmos de tiempo global (NTP, por ola, etc.), los mecanismos de realimentación o *feedback* (como los de RTP/RTCP), la medida de desviaciones, el tratamiento estadístico y la evaluación del protocolo de sincronización. Además, será necesario especificar las relaciones temporales entre los diferentes flujos de información, así como definir una estructura lógica y física de almacenamiento que permita una recuperación

eficiente de la información o la planificación para la recuperación de la misma y todos los demás aspectos de la gestión de objetos multimedia.

- *Aplicaciones Multimedia*. Incluye el desarrollo de herramientas para la creación de aplicaciones multimedia y de la propia aplicación final que interactuará con el usuario (como, por ejemplo, *vic*, *rat*, *Windows Media*, *Netmeeting*, etc.), incluyendo protocolos estándar para el transporte de datos en tiempo real (como, por ejemplo, RTP/RTCP, SIP, RTSP, etc.) para la sincronización, el control de flujo y el control de la congestión.

- *Sistemas de Codificación y Compresión.* Dentro de este campo se pueden incluir todos los aspectos relacionados con las técnicas de codificación y compresión de flujos (JPEG, MPEG-1, MPEG-4, MPEG-7, H.261, H.323, etc.), especialmente de los flujos de vídeo.

- *Control de errores extremo a extremo*. En los sistemas multimedia, típicamente, se han utilizado protocolos 'no seguros', como UDP/IP. Sin embargo, puede ser conveniente introducir cierto grado de seguridad debido a que en algunos sistemas de compresión/descompresión no se toleran pérdidas y a la percepción que pueden producir dichos errores en el usuario. Esto se realiza en dos fases: detección (mediante *checksums* y secuenciamiento o marcas de tiempo) y corrección de errores (*Forward Error Correction* o FEC, codificación multinivel con canal en prioridad o sistemas basados en retransmisiones).

- *Redes y Protocolos de Comunicaciones*. Son aspectos fundamentales la tecnología de las redes utilizadas o que se piense utilizar (ADSL, *Frame Relay*, ATM, MPLS, etc.), la calidad de servicio proporcionada y requerida, el análisis de la congestión de la red o las características de los protocolos de transporte (TCP o UDP).

- *Técnicas de Distribución Multicast*. El trabajo en este campo se centra en el desarrollo y evaluación de protocolos de distribución multicast, como son *Multicast IP*, RMTP, etc.

- *Especificación, cálculo y garantía de Calidad de Servicio*. Dada la heterogeneidad de las aplicaciones multimedia, se deben especificar los requisitos de la aplicación o del usuario, así como las características y calidad de servicio que pueden garantizar los sistemas de comunicaciones. Posteriormente, se tendrá que 'mapear' los requisitos del usuario a los parámetros del sistema de comunicaciones y, por último, si es necesario, se deberán reservar recursos y gestionarse de manera eficiente para mantener dicha calidad (mediante, por ejemplo, *Servicios Integrados* o *Servicios Diferenciados*).

De todos los aspectos anteriores, la presente Tesis se centra en los dos primeros (*Técnicas de sincronización* y *Aplicaciones Multimedia*), ya que propone un *nuevo algoritmo de sincronización* que tiene como objetivo principal el proporcionar un servicio de sincronización de grupo de flujos para *aplicaciones multimedia*, haciendo uso de protocolos de tiempo real.

1.2. Marco y Objetivos de la Tesis

Una vez expuestos los diferentes campos de investigación relacionados, a continuación se describe con detalle cuál es el *marco* donde se centra la Tesis y los *principales objetivos* perseguidos con la realización de la misma.

1.2.1. Marco de la Tesis

Con la presente Tesis se pretende proponer una solución a la problemática que existe en las aplicaciones que necesitan representar, de forma sincronizada, diferentes flujos multimedia provenientes de la misma o de diferentes fuentes, bien en un mismo receptor o bien en varios receptores. Como ejemplo de este tipo de aplicaciones podemos citar todas aquellas de aprendizaje a distancia, las de monitorización de varios sistemas distribuidos o las de televigilancia desde un puesto de control de varios dispositivos distribuidos que estén transmitiendo diferentes flujos de información multimedia (audio, vídeo, datos, texto, etc.).

La idea que se persigue se puede resumir con el siguiente ejemplo: supóngase que se está transmitiendo el audio y el vídeo de una grabación de un concierto que se está realizando con varios micrófonos y varias cámaras de vídeo, distribuidas por un auditorio, y que cada uno de estos dispositivos está conectado, por ejemplo, a un ordenador personal que se encarga de la digitalización y transmisión de cada uno de los flujos, de forma independiente, por la red hacia los receptores reproductores (uno por cada flujo o uno para varios flujos). Lo que se perseguirá será conseguir que todos los flujos se reproduzcan de forma sincronizada, dentro de unos límites de asincronía, cada uno en un receptor independiente (o incluso varios en un único receptor). Si se pudiera lograr que los relojes de todos los sistemas estuvieran sincronizados en un margen de milisegundos, añadir marcas de tiempo a cada uno de los flujos, para poder relacionar las muestras de cada uno en los receptores según el tiempo en que deberían reproducirse dichas muestras, monitorizar la reproducción de todos los receptores y ejecutar las acciones necesarias para corregir asincronías, se podría sincronizar todos los flujos (aunque se reproduzcan en receptores independientes), manteniendo unos márgenes de error permisibles, según el tipo de aplicación de que se trate, durante toda la reproducción.

Por tanto, el *interés principal* de la presente Tesis es conseguir la especificación y evaluación de un algoritmo adaptativo de sincronización entre diferentes flujos de información en aplicaciones multimedia y, a la vez, conseguir una sincronización de grupo, es decir, que todos ellos se reproduzcan a la vez en los diferentes receptores involucrados.

Para conseguir dicho objetivo, nuestro marco de trabajo se ha centrado en los siguientes aspectos:

- *Protocolos RTP/RTCP y NTP* para la distribución en tiempo real de flujos multimedia.

- Entornos de red local (LAN) y entornos de red corporativa.

- Estudio de *Aplicaciones Multimedia* que engloben audio, vídeo, datos, etc. y, que, además, utilicen RTP/RTCP, como, por ejemplo, las herramientas utilizadas en la red MBone (*vic*, *rat*, etc.).

- La *Calidad de Servicio*, en cuanto a calidad de sincronización inter-flujo y de grupo se refiere.

Actualmente, la reserva de recursos y las garantías de calidad de servicio no se pueden ofrecer por todos los nodos de Internet, no obstante, aún cuando esté disponible la reserva de recursos y las garantías de calidad de servicio (como, por ejemplo, en determinadas redes corporativas), también se necesitará de unos mecanismos adaptativos de sincronización en las aplicaciones para tener en cuenta una cierta tolerancia a cualquier error de asignación y reserva de recursos debido a dificultades inherentes en la realización de las estimaciones de tráfico en la red correctamente, así como del propio proceso de reproducción.

1.2.2. Objetivos

Una vez visto el marco donde se sitúa la Tesis, pasamos a describir el objetivo principal y los objetivos operativos asociados que se persiguen en la misma.

El *objetivo general* perseguido, tal como ya se ha mencionado, es el de *especificar y evaluar un algoritmo de sincronización de grupo de flujos multimedia, haciendo uso de RTP/RTCP, en distintos entornos.* Para ello se va a desarrollar un algoritmo de sincronización basado en RTP/RTCP, en redes donde no existe calidad de servicio garantizada pero donde los límites del retardo extremo a extremo son conocidos a priori.

A su vez, este objetivo principal se puede subdividir en cuatro objetivos operativos asociados:

- **Objetivo 1:** *Especificar e implementar un algoritmo de sincronización de grupo de flujos multimedia***,** utilizando la potencia y versatilidad que RTP (*Real-Time Transport Protocol*) y RTCP (*RTP Control Protocol*) brindan y aprovechando la información de realimentación proporcionada por éste último.

- **Objetivo 2:** *Desarrollar una plataforma de evaluación* con una o varias fuentes transmisoras de flujos multimedia y varios receptores (todos interconectados en red y utilizando RTP/RTCP para la transmisión de flujos multimedia y NTP para la obtención de una referencia común de un tiempo global), donde poder *evaluar* el algoritmo diseñado. Para la transmisión de flujos de audio y de vídeo se utilizarán las herramientas de Mbone, *rat* y *vic*, modificadas para incluir el algoritmo diseñado.

- **Objetivo 3:** *Desarrollar una aplicación de transmisión de datos no dependientes del tiempo utilizando RTP/RTCP*. Para transmitir flujos de audio y vídeo existen múltiples herramientas que utilizan RTP/RTCP, sin embargo, para la transmisión de datos no continuos (por ejemplo, de texto) no existe ninguna.

- **Objetivo 4.-** *Evaluar, tanto de forma objetiva* (en base al grado de sincronización, retardo, jitter, etc.) *como subjetiva* (según la evaluación de los usuarios receptores de la aplicación multimedia basada en flujos de audio y vídeo, de la calidad de sincronización y de recepción), *el algoritmo desarrollado en dos entornos diferentes: en red de área local y en red corporativa* (en concreto, en la red del campus universitario de la Universidad Politécnica de Valencia, en adelante, UPV).

  La evaluación se realizará en entorno multicast, con una o varias fuentes y con uno o varios receptores.

  Para la evaluación del funcionamiento del protocolo RTP, se desarrollará un analizador que adquirirá y almacenará datos estadísticos de la sesión multicast para poder evaluarlos, bien en tiempo real o bien posteriormente.

Se tendrá en cuenta que la aplicación del algoritmo está pensada para entornos de red local o corporativa donde el ancho de banda disponible para cada dispositivo de red es suficiente para garantizar una mínima calidad en la transmisión y donde los límites del retardo extremo a extremo están acotados.

1.3. Organización de la Memoria

Una vez realizada la introducción a las principales cuestiones que han motivado la presente Tesis, así como una descripción de los objetivos principales perseguidos en la misma, veamos cómo se va a organizar el resto de la memoria.

En el capítulo 2, se describe el estado del arte mediante un recorrido por las principales contribuciones realizadas hasta la fecha sobre los temas tratados en la Tesis. Se ha intentado clasificar el contenido de las mismas en función de un criterio que considera las diferentes tareas relacionadas con la sincronización multimedia. Se ha realizado un énfasis especial en aquellas publicaciones que aportan ideas estrechamente relacionadas con las presentadas en la presente Tesis.

En el capítulo 3, se resumen los conceptos más importantes relacionados con la Tesis, así como los requerimientos de sincronización en los sistemas multimedia distribuidos. También se incluye una breve descripción de los protocolos, y aplicaciones empleadas en el desarrollo de la Tesis. Primero se presentan los protocolos NTP y RTP/RTCP, base de la misma, así como un mecanismo para intercambiar mensajes entre aplicaciones, denominado *mbus* y, posteriormente, se describen las herramientas utilizadas, *vic* y *rat*.

En el capítulo 4, se describe, de forma detallada, el algoritmo de sincronización de grupo de flujos multimedia propuesto, así como las modificaciones propuestas en los protocolos RTP/RTCP para poder implementarlo, aprovechando las ventajas de los mismos. Al final de dicho capítulo se presenta una comparativa de dicho algoritmo con algunos de los algoritmos existentes más representativos, descritos en el capítulo 2 de Estado del Arte.

En el capítulo 5, se presentan las diferentes posibilidades de transmisión de datos no continuos haciendo uso de RTP/RTCP, publicadas hasta la fecha, y se realiza una propuesta para la transmisión de aquellos datos que se puedan codificar mediante caracteres ASCII. Como demostración de la validez de la propuesta, se presenta una implementación de la misma en una aplicación de transmisión de texto en tiempo real utilizando RTP/RTCP.

En el capítulo 6, se describe cómo se ha evaluado de forma objetiva el nuevo algoritmo de sincronización. En primer lugar, se describe los escenarios donde se ha evaluado el protocolo, las pruebas diseñadas y los parámetros evaluados. Finalmente, se presentan los resultados de dicha evaluación que demuestran la validez del algoritmo desarrollado.

En el capítulo 7, se complementa la evaluación anterior con una evaluación subjetiva del funcionamiento del algoritmo propuesto implementado en una

aplicación de aprendizaje a distancia. Finalmente, también se presentan los resultados de la misma que verifican la validez del algoritmo propuesto.

En el capítulo 8, se expondrán las conclusiones finales, tanto sobre el proceso de diseño e implementación del algoritmo, como de los resultados obtenidos. Además, se mostraran las contribuciones realizadas durante el desarrollo de la Tesis, así como las posibles líneas de trabajo abiertas con la misma.

Finalmente, la Tesis termina con las referencias bibliográficas citadas a lo largo del texto, ordenadas alfabética y cronológicamente para una mejor localización.

Capítulo 2

Estado del Arte

2.1. Introducción

Como se ha comentado en el capítulo anterior, la presente Tesis Doctoral se enmarca en el ámbito de los *Protocolos de Tiempo Real* y, en concreto, dentro de las funcionalidades que proporcionan, en el campo de la *Sincronización Multimedia*. Partiendo de este principio, la exposición sobre el estado del arte pretende dar una visión general de los trabajos de investigación más interesantes que hasta la fecha se han publicado sobre sincronización multimedia.

En este capítulo, en un primer punto, se realiza una breve referencia a trabajos de investigación básicos, dedicados genéricamente al concepto de sistemas multimedia y sus características en cuanto a requerimientos de tiempo y sincronización.

A continuación, se detalla, brevemente, una serie de artículos sobre estándares y protocolos desarrollados para su aplicación en comunicaciones

multimedia distribuidas. Primero se presenta una serie de trabajos dedicados a la sincronización de relojes en red, con soluciones para conseguir una referencia de tiempo global y, posteriormente, se relacionan los protocolos más utilizados en las aplicaciones multimedia.

En tercer lugar, y centrándose en el tema principal de la Tesis, se presentan aquellos trabajos que aportan alguna solución concreta al problema de la sincronización, bien mediante la descripción del funcionamiento de un protocolo, o bien mediante su implementación y presentación de resultados de evaluación. Se han incluido, por similitud con la presente Tesis, aquellas aproximaciones que hacen uso de los protocolos RTP/RTCP en su implementación.

Dado que la consecución de la sincronización en un determinado equipo implica intercambio de información entre procesos, en el cuarto apartado se ha incluido bibliografía sobre mecanismos de intercambio interno de información local, haciendo énfasis en aquéllos más relacionados con la Tesis.

En cuanto a posibles herramientas multimedia, se presentan los dos enfoques existentes (el de la *Unión Internacional de Telecomunicaciones*, en adelante UIT, y el del grupo *Internet Engineering Task Force*, en adelante IETF) y algunas de las más representativas fruto de los grupos de investigación destacados en este campo, evitando las herramientas comerciales que no son abiertas ni manipulables, formando el quinto bloque de referencias.

Por último, y dado que en la Tesis se ha incluido un estudio sobre la evaluación subjetiva del algoritmo de sincronización propuesto, también se comentan, para finalizar este capítulo, los escasos trabajos encontrados dedicados a dicho tema.

2.2. Estado del Arte

2.2.1. Sistemas Multimedia

La sincronización hace referencia al concepto del *tiempo*, el cual no sólo juega un papel fundamental en la sincronización, sino que también lo lleva a cabo en otras áreas de trabajo tan próximas a la presente Tesis como los *Sistemas Distribuidos de Tiempo Real (SDTR)*, o en campos tan diferentes como la filosofía o matemáticas, existiendo distintas aproximaciones y definiciones del concepto de *tiempo* en cada una de ellas ([STA88], [HOO92]). Stankovick, en [STA88], aclara muchos conceptos sobre los nuevos (en aquella época) SDTR y anima a los investigadores a estudiar las restricciones temporales en dichos sistemas. Ya apunta a la necesidad de comunicación en red de algunos sistemas en tiempo real y, para

ello, señala la necesidad de desarrollar soluciones de encaminamiento dinámico con garantías de corrección en cuanto a tiempos garantizados, nuevas técnicas de gestión de *buffers* en la red, comunicaciones tolerantes a fallos y con restricciones temporales, y la necesidad de planificación en la red, combinada, a su vez, con planificación de procesadores para proporcionar soluciones a nivel de sistema. Por otro lado, Hoogeboom et al., en [HOO92], presentan las cuatro principales aproximaciones al estudio del *tiempo*: la *filosófica*, la *física*, la *matemática* y la *tecnológica*. Realmente, el entendimiento teórico del concepto de tiempo está menos desarrollado de lo que nuestro sentido común sobre el tiempo podría sugerir. En general, la definición de tiempo muestra a éste como un concepto que elude una fácil descripción.

Centrándonos en los *Sistemas Multimedia*, se pueden clasificar como sistemas de tiempo real '*no críticos*', es decir, que el no cumplimiento de sus requerimientos temporales no provoca consecuencias catastróficas ([HOO92], [STE95]).

Existen numerosos artículos clásicos dedicados a la descripción de las características propias de los sistemas multimedia, así como de sus implementaciones *y* servicios, de entre los que destacamos, a nuestro entender, los más interesantes, que son [FUR94] y [STE95].

En [FUR94], se describen multitud de conceptos relacionados con los sistemas multimedia distribuidos, centrándose en los requisitos prácticos de los servidores, sistemas operativos, memorias, dispositivos de almacenamiento y en las aplicaciones multimedia. También se incluye una descripción de los sistemas de compresión JPEG, MPEG y H.261, que juegan un papel crucial en las aplicaciones multimedia, clasificándolos en dos grupos: los que recuperan perfectamente la información al descomprimir y los sistemas con pérdidas que pierden precisión (pero obteniendo mayores ratios de compresión), indicando la posibilidad de implementarlos, tanto a nivel hardware como a nivel software, o en combinación para mejorar el rendimiento minimizando costes. Se describen las características de las denominadas *Redes Multimedia* aptas para la transmisión de información multimedia, diferentes de las redes de datos tradicionales que no fueron diseñadas para transmitir dicho tipo de información. Clasifica la *Sincronización Multimedia* en sincronización continua (*Continuous Synchronization*) y sincronización puntual (*Point Synchronization*), y sincronización serie (equivalente a *intra-flujo*) y paralelo (equivalente a *inter-flujo*). El artículo ofrece, además, una descripción de los principales parámetros de la calidad de servicio como son el *jitter*, el retardo, el *BER* y *PER* (*Bit* y *Packet Error Rate*), etc.

En [STE95] se analizan en profundidad las características de los sistemas operativos tomando como criterio la necesidad de incluir determinadas funcionalidades para poder ofrecer servicios multimedia. Muestra cómo el sistema

de gestión de procesos debe tener en cuenta los requerimientos temporales impuestos por los datos multimedia, tanto si son de tiempo real como si no lo son, y, en consecuencia, utilizar los métodos de planificación adecuados. Señala que la principal característica de los sistemas de tiempo real es la *corrección*, no sólo en la computación libre de errores sino también en el tiempo en el que los resultados son presentados. Un sistema de tiempo real puede fallar no sólo debido a la existencia de un número considerable de fallos hardware y/o software sino, también, debido a que el sistema sea incapaz de ejecutar sus tareas críticas a tiempo. En concreto, se analizan los métodos de gestión de recursos, el sistema de ficheros, la planificación de las tareas o procesos y la gestión de memoria. En este trabajo también se describen una serie de algoritmos diseñados para sistemas operativos con características de tiempo real.

De forma genérica, el *problema de la sincronización* representa la necesidad de garantizar la ejecución de determinados eventos de forma coordinada y cumpliendo un conjunto de restricciones temporales.

En adelante, en este capítulo y los siguientes, por *Flujo Multimedia* se entenderá la transferencia de información de un determinado tipo (audio, vídeo, texto, datos, etc.) entre una fuente y un destino receptor (*punto a punto*) o varios destinos receptores (*punto a multipunto*) o desde varias fuentes a varios destinos receptores (*multipunto a multipunto*). La sincronización de flujos multimedia se entiende como el proceso de asegurar, por un lado, que la información contenida en un flujo sea presentada en el momento de su reproducción manteniendo las relaciones temporales que existían entre sus unidades de datos (LDUs o *Logical Data Units*) al crearse, proceso denominado *Sincronización Intra-flujo*, y, por otro lado, en el caso de existir varios flujos relacionados, garantizar que también se mantengan las relaciones temporales que existían entre sus LDUs y las de los demás flujos, en el momento de la generación de todos ellos, proceso denominado *Sincronización Inter-flujo*. Un ejemplo de éste último tipo de sincronización podría ser el problema de controlar la salida de un flujo de voz (audio) y el de un flujo de vídeo para adquirir la sincronización entre el sonido y el movimiento de los labios, lo que es conocido como *Sincronización Labial* o *lip-syncronization* (también, de forma abreviada, *lip-sync*).

En caso de que existan múltiples fuentes y/o múltiples receptores, a la sincronización de todos los flujos en todos los receptores también se la denomina *Sincronización de Grupo*.

Existen muchos trabajos desarrollados en el campo de la sincronización multimedia, entre los que se destacan los siguientes (a pesar de tener la mayoría más de una década de antigüedad): [STE90], [LIT90], [NIC90], [FER90], [LIT91] y [BLA96]. Estos trabajos pueden clasificarse como *básicos y de máxima importancia en el campo de la sincronización multimedia* debido a los conceptos e

ideas que aportan. Prácticamente la totalidad de los artículos sobre sincronización, que se citarán en apartados posteriores, los utilizan como bibliografía y referencia.

El trabajo presentado en [STE90] se centra en las características que deben incorporar los *protocolos de sincronización* en los sistemas multimedia distribuidos. Se definen los conceptos de *multimedia* y *sincronización* y se presenta un *modelo orientado a objetos de un sistema multimedia*. Clasifica la *información* como *dependiente* o *independiente del tiempo*, introduciendo los elementos necesarios para describir correctamente el proceso de sincronización: *objeto multimedia*, *actividad*, *interacción* y *puntos de sincronización*. En este artículo, Steinmetz distingue entre un entorno centralizado y un entorno distribuido, en el cual los flujos de información sufren distintos retardos al utilizar caminos diferentes y, por lo tanto, las restricciones y problemas de sincronización se incrementan, haciendo imprescindible la utilización de un mecanismo de resincronización. También se enfatiza en la descripción de las características de un proceso de sincronización (relación entre flujos, operaciones de sincronización, número de objetos involucrados, etc), siendo el aspecto más interesante el punto dedicado a las propiedades básicas de los mecanismos de sincronización y que denomina: *bloqueo restrictivo*, *aspectos de tiempo real* y *sincronización continua*. El concepto de *bloqueo restrictivo* hace referencia a las acciones que se deben ejecutar cuando un flujo de información debe esperar hasta alcanzar un punto de sincronización. En definitiva, plantea el problema conocido como *gap*, y aporta como solución, en el caso del vídeo, la repetición de la última trama reproducida hasta que se alcance el punto de sincronización y se sincronice con los otros flujos de información. Distingue entre *objetos transitorios*, generados y reproducidos sin ningún tipo de almacenamiento intermedio, y los *objetos persistentes* los cuales se recuperan desde un sistema de almacenamiento. Sobre este segundo tipo es posible aplicar los mecanismos de *bloqueo restrictivo*. Indica que la sincronización puede clasificarse como *óptima*, *buena*, *aceptable* o incluso *no tolerable*, dependiendo de las características de los flujos de información y de la aplicación. En cuanto a los *aspectos de tiempo real*, señala que los sistemas operativos deberían incorporar funciones capaces de especificar *eventos temporales* o *puntos de sincronización* (sincronización paralela o secuencial), así como determinar los retardos *mínimos*, *ideales* y *no aceptables* entre eventos temporales. Por último, define la granularidad de la sincronización en función de la aplicación y de los flujos de información, relacionando la *sincronización continua* con los niveles más altos de granularidad. Ya en este artículo se indica la posibilidad de que la *sincronización continua* podía ser implementada mediante la utilización de marcas de sincronización en los propios datos o de un canal de sincronización adicional.

En [LIT90] se propone un modelo de almacenamiento y sincronización de objetos multimedia en entornos distribuidos con contenidos almacenados previamente. Clasifica los diferentes datos a incluir en una composición multimedia en tres tipos: *estáticos* (se corresponden con el concepto de objetos

independientes del tiempo), *dinámicos* (coincide con el término de objetos dependientes del tiempo), y *mixtos* (formados por estáticos y dinámicos). También diferencia entre relaciones espaciales y temporales, y divide estas últimas en *implícitas* (existen en el momento de la captura) y *explícitas* (se crean artificialmente). Relaciona definitivamente la sincronización con las relaciones temporales y propone tres niveles como los componentes de un sistema multimedia: *nivel físico*, *nivel de servicio* y *nivel de interfaz con el usuario*. La principal contribución del artículo es la propuesta de una técnica para la especificación formal y modelado de la sincronización multimedia y la entrega de objetos multimedia, basada en intervalos de tiempo y *Redes de Petri Temporizadas* (TPN o *Timed Petri Nets*), modificadas para añadir, además, información sobre tiempo, duración y utilización de recursos. El modelo resultante recibe el nombre de OCPN (*Object Composition Petri Net*).

En [NIC90], se consideran como sistemas multimedia los constituidos por un sistema de computación distribuido (DCS, *Distributed Computing System*), o plataforma donde se ejecutan las aplicaciones, y los protocolos de comunicaciones (encargados del transporte de los datos de dichas aplicaciones en tiempo real). Describe los requerimientos necesarios de un sistema de comunicaciones para soportar aplicaciones multimedia en tiempo real y especifica un esquema de sincronización entre diferentes flujos de información. Entre los requerimientos, destaca el retardo que introduce el empaquetamiento (como elemento de más peso dentro del retardo de transmisión extremo a extremo, que depende del tamaño de las unidades de datos o muestras) en un entorno de red de área local. También señala la necesidad de un sistema de almacenamiento o *buffering* en el destino para subsanar los retardos introducidos por la propia red, el *jitter* y la desviación de los relojes de las fuentes y los destinos. Se destaca como característica inherente a los sistemas multimedia la consideración de que un paquete que llegue tarde al destino equivale a un paquete erróneo o perdido, por lo que se propone la utilización de protocolos que no lleven a cabo reconocimientos ni retransmisiones. Por otra parte, señala la tasa de error como el principal factor que degrada la calidad de la presentación, estableciendo un valor de un 1% para el caso de la voz, mientras que para el caso del vídeo considera que dependerá de las técnicas de compresión y codificación utilizadas. Para Nicolaus, el concepto de Calidad de Servicio (QoS) adquiere gran importancia en las aplicaciones multimedia, donde cada flujo de información se diferenciará del resto por necesitar que se cumplan unas determinadas condiciones de QoS (por ejemplo, implementada mediante prioridades), que pueden ser utilizadas, por ejemplo, para diseñar el esquema de *buffering* y minimizar los retardos introducidos en las colas correspondientes a dicho flujo. El sistema distribuido necesitará proporcionar un conjunto de facilidades para implementar aplicaciones distribuidas, entre las que están las llamadas a procedimientos remotos (*RPCs*) y las primitivas de sincronización. El sistema de comunicaciones deberá ser capaz, a su vez, de implementar la sincronización mediante *buffers* y técnicas de control de flujo. Introduce el

concepto de sincronización labial o *lip-sync*. Se compara la posibilidad de transmitir la voz y el vídeo multiplexados en un mismo flujo con la de transmitir cada flujo por separado. Se desecha la primera opción dado que la voz y el vídeo requieren diferentes parámetros de QoS y los dispositivos de reproducción son distintos y específicos. Al utilizar conexiones diferentes, la red y el DCS introducen un *jitter* que debe ser compensado por los mecanismos de sincronización. El esquema de sincronización que se propone consta de dos niveles (físico y lógico) y se basa en la existencia de mensajes de realimentación o *feedbacks*, que indican a la fuente generadora sobre el estado de la comunicación (evitando tener que esperar un *time-out* para recuperarse de una situación de asincronía), y en la utilización de puntos de sincronización que permitan comprobar si existe o no asincronía y, en caso positivo, llevar a cabo las acciones oportunas (parar, reanudar, etc.).

En [FER90], Ferrari recalca la necesidad de que las redes futuras ofrezcan servicios de tiempo real y presenta un método para garantizar retardos en redes WAN basadas en la conmutación de paquetes, basado en el establecimiento de una conexión virtual que denomina *canal de tiempo real* entre la fuente y el destino, caracterizado por los requerimientos del cliente. La importancia de este artículo reside en los conceptos que aporta (por ejemplo, la clasificación de redes deterministas y estadísticas en función de la garantía del retardo máximo que sufren los datos, etc.) y que han sido adoptados en las redes actuales. El esquema propuesto funciona sólo en conjunción con procesos de planificación y control de flujo. Para la planificación, en cada nodo se mantienen tres colas: una de ellas se utiliza para almacenar los paquetes etiquetados como *deterministas*, otra para los datos *estadísticos*, y la tercera para el *resto de paquetes*. Los paquetes deterministas tienen la máxima prioridad y, dentro de este grupo, el orden se establece en función del tiempo de validez temporal o *deadline*. Para el control de flujo, se propone un sistema basado en la modificación de la tasa de transmisión en función del conocimiento de las características del receptor y del camino establecido, sin necesidad de información obtenida mediante reconocimientos. En el caso de que algún usuario no cumpliera las características especificadas, se le penalizaría, con el objetivo de reducir su efecto sobre el resto de paquetes, aumentando su *deadline* en las colas de los nodos cuando en éstos existiera una carga elevada e incluso serían los primeros en ser eliminados de las colas cuando se produjera saturación u *overflow*.

En [LIT91], Little, además de recoger todas las definiciones y clasificaciones aparecidas en artículos anteriores, también describe los modelos conceptuales, lenguajes y aproximaciones gráficas para especificar y representar las relaciones temporales que existen entre los objetos multimedia, así como la problemática del almacenamiento y recuperación de la información en los sistemas multimedia distribuidos. Entre los modelos conceptuales presentados se destacan los *basados en instantes* y los *basados en intervalos* (TIB, *Temporal Interval*

Based). Como métodos de especificación que permitan no sólo modelar las relaciones temporales sino extraer dicha información y utilizarla eficazmente para establecer los instantes de reproducción, se consideran lenguajes de programación (como LOTOS), métodos gráficos (como las *Redes de Petri*: OCPN, propuestas en [LIT90], y PNBH o *Petri Net Based Hypertext*) y lo que denomina *abstracciones temporales* (que permiten controlar o manipular las presentaciones). Por último, se destacan los problemas asociados a un sistema distribuido debido a las latencias que introduce la red, las variaciones en la frecuencia de los relojes en cada una de las fuentes y las características en cuanto al soporte en tiempo real de los sistemas finales.

Uno de los artículos más importantes encontrados sobre sincronización multimedia es [BLA96]. Su importancia radica en el excelente estudio y recopilación de la información que aporta sobre la sincronización y sistemas multimedia en general. Dedica una primera parte a revisar las definiciones y clasificaciones utilizadas en dicho ámbito, estableciendo los criterios para clasificar un sistema como multimedia. Describe los conceptos de *sincronización intra-objetos* e *inter-objetos*, así como el concepto de LDU, clasificando estás últimas como *abiertas* (con duración no predecible) y *cerradas* (con duración predecible). Una aportación fundamental es el estudio sobre los *requerimientos de las presentaciones multimedia* para garantizar la calidad de sincronización desde el punto de vista de la percepción humana. Establece un modelo de referencia para la sincronización, formado por cuatro niveles: el *nivel de media* (*Media Level*), que garantiza la sincronización intra-flujo de un único flujo de LDUs; el *nivel de flujo* (*Stream Level*), que garantiza la sincronización inter-flujo entre diferentes flujos, tanto dependientes como independientes del tiempo; el *nivel de objeto* (*Object Level*), que garantiza la correcta planificación de la presentación de los diferentes objetos totalmente sincronizada; y el *nivel de especificación* (*Specification Level*), que contiene las aplicaciones y herramientas para especificar las relaciones temporales entre los diferentes objetos involucrados en una presentación. Respecto a la especificación, aporta ejemplos de las técnicas más utilizadas (basadas en intervalos, ejes temporales, diagramas de flujo, *Redes de Petri Temporales*, eventos, lenguajes *script*, etc.). Por último, enumera los problemas de sincronización en sistemas distribuidos y presenta un conjunto de casos de estudio (como *MHEG*, *HyTime*, *Firefly*, *MODE*, *ACME*, etc.).

2.2.2. Estándares y Protocolos

2.2.2.1. Protocolos de Sincronización de Relojes en Red

Los trabajos de investigación encontrados sobre *Sincronización de Relojes en Red* están encabezados por los trabajos de Mills sobre el protocolo *Network*

Time Protocol, en adelante *NTP* ([MIL92]), que permite que los relojes de una red puedan estar sincronizados dentro de un margen de error que va desde unos pocos milisegundos en redes de área local, hasta decenas de milisegundos en redes de área amplia, caracterizadas por retardos altamente variables. Si se adquiere esta sincronización con errores de milisegundos se puede garantizar que la diferencia entre tiempos de reloj entre dos sistemas sea más pequeña de lo que el usuario puede percibir. Otros protocolos que se puede encontrar en la literatura sobre sincronización de relojes son el *UNIX 4.3BSD Timed protocol* ([GUS85]), el *Digital Time Service* ([DIG89]), el denominado genéricamente algoritmo *por ola* ([RAY90]) y el algoritmo de sincronización detallado analítica y experimentalmente en [BER00].

Otra opción para conseguir sincronización de relojes en un sistema distribuido puede ser la de utilizar receptores GPS[(1)] (*Global Positioning System* [BAD00]), que obtienen la referencia temporal de la señal obtenida de los satélites de dicho sistema. Los relojes de los satélites GPS más modernos tienen un error de un segundo cada 6 millones de años y la precisión está continuamente mejorándose. La mayoría de los receptores comerciales con capacidad de sincronización, proporcionan una precisión de +/- 1 microsegundo, aunque lógicamente al pasar el tiempo a un host, ésta precisión se verá afectada. En [MEL02] se propone una combinación del sistema GPS y el protocolo NTP para proporcionar una referencia global de tiempos. El abaratamiento de las conexiones a Internet y de los receptores GPS ha contribuido a la posibilidad de alcanzar precisión del orden de milisegundos en nuestros sistemas. En la RFC 2330 ([PAX98]), se pide cautela en el uso de NTP para el cálculo de retardos de red, aunque, por otro lado, en la RFC 2679 ([ALM99]) se indica que una combinación de NTP y GPS puede proporcionar los niveles de precisión requeridos por las aplicaciones multimedia. En [BOR02b], se presenta un experimento consistente en la sincronización, mediante NTP y receptores GPS comerciales de bajo coste, de todos los equipos de una red local. Muestra que con la utilización de este tipo de receptores GPS se obtenía una peor precisión con respecto a la obtenida mediante la referencia de tiempo de servidores NTP de Internet. No obstante, se concluye que estos receptores GPS podrían constituir una fuente de sincronización de backup en caso de caída de la conexión a Internet.

2.2.2.2. Estándares y Protocolos relacionados con las Aplicaciones Multimedia

Para facilitar la aparición de aplicaciones multimedia se han desarrollado una serie de estándares y protocolos que aseguren su interoperabilidad, entre los que se pueden destacar los siguientes: la especificación UIT-T H.323 para telefonía

[1] http://www2.cr.nps.gov/gis/gps.htm

por Internet, el protocolo SIP (*Session Initiation Protocol*, [HAN96]) para inicio de sesión multimedia, SDP (*Session Description Protocol*, [HAN98]) para descripción de la sesión, SAP (*Session Anouncement Protocol*, [HAN00]) para el anuncio de sesiones y RTSP (*Real-Time Stream Control Protocol*, [SCH98]) para controlar servidores multimedia (flujos en tiempo real) en Internet y, sobre todo, los protocolos RTP/RTCP (*Real-Time Transport Protocol/Real-Time Transport Control Protocol*, [SCH96]), propuestos por el IETF para transportar datos en tiempo real, tales como vídeo y audio, sobre una red IP.

De todos ellos, por la importancia que tiene para la Tesis, cabe destacar los protocolos RTP/RTCP. Los protocolos RTP y RTCP se especifican en la RFC 1889 ([SCH96]). El protocolo RTP ha sido propuesto para proporcionar el transporte de la información de los múltiples flujos de datos de las aplicaciones multimedia (por ejemplo, de audio y vídeo) utilizando diferentes formatos según se describe en los perfiles correspondientes, definidos en RFCs (por ejemplo, para audio y vídeo, se sigue la RFC 1890, [SCH96a], mientras que para la transmisión de texto sobre RTP se sigue la RFC 2793, [HEL00]). El protocolo RTCP es utilizado para muchos servicios como, por ejemplo, identificar a los participantes, informar de la QoS que están obteniendo los receptores, transportar información de los transmisores para sincronización entre flujos, etc. Esta información se puede utilizar para diferentes finalidades, como la adaptación de la QoS, sincronización, monitorización de la sesión, etc.

A pesar de haberse detectado problemas al protocolo RTP con respecto a la escalabilidad, es decir, en cuanto a su comportamiento cuando el grupo de receptores aumenta (lo cual afecta a la congestión inicial producida por los paquetes RTCP iniciales, a la capacidad de almacenamiento del estado de la red y al retardo de transmisión de paquetes RTCP) se están introduciendo mejoras al mismo, como, por ejemplo, la indicada en [ROS98] (en este artículo se mejora el problema de la congestión inicial por la transmisión de paquetes RTCP por la entrada de miembros a una sesión RTP mediante un mecanismo nuevo denominado *Reconsideration*), y es muy probable que, tal y como indica dicho artículo, RTP sea el protocolo de transporte para distribuciones multimedia *multicast* en las redes futuras donde podría haber decenas de miles de usuarios conectados. La versión de RTP que se ha propuesto para estandarizar ya incluye todas estas mejoras.

Aunque RTP incluye la denominación de protocolo de transporte, existen ciertas funcionalidades del nivel de transporte que RTP no proporciona, como son las de reservar recursos, garantizar la QoS para servicios en tiempo real, garantizar la entrega de los paquetes, proporcionar mecanismos para asegurar la entrega a tiempo de los paquetes, que pueden llegar demasiado tarde, y prevenir la entrega de paquetes desordenados. RTP ha sido diseñado, deliberadamente, ampliable y, por ello, debe estar integrado en las aplicaciones. Los paquetes RTP son normalmente encapsulados en paquetes UDP (*User Datagram Protocol*), por lo que puede

utilizarse UDP para obtener comprobaciones de *checksum* y multiplexado. RTP también puede utilizarse sobre *IP Multicast* ([DEE89]) para distribuir los paquetes en una sesión *multicast*.

En Internet se encuentra disponible el código fuente de un conjunto de pequeñas aplicaciones que pueden ser utilizadas para procesar datos RTP, así como diferentes librerías y aplicaciones sobre RTP[(2)], que se pueden utilizar para desarrollar todo tipo de aplicaciones utilizando dicho protocolo.

En [PER00] se presenta una evaluación crítica del uso de RTP en aplicaciones multimedia interactivas, presentando sus debilidades con respecto a este tipo de aplicaciones. Concluye que RTP no es apropiado para los requerimientos de las aplicaciones interactivas debido a su poca flexibilidad en aspectos clave para dichas aplicaciones (relojes, marcas de tiempos, implementación de fiabilidad, posibilidad de retransmisiones para recuperar pérdidas de paquetes y el propio formato de algunos paquetes RTCP). Considera el protocolo *Reliable Multicast Framework Protocol o RMFP* ([CRO98]) como una solución mejor para tales aplicaciones por la flexibilidad que supone frente al uso de RTP, pero indica que sigue siendo insuficiente y reclama la necesidad de un nuevo protocolo para las mismas. Actualmente se está trabajando en una nueva versión de dicho protocolo[(3)]. En [MAU99] se propone el *protocolo RTP/I* como un intento de definir un perfil RTP para aplicaciones interactivas, pero que, debido a la dependencia con RTP, hereda los problemas de éste y sigue sin cumplir las expectativas de las aplicaciones interactivas. Actualmente se está trabajando en el desarrollo de dicho protocolo pero como protocolo independiente que reutiliza algunos aspectos de RTP pero adaptado a los requerimientos de las aplicaciones interactivas[(4)].

Son muchos los artículos en los que se implementan diferentes *arquitecturas y aplicaciones multimedia* basadas en RTP/RTCP, entre los que se podría destacar [BIE96], [CHE96a], [KOU96], [MAR96], [MAR96a], [RAM97], [BOL98], [SHI98], [SHI98a], [TAS98b], [DIO99], [LI99], [SHI99], [GON00], [TAS00], [KUO01] e [ISH02]. En los apartados posteriores se comentarán aquellos más relacionados con la presente Tesis.

Para finalizar este apartado, se destaca la existencia de dos trabajos que, al igual que la propuesta de la presente Tesis, basan sus soluciones en modificar los protocolos RTP/RTCP, que son los presentados en los artículos [CHE96a] y [BRA02].

---

[2] En http://www.cs.columbia.edu/~hgs/rtp se pueden encontrar herramientas y librerías que utilizan RTP.
[3] http://www-sop.inria.fr/rodeo/rmfp/#alf
[4] http://www.informatik.uni-mannheim.de/informatik/pi4/projects/RTPI/

En [CHE96a] se presenta una aplicación denominada *Vosaic*, para la que se ha desarrollado una variante de RTP, denominada *Video Datagram Protocol (VDP)*, especializada para manejar tráfico de vídeo en tiempo real sobre la web. Se trata de un protocolo que facilita la transmisión de vídeo en tiempo real sobre Internet reduciendo el *jitter*, incorporando adaptación dinámica según el estado de la red y según la carga de la CPU de los clientes. Además, incluye el tratamiento de los paquetes perdidos mediante un algoritmo de retransmisión por demanda. Se especifica una sintaxis de URL extendida compatible con MBone. También añade extensiones a la sintaxis de HTML para poder especificar los segmentos de audio y vídeo relacionados. Incluye un mecanismo, denominado *logger*, que recoge estadísticas (uso de la red y del procesador de cada petición, datos de QoS como, por ejemplo, pérdidas, tasa de trama, *jitter*, etc.) para caracterizar el patrón de acceso de los usuarios al servidor de vídeo basado en web, sosteniendo que será diferente al presentado por el acceso a servidores web de texto e imágenes fijas. VDP es un protocolo asimétrico punto a punto, en el sentido de que una conexión VDP tiene establecidos, entre el servidor o fuente y el receptor o cliente, dos canales asociados: un canal de transmisión no fiable (para información *feedback* y datos no críticos) y otro para datos e información de control VCR (*play*, *stop*, adelante, atrás, etc.) y de gestión y finalización de la conexión.

En [BRA02], se presenta una solución a la difusión o *broadcast* de flujos multimedia para redes de distribución de contenidos o CDN (*Content Distribution Networks*), utilizando RTP, basada en un mecanismo, denominado *cueing protocol* para proporcionar información temporal, estructura e información de identidad en los flujos multimedia. Los mensajes elementales de dicho protocolo, denominados *cues*, indican eventos cuya temporización precisa es relevante para los receptores, y son enviados sobre RTP. Como ejemplo, se podría citar la inclusión de información del inicio, del contenido y del final de la emisión de un vídeo por Internet que facilitara a los videograbadores del receptor iniciar y parar la grabación. El sistema *Radio Data System* (RDS), que proporciona información de tráfico, de la estación y de las canciones, sería otro ejemplo. Los mensajes *cue*, aunque no es necesario, son independientes de los mensajes de datos y son asociados a un flujo y se pueden enviar tanto en banda (*in-band*) como fuera de banda (*out-of-band*), es decir, junto con los datos o en un flujo separado. Dichos mensajes son creados definiendo un nuevo formato de carga útil del paquete RTP, así como asignando un nuevo significado a determinados campos de la cabecera RTP, utilizando diferentes tipos de señalización codificada en el mismo (inicio o fin de un evento, aviso de un evento por llegar, aviso de un evento en progreso y aviso de cancelación de un evento). En dicho trabajo, se presenta, además, un prototipo de una estación de radio en Internet, transmitiendo flujos mejorados con información contenida en los mensajes *cue*.

2.2.3. Soluciones Particulares al Problema de la Sincronización de Flujos Multimedia

En las pasadas décadas se ha realizado, y se sigue realizando, un importante trabajo para solucionar los problemas de sincronización en entornos distribuidos. Existen multitud de trabajos, sobre todo en sincronización de flujos de tasa constante, en configuraciones de uno a muchos o de muchos a uno, en arquitecturas multimedia distribuidas, en métodos para planificación (*scheduling*) en la entrega de objetos multimedia, etc. También se ha realizado un trabajo interesante para poder lograr la restauración de la temporización dentro de un flujo (*sincronización intra-flujo*) de paquetes que haya sido alterada por los retardos sufridos en las colas de los nodos, así como en sincronización entre flujos (*inter-flujo*) en comunicaciones punto a punto o entre varias fuentes y varios destinos receptores. Además, sobre todo en aplicaciones como el trabajo cooperativo, la teleenseñanza o los juegos en red, hace falta sincronizar grupos de flujos para que todos los participantes sean tratados por el sistema de forma justa y equitativa.

Por tanto, en una primera clasificación podríamos dividir los artículos seleccionados en tres grupos: los que tratan de la *sincronización intra-flujo*, los que tratan de la *sincronización inter-flujo* y los que tratan de la *sincronización de grupo*.

Entre los primeros podemos destacar [ALV93], [RAM93b], [KÖL94], [RAM94], [ROT95], [ISH95], [SHI95], [STO95], [BIE96], [GEY96], [KOU96], [LIU96], [ISH97], [ISH99], [ROT97], [MOO98], [XIE99], [BAL00], [GON00], [ISH01a], [KUO01], [LAO01a], [LAO01b], [ROC01], [JEO02] y [LAO02].

Entre los artículos del segundo grupo podemos destacar [YAV92], [LIT92], [LIT93], [YAV94], [ESC94], [ISH95], [ROT95], [SHI95], [AKY96], [BAQ96], [BIE96], [CHE96], [KOU96], [LAM96], [LIU96], [RAN96], [SON96], [SRE96], [STE96], [YAN96], [ZAR96], [GUE97], [QIA97], [ROT97], [TAS98a], [TAS98b], [XIE99], [GON00], [ISH01a] y [KUO01].

Entre los del tercer grupo podemos citar [ESC94], [AKY96], [ZAR96], [ISH97], [ISH99], [DIO99], [ISH01a] y [ISH02]. Como se puede apreciar existen trabajos que aparecen en más de un grupo ya que algunos de los algoritmos propuestos en los mismos incluyen varios tipos de sincronización.

2.2.3.1. Sincronización Intra-flujo

La transmisión de flujos de tiempo real sobre redes *best-effort* ha sido un área de gran interés durante la pasada década. Un objetivo importante de la

comunidad investigadora ha sido desarrollar métodos que se ocupen del *jitter* (variaciones del retardo de la red), que es una característica propia de las redes *best-effort*. Existen muchos algoritmos de adaptación del punto de reproducción de un determinado flujo que se encargan de reconstruir el flujo original en el receptor, lo cual se conoce como restauración de la calidad de la sincronización intra-flujo. Los sistemas planificadores del punto de reproducción (*playout schedulers*) se diferencian, según utilicen o no información de tiempo, en dos grupos:

- Los planificadores *time-oriented*, que son los que utilizan información de tiempo, colocan marcas de tiempo (*timestamps*) en las LDUs y utilizan relojes en el emisor y en el receptor para medir el retardo en la red y el *jitter*.

- Los planificadores *buffer-oriented*, que son los que no utilizan información de tiempo.

Entre los planificadores *time-oriented* podemos destacar los descritos en [ALV93], [RAM94], [SHI95], [GEY96], [LIU96], [MOO98], [BAL00] y [ROC01]. Entre los planificadores *buffer-oriented* podemos destacar los descritos en [KÖL94], [ROT95], [STO95], [BIE96], [KOU96], [ROT97], [MOO98], [XIE99], [SRE00], [KAN01], [KUO01], [LAO01a], [LAO01b], [JEO02] y [MEL02].

Laoutaris y Stavrakakis, en [LAO02], presentan una recopilación de los trabajos de investigación orientados a la adaptación del punto de reproducción (intra-flujo), organizando las ideas que han sido presentadas por los investigadores de forma aislada e identificando las principales diferencias entre los diferentes esquemas presentados hasta la fecha. También se examinan brevemente las técnicas FEC (*Forward Error Correction*) de corrección de errores y el uso de sistemas caché en los sistemas multimedia y su importancia en la adaptación del punto de reproducción. La comparativa discute diferentes aspectos relacionados con la información de tiempo, manejo de las LDUs que llegan tarde, métricas de evaluación de la calidad de la sincronización y la adaptación a los cambios del retardo en la red. Señalan que para la evaluación de la calidad de la sincronización intra-flujo se han propuesto varias métricas que se pueden englobar en *métricas de primer y segundo orden*. Las de *primer orden* cuantifican cuántas veces se han producido pérdidas de sincronización (frecuencia esperada de vacíos o *gaps* en la sincronización, duración acumulada de dichos vacíos, etc.). En los primeros planificadores diseñados para aplicaciones de transmisión de paquetes de voz, la métrica utilizada era la probabilidad o frecuencia en que un paquete era descartado por haber llegado tarde al receptor. En planificadores orientados a *buffers* (*buffer-oriented*) la calidad continua del flujo se ve afectada por discontinuidades *underflow* (si el *buffer* se queda vacío) u *overflow* (si se llena y no acepta más paquetes), por lo que la métrica utilizada debe tener en cuenta ambos eventos.

También, en los sistemas que modifican el *playout delay* o retardo de reproducción, afectando a la duración de las LDUs, la métrica también debe considerar la discontinuidad introducida por el propio sistema (por, ejemplo, debido a la expansión o reducción de la presentación de una trama de vídeo). En las métricas de *segundo orden* se captura el patrón de apariciones de pérdidas de sincronización. También se podría utilizar métricas híbridas que agruparan las de primer y segundo orden mencionadas.

2.2.3.2. Sincronización Inter-flujo y de Grupo

Como el propósito de la presente Tesis es la sincronización de un grupo de flujos y la sincronización entre flujos (inter-flujo), se profundizará en los trabajos encontrados relacionados directamente con la *sincronización inter-flujo* y la *sincronización de grupo*.

La sincronización entre flujos (inter-flujo) es muy importante, ya que la calidad de un entorno multimedia no sólo depende de la calidad de los entornos visuales y auditivos sino también de la calidad de la sincronización entre los dos entornos. De hecho, los oídos y la vista son fuertemente dependientes entre sí en los seres humanos. Una aproximación común en la mayoría de algoritmos dedicados a la sincronización inter-flujo en el receptor es la de utilizar un *reloj globalmente sincronizado* para relacionar los flujos provenientes de diferentes fuentes. Mientras algunos estudios asumen esta condición como preexistente ([LIT91a], [FER92], [LIT92], [LIT93], [ESC94], [SHI95], [GUE97], [ISH97], [ISH99], [KOU96], [ISH01a], [ISH02] y [MEL02]), otros incluyen, en sus propios algoritmos, mecanismos de estimación de las diferencias de tiempo de los relojes ([AND91], [RAN92], [RAM92], [ALV93], [RAM93], [RAM93b], [RAM94], [ISH95], [RAN95a], [RAN95b], [LAM96], [RAN96], [SON96], [ZAR96]). En [ROT95] y [ROT97] se propone un mecanismo capaz de utilizar ambas aproximaciones. En [ISH95] y [GON00] se presentan modelos que no requieren un reloj globalmente sincronizado para conseguir la reproducción de múltiples flujos de forma sincronizada.

A continuación se va a pasar a describir los artículos más importantes, de los enumerados anteriormente, que tratan sobre soluciones particulares en la sincronización entre flujos multimedia, ordenados cronológicamente.

En [AND91], se propone un mecanismo de recuperación de pérdidas de sincronismo en un entorno monolítico (no distribuido) producidas en dispositivos de entrada/salida (E/S) controlados por interrupciones y que, a su vez, están conectados a un servidor de flujos continuos. A dicho mecanismo se le ha asignado el nombre de *Algoritmo ACME* (*Abstractions for Continuous Media*). Se implementa haciendo uso de TCP para las comunicaciones. El mecanismo de

sincronización utilizado para soportar la sincronización de flujos continuos de información se basa en la utilización de un sistema de tiempo local o LTS (*Logical Time System*) y no asume la presencia de relojes globales. Cada dispositivo lógico es ligado a un LTS, de forma que un solo LTS puede acaparar a varios dispositivos lógicos. Mientras un LTS está en funcionamiento, su valor de tiempo se incrementa aproximadamente a la misma tasa que el tiempo real. La E/S de cada dispositivo lógico es sincronizada al LTS. La dificultad consiste en detectar la asincronía antes de que el usuario la perciba cuando la red tiene una elevada carga de tráfico. Uno de los dispositivos es elegido como *maestro* (el más restrictivo) al que se sincronizan el resto de dispositivos, tomados como *esclavos*. En el sistema propuesto, las diferencias entre flujos también se ajustan mediante modificaciones en el proceso de reproducción de los datos, mediante '*saltos*' y '*pausas*' en el mismo. Además, también se propone utilizar otras dos técnicas: variación de la tasa de E/S hardware de los dispositivos físicos y la interpolación de datos para ajustar la tasa de E/S efectiva.

En [LIT91a], Little y Ghafoor, presentan un esquema de sincronización de dos niveles, para proporcionar sincronización de llamadas multimedia. Los algoritmos presentados se corresponden con los niveles de sesión (NSP o *Network Synchronization Protocol*) y de aplicación (ASP o *Application Synchronization Protocol*) del modelo OSI. NSP proporciona la funcionalidad de establecer y mantener conexiones individuales con características específicas de sincronización, mientras que ASP soporta un servicio integrado de sincronización para aplicaciones multimedia. El esquema propuesto es válido para varias fuentes de datos multimedia que envían a un único destino donde se sincronizan los diferentes flujos. La especificación de las relaciones temporales se realiza mediante *Redes de Petri OCPN* ([LIT90]). Distingue entre fuentes de datos almacenados con sincronización *sintética* y fuentes de datos capturados *en directo* con sincronización *continua*. Las fuentes de información realizan una planificación de los instantes de transmisión antes de enviar la información, garantizando, con una cierta probabilidad, que los datos alcanzan el destino antes de que se supere su límite de validez y, también con una cierta probabilidad, que no existan fallos en la sincronización. Se describe con cierto detalle un algoritmo para calcular los instantes de reproducción y de transmisión, dividiendo el objeto multimedia en diferentes tipos de tráfico. Debido a los retrasos aleatorios introducidos por la red, la planificación descrita y el conocimiento a priori del tráfico no son suficientes para mantener la sincronización entre los flujos, y por ello es necesario que el receptor corrija la pérdida de sincronización. Parte del supuesto de que el destino controla la tasa del reloj de las fuentes, bien por medio de un reloj global, bien mediante correcciones periódicas del *drift* producido, o bien mediante mensajes de realimentación o mensajes *feedback*. Por último, se comprueba el funcionamiento de los algoritmos sobre una aplicación multimedia de aprendizaje sobre anatomía y fisiología.

Por otro lado, Little, en [LIT92] y [LIT93], propone una técnica de sincronización basada en el receptor que, principalmente, se ocupa de la sincronización inter-flujo, aunque algunos aspectos están relacionados con la sincronización intra-flujo. El modelo propuesto incluye varias fuentes y varios destinos. Se trata de una técnica de control de la asincronía entre flujos debida a situaciones anómalas de la red como, por ejemplo, sobrecarga del canal, pérdidas de datos y las variaciones de las tasas de los relojes en las fuentes y los destinos, que pueden provocar situaciones de *overflow* y *underflow* del *buffer* del receptor. Sigue el método de planificación de la transmisión propuesto en [LIT91a]. Los mecanismos de sincronización se implementan tanto en la fase de inicialización y transmisión/recuperación, como en la fase de presentación, control y monitorización del *buffer*. La sincronización de un único flujo de información se intenta garantizar en la fase de establecimiento mediante la reserva de recursos estadística (fundamentalmente el tamaño del *buffer*), para compensar el retardo introducido por la red. Se utiliza la técnica de inserción y descarte de LDUs (*LDUs stuffing*), en vez del ajuste de relojes. Con el fin de evitar situaciones de *overflow* y *underflow*, cuando el nivel del *buffer* de recepción alcanza un umbral superior o inferior, las LDUs son eliminadas o duplicadas, respectivamente. La función de corrección que controla la eliminación y duplicación de LDUs es arbitraria, pero sólo se investiga una función de corrección de tasa constante (*constant rate-based correction function*). El mecanismo está basado en asumir recursos de red garantizados y reacciona sólo ante violaciones de la reserva o pérdidas de LDUs, asumiendo una ligera pérdida de calidad de la presentación ante situaciones anómalas de la red.

En [RAN92], [RAM92] y, posteriormente, en [RAM93], [RAM93b], [RAM95], [RAN95a] y [RAN95b], se presenta el concepto de servidor multimedia bajo demanda y los requisitos de un servicio multimedia bajo demanda. Formula el problema de mantener la continuidad de la reproducción de cada flujo en presencia de múltiples peticiones de los suscriptores y presenta algoritmos de control de admisión que permiten a un servidor satisfacer el máximo número de suscriptores simultáneo posible. Se propone un sistema de sincronización entre audio y vídeo para servicios bajo demanda sobre redes integradas con ausencia de un tiempo global. Se argumenta que para sincronizar la reproducción de flujos multimedia almacenados no es suficiente la sincronización de los relojes de los reproductores y, además, que la sincronización de relojes de los equipos de captura o grabación y de los reproductores puede ser suficiente pero tampoco es necesaria ni factible en la mayoría de entornos de trabajo. En [RAN92] se propone la utilización de mensajes de realimentación o *feedback* para conseguir la sincronización, que también serán empleados en los artículos posteriores del autor. Es por ello que, otros investigadores, hayan denominado *Feedback Protocol* al mecanismo propuesto. El servidor o fuente multimedia, que también tiene el papel de sincronizador, aprovecha los mensajes *feedback*, que le envían los dispositivos reproductores sencillos (que se denominan *mediáfonos*) de forma periódica, para

detectar la asincronía entre los diferentes flujos. El servidor indica cuándo se envían los mensajes *feedback* para controlar la reproducción continua en los receptores. El principal objetivo es corregir la asincronía debida a la diferencia en las tasas de consumo entre los diferentes reproductores. El mecanismo propuesto es especialmente interesante por tres razones: por su sencillez de implementación y funcionamiento, por la baja carga de tráfico que introduce y, en tercer lugar, por la forma de resincronizar los flujos de información mediante '*saltos*' y '*pausas*' en los dispositivos reproductores. Emplea un *esquema de sincronización maestro/esclavo*, en el que se elige un flujo como *maestro* que será utilizado como referencia de sincronización para el resto de flujos denominados *esclavos*. Como flujo *maestro* suele escogerse el audio debido a sus restricciones temporales. La corrección la realiza el servidor multimedia, incrementando o disminuyendo la velocidad de reproducción de los receptores mediante '*saltos*' o '*pausas*' en el consumo de las LDUs. Se presentan dos técnicas de sincronización inter-flujo: una utiliza *buffers* de tamaño limitado con mensajes *feedback* enviados por el *maestro*, y la otra utiliza mensajes *feedback* provenientes de todos los dispositivos *mediáfonos* (que, como información básica, incluyen únicamente el número de la última LDU que se ha reproducido en el *mediáfono* en el momento del envío). El servidor indica en qué momentos se deben enviar los mensajes mediante un bit en la cabecera de las LDUs que envía. El algoritmo se evalúa con diferentes políticas de resincronización (*conservadora*, *agresiva* y *probabilística*), emulando diferentes entornos (LAN/MAN y WAN), mediante distribuciones exponenciales ([RAN92]) y normales ([RAM93]) del retardo. Las prestaciones del protocolo se miden a través de un conjunto de parámetros objetivos (número de asincronías, tráfico generado, etc.). Como desventaja se puede afirmar que la precisión viene limitada por el valor del *jitter*, lo cual es realmente importante en redes WAN, caracterizadas por un elevado valor del mismo, lo cual es motivo para que en ese caso el protocolo no ofrezca buenos resultados.

En [RAM93] y [RAM95] se propone una extensión del protocolo *Feedback* ([RAN92]) para entornos WAN, mediante un esquema jerárquico de sincronización que alivia al servidor multimedia de la recepción de múltiples mensajes *feedback* de los receptores y de la necesidad de resincronizarles a todos. La carga computacional y de recepción/transmisión de *feedbacks* se distribuye entre los nodos intermedios, integrándose esta función con la de encaminamiento. En [RAM93b], también se describe la técnica de utilización de *buffers* de capacidad limitada en los mediáfonos para asegurar una reproducción continua (sincronización intra-flujo) y que la asincronía entre mediáfonos no sobrepase un determinado límite. Esta técnica, aunque es simple y fácil de implementar, puede llevar a obtener una asincronía media elevada entre flujos, por lo que no es apropiada para la sincronización inter-flujo. Se propone, para ello, la técnica de utilización de mensajes *feedback* tanto del flujo *maestro* como de los flujos *esclavo*s para que el servidor pueda controlar la sincronización forzando una tasa de envío de mensajes *feedback*, tal y como ya se proponía en [RAM93]. También

se intenta integrar dichos algoritmos en el nivel de sesión del modelo OSI, para lo cual divide dicho nivel en dos subniveles: *subnivel de continuidad* (inferior) y *subnivel de sincronización* (superior). Por último, se evalúan las dos propuestas en un entorno de pruebas con transmisión de audio y vídeo, verificándose el anterior comportamiento en el primer esquema y apreciándose cómo es corregido por el segundo. Por otra parte, el trabajo presentado en [RAN95a] se centra en la sincronización inter-flujo y se añade la política de sincronización *predictiva* a las 3 presentadas en [RAN92] y [RAM93] (conservadora, agresiva y probabilística) y se compara con las anteriores en un entorno de pruebas similar al de los artículos anteriores.

En [YAV92] y [YAV94], Yavatkar hace hincapié en la importancia de la sincronización, tanto *temporal* como *causal*, en las aplicaciones multimedia de colaboración distribuida, entendiendo por *Sincronización Causal* en una aplicación colaborativa el conseguir que las relaciones causales entre los mensajes enviados en uno o varios flujos deben ser mantenidas para preservar el contexto en que son enviados. El grado de causalidad varía en función de los flujos involucrados y de las aplicaciones. Se propone el protocolo *Multi-Flow Conversation Protocol* (*MCP*) diseñado para soportar gran variedad de aplicaciones colaborativas. Primero, incluye un mecanismo basado en tokens para control de la concurrencia entre participantes, y, segundo, también incluye una abstracción de comunicación denominada *multi-flow conversation* para permitir la sincronización temporal entre múltiples flujos independientes. Considera una *conversación* como una entidad lógica formada por una o más flujos, y MCP fuerza la sincronización temporal en la entrega de tráfico de cada flujo. Introduce el concepto de causalidad-Δ (*Δ-causality*), donde Δ define una ventana de tiempo para cada LDU, teniendo en cuenta las restricciones de tiempo asociadas con el tráfico de tiempo real. El orden causal se garantiza dentro de cualquier duración dentro de los Δ segundos, para tener en cuenta dichas restricciones. MCP incluye un canal de señalización fuera de banda (*out-of-band*) para información de control y una interface de usuario asíncrona. En [YAV94] se presenta un prototipo con MCP implementado sobre UNIX 4.3 BSD, ejemplos de contenidos de los mensajes intercambiados y una evaluación del funcionamiento del protocolo sobre el mismo.

En [LIT93], Little describe un esquema integrador de soluciones (propias y de otros autores) para gestionar datos dependientes del tiempo en un sistema de información multimedia, que incluye la identificación y especificación (a nivel de autor o *authoring*) de las relaciones temporales entre objetos multimedia, el desarrollo conceptual de un sistema de almacenamiento de datos (base de datos) temporal, el diseño del sistema físico y los métodos de acceso para una recuperación de la información y transmisión síncrona.

En [ESC94], Escobar et al., describen una solución al problema de sincronizar varios flujos, incluso si las fuentes y destinos de los diferentes flujos

son diferentes. Aprovecha la existencia de relojes de red sincronizados ([MIL92]) y tiene en cuenta que el retardo de red es variable con el tiempo y que es, en el interés de las aplicaciones, necesario adaptarse a dichos cambios. La idea reside en que en dichas aplicaciones existen determinados puntos (instantes) en el flujo donde el tiempo puede ser 'expandido' o 'comprimido' y el usuario no lo notará (por ejemplo, un período de silencio en una conversación). De forma similar, la cantidad de tiempo que, en el flujo de vídeo, una imagen es reproducida puede ser ligeramente incrementada o reducida sin ser apreciado por el usuario. Se presenta un protocolo denominado *Flow Synchronization Protocol* (FSP), para sincronización de flujos multimedia basado en relojes de red sincronizados que proporcionan sincronización extremo a extremo y se adapta a los cambios en los retardos de cada flujo en diferentes topologías. Mientras que en [YAV92] se fijaba un límite máximo para el retardo de un flujo, descartando los datos que llegaban tarde y producían pérdidas de sincronismo, este protocolo asume que es posible fijar un límite máximo razonable al retardo entre los diferentes flujos. Ofrece mucha flexibilidad, permitiendo sincronizar flujos enviados de una fuente a muchos receptores, de muchas fuentes a muchos receptores o de muchas fuentes a un solo receptor. Además, también es capaz de sincronizar flujos por tipo (por ejemplo, voz con voz, datos con datos, etc.), por grupos relacionados geográficamente (por ejemplo, el audio y el vídeo coincidiendo en un mismo destinatario, es decir, *lip-sync*) o por cualquier otro criterio de agrupamiento de flujos. El protocolo no define un nivel específico para su implementación debido a la poca experiencia que había en su época con aplicaciones de sincronización pero ya sugiere que se implemente sobre el nivel 3 del modelo OSI, en el nivel de transporte de la arquitectura TCP/IP o en el nivel de adaptación de la arquitectura ATM. El protocolo propuesto se probó en dos aplicaciones de test: una aplicación para probar la sincronización labial, donde una asincronía entre flujos de hasta 300 milisegundos resultaba aceptable; y una aplicación de un trío de música distribuido ([SCH93]) de características más restrictivas, donde la asincronía máxima permitida para una calidad aceptable era de 100 milisegundos. También remarca el efecto negativo del aumento de carga de los sistemas implicados o de eventos inesperados en las comunicaciones sobre la sincronización de flujos. En las aplicaciones, el protocolo fue implementado sobre UDP/IP, pero sin soporte *multicast*. Según los resultados experimentales, se concluye que los requerimientos de sincronización entre flujos pueden ir desde los 100 milisegundos (en música) hasta, aproximadamente, los 300 milisegundos (*lip-sync* y baja fidelidad). Estos valores están en el rango de los retardos y desviaciones del retardo de las redes WAN y MAN. Además, los estudios en la percepción humana sugieren que los requerimientos de sincronización serán más restrictivos a medida que los parámetros de fidelidad como los factores de resolución y tasa de muestreo aumenten. Esta literatura enfatiza la importancia de las herramientas de sincronización de flujos con posibilidades de llegar al rango de milisegundos.

En [EHL94], se clasifica la mayoría de las técnicas anteriores de sincronización multimedia para controlar la asincronía entre flujos multimedia, presentando sus ventajas e inconvenientes. Se describen las técnicas en función de la topología y se consideran dos esquemas de sincronización: el *distribuido* (en red) y el *local* (dentro de la estación de trabajo). Sin embargo, en el artículo no se tratan los algoritmos evaluados extensivamente.

En [KÖH94] también se presenta una clasificación de algoritmos de sincronización multimedia en función de varios factores: *relojes* (relojes globalmente sincronizados y relojes disponibles localmente *–locally available clocks-*), *localización* (fuente o destino) y *técnicas de control de sincronización* (ajuste de frecuencia de reloj y realización de '*saltos*' y/o '*pausas*').

Ishibashi y Tasaka proponen, en [ISH95], el algoritmo de sincronización *VTR* (*Virtual-Time Rendering*), aplicable tanto a flujos continuos como no continuos, fuertemente acoplados o no (*tightly o losely-coupled streams*), que no necesita información sobre los valores mínimo y máximo del retardo de la red. Contiene tanto mecanismos de sincronización intra-flujo como inter-flujo, siguiendo este orden de ejecución y teniendo preferencia el primero. Sin embargo, no contempla mecanismos de sincronización entre destinos en comunicaciones *multicast*. Considera un nivel de sincronización ubicado entre los niveles de aplicación y de transporte de la pila OSI (ocupando el lugar de los niveles de sesión y de presentación) dividido en tres subniveles: los de *sincronización intra-flujo*, *inter-flujo* y *controlado por eventos* (*event-driven*). Puede realizar un control fino de la sincronización mediante la inclusión de marcas de tiempo (*timestamps*) en las LDUs transmitidas en el punto de acceso al servicio de transporte (*TSAP*) en el momento de la generación en la fuente y la utilización de varios valores umbrales, de tal manera que la calidad de la sincronización puede ser ajustada según los requerimientos de cada aplicación. Supone un protocolo de transporte completamente fiable y asume que cada LDU generada en la fuente estará disponible, y en orden, en el correspondiente *TSAP* del destino. No se basa en relojes globalmente sincronizados sino en relojes locales. Supone que los *ticks* de reloj en la fuente y el destino tienen la misma cadencia pero los tiempos locales pueden ser diferentes (*Locally Available Clocks*). La idea del algoritmo es bastante simple: en adición al tiempo actual, se añade un tiempo virtual que se expande o se reduce, en función del *jitter* sufrido por las LDUs recibidas en los destinatarios (idea similar al concepto de LTS descrito en [AND91]). El algoritmo consigue la sincronización modificando el tiempo de reproducción de las LDUs y también adopta el mecanismo de '*saltos*' y '*pausas*' para recuperarse de situaciones de asincronía. Para la sincronización inter-flujo, el algoritmo selecciona un flujo como *maestro* y los otros como *esclavos*. En una primera fase denominada periodo de aprendizaje (*learning period*) que comprende los T milisegundos posteriores a la reproducción de la primera LDU, se realiza sólo control intra-flujo, y en ella se calcula una estimación del retardo de red. Después de dicho periodo, se realiza

tanto control intra como inter-flujo. Incluye dos alternativas para corregir las asincronías detectadas: *Quick Recovery* y *Graceful Recovery*. La primera trata de recuperar la asincronía lo más rápido posible pero puede degradarse la calidad de la reproducción. La segunda recupera la sincronización de forma gradual sin degradar la presentación. En ambos esquemas, el algoritmo recupera la sincronización mediante modificaciones del denominado *target output time* de las LDUs, que es el instante en el que el destinatario debería reproducir una LDU. Se distingue entre flujos fuertemente acoplados (*tightly-coupled streams*) y poco acoplados (*loosely-coupled streams*). En los primeros, cada LDU de los flujos *esclavo*s incluye, además de las marcas de tiempo (*timestamps*) necesarias para la sincronización intra-flujo, el número de secuencia de la LDU correspondiente del flujo *maestro* (aunque puede ser eliminada si dicho flujo produce LDUs de forma periódica). En el segundo tipo, sólo incluyen las marcas de tiempo.

Rothermel y Helbig, en [ROT95] y [ROT97], proponen un algoritmo de sincronización intra e inter-flujo adaptativo, denominado *Adaptive Synchronization Protocol (ASP)*, basado en un mecanismo de control del nivel de los *buffers* en recepción, permitiendo ajustes inmediatos cuando se detecta peligro de situación de *underflow* y *overflow* del *buffer*. Se basa en la existencia de relojes globalmente sincronizados, por ejemplo, usando NTP ([MIL92]) o relojes radio. Las principales características del protocolo son el soporte de fuentes y receptores distribuidos, una reacción inmediata a los cambios de las condiciones de la red o de los requerimientos de calidad de servicio, baja carga de mensajes de control y flexibilidad. Soportar cualquier distribución de flujos a sincronizar, las fuentes y los receptores pueden residir en diferentes nodos y los flujos pueden ir punto a punto o punto multipunto. A diferencia del algoritmo propuesto en [ESC94], los mensajes en ASP no se envían de forma periódica sino cuando se produce algún cambio. Las adaptaciones están coordinadas por un algoritmo *maestro*/*esclavo* dinámico, basado en la existencia de un flujo, denominado *maestro*, que toma decisiones sobre el punto de reproducción y las difunde al resto de flujos, denominados *esclavos*. Cualquier flujo, detecta que el estado de su *buffer* está entrando en la zona crítica de *overflow* o *underflow*, se puede convertir en *maestro* y realizar las acciones de adaptación pertinentes. Si se da el caso, el protocolo soporta la existencia de varios flujos *maestros*, dejando finalmente sólo a uno de ellos con dicha función. Cada receptor realiza la adaptación mediante variaciones de la tasa de reproducción durante determinados intervalos de tiempo, lo que puede conseguir de varias formas, como, por ejemplo, variando su tasa de consumo, o también '*saltando'* o duplicando unidades de datos o LDUs. También se podrían utilizar otros métodos dependiendo del tipo de flujo, como, por ejemplo, la modificación de los periodos de silencio en los flujos de voz. ASP está formado por cuatro subprotocolos independientes: el *de inicio*, el *de control del buffer*, el *de sincronización maestro/esclavo* y el *de conmutación de maestro*. Presenta dos políticas de sincronización: *la de retardo mínimo* y *la de pérdidas mínimas*. Ésta última es más apropiada para aplicaciones para las que sea más importante una

transmisión perfecta que un retardo extremo a extremo bajo. Finalmente, se presentan resultados de simulación del algoritmo tomando trazas de retardo de Internet, además de una distribución normal del retardo.

En [SHI95], Shivakumar et al., presentan el denominado *Algoritmo Concord*, que se ocupa, por un lado, de la sincronización intra-flujo y, por otro, de la sincronización inter-flujo en comunicaciones con varias fuentes transmitiendo hacia un único receptor. El algoritmo acepta tanto el uso de un tiempo global como la estimación de los retardos localmente. Consideran los posibles problemas de las desviaciones de los relojes del emisor y del receptor a la hora de dimensionar los *buffers* de recepción y de calcular el retardo total extremo a extremo, teniendo en cuenta el comportamiento dinámico de la red. El algoritmo describe la sincronización intra-flujo a través de *buffers* circulares en recepción teniendo en cuenta el *máximo retardo* y la *máxima tasa de pérdidas permitidos*, dejando que la elección de la configuración del compromiso entre uno u otro se decida por la propia aplicación, de forma estática. Utiliza una función de distribución normal del retardo sufrido por los paquetes, estimada según determinados parámetros, como son las condiciones del tráfico existente en el momento de inicio, información histórica del sistema, características negociadas del servicio, etc. El algoritmo parte de la premisa del conocimiento del retardo sufrido por el primer paquete enviado, para lo cual se podría hacer uso de NTP. Para el resto de paquetes el algoritmo ya no necesita el uso de marcas temporales. En caso de no poder conocer el retardo sufrido por dicho primer paquete, se propone la utilización de *Paquetes Especiales de Control*. El algoritmo puede variar la función de distribución del retardo a partir de las medidas del retardo realizadas para cada paquete, ya que el usuario tendrá acceso a estos datos y podrá incrementar o decrementar el tamaño del *buffer* en función de los requerimientos de la aplicación. Para la sincronización inter-flujo, el *Algoritmo Concord* considera N flujos de datos transmitidos desde N fuentes hacia un único receptor y trata a cada flujo de forma independiente con su *máxima tasa de pérdidas permitida* y se elige como *retardo máximo* el mínimo que permita que no se supere dicha tasa para ninguno de los flujos. En el algoritmo también se contempla el caso de aquellos flujos que, como es el caso de las aplicaciones multimedia, no necesitan una sincronización estricta sino que toleran algunos milisegundos de asincronía, mediante la definición del parámetro *asincronía máxima relativa* entre flujos. Se tiene en cuenta las variaciones del retardo extremo a extremo entre flujos. En [SRE96] se muestran los resultados obtenidos por el *algoritmo Concord* en un experimento realizado en Internet interconectando varios lugares de Estados Unidos y uno en el Reino Unido utilizando trazas obtenidas de Internet y UDP como protocolo de transporte, añadiendo números de secuencia y marcas temporales en el momento de la captura de los datos en el emisor. El experimento se realizó sin sincronización de relojes y consistió en la transmisión de flujos independientes y flujos dependientes en parejas de flujos de audio/vídeo, con parámetros típicos de transmisiones en la red MBone (*Multicast Backbone,* [MAC94, CAS98, GAR98, RUI99]). A diferencia de otros protocolos, el algoritmo

Concord no es adaptativo por sí mismo, sino que es el propio usuario el que, a través de la aplicación, puede realizar sus adaptaciones.

En [AKY96] se plantea la necesidad de la sincronización, tanto en entornos unicast (punto a punto), *multicast* (punto a multipunto), de recuperación de información y de multipunto a multipunto. Como principales fuentes de asincronía señala el *jitter*, las desviaciones en las tasas de consumo de los reproductores, los diferentes instantes de transmisión de las fuentes de diferentes flujos y los diferentes instantes iniciales de reproducción. El entorno que propone parte de las siguientes hipótesis: cada nodo reproductor se encarga de presentar todos los flujos de información, existiendo un proceso común a todos ellos y, además, supone que la distribución del retardo no cambia mientras dura la presentación. Para compensar el efecto del *jitter*, se utilizan *buffers* de reproducción. El problema de la desviación en las tasas de reproducción de los receptores lo elimina utilizando una referencia de tiempo global (*Tiempo Global Virtual* o *VGT*), de forma que todos los reproductores trabajan a la misma frecuencia. Se toma como referencia de sincronización, y de tiempos, uno de los usuarios del grupo denominado *chairman*, elegido en función de un sencillo algoritmo, basado en la elección de un número aleatorio. El protocolo presentado envía información temporal de forma implícita, a diferencia del envío explícito de otros protocolos como es el caso de NTP. Para el cálculo del instante inicial de reproducción conjunta, se toma el retardo de reproducción máximo de entre todos los flujos y todos los receptores, para lo cual es necesario un intercambio de los retardos de reproducción de cada flujo entre todos los receptores. Para asegurar asincronías del orden de varios milisegundos, los ajustes se producen con un periodo de cincuenta LDUs, lo que implica la generación de un tráfico considerable. Los resultados presentados, que se corresponden a una simulación, midiendo las derivas entre los relojes y el número de discontinuidades debidas a una situación de *underflow* en los *buffers*, indican que el protocolo propuesto se podría utilizar tanto para entornos LAN como WAN.

En [BAQ96] se propone un mecanismo de sincronización de flujos multimedia en entorno cliente/servidor y *multicast*, partiendo de la suposición de que la red utiliza un esquema de reservas estáticas y proporciona múltiples canales lógicos, cada uno con ancho de banda y límites de retardo conocidos diferentes. Además, supone el conocimiento del tamaño de las LDUs y de los *deadline* de reproducción de las mismas. El algoritmo presenta cinco métodos heurísticos para obtener soluciones aproximadas al problema de la planificación de la transmisión de LDUs. Los cinco planifican por adelantado la transmisión de LDUs, de tal manera que éstas puedan llegar al destino antes de su *deadline* de reproducción. Debido a las hipótesis de partida, el algoritmo propuesto es más apropiado para contenidos almacenados que para flujos *en directo*. También se identifican varios aspectos funcionales internos de los sistemas multimedia distribuidos y se presenta una arquitectura unificada de cinco niveles capaz de soportar diferentes aplicaciones multimedia distribuidas.

En [BIE96], Biersack et al., presentan un esquema para la reproducción continua y sincronizada de flujos multimedia almacenados distribuidos, enviados a través de una red de comunicaciones. Se ocupa tanto de la sincronización intra como inter-flujo. El esquema utiliza *timestamps* y número de secuencia en las LDUs (propone la utilización de RTP que incluye esta información), no requiere relojes sincronizados y está basado en el receptor. El esquema propuesto asegura la sincronización en un entorno con diferentes retardos, *jitter*, caídas del servidor, *drift* de los relojes y alteraciones del retardo promedio, para lo cual se presentan tres modelos: el *modelo 1* parte de la suposición de un *jitter* nulo, resuelve el problema de retardos diferentes pero fijos en las conexiones de red para cada subflujo y permite iniciar la reproducción sincronizada de un flujo formado por varios subflujos; el *modelo 2* se ocupa de la sincronización intra e inter-flujo y suaviza el efecto del *jitter* extremo a extremo (que considera limitado) mediante *buffers* elásticos de reproducción; y, por último, el *modelo 3* considera que el *jitter* no está acotado y se ocupa de resolver el problema del drift de los relojes, los cambios de las condiciones de la red y las caídas de los servidores mediante el control del nivel del *buffer* y el envío de mensajes de realimentación (*feedback*) a los servidores en caso de perturbaciones para recuperar la sincronización. Los mensajes *feedback* contienen un valor de *offset* que indica la cantidad de LDUs que el servidor debe bien '*saltar*' o bien '*pausar*' en la transmisión. El algoritmo se prueba sobre un prototipo de servidor de vídeo, denominado *Server Array*, consistente en *n* nodos servidores distribuidos, en el que un vídeo es distribuido entre los servidores utilizando una técnica denominada *Sub-frame Stripping*.

En [CHE96] se plantea el problema de la sincronización multimedia cuando los procesos reproductores se ejecutan en sistemas operativos multiproceso con tiempo compartido (*time-sharing multiprocessing O. S.*), pero que no incorporan características de tiempo real (por ejemplo, UNIX). Indica que la tarea de garantizar la sincronización en los sistemas multimedia debe ser llevada a cabo por los niveles de transporte y de aplicación. Supone la existencia de un nivel de transporte que garantice unas funciones básicas en cuanto a sincronización y se centra en el nivel de aplicación. Los resultados de los mecanismos tradicionales de sincronización entre flujos en las estaciones de trabajo varían en función de la carga del sistema, provocando degradaciones en la sincronización y disminuyendo la calidad de la presentación. Cuando el sistema se encuentra con una situación de sobrecarga, se producen dos efectos perjudiciales desde el punto de vista de la percepción: *discontinuidades* en flujos continuos y *asincronías* entre flujos. Se propone un modelo denominado *MultiSync*, que reparte las tareas de sincronización entre procesos (*threads*) dentro de un sistema multiproceso, y que garantiza la sincronización mediante la *asignación de prioridades* a cada flujo. A las LDUs de los flujos más importantes (audio), se les asigna prioridades más altas garantizando una reproducción continua, mientras que con las LDUs de los flujos menos importantes (vídeo) se utiliza una política de retardos o eliminación (*delay-or-drop*) por lo que su reproducción sufrirá '*saltos*' y/o '*pausas*', pudiendo, incluso,

llegar a eliminarse dichos flujos. La sincronización entre flujos se consigue mediante el intercambio de información entre procesos, o bien, mediante la estampación de tiempos. Las medidas se han realizado con flujos de vídeo (JPEG) y audio (8KHz) sobre un módulo, denominado CMP (*Continuous Media Playback*), implementado en una estación SUN Sparc10, con UNIX. Se presentan tres tipos de medidas: con carga en la CPU, con carga en el sistema de E/S y, simultáneamente, con carga en la CPU y en el sistema de E/S. Mediante una asignación adecuada de prioridades y una reducción de la tasa de reproducción hasta un valor de 13 tramas/seg se comprueba cómo el modelo propuesto disminuye el número de ocasiones en que se producen discontinuidades en el audio y el número de ocasiones en que se produce asincronía entre flujos.

El trabajo presentado en [KOU96] se considera fundamental por la similitud, en algunos aspectos, a la propuesta realizada en la Tesis. Presenta la primera implementación de sincronización entre flujos *multicast* en la red MBone. Describe un modelo o arquitectura para proporcionar sincronización de vídeo intra-flujo y sincronización inter-flujo basada en RTP/RTCP y NTP. Además, utiliza un bus de conferencia local basado en el denominado *Conference Control Channel Protocol* (CCCP, [HAN95]) desarrollado en el UCL (*University College of London*). Dicho mecanismo de sincronización se integra dentro de las aplicaciones *vic* ([CAN95]) y *rat* ([HAR95], [HAR96] y [KOU96]) que hacen uso de RTP/RTCP. Puesto que MBone estaba implementada sobre una red de conmutación de paquetes y los routers solían utilizar una disciplina de servicio FIFO y una multiplexación estadística, se introducía *jitter* en el tráfico de tiempo real. Se propone un sistema de sincronización intra-flujo, mediante la adición de un retardo artificial para eliminar dicho *jitter* que provocaría que el audio fuese ininteligible. Se centra en el concepto de sincronización labial o *lip-sync*. Indica que no es necesario que los dos flujos estén sincronizados exactamente sino que se puede tolerar una asincronía entre 80-100 milisegundos, basándose en los resultados obtenidos en [STE96] y [LAM96]. Señala que el retardo sufrido por los flujos de audio y vídeo en la reproducción es diferente, debido a que utilizan distinto hardware y distintos procesos. La reconstrucción del vídeo en recepción se realiza en función de las marcas de tiempo que transportan los paquetes RTP y que, en definitiva, facilitan el valor del tiempo proporcionado por el protocolo NTP en el instante de generación de la trama. La consecución final de la sincronización labial se puede obtener gracias a la utilización de *buffers de reconstrucción* tanto para el audio como para el vídeo (se añade a la herramienta *vic* para suavizar el efecto del *jitter* en el flujo de vídeo) y un sistema de comunicaciones entre procesos para poder realizar la sincronización dentro de los límites mencionados. Los procesos reproductores de audio y vídeo se intercambian sus retardos de reproducción. Así, una vez obtenido el retardo del otro flujo, cada proceso estará en disposición de realizar los ajustes pertinentes para obtener la sincronización deseada. En este caso, la decisión elegida es la de forzar en ambos el retardo de reproducción máximo. Kouvelas indica que para obtener la sincronización labial

hace falta añadir a las aplicaciones tradicionales de audio y vídeo las siguientes características: medidas de retardo extremo a extremo del audio y del vídeo, *buffers* de reconstrucción de vídeo, facilidades de negociación y cálculo del punto de reproducción deseado. Finalmente, muestra una *evaluación subjetiva* del modelo propuesto. También señala los problemas asociados a las pérdidas, al retardo extremo a extremo y al *jitter* introducido por la propia red y, a pesar de las ideas propuestas, hace hincapié en la problemática añadida de la falta de soporte de características de tiempo real en los sistemas operativos tradicionales y del *drift* o desviación de los relojes en los reproductores.

En [LAM96] se describe una arquitectura software de control para sincronizar diferentes flujos generados por un servidor de bases de datos multimedia y se integra en una aplicación de noticias bajo demanda (*News on Demand*). El diseño propuesto se centra en la entrega de información multimedia y en la sincronización de flujos. En los receptores se ejecuta un protocolo de sincronización de flujos con el fin de compensar los retardos introducidos por la red, denominado SSP (*Stream Synchronization Protocol*), en el que no es necesaria la utilización de un tiempo global. Éste se basa en la especificación de ciertos parámetros de QoS para garantizar la entrega y reproducción simultánea de los diferentes flujos. Además, también se presenta un interesante mecanismo de sincronización de flujos MPEG-2 basado en prioridades (según la importancia de las tramas I, P y B). El funcionamiento del protocolo se analiza formalmente mediante el uso de *DSPN* (*Deterministic Stochastic Petri Nets*) y es simulado mediante la herramienta *UltraSAN*. Para obtener las presentaciones multimedia desde el servidor, se establecen conexiones independientes entre el servidor y las estaciones receptoras. En primer lugar, realiza una distinción entre los esquemas de transmisión de datos multimedia, diferenciando entre la entrega en tiempo real, donde es necesaria una planificación de dicha entrega (normalmente en las fuentes), y la entrega basada en almacenar y reproducir (*store-and-forward*), donde la sincronización es más simple pero se necesita la utilización de grandes *buffers* de almacenamiento en el receptor y en la que el tiempo de respuesta es elevado cuando el usuario interactúa con el servidor. Es por ello que escoge el primer esquema para la transmisión de flujos multimedia. Se basa en la existencia, tanto en el servidor como en el cliente, de un planificador que crea los denominados *controladores de sincronización* o *MSC* (*Media Synchronization Controllers*) y los *Gestores de Calidad de Servicio* (*Client QoS Manager*) encargados de especificar las características de la transmisión de cada flujo. La planificación de la reproducción se basa en parámetros de QoS de la conexión como, por ejemplo, el retardo, la varianza del mismo, el *throughput*, etc. Se establece una negociación a tres bandas entre los clientes, los servidores y el sistema de transporte y, según el resultado, éste último debe ser capaz de reservar recursos para garantizar la QoS deseada o, en caso contrario, la petición de información será rechazada, pudiéndose iniciar una renegociación. Una vez se inicia la trasmisión de información desde varios servidores, se arranca un mecanismo para asegurar que no se inicie la

presentación hasta que no hayan llegado todos los elementos de los flujos que se deben reproducir en el instante inicial de la misma y, a continuación se sigue un diagrama de tiempos de la presentación garantizando así la sincronización. Ésta la obtiene el servidor compensando los retardos aleatorios mediante la introducción de *'LDUs duplicadas'*, *'saltos'* (video) y/o *'retardos intencionados'* (resto de flujos), técnica también conocida como *Política de Expansión de Tiempos* (*Time Expanding Policy*). Se presenta un prototipo de un servicio de noticias multimedia bajo demanda, con audio, vídeo y texto (subtítulos), en el cual se implementaba la sincronización labial entre audio y vídeo, y el texto se sincronizaba al flujo de audio. Se basa en un modelo cliente/servidor y utiliza llamadas a procedimientos remotos (RPC) para la comunicación entre componentes. La red de comunicaciones utilizada fue una red heterogénea ATM en la que se utilizaba el protocolo XTP (*eXpress Transport Protocol*).

En [LIU96] se presenta un sistema real que soporta una aplicación de teleconferencia multimedia distribuida multipunto. Describe la arquitectura, la gestión del establecimiento de la conexión (*sockets TCP/UDP*), el mantenimiento de la comunicación y el proceso de sincronización adaptativo intra e inter-flujo. Clasifica a los participantes en tres grupos: *participantes regulares* (pueden transmitir y recibir objetos multimedia), *observadores* (sólo pueden recibir) y *gestores* (administradores de la sesión). El mecanismo de compartición de recursos se basa en la utilización de un testigo (*token*) para asignar los turnos de intervención. Las conexiones utilizan TCP para el transporte de gráficos, texto y mensajes de control que requieren más fiabilidad, mientras que se utiliza UDP para la transmisión de audio y vídeo que requieren menores retardos. Para conseguir la sincronización intra-flujo, el emisor, en el sistema de transmisión, utiliza marcas de tiempo (*timestamps*) en las LDUs que permiten que el receptor pueda comparar los instantes de generación con el tiempo de un reloj virtual de reproducción, que emula al reloj del emisor y que se denomina PBC (*PlayBack Clock*). La diferencia entre los dos valores se utiliza para descartar las LDUs que llegan tarde y reajustar el valor del PBC. Para ello, para cada LDU, el tiempo en el receptor se divide en tres regiones: la *de espera* (la LDU se almacena en un *buffer*), la *de no espera* (se reproduce inmediatamente) y la *de descarte* (se descarta). Para la sincronización inter-flujo se necesita un PBC de grupo, que normalmente será el PBC más lento de los diferentes flujos, y las decisiones de descarte de LDUs se tomarán de acuerdo con dicho PBC. El objetivo del mecanismo propuesto es encontrar el punto de reproducción óptimo basándose en que los objetos multimedia originales pueden ser reconstruidos fielmente con una distorsión aceptable, manteniendo la tasa de pérdidas por debajo de un cierto límite. Se muestran medidas realizadas sobre una red FDDI y dos segmentos Ethernet, demostrando la adaptabilidad del algoritmo cuando cambian las condiciones de la red.

Rangan et al., en [RAN96], de forma similar a [RAN92] y [RAN93], también aportan nuevas ideas en el campo de la sincronización entre flujos

multimedia. En la mayoría de artículos referenciados en la presente Tesis no se especifican algoritmos de sincronización para flujos comprimidos MPEG debido a las dificultades que presentan. En éste, primero se describe, de forma detallada, el proceso de codificación y de decodificación MPEG, y, posteriormente, se presenta un método de sincronización entre flujos MPEG. Supone un sistema distribuido, con flujos MPEG almacenados en un servidor multimedia DSM (*Digital Storage System*), que serán transmitidos a los destinatarios, bajo demanda. En primer lugar, considera la sincronización entre varios flujos MPEG que se reproducen en un mismo destino, una vez recibidos desde el servidor. La sincronización entre los flujos de audio y vídeo, se basa en marcas de tiempo PTS (*Presentation Time Stamp*) y DTS (*Decoding Time Stamp*) que se incluyen en determinados paquetes (de forma periódica, no excediendo los 0,7 segundos, para actualizar los relojes de los decodificadores) y un 'mapeado' de tiempos entre emisor y receptor, basado en la referencia de reloj del sistema o SCR (*System Clock Reference*). En el receptor, el decodificador STD (*System Target Decoder*) se encarga de separar los flujos de audio y vídeo y decodificarlos por separado. Supone que el retardo de dicho proceso es nulo o que se puede compensar en el propio decodificador. Para la sincronización entre flujos propone, bien la utilización de uno como *maestro*, tomando como referencia su señal PTS, o bien la utilización del DSM como sincronizador. En la primera propuesta, un flujo *esclavo* como, por ejemplo, el de vídeo, se sincroniza con el *maestro* como, por ejemplo, el de audio, de forma que el decodificador de vídeo compara el valor de los PTS de las tramas de vídeo, con el reloj del decodificador de audio, eligiendo un mecanismo basado en '*saltos*' y/o '*pausas*', para corregir las asincronías en el proceso de decodificación. En la segunda propuesta se muestra la sincronización entre varios flujos MPEG considerando que los decodificadores de audio y de vídeo se encuentran distribuidos físicamente en distintos lugares dentro de la red de comunicaciones. En este contexto, en el que no se puede tomar uno de los flujos como *maestro*, ambos decodificadores reciben ambos flujos multiplexados MPEG, y descartan el flujo que no les concierne, según el caso. A partir de las marcas de tiempo del flujo que van a decodificar obtienen el instante de inicio de la decodificación. Como los decodificadores experimentan retardos aleatorios y diferentes en el consumo, ello conlleva a la aparición de valores de asincronía que se incrementan con el tiempo. En este caso, es el servidor DSM quien actúa como sincronizador y para ello solicita mensajes de realimentación o *feedback* a los decodificadores (igual que en [RAN92] y [RAM93]), conteniendo el número de trama y el valor de su PTS, de los que obtiene el progreso de cada proceso decodificador y, así, detecta las asincronías existentes y ejecuta las acciones correctoras oportunas. Éstas consisten en dejar de transmitir el flujo adelantado hasta que el atrasado alcance su estado de reproducción. No se presentan resultados sobre el método de resincronización.

Son y Agarwal, en [SON96], presentan un trabajo con bastantes similitudes respecto al trabajo presentado en la Tesis ya que se centra en la sincronización inter-flujo y, además, utiliza como base el protocolo *Feedback* propuesto en

[RAN92]. Utilizan un modelo cliente/servidor en el que se emplea un sistema de *tiempos de reloj normalizados*, que consiste en ejecutar, de forma periódica, un algoritmo de normalización del tiempo de reloj de los destinos al del propio servidor (ejecutado en el servidor), para subsanar el problema asociado a tener diferentes tiempos de reloj en cada uno. El servidor añade, por su parte, las mismas *marcas de tiempo relativas normalizadas* a las LDUs de diferentes flujos que hayan sido generadas en el mismo instante. El esquema es adaptable, simple de implementar e introduce muy poca sobrecarga en la red. Los receptores sólo tienen que enviar mensajes de realimentación o *feedback*, indicando el instante de reproducción de las LDUs que contienen las marcas relativas de tiempo (RTS). En función de la asincronía detectada por el servidor, activará políticas de resincronización agresivas o conservativas ([RAN92]). El trabajo se centra en un modelo de especificación basado en el modelo de *Redes de Petri* OCPN ([LIT90]), siendo capaz de mapear los requerimientos temporales provistos por las aplicaciones en los procesos de comunicaciones para obtener la información necesaria para asegurar la sincronización requerida. Hace hincapié en dicha posibilidad de adaptar el protocolo a los requerimientos de la aplicación, indicando que la precisión de los protocolos para detectar y corregir asincronías debería depender de las aplicaciones. El modelo inicialmente propuesto comprende varias fuentes distribuidas generadoras de flujos de datos, un servidor multimedia, donde se almacenan dichos datos, y destinos distribuidos que reciben dichos flujos bajo demanda. El trabajo también presenta una extensión del modelo para soportar flujos *en directo*, sin necesidad de utilizar equipos de almacenamiento masivo, pero incluyendo la existencia de un equipo *servidor de sincronización* que ejecute el protocolo de sincronización. Se analizan mediante simulación, las desventajas del modelo cuando se ejecuta en redes WAN, concluyendo únicamente que el error y su dependencia con el *jitter* disminuirán. Los resultados, presentados mediante gráficos, indican los errores cometidos por dicho protocolo de tiempo normalizado, para diferentes valores de *jitter* y diferentes intervalos entre resincronizaciones.

En [YAN96] se presenta la implementación de un mecanismo de sincronización inter-flujo, incorporado en un protocolo de transporte, para redes de conmutación de paquetes (o celdas), denominado *Multimedia Synchronization Transport Protocol (MSTP)*. Se basa en un nuevo modelo de sincronización en tiempo real, denominado RTSM (*Real Time Synchronization Model*), basado en la especificación de *Redes de Petri* OCPN extendidas (*XOCPN*). La modificación introducida respecto al modelo OCPN, consiste en obligar a que se produzca una transición en la *Red de Petri* cuando el flujo de información más importante o más sensible al *jitter* (*flujo clave*, concepto similar a *flujo maestro*) se encuentre en disposición de activar una transición, independientemente del estado de reproducción del resto de flujos, los cuales sufrirán reajustes en la reproducción (descarte de LDUs). Incluso, para evitar que un retardo mayor del esperado en el *flujo clave* genere asincronía, se utiliza un *flujo virtual* (flujo temporal , referencia de tiempo absoluto) que obliga a disparar dicha transición cuando se supere su

duración temporal (garantiza restricciones temporales). La introducción del mecanismo en el protocolo de transporte, facilita la implementación de las aplicaciones, las cuales únicamente deben preocuparse de la reproducción, pero, por el contrario, se limitan las relaciones de sincronización. En el transmisor existe un *gestor MSTP* que soporta el modelo RTSM y que obtiene los parámetros de QoS para cada flujo procedentes de las aplicaciones. Con estos parámetros establece conexiones de transporte con el *gestor MSTP del receptor*. Una vez la conexión está establecida, el *gestor MSTP* del transmisor enviará paquetes de acuerdo con el modelo MSTP. En el receptor, el *gestor MSTP* ejecuta las acciones de resincronización para garantizar las relaciones temporales y, por otra parte, la entrega de las LDUs de cada flujo a la aplicación correspondiente. De esta manera, las aplicaciones no necesitan controlar la sincronización en ambos extremos ya que se ocupa de ello el gestor MSTP. En el trabajo se presenta el formato de los paquetes de control y de datos del protocolo propuesto. Estos últimos, contienen el número de secuencia de la LDU del *flujo clave* con el que deben ser sincronizados. Se evaluó el mecanismo propuesto para una aplicación de videotelefonía, sobre una red Ethernet, pero simulando retardos (distribución normal) y tasa de error (1% para el vídeo, y 0% para el audio, tomado como *flujo clave*) propios de redes WAN. Se compara el comportamiento de la aplicación utilizando los modelos RTSM frente a OCPN, para protocolos subyacentes con y sin control de errores, reduciendo la asincronía y garantizando la sincronización del *flujo clave*.

En [ZAR96] se propone un mecanismo de sincronización de paquetes provenientes de diferentes fuentes a nivel de aplicación, que ataca el problema conocido como la *sincronización inter-participante* (entendida como la sincronización de los paquetes que llegan a un receptor desde el resto de participantes). El mecanismo propuesto se ejecuta periódicamente y no necesita de la existencia de una sincronización global de todos los relojes. También utiliza mensajes *feedback* y escoge un modelo estadístico para obtener la referencia de tiempos de las fuentes, a partir de la recepción de un determinado número de paquetes (N, escogido según la precisión deseada). A diferencia de [RAN92], en este mecanismo los mensajes *feedback* no son utilizados para asegurar la reproducción sincronizada sino para minimizar el tiempo de espera para la reproducción de los paquetes. El algoritmo propuesto se basa en el tiempo de llegada esperado de los paquetes, de tal manera que el receptor determina qué paquetes pertenecen a un mismo conjunto y, por tanto, deben ser reproducidos simultáneamente, a partir de sus tiempos de llegada, en vez de a partir de sus tiempos de generación. Se divide el eje de tiempo en intervalos periódicos, de tal manera que todos los paquetes que lleguen dentro de un mismo intervalo (más un tiempo de guarda adicional, para poder recibir paquetes retrasados que deberían haber llegado en dicho intervalo) son considerados como pertenecientes al mismo grupo de reproducción. Los receptores detectan los desvíos entre su reloj y el reloj de las fuentes y se lo comunican a éstas mediante mensajes *feedback*. Las fuentes bien realizan una '*pausa*' en su transmisión o bien insertan datos de ruido (*noisy*

data) dependiendo de si su reloj es más rápido o más lento que el reloj del receptor, respectivamente. Una vez determinada la referencia de tiempos, el *jitter* máximo y el tiempo de guarda, asegura la sincronización para los próximos N paquetes.

En [GUE97] y [GUE99], Guerri presenta el *protocolo Feedback-Global* de sincronización de flujos multimedia, que toma una referencia de tiempos global proporcionada por el protocolo NTP ([MIL92]) y se basa en mecanismos *feedback*, definiendo sus propios mensajes de control. En dicha Tesis se presenta una evaluación objetiva del protocolo propuesto, comparando los resultados obtenidos con el protocolo *Feedback* ([RAN92]), así como una evaluación subjetiva del mismo. Este trabajo de investigación ha constituido una base muy importante para la presente Tesis ya que ésta se enmarca en una de las líneas de investigación abiertas por el mismo.

En [ISH97], [ISH99] y [ISH01c], se presenta un mecanismo *maestro*/*esclavo* de *sincronización de grupo*, denominado esquema de sincronización *maestro* (*synchronization maestro scheme*) que sincroniza los diferentes destinos considerados como *esclavos* con un destino *maestro*, para contenidos multimedia almacenados en comunicaciones *multicast*. Contempla la sincronización intra e inter-flujo, tanto en el destino *maestro* como en los destinos *esclavos*, basándose en un mecanismo *maestro*/*esclavo* propuesto por el propio autor en [ISH95]. Considera que los relojes de los equipos están globalmente sincronizados (con NTP) y que no sufren desviaciones. La sincronización de grupo se consigue ajustando los tiempos de reproducción del flujo *maestro* en cada uno de los destinos *esclavos* de acuerdo con el tiempo de reproducción del mismo flujo en el destino *maestro*. Se basa en el envío de paquetes de control del destino *maestro* al resto de destinos, informando de su estado de reproducción. Los destinos *esclavos* también informan de su estado al destino *maestro*, que podrá modificar su estado de reproducción si detecta un elevado retraso con respecto a aquellos. Si detecta que está retrasado con respecto a un número determinado de destinos *esclavos*, adelantará su estado de reproducción. Para la sincronización intra e inter-flujo se utiliza el algoritmo de sincronización VTR (*Virtual-Time Rendering*, [ISH95]), en el que se establecen 15 posibles casos de desviaciones en los estados de reproducción para sincronización intra-flujo y 6 casos para sincronización inter-flujo y se indica cómo corregirlos mediante la política *Graceful Recovery* de VTR. Cabe destacar que el umbral seleccionado en este artículo para detectar asincronía entre un flujo de audio (*maestro*) y otro de vídeo (*esclavo*) es de 160 milisegundos. En [ISH97], Para evitar la congestión de la red, el flujo de vídeo MPEG se divide en tres flujos (de tramas I, tramas P y tramas B, respectivamente) que se envían, cada uno por un grupo *multicast* distinto. Según el estado de la red, los destinos estarán en uno o varios grupos *multicast*, recibiendo todas las tramas o sólo las de determinados flujos. Se toma como medida de la posible congestión de la red el tamaño de los *buffers* de recepción. Si se supera un cierto umbral de ocupación, se abandona algún grupo *multicast* con información

secundaria. Cuando el tamaño de los *buffers* descienda por debajo de un determinado umbral, de nuevo entrará en el grupo *multicast* abandonado previamente. En [ISH01c] se incluye el control causal de [ISH01a] al mecanismo de sincronización de grupo propuesto. En cada trabajo se presentan resultados de una prueba experimental, que consistió en el envío desde una fuente de dos flujos (audio como flujo *maestro* y vídeo como flujo *esclavo*) a través de una red a dos destinos (*maestro* y *esclavo*) en la cual se simularon retardos mediante diferentes distribuciones y diferentes valores medios, añadidos al enviar las LDUs de ambos flujos.

En [QIA97] se presentan protocolos, servicios y algoritmos para obtener sincronización labial (*lip-sync*), en una aplicación cliente/servidor adaptativa de Vídeo MPEG-1 bajo demanda (VoD) sobre un entorno *best-effort*. Muestra la situación en la que el sistema que soporta la aplicación no tiene siempre suficientes recursos para reproducir audio y vídeo con la misma tasa con la que fueron generados. Sigue la política de preservar la calidad del audio y eliminar intencionada y selectivamente información del vídeo en los sistemas finales. La arquitectura propuesta incluye la especificación de tasas, la adaptación del flujo de vídeo (con esperas o eliminación de tramas), sincronización labial, especificación de la sincronización (basada en ejes de tiempos) y un esquema adaptativo de sincronización tanto en los clientes como en los servidores. Debido a la sensibilidad del audio a las pérdidas, se toma el flujo de audio como *maestro* y divisor del eje de tiempos para controlar la reproducción y el flujo de vídeo como flujo *esclavo*. El protocolo tiene dos partes: *protocolo de negociación*, durante la fase de establecimiento de la comunicación, y *protocolo de adaptación/renegociación*, basado en mensajes de realimentación o *feedback*, durante la fase de transmisión. En esta aproximación, el flujo de comunicación entre el cliente y el servidor utiliza dos canales de comunicaciones: un canal TCP bidireccional utilizado para transmitir información sobre QoS, sobre el vídeo solicitado, etc., y otro UDP para transmitir audio y vídeo multiplexados.

Tasaka e Ishibashi, en [TAS98a] estudian un conjunto de esquemas de sincronización labial (*lip-sync*) para entornos con redes heterogéneas. El conjunto consiste en 4 tipos de esquemas (tipos 0 a 3), que son clasificados dentro de la aproximaciones de flujo único y de flujo múltiple (*single-stream* y *multi-stream*). Los tipos 0 y 1 pertenecen a la aproximación *single-stream*, que entrelaza voz y vídeo para formar un único flujo de transporte para la transmisión. Por otro lado, los tipos 2 y 3, que pertenecen a la aproximación *multi-stream*, forman flujos de transporte separados para los flujos de datos individuales. Mientas que los tipos 0 y 2 no realizan control de la sincronización en el destino (por lo que serán apropiados para redes que garanticen calidad de servicio extremo a extremo), los tipos 1 y 3 sí lo ejercen (por lo que serán apropiados para cualquier tipo de redes). En primer lugar, se presentan las características de cada tipo en términos de calidad de la sincronización requerida, la localización física de las fuentes de transmisión y la

complejidad de la implementación. A continuación, se describe algoritmo VTR ([ISH95]) para flujos de datos almacenados y se propone una modificación de dicho algoritmo para la transmisión de flujos MPEG, ya que se utiliza una codificación *inter-frame* (tramas I, P y B) y se debe tener cuidado al realizar ajustes mediante '*saltos*' de LDUs, ya que la decodificación de las tramas recibidas posteriormente puede depender de las tramas eliminadas. Se presenta la implementación de los 4 tipos, primero en una red ATM con servicios CBR y UBR y, posteriormente, en una red ATM con conexión inalámbrica (donde no necesariamente se pueda garantizar la calidad de servicio), haciendo uso del protocolo TCP (por razones de fiabilidad y simplicidad de la implementación) para transmitir flujos de voz y vídeo MPEG almacenados. Se obtiene el *error cuadrático medio* como el cuadrado de la diferencia entre el tiempo de reproducción real de una LDU (resultado de la sincronización intra e inter-flujo) y su tiempo calculado según el algoritmo de sincronización intra-flujo, además del tiempo total de pausa (debido a las acciones de ajuste mediante '*pausas*'), la tasa de bits de información (de audio o vídeo) reproducidos por segundo y el tiempo total de reproducción (que puede ser mayor al de la trasmisión debido a los retardos y a las pausas) y se comparan los resultados de los cuatro tipos, determinando la idoneidad de cada uno para diferentes condiciones y tráfico de la red de comunicaciones. También se realiza un análisis subjetivo pero no se presentan resultados sino únicamente comentarios del autor. En el caso de una red ATM con garantías de calidad de servicio (servicio CBR), concluye que se podría utilizar cualquiera de los 4 esquemas para obtener la sincronización labial, desde el punto de vista del funcionamiento y que, por tanto, se debería escoger uno en función de otros aspectos como la complejidad de implementación, calidad de la sincronización requerida y la localización de las fuentes. Para el caso de la red ATM con conexión inalámbrica se deberían seleccionar los tipos 1 ó 3, siendo el 3 el que proporciona mejor calidad en el flujo de audio.

En [TAS98b], además de los 4 esquemas propuestos en [TAS98a], se incluyen dos esquemas más (variantes del 1 y del 3, llamadas 1+ y 3+, respectivamente), que ejercen control de resolución dinámica de vídeo JPEG, haciendo uso, en este caso, del protocolo RTCP. Estos dos últimos esquemas cambian la resolución espacial o temporal del flujo de vídeo JPEG (cambiando el factor de quantificación *Q*), según la carga de la red. Se utiliza la información que contienen los paquetes RTCP RR (informes del receptor) para conocer la cantidad de paquetes RTP perdidos durante un cierto intervalo de tiempo frente a los enviados en ese mismo periodo de tiempo. En este caso, se implementan dichos esquemas en los mismos entornos que en [TAS98a] pero ahora haciendo uso de RTP/RTCP sobre UDP y transmitiendo tanto flujos JPEG almacenados como *en directo*. Se muestran sólo los resultados en el entorno de interconexión inalámbrica que son los más significativos. Además de evaluar los mismos parámetros que en [TAS98a], se incluye uno nuevo para la evaluación de la sincronización de flujos multimedia *en directo*: el *retardo medio de reproducción* (*Average LDU Delay*).

Éste consiste en el tiempo que transcurre desde que se captura una LDU en el emisor hasta que es reproducida en el receptor. Plantea la desventaja que tiene el uso de UDP como protocolo de transporte debido a la aparición de pérdidas de paquetes asociada al uso del mismo. Sugiere el uso de un protocolo de transporte con mecanismo de retransmisiones en aquellos casos donde las pérdidas de LDUs sean determinantes.

En [DIO99], Diot y Gautier, presentan el diseño, la implementación y evaluación de un juego multi-jugador distribuido sobre Internet, denominado *MiMaze*. En primer lugar se describe el juego y su arquitectura, que es una arquitectura de comunicaciones distribuida basada en la pila de protocolos RTP/UDP sobre *IP Multicast*. Se trata de un juego bastante simple, que representa en 3D a cada uno de los jugadores dentro de un laberinto y tienen que ir eliminándose unos a otros. También se explican las diferencias de un sistema distribuido frente a un sistema basado en servidor/es, en cuanto a robustez, escalabilidad, retardos mínimos, consistencia en la visión del juego por todos los participantes, sincronización, posibilidad de seguimiento del tiempo de juego de cada jugador (importante para los proveedores de juegos) y detección de trampas en el juego. En el trabajo se describen los mecanismos de control de la transmisión y las estructuras de datos utilizados. Cada participante envía periódicamente información del estado del juego en unidades de datos con identificación de usuario, números de secuencia y las correspondientes marcas de tiempo. Se trata del primer trabajo encontrado en la literatura que analiza una aplicación de juego interactiva y distribuida utilizando *multicast* en Internet. La principal contribución del trabajo presentado es mostrar, mediante un ejemplo, la factibilidad de esta familia de aplicaciones sobre una red que ofrece un servicio *best-effort*, como es el caso de Internet, actualmente. Para asegurar la consistencia en la visión del estado del juego de cada participante en cada momento, es necesario un mecanismo de sincronización que garantice los requerimientos de tiempo real de un juego interactivo. Proponen el *algoritmo de sincronización denominado bucket*, que trata de mantener la causalidad y sincronización de grupo de los datos de los jugadores, apoyándose en una infraestructura de tiempo global (por ejemplo proporcionado por el protocolo NTP). El tiempo se divide en intervalos de duración fija a los cuales se les asocia un *bucket* (cubo). MiMaze requiere que la información sea entregada a los jugadores en menos de 100 milisegundos. Todos los datos enviados por un participante en instantes de tiempo dentro de un intervalo se guardan en el *bucket* correspondiente a dicho intervalo y son evaluados por el algoritmo 100 milisegundos después de haber terminado el tiempo del intervalo. Se implementa, además, un sistema de recuperación de pérdidas basado en la estimación del estado actual de un jugador cuando se pierde la información enviada por el mismo. Se evalúa sobre MBone y se presentan aproximaciones para monitorizar y evaluar este tipo de aplicaciones. Cabe destacar que no se ha evaluado la escalabilidad del juego pues no se trata de un juego realista ya que los requerimientos en cuanto a tamaño de LDUs y capacidad de CPU son mínimos. Además, el algoritmo propuesto no

puede preservar la consistencia del juego, la causalidad y la igualdad entre jugadores cuando la variación en el estado de carga de la red es grande (lo que se traduce en pérdidas elevadas).

Xie et al., presentan, en [XIE99], el diseño e implementación de un esquema *buffer-oriented* de sincronización multimedia adaptativo, que incluye sincronización intra e inter-flujo para aplicaciones en tiempo real. Se propone que en el receptor se fuercen, dinámicamente, retardos ecualizados en los flujos entrantes para suavizar las variaciones del retardo de red y sincronizarlos entre sí. El esquema propuesto no se basa en relojes distribuidos: el *Reloj de Fuente* (*SC o Source Clock*) y los denominados *Relojes de Reproducción* (*POCs* o *PlayOut Clocks*). Como el *offset* entre ambos tipos de relojes determina los retardos extremo a extremo de las LDUs, el mecanismo adaptativo no se basa en estadísticas de estimación del retardo sino en un algoritmo contador de eventos (*event-counting algorithm*) utilizado para calibrar de forma adaptativa los *POCs*, que son los encargados de gestionar la presentación de los datos multimedia en el receptor, en respuesta a la dinámica de funcionamiento de la red. De esta forma, el algoritmo es inmune al *drift* de frecuencia de los relojes y a la pérdida de LDUs. La pérdida de sincronismo es cuantificada en términos de distorsión de la fase de sincronización. Además, el algoritmo también permite incorporar especificaciones de QoS definidas por el usuario dentro de los parámetros de diseño del mismo. La sincronización inter-flujo se realiza siguiendo un esquema *maestro*/*esclavo*, donde uno de los flujos se considera *maestro* (normalmente el más retrasado con respecto al reloj de la fuente o *SC*) y el resto serán los *esclavos*. Cuando se realiza cualquier calibración en el reloj *POC* del *maestro*, ésta repercutirá en las calibraciones de los *POCs* de los flujos *esclavos*. Se incluye una implementación del algoritmo en un sistema de teleconferencia punto a punto, con soporte de trabajo cooperativo y cara a cara (*face-to-face*), que consiste en canales de audio y vídeo controlados separadamente. Se presentan los resultados de la implementación y la idoneidad del esquema propuesto con respecto al tráfico multimedia a través de una red FDDI/ethernet. Dicho esquema se sustenta en la medida de tiempos relativos en el manejo de eventos relacionados con los objetos multimedia. La idea de utilizar un tiempo relativo coordinado se ha extraído del algoritmo VTR ([ISH95]), con la excepción de que a cada objeto se le añade una marca de tiempo en el instante de generación (inicio de la muestra) en vez de en el Punto de Acceso al Servicio o SAP (al final de la muestra).

En [GON00] se propone un algoritmo para conseguir tanto la sincronización intra-flujo como inter-flujo. Introduce la idea de un '*observador virtual*' colocado en el lugar de la escena para definir las relaciones temporales que deberían ser mantenidas dentro y entre diferentes flujos activados según diferentes eventos. También señala la existencia de dos modelos de sincronización: el *modelo de sincronización global* y el *modelo de sincronización diferenciada*. El primero trata de simular un encuentro '*cara a cara*' sintetizando en cada extremo una única

vista para todos los miembros de la sesión ([ESC94] y [ROT97]), y se consigue forzando en los receptores el máximo retardo existente para todos los flujos, entre la fuente y cada receptor. Este método tiene dos ventajas: los receptores experimentan la sensación de estar todos en una misma sala y se garantiza una ordenación global. Por otro lado, tiene los inconvenientes de imponer el peor retardo a todos los receptores y la necesidad de una sincronización global entre los relojes de todos los participantes de la sesión. El segundo modelo se basa en ecualizar a un retardo común sólo aquellos flujos producidos o controlados por un determinado participante, es decir, los flujos producidos o controlados por un miembro son presentados de una forma sincronizada a los otros miembros de la sesión, de acuerdo a los retardos emisor-receptor individuales. Al conjunto de flujos controlados por un miembro se le denomina *presencia multimedia* del participante en la sesión. Tiene la ventaja de no necesitar un reloj globalmente sincronizado pero el inconveniente de la falta de ordenación global, que lleva a tener diferentes condiciones para cada receptor. Ya que el retardo de comunicación (desde que se añaden las marcas de tiempo en los paquetes antes de transmitirlos hasta su recepción) es desconocido y variable, y no puede ser estimado a partir de medidas en un sentido (*one-way measurements*), el algoritmo no se ocupa de este retardo sino de preservar las diferencias entre escenas mientras se controla la ecualización del retardo medio. Para la sincronización intra-flujo, se diferencia entre los distintos tipos de flujos. En el caso del flujo de audio se indica que lo normal es la utilización de técnicas de reducción de los periodos de silencio y el descarte de paquetes que lleguen tarde, técnicas que funcionan adecuadamente en flujos de audio con discontinuidades o silencios (conversacional). Sin embargo, en caso de flujos audio continuos (por ejemplo, música) la reducción de períodos de silencio afecta considerablemente a la inteligibilidad de los mismos. En este trabajo, por tanto, se propone una técnica híbrida: si existen pausas, se utiliza la técnica denominada *Entrega Temprana* (*Early Delivery*), consistente en reducir el tiempo planificado para el paquete más antiguo del *buffer*; o bien, si transcurre un cierto tiempo sin la existencia de pausas (controlado por un temporizador), se utiliza una técnica de descarte de paquetes (eliminándolos directamente del *buffer* de recepción). No se contempla como descarte de paquetes sino como una política de resincronización y una reducción del retardo. Para el flujo de vídeo se proponen varias técnicas: para los paquetes que lleguen tarde se propone la técnica de *Entrega Tardía* (*Late Delivery*), consistente en reproducir los paquetes tardíos con el mínimo retardo de ecualización a pesar de la pérdida de sincronización que ello conlleve; para reducción del retardo se propone la *Entrega Temprana*; y, por último, para aumentos del retardo se propone la técnica de inserción de *gaps* o *Gap Insertion*. Para la sincronización inter-flujo se define el denominado *retardo virtual multimedia*, como el retardo común utilizado para representar todos los paquetes, independientemente del flujo al que pertenezcan. Se toma el mayor retardo de entre todos los flujos de la sesión multimedia. Se propone la utilización de un objeto centralizado para el cálculo y mantenimiento de dicho parámetro. Cada módulo de sincronización le envía a dicho objeto su retardo calculado pero tomará como

referencia el retardo virtual multimedia obtenido del mismo. La precisión de la sincronización inter-flujo dependerá de lo que se pueda aproximar cada flujo a dicho retardo.

En [ISH00] se presenta una comparativa de 38 algoritmos de sincronización multimedia en función de las técnicas de sincronización utilizadas en cada uno de ellos y de dónde se localicen (en la fuente o en el receptor).

En [TAS00], Tasaka e Ishibashi proponen un esquema de recuperación de errores a nivel de transporte utilizando retransmisiones, para transmisión de audio y vídeo a través de redes que no son capaces de garantizar una determinada QoS. Dicho esquema emplea una versión mejorada del algoritmo VTR ([ISH95]), denominada RVTR (*Retransmissin with VTR*), que ajusta el punto de reproducción de acuerdo con las condiciones de la red. Se aplica a sincronización inter-flujo, mientras que para la sincronización inter-flujo propone utilizar VTR. A diferencia de VTR, la nueva versión utiliza relojes globalmente sincronizados. Se evalúa con audio y vídeo JPEG *en directo*, utilizando RTP/RTCP sobre UDP/IP y ejecutando un mecanismo de retransmisión utilizando reconocimientos negativos sobre RTP/UDP. Los destinos solicitan la retransmisión de ciertas LDUs mediante peticiones y la fuente, si determina que la LDU solicitada puede llagar antes de su *output deadline* (límite del tiempo de reproducción), la retransmite. Por tanto, el fin del mecanismo de retransmisión será que cada LDU retransmitida llegue al destino antes de su *output deadline*, que vendrá determinado por el algoritmo de sincronización que se esté utilizando en el destino. Por tanto, podrán perderse LDUs debido no disponer de tiempo insuficiente para la retransmisión y, además, el orden de llegada de las tramas puede ser diferente al de su generación en la fuente. En el receptor se utilizan dos *buffers*: uno para las LDUs que llegan de forma normal y otro para las LDUs que han sido retransmitidas. El proceso de reproducción busca una LDU primero en el primer *buffer* y, si no lo encuentra, lo buscará en el segundo *buffer*. Si no la encuentra en ninguno de los dos, espera hasta que venza el tiempo límite de su reproducción. El receptor puede realizar dos acciones para mejorar el funcionamiento del esquema propuesto: bien incrementar el tiempo de reproducción calculado (dentro de unos límites) si se detectan pérdidas, para dar tiempo a que se puedan retransmitir las LDUs perdidas, o bien decrementar el tiempo de reproducción si detecta que la red no está congestionada. El algoritmo RVTR se evalúa en un entorno punto a punto, variando al carga de la red y añadiendo diferentes retardos con un simulador de enlace de datos (*Data Link Simulator*), llegando a la conclusión de que mejora la calidad de la sincronización tanto intra como inter-flujo, siempre que el *RTT* sea reducido, ya que la mejora introducida por el algoritmo disminuye a medida que éste aumenta.

En [ISH01a] se presenta un esquema de sincronización multimedia con control causal y de sincronización para aplicaciones multimedia distribuidas, en el que existe una relación temporal y causal entre flujos multimedia, tales como datos,

voz y vídeo. Además del control de la sincronización intra e inter-flujo, para el que se utiliza el protocolo VTR ([ISH95]), se ha implementado un control de causalidad para mantener las relaciones causales entre los objetos multimedia. El algoritmo de sincronización inter-flujo sigue un esquema *maestro*/*esclavo* y se aplica después de la sincronización intra-flujo. En este trabajo se toma al flujo de datos como el flujo *maestro*, ya que es el más importante en aplicaciones de juegos en red, siendo los *esclavos* los flujos de audio y video. Los equipos involucrados en la aplicación distribuida, envían el flujo de datos a un servidor que los entrelaza y los envía de forma *multicast* (UDP/*IP Multicast*) a todos los demás equipos. Por otro lado, los otros dos flujos se envían directamente (UDP/IP) entre los equipos involucrados. Se presenta una aplicación de juego de batalla en red para probar la efectividad del esquema propuesto. El control de causalidad se denomina *control de causalidad-Δ* y consiste en que cada LDU tiene un límite temporal o *deadline*, que se actualiza con el tiempo de generación de dicha LDU, más Δ segundos. Una LDU es descartada si llega después de su *deadline*, mientras que si llega antes es almacenada en un *buffer* para compensar el efecto del *jitter*.

En [ISH01b] y [ISH03], se presenta una evaluación, tanto objetiva como subjetiva de 9 combinaciones de 4 de las técnicas de sincronización reactivas más utilizadas por los autores anteriores (‘*saltos’*, *descarte de LDUs*, *estiramiento y acortamiento de la duración de la reproducción* y *contracción y expansión del tiempo virtual*) sobre una simulación con flujos de audio y vídeo. Concluye, en primer lugar, que un esquema que utilice conjuntamente acortamientos y estiramientos de la duración de la reproducción y la contracción y expansión del tiempo virtual para el flujo de voz, y, a su vez, acortamientos y estiramientos de la duración de la reproducción para el flujo de vídeo, es la que mejor calidad de sincronización labial produce. En segundo lugar, también concluye que la técnica de ‘saltos’ y descartes no es adecuada para el flujo de audio ya que afectan a la inteligibilidad del mismo.

En [KUO01] se presenta un esquema de transmisión adaptativo para asegurar la reproducción continua y sincronizada de flujos de audio y vídeo, basada en los protocolos RTP/RTCP y en la existencia de un temporizador global. El esquema está compuesto por una serie de operaciones en tres fases bien detalladas: *mecanismo de reordenación dinámica*, *mecanismo de decodificación y recuperación* y *mecanismo de sincronización adaptativa*. Este último mecanismo se realiza utilizando un temporizador global para ambos flujos en el receptor. Con el esquema propuesto se consigue minimizar el retardo medio en las colas de recepción controlando la velocidad de reproducción, manteniendo la sincronización entre flujos y manteniendo una reproducción estable. Además, mediante el proceso de sincronización adaptativa propuesto se consigue minimizar el retardo extremo a extremo, dependiendo del estado de la red. Se presentan los resultados de la simulación de dicho esquema y se analizan los efectos del *jitter*, el retardo extremo

a extremo y el tamaño del *buffer* requerido para ampliar la aplicabilidad del esquema a otras aplicaciones que requieran de la transmisión de datos multimedia.

Ishibashi et al., en [ISH02] proponen un esquema de sincronización conjunta entre contenidos almacenados con control interactivo y contenidos *en directo*, en comunicaciones *multicast*. A este tipo de sincronización entre estos dos tipos de contenidos (almacenados y *en directo*) lo denomina *joint synchronization*. Trabaja con control de búsqueda visual, tales como *fast-forward* y *fast-reverse* (por ejemplo, adelantar o retrasar, pausar o detener un vídeo pregrabado), como control interactivo. El esquema permite búsqueda visual mejorando el algoritmo de sincronización multimedia VTR ([ISH95]), y ajusta la temporización de cambio del modo de búsqueda visual entre los destinos llevando a cabo el control de la sincronización conjunta. Ya que VTR funciona de forma diferente para ambos tipos de contenidos, no sirve para la sincronización conjunta, por lo que los autores presentan un nuevo esquema en este trabajo y lo evalúan en un experimento en entorno WAN para demostrar la efectividad del mismo. Para la transmisión de los datos de los flujos se utiliza RTP/UDP sobre *IP Multicast* y se implementa el sistema de retransmisión RVTR ([TAS00]). Por simplicidad y para proporcionar interactividad, sólo contempla una retransmisión por LDU. En el trabajo se propone el uso de RTCP para el envío de paquetes de control para la sincronización de grupo, pero no la utiliza en la implementación por simplicidad. La información de control visual sobre los contenidos almacenados se lleva a cabo utilizando TCP. Para evitar la implosión de mensajes *feedback* cuando el número de destinos es grande, se propone utilizar el método de control de *feedback* propuesto en [BOL94]. Para la sincronización de grupo se adopta el método *maestro*/*esclavo* propuesto en [ISH97], mejorado para su utilización en entornos WAN. Cada destino envía al destino *maestro* de la sincronización la estimación del tiempo de reproducción de la primera LDU de los flujos *en directo*, o bien la estimación del *jitter* de red cuando éste sufra alguna variación importante. Con esta información el *maestro* de la sincronización enviará a los demás destinos, de forma *multicast*, el tiempo de reproducción de referencia elegido (en el artículo escoge el del destino más lento en reproducir la primera LDU o aquel que sufra un mayor *jitter*) para que ajusten sus tiempos de reproducción. Se toma como flujo *maestro* el flujo de audio *en directo*. En este trabajo se presenta el problema de la sincronización de flujos cuando se realiza control visual del flujo de datos almacenados (por ejemplo, un video). No se puede seguir el método tradicional de calcular el tiempo de reproducción para cada LDU almacenada, debido a tres problemas principalmente: la *dirección de la reproducción* (adelante, atrás), la *velocidad de reproducción* (lenta, rápida) y *el 'pausado' del flujo*. Se aporta solución a los tres problemas mencionados.

2.2.4. Mecanismos de Control Local de Conferencias

Al principio, la UIT disponía de un sistema sofisticado para el control de conferencias de flujos '*poco acoplados*' (*loosely-coupled*) basadas en sistemas de codificación de vídeo H.261 y redes RDSI, incorporando los protocolos T.120, utilizando unidades de control multipunto (MCUs o *Multipoint Control Units*) y un servicio de comunicación *multicast*. En su diseño original, dicho esquema dependía de un modelo de control centralizado y de la fiabilidad y de la naturaleza de tasa constante de dichas redes. Más recientemente, el sistema ha evolucionado para poder incluir las redes de conmutación de paquetes y una aproximación *loosely-coupled* y distribuida para el control de conferencias haciendo uso de los esfuerzos de investigadores de los grupos de trabajo de la UIT y del IETF ([CRO99]).

El IETF ha desarrollado una aproximación basada en el uso de *IP Multicast*, que se puede utilizar desde para el control local de un sistema (por ejemplo, enviando de forma *multicast* paquetes con TTL igual a 0 para controlar un conjunto de procesos asociados en la misma sesión dentro de un mismo sistema local) hasta para la formación de la sesión de forma *multicast* en un entorno de área amplia, así como para el control de la conferencia, por ejemplo, utilizando un sistema como el *Conference Control Channel Protocol (CCCP)* descrito en [HAN95], y desarrollado como parte del proyecto MICE (*Modular Integrated Communication Environment*, [HAN95a]), que permite construir varios tipos de aplicaciones de control. Normalmente, se utiliza esta aproximación para permitir la escalabilidad a grandes números de participantes en vez de utilizar aproximaciones basadas en MCUs. MICE fue un proyecto del UCL, fundado entre 1992 y 1995, que surgió con la necesidad de promover conferencias de audio y vídeo mediante herramientas *multicast* utilizadas en MBone. Debido a la utilización de varias herramientas simultáneamente, se detectó la necesidad de utilizar un canal común mediante el cual dichas herramientas pudieran intercambiar mensajes de forma local (sólo entre las herramientas de una misma máquina), además de la utilización de una herramienta que se encargara, de forma centralizada, de coordinar todos los mensajes de control, necesarios para un buen funcionamiento de todo el sistema.

En [HAN95] y [CRO99] se define el Canal de Control de la Conferencia CCC (*Conference Control Channel*), que es un canal de comunicaciones común para unir los constituyentes de la conferencia y que les ofrece facilidades y servicios para intercambiar información entre sí. Es similar a las facilidades de comunicación entre procesos ofrecidas por los sistemas operativos. También se define el protocolo que actúa sobre el canal CCC, denominado CCCP (*CCC Protocol*), sus requerimientos, nombres utilizados, sistema de direccionamiento, intercambio de mensajes fiables y no fiables, incluso se proporcionan ejemplos de aplicación.

Dentro de la comunicación entre procesos locales dentro del mismo sistema, se puede destacar el bus de mensajes denominado *Message Bus o mbus* ([PER98], [OTT99a], [OTT99b], [KUT00] y [OTT00]) que está basado en el CCCP y que será utilizado en la presente Tesis. En [PER98] se describe, brevemente, cómo utilizar la librería de funciones de *mbus*. En [OTT99a], aparecen explicados los mensajes y comandos más comunes definidos en la especificación de *mbus*. En [OTT99b] se presentan los requerimientos que se deben cumplir para realizar un control local de una conferencia donde existen diferentes aplicaciones para cada flujo e incluso herramientas de control de la misma (de accesos, de participantes, etc.), que necesiten intercambiar datos entre sí. En [OTT00] se presenta la infraestructura de coordinación utilizando mensajes locales para las comunicaciones entre aplicaciones agrupadas dentro de aplicaciones que las engloban de forma coordinada. Se define el formato de los mensajes, el sistema de direccionamiento, la sintaxis, el tipo de mensajes y cómo se realiza el transporte de los mismos. En el *White Paper* [KUT00] se explican las características y las ventajas de la utilización de *mbus* en sistemas basados en componentes (*Component-based Systems*) y cómo se puede utilizar en sistemas CTI (*Computer Telephony Integration*).

En otro proyecto, explicado posteriormente, denominado *RELATE*, en vez de utilizar un método tradicional de comunicaciones entre procesos, ya se utilizó el sistema *mbus* para la integración de información del audio y del vídeo en una misma conferencia.

El sistema local de mensajes *mbus* se explica más detalladamente en el capítulo 3, ya que ha sido utilizado en el trabajo desarrollado en la Tesis.

2.2.5. Aplicaciones y Herramientas

Existen dos enfoques a la hora de definir aplicaciones para la transmisión de flujos multimedia: el de la UIT (que propone diferentes estándares, siendo el más conocido e implementado el H.323) y el del IETF (que propone la utilización de *IP Multicast* [DEE89]), tal y como se ha venido haciendo sobre MBone en los últimos años. Ambos enfoques se describen a continuación, centrándose más en el segundo ya que es el elegido para implementar el algoritmo propuesto en la Tesis.

2.2.5.1. Enfoque UIT: H.323

El estándar H.323 está bajo el amparo de la UIT y proporciona una base para las comunicaciones de audio, video y datos a través de una red IP. Se trata de un conjunto de estándares para la comunicación multimedia sobre redes que no

proporcionan QoS. Estas redes son las que predominan hoy en día, como redes de conmutación de paquetes TCP/IP e IP sobre Ethernet, Fast Ethernet, etc. Los estándares H.323 son bloques importantes de desarrollo para un amplio abanico de aplicaciones basadas en redes de paquetes para la comunicación multimedia y el trabajo colaborativo. Incluyen desde dispositivos específicos hasta tecnologías integradas en ordenadores personales, además de servir para comunicación punto a punto o conferencias punto multipunto. H.323 trata, también, el control de llamadas, la gestión multimedia y la gestión del ancho de banda, además de las interfaces entre redes de paquetes y otras redes (RTC, por ejemplo). H.323 forma parte de una gran serie de estándares que permiten la videoconferencia a través de redes de datos. Conocidos como H.32X, esta serie incluye H.320 y H.324, que permiten las comunicaciones sobre RDSI y RTC, respectivamente.

Inicialmente, el esquema de H.323 funciona muy bien en conferencias punto a punto pero, a medida que aumenta el número de participantes en la videoconferencia, se requiere de elementos extra, como, por ejemplo, las unidades de control MCUs, que se encargan de replicar los datagramas para los diferentes usuarios. Cuando, además, se requiere un cierto control sobre las conferencias es necesario otro nuevo elemento llamado pasarela o *Gatekeeper*, que es el encargado del control de llamadas, traducción de direcciones, control del ancho de banda, etc.

H.323 ha tenido gran éxito debido a la existencia de numerosas aplicaciones que lo implementan, y cuyo uso se ha popularizado. Sin embargo, para videoconferencias con un gran número de usuarios H.323 acaba siendo ineficaz y se requieren otros mecanismos basados en *IP Multicast*.

2.2.5.2. Enfoque IETF: *IP Multicast*

La otra alternativa al uso de H.323 es el uso de *IP Multicast* ([DEE89]), cuyo uso se ha venido experimentando en los últimos años sobre la red MBone. La distribución de paquetes *multicast*, que lleva ya mucho tiempo funcionando sobre Internet (*IP Multicast*), no solamente se está utilizando para las comunicaciones tradicionales de datos entre ordenadores sino que, con la aparición de las aplicaciones multimedia, se está utilizando también para distribuir tráfico en tiempo real. Los grupos *multicast* para distribución de voz y vídeo pueden estar formados por grupos que van desde pocos usuarios o receptores hasta cientos o miles de receptores, como puedan ser las aplicaciones de distribución de televisión. La transmisión incontrolada de flujos de audio y vídeo fácilmente ocuparía todos los recursos de Internet, causando congestión y degradaría la QoS ofrecida para todos los usuarios de la red.

En [BRA98] se presentan una serie de protocolos de transporte *multicast* y en [LI99], se analiza el problema de la transmisión de vídeo *multicast* sobre

Internet y hace una revisión de las ideas y técnicas desarrolladas hasta la fecha para proporcionar transmisión *multicast* de vídeo adaptativa, exponiendo sus ventajas e inconvenientes, sobre todo, en cuanto a ancho de banda empleado y a escalabilidad.

2.2.5.3. Aplicaciones y Herramientas.

Se van a tratar algunas de las aplicaciones más conocidas, aunque, posiblemente, existan muchas más que son desconocidas para el autor de la presente Tesis ya que hay gran cantidad de proyectos dedicados a la distribución de contenidos multimedia y se están mejorando constantemente las herramientas existentes y desarrollando nuevas y mejores aplicaciones cada día.

Los dos enfoques anteriores (H.323 e *IP Multicast*), como es lógico, tienen sus ventajas y sus inconvenientes. Sin embargo, el gran reto está en ser capaces de unir lo mejor de los dos enfoques en un mecanismo común y estable. Muchos de los esfuerzos que se realizan actualmente en el tema de videoconferencia van encaminados al desarrollo de pasarelas (*gateways*) que faciliten precisamente esto. Tanto con un enfoque como con el otro se dispone de una serie de herramientas que permiten, de un modo cada vez más estable, la transmisión de eventos, congresos, teleseminarios, teleconferencias, cursos a distancia, etc.

Herramientas H.323

Existen multitud de herramientas basadas en H.323 pero, en realidad, las diferencias entre unas y otras son mínimas. Casi todas estas herramientas son comerciales y ofrecen tanto audio como videoconferencia y, en algunos casos, compartición de datos. De todas, la más famosa es *Microsoft NetMeeting*, sin embargo hay otras muchas aplicaciones desarrolladas por otras empresas que ofrecen una funcionalidad similar. El mayor problema que presentan este tipo de herramientas para la formación en línea es la escalabilidad, siendo solamente útiles para un número muy reducido de participantes.

Otra herramienta que se basa en H.323 es *Real Server/Player*, que también se basa en el empleo de RTP y tiene la ventaja de que puede iniciarse vía web. Su principal problema, sin embargo, está en que no permite interacción. Los receptores deben limitarse a atender a las sesiones sin poder, en ningún momento, interactuar con el interlocutor. Por otro lado, tiene la ventaja adicional de permitir que los contenidos estén almacenados en un servidor y se puedan emitir cuando se considere oportuno.

Herramientas MBone

En los últimos años, se ha producido un esfuerzo de colaboración en la comunidad de investigadores que ha dado lugar a un gran abanico de aplicaciones de teleconferencia en Internet. El aumento de potencia de las estaciones de trabajo y de los PCs, junto con la aparición de dichas aplicaciones de audio y vídeo implementadas y probadas en MBone, han incrementado un elevado porcentaje el tráfico de tiempo real a través de Internet. El desarrollo de *IP Multicast* ha promovido el desarrollo de un conjunto de aplicaciones, colectivamente denominadas herramientas MBone, para conferencias multimedia a través de Internet.

Dentro de este amplio conjunto de aplicaciones que han aparecido en estos últimos años (gran parte de ellas continúan en desarrollo), las que más éxito han tenido en MBone han sido las que permiten la realización de conferencias de audio y vídeo. Entre dichas herramientas se pueden citar *nv*, *vic*, *vat*, *rat*, *wb*, *nt*, *SDR, nevot*, *nv*, *IVS*, *Cu-SeeMe*, *IPTV*, *Rendez-Vous*, etc., y todas ellas trabajan sobre RTP/RTCP. De todas ellas se van a comentar brevemente algunas que se han estudiado durante el desarrollo de la Tesis. Se puede encontrar información sobre la mayoría en la página web del UCL[(5)].

Las primeras aplicaciones de transmisión de vídeo sobre MBone fueron las herramientas de vídeo *Xerox PARC Network Video Tool*, *nv* ([FRE94]), el sistema de videoconferencia de INRIA (*Institut National de Recherche en Informatique et en Automatique*), *ivs* ([TUR94]), y la aplicación desarrollada en UCL/LBL de transmisión de vídeo, *vic* ([CAN95]). En [FRE94] se describen los objetivos de diseño de *nv*, su algoritmo de compresión de vídeo y se muestran algunas experiencias de transmisión de vídeo sobre Internet, utilizando dicha herramienta. Se trata de una herramienta que permite transmitir vídeo de mayor o menor calidad según el ancho de banda disponible, que permite ser utilizado en redes LAN y/o en redes WAN con accesos de baja velocidad. El *IVS* (*INRIA Videoconference System*) [TUR94], desarrollado por el INRIA, permite la integración de los canales de audio y vídeo sobre una misma aplicación. Esta aplicación fue desarrollada en 1992 y dejó de ser mantenida para dar paso a otras nuevas aplicaciones como el *Rendez-Vous* (nueva generación del *IVS*). La aplicación *vic* ([CAN95]) se explicará más detalladamente en el próximo capítulo ya que ha sido utilizada en el trabajo desarrollado en la Tesis.

Entre las herramientas de audio, se pueden destacar, *vat* ([JAC92]), *nevot* ([SCH92]) y *rat* ([HAR95], [HAR96] y [KOU96]), que son las más empleadas en MBone. También se puede mencionar la herramienta *FreePhone*, objeto de evolución de *IVS*. Todas ellas permiten la transmisión de audio de cierta calidad,

[5] http://www-mice.cs.ucl.ac.uk/multimedia/software/documentation/

tanto en entornos uno-a-uno (*unicast*) como en entornos uno-a-muchos (*multicast*). La herramienta *rat* también será explicada más detalladamente en el próximo capítulo, ya que también ha sido utilizada en la Tesis.

En cuanto a herramientas de pizarra compartida podemos destacar *wb* ([CAN92]) y *DLB*[6]. *Wb* es una herramienta que ofrece una pizarra compartida por todos los usuarios de la sesión *multicast*, que puede emplearse como una pizarra usual de las que se dispone en un aula pero de modo distribuido. Dispone de una serie de utilidades tanto para escribir texto con diferentes formatos (tamaños y colores), así como para realizar dibujos a mano alzada y formas geométricas sencillas. Permite, además, incorporar archivos en formato *PostScript* a la pizarra. Por otra parte, *DLB* es una herramienta de pizarra electrónica bastante mejoraque permite trabajar en dos modos bien diferenciados *on-line* y *off-line*. De este modo se permite la preparación de contenidos en modo *off-line* para posteriormente presentarlos *on-line*.

El editor de texto *nte* (*Network Text Editor*)[7] ofrece la posibilidad de editar textos de un modo distribuido. Soporta *tokens* para pedir el turno y es bastante completa. Por otro lado, permite a múltiples usuarios trabajar en tiempo real sobre un mismo documento de texto: crear nuevos párrafos, borrar, cambiar y mover los existentes, etc.

Estas son las herramientas MBone más comunes, no obstante, este campo sigue evolucionando y va más allá, ofreciendo cada vez herramientas más potentes. El código fuente de las herramientas MBone (*MBone tools*) se pueden encontrar de forma actualizada en la página del UCL[8], así como los ejecutables de distintas versiones. La información sobre este software utilizado generalmente en MBone se puede encontrar en el web de la Red Iris[9] o del UCL[10].

También se han desarrollado interfaces gráficas capaces de integrar todo este tipo de aplicaciones de audio, vídeo, texto y pizarra (que pueden funcionan independientemente) como si se tratara de una sola aplicación y que pueden ser ejecutados vía web. Se pueden destacar algunos grandes proyectos como, por ejemplo:

- El proyecto MASH[11] fue un proyecto muy ambicioso iniciado en Berkeley que, además de intentar agrupar todas las herramientas típicas de videoconferencia MBone en una interfaz común, ofrecía la posibilidad de

[6] http://www.informatik.uni-mannheim.de/informatik/pi4/projects/dlb/project.html#5
[7] http://www-mice.cs.ucl.ac.uk/multimedia/software/nte/
[8] http://www-mice.cs.ucl.ac.uk/multimedia/software/
[9] http://www.rediris.es/mmedia/
[10] http://www-mice.cs.ucl.ac.uk/multimedia/software/documentation/
[11] http://www-mash.cs.berkeley.edu/mash/

grabar sesiones y reproducirlas, así como integrarlo todo en web, de forma que los usuarios podían acceder directamente a los contenidos desde el navegador.

- *MiNT* (*Multimedia Internet Terminal*)[12] es otra aplicación del GMD FOKUS (*Gesellschaft für Mathematik und Datenverarbeitung mbH Bonn, Institute for Open Communication Systems*)[13] muy completa, que soporta parcialmente el protocolo SIP (*Session Initiation Protocol*), incorpora un agente de reservas RSVP ([ZHA93], [BRA97], [WHI97]), un agente que permite el establecimiento y control de sesiones multimedia a través de Internet e incluso una interfaz común para las herramientas de audio y vídeo. Presenta una arquitectura totalmente distribuida, sin componentes centralizadas. La transmisión de audio se realiza mediante las herramientas *vat* o una nueva herramienta desarrollada en este proyecto denominada *nevot*[14]. La transmisión de vídeo se realiza mediante la herramienta *vic*.

- *RELATE* (*REmote LAnguage Teaching over SuperJANET*)[15]. Se trata de un proyecto, realizado conjuntamente por el UCL y la *Exeter University*, de desarrollo de una aplicación con la intención de transmitir audio y vídeo sincronizado (*lip-sync*) para la enseñanza de diferentes lenguajes vía MBone, utilizando *vic*, para vídeo, *rat* para audio y las aplicaciones de edición de texto y/o pizarra electrónica (*nte* y/o *wb*). Sirvió también para evaluar el uso de Internet y MBone para proporcionar tutorías a distancia en un campo de prueba tanto para estudiantes como para profesores.

Otra utilidad fundamental en MBone es el *SDR* (*Session Directory Tool*)[16], que permite conocer las sesiones MBone que estaban activas en todo momento, para permitir a los usuarios conectarse a cualquiera de ellas, o definir una sesión MBone propia. La primera implementación del *SDR*, llamada *SD*, fue desarrollada por Van Jacobson en el *LBNL* (*Lawrence Berkeley National Laboratory*). El desarrollo posterior de la misma ha sido llevado a cabo dentro del proyecto europeo de investigación en herramientas multimedia *MERCI-ESPRIT* (continuación del proyecto MICE [HAN95a]), dando lugar al actual *SDR*. Ciertos cambios fundamentales en su estructura han hecho que ambas versiones sean incompatibles y los anuncios de sesiones se hacen de forma generalizada con la nueva versión desarrollada por el *MERCI*, el *SDR*. El *SDR* emplea una dirección *multicast* asignada por la *IANA*, la 224.2.127.254 y un puerto UDP específico (el 9875) para retransmitir los anuncios de sesiones multimedia. Esto se implementa

[12] http://www.fokus.gmd.de/step/mint/content.html
[13] http://www.fokus.gmd.de/
[14] http://www.cs.columbia.edu/~hgs/rtp/nevot.html
[15] http://www-mice.cs.ucl.ac.uk/multimedia/software/relate-ui/
[16] http://www-mice.cs.ucl.ac.uk/multimedia/software/sdr/

por medio de un protocolo muy simple, en el que los datos de la sesión van incluidos en forma de texto ASCII tras una breve cabecera UDP. Esta simple implementación permite que estos datos puedan ser capturados fácilmente por una aplicación y puedan emplearse para lanzar las aplicaciones necesarias para unirse a una determinada conferencia.

También hay que destacar un nuevo tipo de herramientas denominadas de trabajo cooperativo o CSCW (*Computer Support for Colaborative Work*), que son muy interesantes a la hora de coordinar a diferentes personas situadas en diferentes localizaciones para trabajar en equipo. De todo este tipo de herramientas, una de las destacadas es *BSCW*[17] (*Basic Support for Cooperative Work*) desarrollada en el GMD ([APP95]) que tiene la ventaja de manejarse totalmente vía web y permitir la integración con herramientas síncronas de audio y videoconferencia. BSCW es la herramienta que se ha utilizado en la Red Iris para integrar las aplicaciones de videoconferencia ofrecidas por MBone y favorecer la formación de CVUs (*Comunidades Virtuales de Usuarios*[18]) en la comunidad académica y científica nacional.

Dado el carácter experimental de MBone, no son numerosas las aplicaciones comerciales que aprovechen las ventajas que esta red ofrece. Se han desarrollado gran variedad de aplicaciones comerciales de intercambio multimedia y videoconferencia basadas en los protocolos más establecidos de encaminamiento uno-a-uno o unicast, aunque también es cierto que cada vez más irán integrando ambos modos de operación, tanto uno-a-uno como uno-a-muchos. Muchas empresas ya han anunciado sus productos basados en RTP como, por ejemplo, *LiveMedia* presentada por *Netscape Communications Corporation*, y, por su parte, *Microsoft Corporation* ya ha anunciado la compatibilidad de sus productos con RTP. Como es lógico, está fuera del propósito del presente capítulo enumerar y describir las aplicaciones comerciales que, continuamente, van apareciendo y/o mejorándose.

2.2.6. Evaluación Subjetiva

En los sistemas multimedia, la sincronización es esencial en la reproducción en el interfaz de usuario de los diferentes flujos multimedia involucrados en el mismo. No es posible proporcionar una medida objetiva de la sincronización desde el punto de vista de la percepción subjetiva del ser humano. Ya que la percepción varía de una persona a otra, sólo criterios heurísticos pueden determinar si la presentación de un flujo es correcta o no.

[17] http://orgwis.gmd.de/projects/BSCW/
[18] http://www.rediris.es/cvu

En el capítulo 7, se ha incluido un estudio sobre la *evaluación subjetiva* del algoritmo de sincronización propuesto, considerando dicho tipo de evaluación como una forma alternativa y complementaria a la evaluación objetiva de la calidad obtenida con dicho algoritmo. En la mayoría de la aplicaciones multimedia, el usuario, a través de su percepción, es el que, en último término, acepta o no la presentación.

En cuanto a estudios sobre valoración subjetiva de la calidad de la sincronización, se han encontrado los siguientes trabajos: [GIL95], [STE96], [KOU96], [GUE97], [WAT97a], [WAT98], [KAL99], [WIL00], [ISH01b] y [ISH03]. Además, existen otros trabajos sobre la calidad subjetiva de la imagen como, por ejemplo, [GIL95].

En [STE96] se estudia de forma muy exhaustiva y completa el efecto de la asincronía entre flujos multimedia en la percepción humana. En dicho trabajo se busca fijar una serie de requerimientos de QoS referida a la sincronización entre flujos multimedia (inter-flujo). Un primer experimento consistió en reproducir secuencias de audio y vídeo sincronizadas y no sincronizadas. Estas últimas se obtuvieron variando, mediante equipos profesionales de edición, la asincronía o *skew* entre sendos flujos, desde -320 milisegundos, hasta 320 milisegundos, en saltos de 40 milisegundos. Para el test se consideraron tres planos diferentes del locutor (primer plano, plano medio y plano lejano) y, a partir de las respuestas que un conjunto de sujetos (107 personas, de ambos sexos y de todas las edades) contestaron después de visualizar una serie de secuencias, se obtuvieron los requerimientos mínimos para garantizar una sincronización aceptable por los usuarios. La principal conclusión del trabajo es la estimación del intervalo dentro del cual se puede considerar que los flujos de audio y de vídeo se encuentran sincronizados, que es [-80, +80] milisegundos. También se dedujo del experimento que la percepción humana tolera mejor que el vídeo se adelante al audio que viceversa. Por otro lado, un segundo experimento consistió en comprobar los efectos de las asincronías en presentaciones multimedia distribuidas, cuando existen punteros que remarcan algún punto de un gráfico (en definitiva, datos). En este caso, es más difícil la detección de asincronías y la conclusión del trabajo fue que se consideraba el flujo de audio sincronizado con el puntero si el audio no se adelantaba más de 750 milisegundos con respecto al posicionamiento del puntero o bien si, por otro lado, el posicionamiento del puntero no se adelantaba más de 500 milisegundos con respecto al audio. Además, se recogen una serie de valores de asincronías máximas permitidas para otras configuraciones diferentes a las anteriores, como, por ejemplo, entre pistas de audio de una grabación; en presentaciones con distintas combinaciones: audio y animaciones, audio e imágenes estáticas o gráficos (transparencias o diapositivas), audio con texto, video con texto o imágenes estáticas; e incluso en el procesamiento en tiempo real de datos de control (telecirujía). Todos los resultados mostrados, así como las conclusiones, son de gran utilidad para la evaluación de cualquier protocolo de

sincronización. En dicho trabajo también describe cómo calcular los parámetros de calidad de asincronía cuando en el sistema multimedia se agrupan más de dos flujos y la aplicación impone restricciones en cuanto a máximas asincronías permitidas para cada pareja de ellos.

En [KOU96] además de describir la técnica de sincronización ya comentada, se realizan unas medidas subjetivas sobre una población de 8 personas, para diferentes tasas de reproducción (2, 4, 6, 8, 10 y 12 tramas por segundo) y se comparan los resultados con el caso de no utilizar ningún protocolo de sincronización. La escala de valores utilizada, menos compleja que en [STE96], consiste en una valoración de 1 a 5 (donde 5 significa totalmente sincronizado) y permite emitir un juicio valorativo a los sujetos después de visualizar una secuencia de audio y vídeo. Los experimentos se realizaron en una estación de trabajo UNIX, con vídeo codificado según H.261. Se concluye que a partir de 5 tramas por segundo se obtienen buenos resultados en la evaluación subjetiva.

En [GUE97] se presenta una evaluación subjetiva del comportamiento del protocolo *Feedback Global*, basándose en los dos trabajos anteriores. La evaluación se realizó sobre una población de 20 personas y se comparó el resultado obtenido con la utilización del protocolo con respecto otros dos casos: el caso 'ideal' de una secuencia de audio y vídeo totalmente sincronizados y el caso de la misma secuencia en ausencia de sincronización entre el audio y el vídeo. También se valoró el efecto sobre la percepción de los individuos encuestados de la desviación de los relojes y de la tasa de reproducción, así como el efecto de la asincronía entre flujos dependiendo del que estuviera adelantado respecto al otro.

En los últimos años, las técnicas de compresión MPEG de imagen y sonido se están convirtiendo en un estándar de facto debido, principalmente, a su buen rendimiento en términos de calidad percibida, ratio de compresión y su adecuación para su uso en una gran variedad de aplicaciones. En [GIL95] se describe un conjunto de experimentos psicofísicos con el objetivo de comparar la calidad subjetiva de la imagen proporcionada por la codificación MPEG (MPEG1 y MPEG2) a diferentes velocidades, frente a la proporcionada por los sistemas tradicionales de difusión de televisión y Betacam-SP. El trabajo obtiene la conclusión de que la compresión MPEG es apropiada para transmisión de vídeo, que MPEG a 4 Mbps ofrece una calidad subjetiva similar a Betacam, mientras que MPEG a 1,5 Mbps es aceptable para aplicaciones y servicios en un entorno no profesional. Además de los resultados, el artículo explica con profundidad la metodología utilizada para llevar a cabo los experimentos.

Por otro lado, en la recomendación UIT-R BT. 500-7 se puede encontrar la metodología, recomendada por la UIT, a seguir para realizar la *evaluación subjetiva de la calidad de las imágenes de televisión.* Se utiliza una escala con dobles estímulos.

Watson, Sasse y Wilson, del UCL, en [WAT96], [WAT97], [WAT98] y [WIL00], indican que la percepción subjetiva de la calidad se ve influenciada por varios factores y varía de acuerdo a la tarea a realizar (no evalúa por igual un sujeto que simplemente escucha y mira pasivamente y, posteriormente, evalúa, que otro que está interactuando con la herramienta de evaluación como, por ejemplo, participando en una videoconferencia), de forma que determinar la calidad subjetiva para una aplicación dada no es una tarea trivial. Indica que la evaluación subjetiva debe ir encaminada a encontrar los niveles mínimos de calidad a partir de los cuales el usuario no encuentra una mejoría notable en su percepción. En [WAT96] se realiza una evaluación subjetiva de una aplicación de audio sobre 21 personas, variando la tasa de pérdidas y utilizando tres métodos de corrección (rellenar con silencios el contenido de los paquetes perdidos, repetir el último paquete recibido correctamente y LPC o *Linear Predictive Coding*) mediante escalas de valores discretos (calculando medias *MOS* o *Mean Opinión Scores* con una escala con 5 posibles valores: *Excellent*, *Good*, *Fair*, *Poor* y *Bad*). También se evalúa subjetivamente los resultados de una serie de cursos de enseñanza de idiomas a través del proyecto RELATE a nivel de calidad de audio y vídeo y a nivel de calidad del sistema de enseñanza y satisfacción, tanto de estudiantes como de profesores. En [WAT97] se argumenta que la evaluación mediante escalas unidimensionales de valores discretos recomendadas por la UIT (por ejemplo, calculando *MOS*) no son adecuadas debido al significado diferente que cada sujeto evaluado le asigna a cada valor (pueden ser de diferentes nacionalidades y asignar a estas 5 palabras diferentes significados según traducciones) y, también, debido a la variación inesperada que puede surgir en las condiciones de la red en aplicaciones multimedia sobre redes de conmutación de paquetes. Dichas escalas están pensadas para evaluar los servicios de telefonía y TV tradicionales. Estos servicios, unidos a los servicios multimedia actuales (videoconferencia, etc.) ofrecidos a través de redes de conmutación de paquetes tienen características diferentes, además de verse afectados por la congestión, pérdidas de paquetes (afectan de diferente forma dependiendo del contenido de los paquetes), etc. Por ejemplo, si evaluamos con técnicas MOS la calidad de una videoconferencia a través de Internet, es muy probable que nunca se otorgue la escala *Excellent* si se compara con la calidad del servicio de telefonía o TV tradicionales. En [WAT97] se presenta como solución la utilización de una escala de evaluación sin etiquetas. Se muestran los resultados de una evaluación de la calidad percibida e inteligibilidad de la voz en una aplicación de audio (*rat*) variando la tasa de pérdidas y utilizando dos métodos de corrección (repetición del último paquete recibido correctamente y LPC). Se utilizó una muestra de 24 individuos y se utilizaron ambos tipos de escalas comparando los resultados. En [WAT98], tras presentar un breve resumen de las medidas típicas de las recomendaciones de la UIT sobre evaluación subjetiva de vídeo y audio, concluye que las escalas subjetivas de clasificación recomendadas por la UIT no son adecuadas para las comunicaciones multimedia que requieren muchos de los nuevos servicios y aplicaciones sobre redes de conmutación de paquetes. [WAT98] presenta un perfil

de lo que sus autoras creen que sería una metodología más adecuada y que reconoce la naturaleza multidimensional de la calidad percibida del audio y del vídeo (que no se puede medir con escalas unidimensionales). A diferencia del trabajo anterior, en éste presentan una herramienta de evaluación basada en una barra deslizante denominada QUASS (*QUality ASsessment Slider,* [BOU98]), que el usuario controla mediante el ratón y del que se toman muestras cada segundo, para registrar la percepción del sujeto en cada instante y, así, evitar problemas de memoria del sujeto que normalmente retiene lo último que aprecia.

Sin embargo, Wilson, en [WIL00], indica que la evaluación continua mediante la herramienta QUASS tiene el inconveniente de interferir en la tarea a realizar por el evaluador. Introduce el factor coste del usuario (*user cost*) a tener en cuenta como factor determinante en las evaluaciones subjetivas de las aplicaciones. En su aproximación investigan el uso de métodos objetivos para evaluar el impacto de la calidad multimedia en el usuario. Para ello, monitorizan respuestas psicológicas que son indicativas de estrés e incomodidad y cuya monitorización no interfiere con la tarea que están realizando los sujetos evaluados. Presenta el proyecto ETNA (*Evaluation of Taxonomy for Networked Multimedia Applications*), que tiene como objetivo obtener la taxonomía de aplicaciones multimedia de tiempo real y sus requerimientos de calidad.

En [KAL99] se presentan los resultados de una evaluación subjetiva de la calidad de sincronización multimedia obtenida por el algoritmo VTR ([ISH95]). Para ello, se transmitieron dos flujos de voz y vídeo JPEG almacenados de forma separada en dos flujos de transporte independientes mediante conexiones TCP, desde una fuente a un único destinatario bajo diferentes condiciones de tráfico de red. Se eligen dos vídeos con diferencias en cuanto a movimientos y efectos visuales especiales, para ver el efecto sobre la percepción humana de los mismos: un vídeo de un primer plano de una persona hablando y un fragmento de la película *Top Gun* (con vuelo de aviones, diálogos, disparos y explosiones). En el destino se reprodujeron los dos ejecutándose el algoritmo de sincronización y se comprobó subjetivamente la calidad de la sincronización de una forma sistemática. Ya que la evaluación subjetiva es no-sistemática, la relación entre la evaluación objetiva y la percepción humana no están claras, el artículo también aclara las relaciones entre las medidas de funcionamiento analíticas y subjetivas. Para ello realiza una evaluación subjetiva sistemática con una muestra de 20 sujetos, de entre 21 y 30 años y sin tener un trabajo cotidiano directamente relacionado con la calidad de audio y vídeo. Mediante pruebas preliminares deduce que es suficiente con tests de 40 segundos de duración. En cada sesión, de aproximadamente 35 minutos de duración, se presentaba a un individuo una serie de muestras de tests, transmitidas bajo diferentes condiciones de red y elegidas de forma aleatoria. Se evalúan las políticas *Quick Recovery* y *Graceful Recovery* frente a una transmisión sin sincronización, así como la percepción de la sincronización de los individuos en los

dos tipos de vídeo. Lógicamente, se comprueba que el efecto de las asincronías se nota menos cuantos más movimientos y variabilidad tenga la escena.

En [ISH01b] y [ISH03], se evalúan tanto objetiva como subjetivamente, nueve combinaciones de técnicas de control de la sincronización reactivas. La evaluación subjetiva la realizaron 16 sujetos, de 22 a 24 años, no directamente relacionados con la calidad de audio y vídeo en su trabajo cotidiano y las sesiones tenían una duración de 40 minutos, en las que se les pasaban diferentes tests, de un mínimo de 25 segundos cada uno. Se utilizó el método del estímulo simple (*single stimulus method*) de la recomendación ITU-R BT.500-5.

Todos estos estudios muestran las dificultades de 'medir' la calidad percibida por el usuario y que, además de tener un fuerte carácter subjetivo, también se ve afectado por otros factores, como, por ejemplo, el coste económico y personal que le supone y la influencia de las tareas realizadas durante la evaluación. La apreciación de la calidad de una comunicación tiene una buena parte subjetiva pero también depende de la tarea a realizar y la relación calidad/precio de dicha comunicación y el grado de comodidad del usuario.

Para finalizar con este apartado, simplemente comentar que dado el fuerte carácter subjetivo de este campo, la mayoría de las investigaciones que se están llevando a cabo están siendo realizadas por psicólogos, aunque en el campo de las telecomunicaciones también existen trabajos relacionados con la percepción del usuario y la caracterización del rendimiento, buscando una caracterización más objetiva.

2.3.- Conclusiones

La Internet de nuestros días ha sufrido un profundo cambio desde su creación hasta momento actual y ha pasado a ofrecer a los usuarios posibilidades que en su día eran impensables incluso para sus creadores. Han surgido nuevos protocolos para dar soporte al tiempo real, a las videoconferencias, a la simulación distribuida y aún quedan otras muchas posibilidades por explotar, como la teleenseñanza y la televigilancia por las que tanto están apostando cada vez más y más empresas. Para garantizar las características de tiempo real de estas aplicaciones será necesario integrar algoritmos o mecanismos que garanticen el mantenimiento de las relaciones temporales en el momento de la reproducción, conocidos como *Algoritmos o Protocolos de Sincronización Multimedia*.

A lo largo de este capítulo se ha estudiado el estado del arte en la materia que constituye el ámbito de aplicación de esta Tesis, que es, precisamente, los *Algoritmos y Protocolos de Sincronización de Flujos Multimedia*, cuyo objetivo es garantizar las relaciones temporales entre los flujos de información multimedia,

manteniendo una determinada calidad de servicio en lo que a sincronización se refiere.

Como se ha visto, hay multitud de propuestas para resolver el problema de la sincronización multimedia, algunas de las cuales ya se han implementado tanto en los clientes como en los servidores de ciertas aplicaciones comerciales. Además, en los últimos años ha habido movimiento hacia la estandarización, con la aparición de *IP Multicast* y RTP/RTCP, que ha ayudado a la aparición de más y mejores aplicaciones.

En el campo de la sincronización de flujos multimedia podemos considerar tres tipos de sincronización: la *sincronización intra-flujo*, que garantiza las relaciones temporales entre LDUs de un mismo flujo, corrigiendo los retardos de red normalmente aleatorios; la *sincronización inter-flujo*, entre LDUs de distintos flujos; y *la sincronización de grupo* o entre participantes, que necesita una solución más compleja. Los dos primeros tipos han sido extensamente tratados en la literatura consultada. Además, el problema de la sincronización inter-flujo está solucionado con la utilización de RTP/RTCP. Sin embargo, *la sincronización de grupo ha sido poco tratada en la bibliografía existente y los autores que se han ocupado de la misma no han utilizado RTP/RTCP*.

Si nos centramos en el estudio de la sincronización en los sistemas multimedia distribuidos y considerando las principales ideas aportadas en las referencias analizadas, se puede concluir que, *mientras la red no garantice la calidad de servicio mediante la reserva de recursos, no se utilicen sistemas operativos de tiempo real o se produzcan desviaciones durante la presentación por parte de los reproductores, será necesaria la ejecución de algoritmos o protocolos de sincronización, que involucren bien a las fuentes transmisoras de información, bien a los receptores o bien a ambos*.

Hay que tener en cuenta que las acciones de dichos protocolos dependerán, en gran medida, de los servicios ofrecidos por la red y de las características de los reproductores. Ante situaciones de carga o congestión, la única solución será la de degradar la calidad de la presentación, intentando que se mantenga un mínimo de calidad. Respecto a la desviación de las tasas de consumo y efecto del *jitter*, la solución mayoritariamente utilizada por los investigadores es la de ejecutar ciertas acciones de resincronización (bien en los receptores o bien en las fuentes) como, por ejemplo, la realización de '*saltos*' y/o '*pausas*' en la reproducción, acciones que han sido también incluidas en la propuesta realizada en este trabajo de investigación.

En este capítulo también se han analizado las escasas aportaciones encontradas sobre la *evaluación subjetiva* de algunos algoritmos o protocolos de sincronización, prueba, a nuestro entender, muy importante para demostrar la

apreciación de los usuarios del buen funcionamiento del algoritmo de sincronización propuesto en las aplicaciones donde se implemente (que, al fin y al cabo, es lo que marcará el éxito o no del mismo).

A partir de la bibliografía existente, y para terminar el capítulo de estado del arte, *podemos concluir* que existen dos aproximaciones que se pueden seguir para acomodar los requerimientos de tiempo real de la distribución *multicast* de flujos multimedia por Internet. Una es utilizando *Soporte de Calidad de Servicio* dentro de la red para permitir a las aplicaciones de distribución de información multimedia ser capaces de reservar recursos para establecer límites en el retardo de entrega de datos, el *jitter* de dicho retardo y las pérdidas. La otra consiste en utilizar *Técnicas o Algoritmos de Control Adaptativos* para ajustar las características del tráfico multimedia de acuerdo con los recursos disponibles en la red. Hay que destacar que, en la Internet actual, no se dispone de realimentación de la red sino de realimentación que generan los sistemas finales.

Como ya se ha comentado en puntos anteriores, *en la realización de la Tesis, nos hemos centrado en las técnicas de sincronización de grupo e inter-flujo,* dentro de la segunda aproximación, puesto que *dichas técnicas son compatibles con la Internet actual* y, además, *aún cuando esté disponible la reserva de recursos y las garantías de calidad de servicio, también se necesitará de unos mecanismos de control adaptativos, en las aplicaciones, para tener en cuenta una cierta tolerancia a cualquier error de asignación y reserva de recursos debido a dificultades inherentes en la realización de la estimación de tráfico en la red correctamente, así como de los propios procesos de reproducción*.

Capítulo 3

Sincronización, Protocolos y Aplicaciones utilizadas en la Tesis

3.1. Introducción

Tal como se ha comentado en el capítulo de introducción, la idea básica y fundamental aportada en la Tesis se centra en la descripción, estudio y evaluación de un *algoritmo de sincronización de grupo de flujos multimedia* que hace uso simultáneamente de la existencia de un tiempo global (proporcionado por el protocolo NTP), así como de marcas de tiempo en las LDUs y de un mecanismo de realimentación (proporcionados por los protocolos RTP/RTCP), para ejecutar las acciones de sincronización.

En el presente capítulo, en primer lugar se presenta la definición de los *Sistemas Multimedia*, así como sus características y requerimientos, en cuanto a sincronización se refiere. Además, se presentan algunos conceptos básicos de sincronización multimedia que serán utilizados a lo largo de la Tesis.

En segundo lugar, se presentan de forma genérica, sin entrar en mucho detalle, los protocolos, mecanismos de comunicación local entre procesos y aplicaciones utilizados en la Tesis, para facilitar una mejor comprensión de los capítulos posteriores.

3.2. Sincronización Multimedia

3.2.1. Sistemas Multimedia

Los *Sistemas Multimedia* se caracterizan por la integración controlada por ordenadores de la generación, almacenamiento, comunicación, manipulación y presentación de diferentes objetos multimedia. En la presente Tesis, se hace uso del concepto *'multimedia'* para hacer referencia al proceso de integración, en el instante de la reproducción (punto de reproducción o *playout point*), de diferentes tipos de objetos multimedia como, por ejemplo, audio, vídeo, texto, imágenes estáticas, etc.

En principio, se puede distinguir entre *objetos dependientes* e *independientes del tiempo*. Un objeto dependiente del tiempo está representado por un flujo de información y suele consistir en una secuencia de unidades de datos o información denominadas LDUs (*Logic Data Units*), en la que existe una relación temporal entre unidades consecutivas. Si la duración de todas las unidades es la misma, se denominan *objetos continuos*. Como ejemplos típicos de este tipo de objetos se pueden citar las secuencias de vídeo y de audio. Por otra parte, se puede considerar como objetos de información independientes del tiempo, por ejemplo, la información tradicional en forma de texto e imágenes estáticas, donde no existe ninguna relación entre la representación de la misma y el tiempo.

En los artículos referenciados en el capítulo anterior se pueden encontrar diferentes definiciones para los términos de aplicación y sistema multimedia. No obstante, se pueden distinguir tres criterios para clasificar un sistema como multimedia como son el *Número* de tipos de objetos o información, el *Tipo* de objetos soportados y, por último, el *Grado de integración* de los tipos de información. De esta forma, haciendo uso únicamente del primer criterio, que es el más simple, incluso un procesador de documentos que soporte tanto texto como gráficos puede ser considerado como un sistema multimedia. En cuanto a los *tipos de objetos soportados* por la aplicación, en este caso, se puede distinguir, tal como se ha comentado, entre objetos *dependientes e independientes del tiempo*. En cuanto al tercer criterio (*grado de integración*), dicha integración implica que los diferentes tipos de objetos, aunque permanezcan independientes, puedan ser procesados y presentados conjuntamente. Combinando los tres criterios anteriores, Blakowski y Steinmetz, en [BLA96], proponen como *Sistema Multimedia* a *'aquel*

sistema o aplicación que soporta el procesamiento integrado de varios tipos de objetos o información siendo al menos uno de ellos dependiente del tiempo'. La figura 3.1 muestra diferentes aplicaciones en función de los tres criterios enumerados anteriormente.

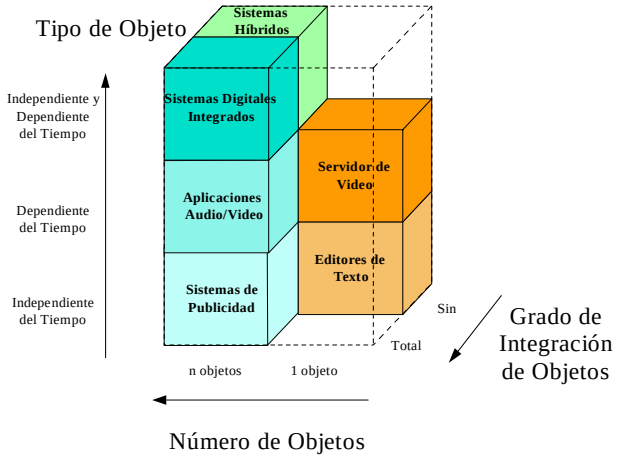


Figura 3.1. Clasificación de los sistemas multimedia ([BLA96])

3.2.2. Sincronización en los Sistemas Multimedia

El *procesamiento integrado de objetos* es una de las principales características de los sistemas multimedia. La principal razón para requerir dicha integración se encuentra en las dependencias inherentes que suelen existir entre la información que contiene cada uno de los objetos que participan en la aplicación. Estas dependencias deberán mantenerse y reflejarse en todo el procesamiento integrado de la información, incluyendo el almacenamiento, manipulación, comunicación y captura, especialmente, durante la presentación.

La palabra *sincronización* hace referencia al concepto de tiempo, sin embargo, en la literatura estudiada, varios autores utilizan el concepto de sincronización en sistemas multimedia en un sentido más amplio, haciendo referencia a 3 tipos de relaciones: las *relaciones del contenido*, las *relaciones espaciales* y las *relaciones temporales*.

Las *relaciones de contenido* definen una dependencia de determinados objetos sobre ciertos tipos de datos. Un ejemplo de este tipo de relación sería la que existiría entre una tabla de datos estadísticos y un gráfico que sea una representación de dichos datos. Las *relaciones espaciales* definen el espacio

utilizado por la presentación de un objeto en un dispositivo de salida, en un determinado instante de la presentación multimedia. Por ejemplo, si el dispositivo de salida es bidimensional, como es el caso de la pantalla o monitor de un ordenador, se especificará el área asignada a dicho objeto. Por último, las *relaciones temporales* definen las dependencias entre objetos de información en el dominio del tiempo. Serán de gran interés cuando existan objetos dependientes del tiempo en la aplicación. El ejemplo más claro, es la relación temporal que existe entre una secuencia de vídeo y otra de audio que son grabadas simultáneamente. En el momento de la representación de la información grabada, la relación temporal de los dos objetos debe corresponder con la relación temporal que existía en el momento de la captura o grabación. Evidentemente, en este ejemplo, las relaciones temporales ya existen en el momento de la captura y reciben el nombre de *relaciones naturales*. Sin embargo, las relaciones temporales pueden crearse en el momento de la edición de una presentación multimedia (por ejemplo, en la creación de una aplicación formada por imágenes y texto explicativo de las mismas), recibiendo la denominación de *relaciones sintéticas*.

Debido a la dependencia temporal entre los diferentes objetos, resultará necesaria una coordinación y ordenación en el tiempo de los diferentes flujos de información. Dicho proceso de mantener las relaciones temporales y garantizar una presentación ordenada en el tiempo entre las LDUs de diferentes objetos o flujos multimedia es denominado proceso de *Sincronización Temporal Multimedia*. En adelante, se utilizará el término '*sincronización*' para referirse a dicho concepto. Por otro lado, se denominará *Algoritmos o Protocolos de Sincronización Multimedia* a aquellos algoritmos o protocolos que incorporen la funcionalidad de mantener las relaciones temporales y garantizar una presentación ordenada en el tiempo de los diferentes flujos multimedia.

Los objetos multimedia que deben ser sincronizados pueden ser de diferente naturaleza como, por ejemplo, voz, vídeo, imágenes, texto, etc. y, en todos los casos, se deben mantener las relaciones temporales.

En la figura 3.2 se muestra un ejemplo de una presentación multimedia con tres flujos de información diferentes (audio, vídeo y datos).

Centrándonos en la sincronización temporal, se puede distinguir entre *sincronización intra-objetos* (o *intra-flujo*), *sincronización inter-objetos* (o *inter-flujo*) y *sincronización de grupo* (o *inter-destinatario*). La distinción entre los tres tipos de sincronización permitirá, además, clasificar o diferenciar los mecanismos encargados de soportar dichos tipos de relaciones que, en general, son diferentes.

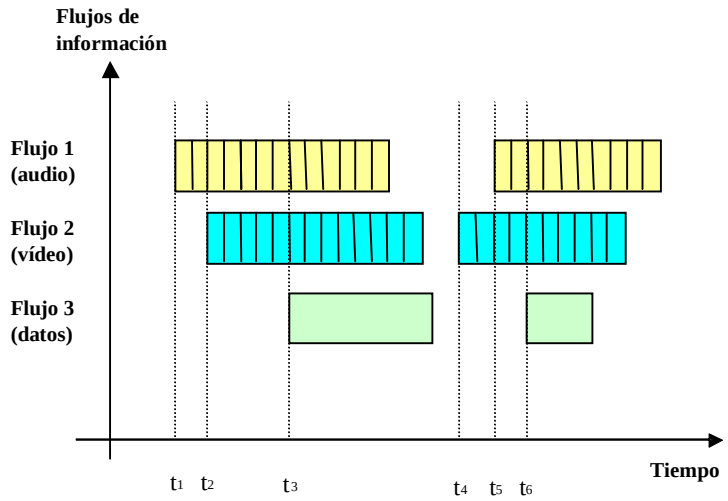


Figura 3.2. Relaciones temporales

3.2.2.1. Sincronización Intra-flujo

La *sincronización intra-objetos* o *intra-flujo*, se refiere a la relación temporal entre varias unidades de datos o LDUs de un mismo flujo de información dependiente del tiempo. Como ejemplo, se puede citar la relación temporal entre las tramas de una secuencia de vídeo. Si suponemos que la tasa de grabación del vídeo es de 25 tramas por segundo, cada una de las tramas deberá representarse durante 40 milisegundo en el monitor o dispositivo de visualización. En la figura 3.3 se representa dicho requerimiento de sincronización entre las tramas de una secuencia de vídeo mostrando una pelota saltando. Las LDUs, una vez han atravesado la red, serán almacenadas en un *buffer* de recepción, de tal manera que para garantizar la sincronización intra-objetos será necesario garantizar la existencia de LDUs en el *buffer* en los instantes de reproducción (evitando que se vacíe y/o se llene, es decir, evitando situaciones de *underflow* u *overflow* del *buffer*, respectivamente) y, además, el proceso de reproducción deberá ser capaz de consumir las LDUs con la tasa adecuada.

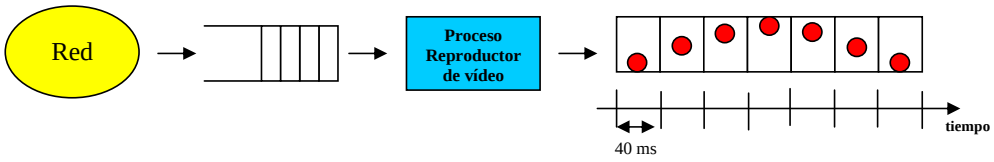


Figura 3.3. Sincronización intra-flujo

3.2.2.2. Sincronización Inter-flujo

Por otro lado, la *sincronización inter-objetos* o *inter-flujo* hace referencia a la sincronización, en el momento de la presentación, de diferentes objetos o flujos de información (tanto dependientes como independientes del tiempo). La figura 3.4 representa un ejemplo de las relaciones temporales en una aplicación multimedia que empieza con una secuencia de audio y vídeo, seguida de varias imágenes estáticas (diapositivas) y una animación comentada por una secuencia de audio. En este caso, también será necesario, para garantizar la sincronización inter-flujo, en primer lugar, garantizar la sincronización intra-flujo para cada uno de los flujos involucrados y, en segundo lugar, corregir posibles errores de sincronización entre los mismos.

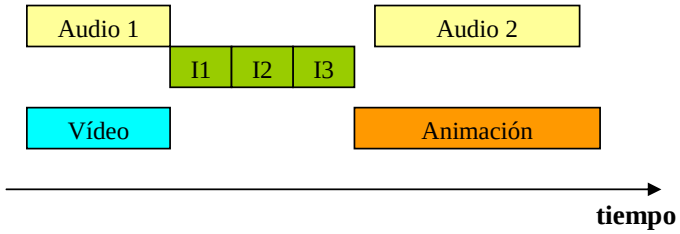


Figura 3.4. Sincronización inter-flujo

A continuación se presentan una serie de ejemplos típicos de sincronización:

- Sincronización labial o *lip-synchronization* (*lip-sync*). Como se ha mencionado en el capítulo anterior, la sincronización entre el audio y el movimiento de los labios que existe en un discurso se conoce como *sincronización lip* (figura 3.5) y requiere una sincronización muy exacta entre los flujos de audio y vídeo. La sincronización puede especificarse definiendo una *desviación* o *asincronía* máxima permitida entre los dos flujos, de forma que, dentro de dicho rango, la falta de sincronización no sea percibida por el usuario.

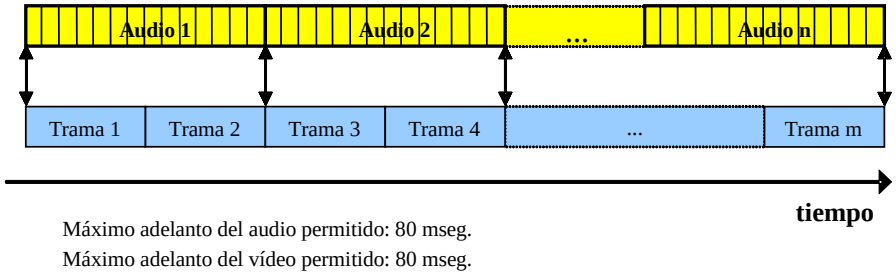


Figura 3.5. Sincronización labial o *lip-sync*

- Sincronización de la presentación de imágenes fijas (transparencias o diapositivas) con una secuencia de audio como comentario a dichas imágenes. En este caso, el cambio de imagen deberá coincidir temporalmente con determinados instantes de la secuencia de audio. A diferencia del ejemplo anterior, los requerimientos de sincronización suelen ser menores, tal y como se indica en la figura 3.6.

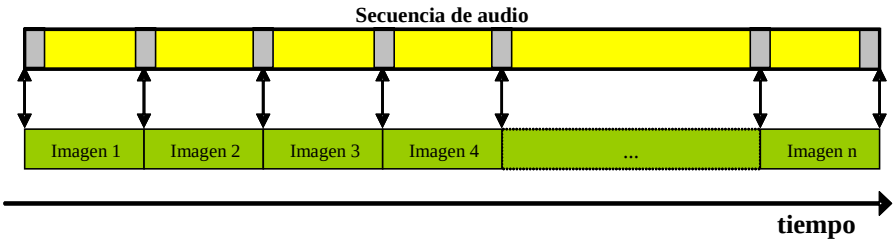


Figura 3.6. Sincronización entre imágenes y audio

- Un ejemplo completo sería el mostrado en la figura 3.7, en el que secuencias sincronizadas de audio y vídeo (*lip-sync*), son acompañadas de imágenes estáticas, animaciones, incluso acciones del propio usuario (interacción) que también deben reproducirse de forma sincronizada. Como ejemplo de interacción del usuario podría ser una pregunta de selección entre diferentes opciones y, dependiendo de que el usuario realice la elección apropiada, se podría mostrar o no la imagen 4.

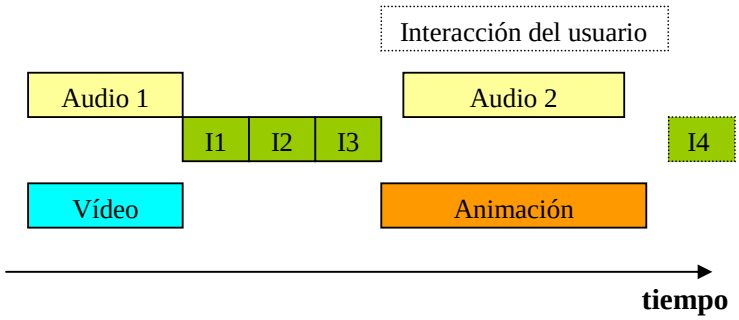


Figura 3.7. Sincronización en una presentación multimedia

3.2.2.3. Sincronización de Grupo

Además de los dos tipos de sincronización anteriores (intra e inter-flujo), en las comunicaciones *multicast* aparece otro tipo de sincronización, la *sincronización de grupo* de flujos en diferentes receptores al mismo tiempo. Por ejemplo, en aplicaciones de aprendizaje a distancia, un profesor podría enviar un vídeo de una película (flujo de contenido almacenado) y, al mismo tiempo, en

algunas ocasiones, podría hacer comentarios sobre la película (flujo de contenido en directo). En este caso, es importante conseguir una reproducción simultánea de los flujos, tanto de contenidos almacenados como en directo, para todos los estudiantes. Incluso aunque sólo se enviara la película, es decir, el flujo de contenidos almacenados, cada LDU (por ejemplo, una trama de vídeo) debería ser reproducida simultáneamente en los diferentes destinos (estudiantes) y los estudiantes pueden discutir el contenido de la misma. A este tipo de sincronización se la denomina *Sincronización de Grupo o Sincronización Inter-destinatario* (*Group or Inter-destination Synchronization*).

Este proceso se encargará de que cuando un mismo flujo sea enviado desde una fuente a varios receptores de forma *multicast*, éstos reproduzcan las LDUs de dicho flujo en los mismos instantes, es decir, de forma sincronizada. Será importante el *instante inicial de consumo* o de reproducción, que deberá ser el mismo en todos los receptores, cada vez que la fuente transmisora inicie un nuevo proceso de envío de paquetes.

Una vez que cada uno de los receptores haya iniciado su proceso reproductor sincronizado con el resto, las relaciones temporales entre las LDUs de un mismo flujo reproducidas se mantendrán debido al proceso de sincronización intra-flujo para dicho flujo de datos. No obstante, debido a la diferencia entre las tasas de consumo de los diferentes procesos reproductores de los diferentes flujos en un mismo sistema local, serán necesarios mecanismos de resincronización durante la reproducción para que todos los receptores reproduzcan las mismas LDUs en los mismos instantes de tiempo (sincronización de grupo). En la figura 3.8 se muestra la reproducción de la secuencia de vídeo de la pelota saltando, sincronizada en todos los receptores (se supone un eje de tiempos común en todos los receptores).

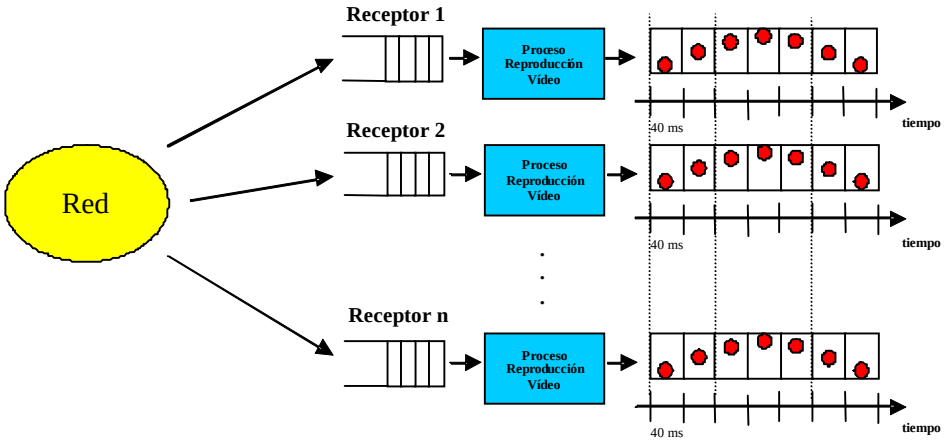


Figura 3.8. Sincronización de grupo

Una vez definido un sistema multimedia y dejada patente la necesidad de un proceso de sincronización, en el siguiente punto se describen las principales causas generadoras de asincronía y sus efectos.

3.2.3. Requerimientos de Sincronización

El problema de mantener la sincronización entre diferentes flujos de información es más complejo en un entrono distribuido que en un entorno local o centralizado. En un entorno local se exige el desarrollo cuidadoso de un conjunto de procesos puesto que la tarea de sincronización se encuentra relacionada y soportada por un elevado número de componentes, como son el sistema operativo, las comunicaciones entre procesos y las propias aplicaciones. En los entornos distribuidos, además de estos aspectos, aparecen otros parámetros que aumentan la problemática asociada a la sincronización. Esto es debido principalmente al carácter descentralizado tanto de la captura de la información como del almacenamiento de la misma y la localización distribuida de los objetos multimedia involucrados en el sistema.

En entonos distribuidos, los retardos de red pueden abarcar desde un milisegundo en una red local (LAN) hasta cientos de milisegundos en redes de área amplia (WAN). Debido a consideraciones de calidad de servicio, los nuevos algoritmos de encaminamiento pueden incluso proporcionar diferentes rutas para diferentes flujos. Incluso dentro de una misma ruta, se pueden utilizar distintas colas o filosofías de colas para diferentes tipos de flujos y, además, en una misma ruta y con colas idénticas, se pueden observar variaciones del retardo que pueden ir desde decenas de milisegundos hasta llegar a superar los cien milisegundos. Por tanto, la red puede destrozar cualquier tipo de relación temporal que exista entre los flujos multimedia que por ella circulan, incluso compartiendo las mismas rutas y colas.

Por tanto, las redes de comunicación a las que se conectan las fuentes y los receptores de información (tanto si ésta está almacenada como si se captura en directo) introducen retardos adicionales que provocan asincronías. Algunas de las fuentes más importantes que generan dicho efecto y que, por tanto, deberán ser corregidas en la medida de lo posible por los protocolos de sincronización multimedia son el *jitter* que introduce la red y la *diferencia en las tasas de consumo de los elementos reproductores*.

3.2.3.1. Latencia y *Jitter*

En la mayoría de las aplicaciones multimedia, los parámetros más importantes a tener en cuenta son, posiblemente, la *latencia* y el *jitter* que se considerarán como una medida objetiva válida de la calidad de servicio (QoS) ofrecida. La *latencia* se define como el tiempo transcurrido desde la generación de la señal (o el momento de su recuperación de un sistema de almacenamiento) hasta su instante de reproducción. El *jitter* es la variación del retardo respecto a su valor nominal entre LDUs consecutivas. Por otra parte, el *tamaño del buffer* requerido en la red, y sobre todo en los clientes, también se puede considerar como un parámetro de considerable importancia.

Según el tipo de aplicación distribuida, la importancia de dichos parámetros varía en función de los requerimientos de la aplicación, del tipo de red y de los objetos de información involucrados en la misma. Por ejemplo, en las aplicaciones de tiempo real como la videoconferencia, se requieren pequeñas diferencias entre los instantes de captura, codificación, almacenamiento, transmisión, recepción y el instante de reproducción, siendo normalmente dicho intervalo de tiempo inferior a decenas de milisegundos. Por otra parte, la cantidad de tramas almacenadas en los *buffers* debe ser pequeña para minimizar dicho retardo. Sin embargo, en las aplicaciones basadas en la recuperación de información como el vídeo bajo demanda, puesto que el principal requisito es reproducir la información a la misma tasa que la tasa original de captura, el tiempo entre el instante de petición de la información y el instante de reproducción puede ser superior a varios segundos.

3.2.3.2. Desviación de la Tasa de Consumo de los Procesos Reproductores

En algunos sistemas distribuidos es importante procurar obtener una sincronización precisa entre los relojes de los transmisores y de los receptores. Muchos esquemas de sincronización demandan un conocimiento entre las relaciones temporales. Este conocimiento es la base para los esquemas de sincronización basados en un tiempo global, así como para los esquemas en los que los nodos distribuidos necesiten realizar acciones coordinadas temporalmente para asegurar, por un lado, la entrega a tiempo de las LDUs de cada flujo y, por otro lado, que las operaciones no sean realizadas demasiado rápidas como para producir problemas de *overflow* en los *buffers* de recepción. Este problema será especialmente importante en el caso de múltiples fuentes (figura 3.9).

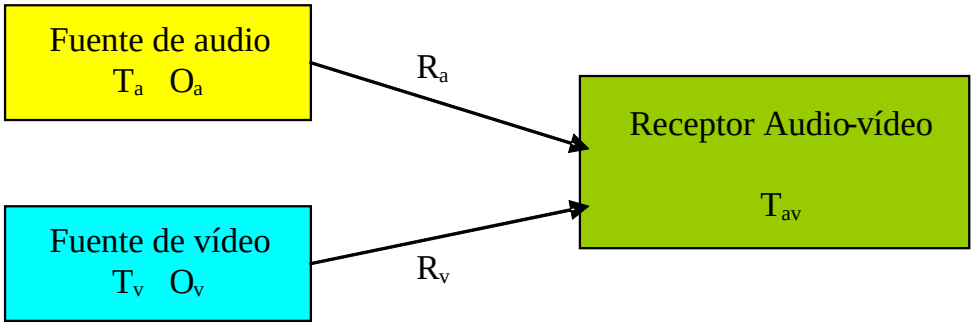


Figura 3.9. *Offsets* de reloj en un entorno distribuido

Supongamos que una presentación de audio y vídeo debe representarse en un reproductor en el instante T_{av}. Para que ello se realice de este modo, la transmisión del flujo de audio de la fuente de audio deberá empezar en el instante

$$T_a = T_{av} – R_a - O_a \qquad \text{[Ec. 3.1]}$$

, donde R_a es el retardo de red entre la fuente de audio y el receptor y O_a es el *offset* que existe entre el reloj de la fuente de audio y el reloj del receptor.

Del mismo modo, el transmisor del flujo de vídeo deberá empezar a transmitir en el instante

$$T_v = T_{av} – R_v – O_v \qquad \text{[Ec. 3.2]}$$

, donde R_v es el retardo de red entre la fuente de vídeo y el receptor y O_v es el *offset* que existe entre el reloj de la fuente de vídeo y el reloj del receptor.

Los valores de los *offsets*, O_a y O_v, no son conocidos a priori, pero el problema de entrega a tiempo de las correspondientes LDUs al receptor podría ser solucionado si se conocieran los valores máximos de dichos *offsets*. Sería posible asignar suficiente capacidad a los *buffers* de recepción e iniciar la transmisión de audio y vídeo lo suficientemente antes para garantizar que las LDUs estén disponibles en el receptor cuando se necesiten para su reproducción. Según esto, la necesidad de capacidad en los *buffers* de recepción depende del *offset* posible entre los relojes de la fuente y del receptor, sin embargo, dicha capacidad es limitada, por lo que será necesario limitar el *offset* de alguna manera.

Se podría pensar en hacer uso de las soluciones hardware clásicas propuestas en la bibliografía dedicada a los protocolos de sincronización de relojes que hacen uso de un hardware especial en cada uno de los nodos para alcanzar una sincronización fina entre los relojes hardware, mediante el ajuste de la frecuencia y la fase de los mismos. Estas soluciones hardware requieren, en general, el uso de

una red de relojes entre los nodos del sistema distribuido separada de la red utilizada para el intercambio de información.

En los algoritmos hardware, el circuito encargado de la sincronización monitoriza la frecuencia y la fase de todos lo relojes y los actualiza de forma continua, por lo que la sincronización se alcanza de forma rápida y permanente, pero a costa de un hardware adicional, lo cual no lo hace recomendable en grandes sistemas distribuidos o en sistemas donde no se requieran sincronizaciones de relojes extremadamente exactas, como es el caso de los sistemas multimedia distribuidos. De hecho, la mayoría de las soluciones propuestas en la bibliografía dedicada a la sincronización multimedia, descartan la posibilidad de disponer de un reloj hardware común y, en definitiva, de una señal hardware de referencia que sincronice los dispositivos reproductores. Esta filosofía, es utilizada ampliamente en las redes telefónicas, en redes privadas caracterizadas por un elevado grado de control y homogeneidad y, actualmente, por las redes de transporte síncronas (JDS, SONET, etc.). Sin embargo, no es factible en los entornos típicos de las aplicaciones multimedia, básicamente por las siguientes razones: las aplicaciones se ejecutarán sobre redes LAN o WAN donde las comunicaciones son asíncronas; los protocolos y señales utilizadas no disponen de las características necesarias para extraer una señal de reloj global; las aplicaciones y los protocolos de sincronización funcionan en un nivel alto de la pila de protocolos haciendo difícil el acceso al hardware dedicado a la reproducción; existencia de un entorno heterogéneo formado por estaciones de trabajo, PCs, dispositivos multimedia, etc.; implementaciones diferentes e incompatibles entre los procesos reproductores; necesidad de sencillez en las aplicaciones y protocolos de sincronización; y también, fundamentalmente, porque, en general, las bases de tiempo de los reproductores no pueden estar sincronizadas dada su diferente implementación o ubicación física.

Si se descarta la solución hardware por las razones enumeradas anteriormente, los protocolos de sincronización se caracterizan por ser protocolos software, con la ventaja de ser una opción más flexible y económica. Sin embargo, este tipo de solución requiere el intercambio de mensajes entre los diferentes nodos (según [KOP87] el tráfico y la carga de la CPU dedicada a la sincronización debe ser menor de un 1 % del tráfico y carga generada por el resto de aplicaciones de datos para no interferir en el funcionamiento normal).

En cuanto a los protocolos de sincronización multimedia software encontrados en la bibliografía consultada, se pueden distinguir básicamente dos grupos: por un lado, aquellos que hacen uso de un *tiempo global* que en definitiva se traduce en la existencia de una variable común a todos los nodos y sobre la cual se pueden realizar acciones de lectura y escritura; y, por otro lado, aquellos protocolos que, con ausencia de un tiempo global, sincronizan los receptores haciendo uso de la información de realimentación (*feedback*) enviada por éstos.

Relacionado con la sincronización de relojes, otra fuente importante que genera asincronía entre los flujos de información es la *desviación de las tasas o periodos de consumo de los procesos reproductores*. Dicho problema surge cuando los dispositivos reproductores trabajan a la misma frecuencia pero sufren desviaciones respecto a su frecuencia nominal. Evidentemente, en la existencia de dicha desviación influyen un gran número de factores, entre los cuales podemos citar la falta de características de tiempo real de la mayoría de los sistemas operativos, la carga de la estación de trabajo, una planificación de tareas no apropiada para procesos multimedia, etc. Sin embargo, se considera que el principal factor que contribuye a dicho efecto es la desviación de la *base de tiempo o relojes hardware asociados al proceso de reproducción*. El análisis de la influencia que tiene dicha sobre el consumo de las tramas de vídeo y audio, es muy similar y está estrechamente relacionado con la definición de algunos conceptos propios de la sincronización de relojes.

La figura 3.10 ilustra la influencia que tiene la *desviación de la tasa de reproducción* debido al *drift* del reloj, sobre el consumo de las tramas de vídeo y audio. Un reloj 'perfecto', y por lo tanto un proceso de reproducción 'perfecto', es aquel en el que una unidad de tiempo de reloj dura exactamente una unidad del tiempo real. Si durante una unidad de tiempo real, el reloj local ha avanzado más o menos de una unidad de tiempo de reloj, se dice que el reloj y, por lo tanto, el proceso reproductor funciona 'rápido' o 'lento', respectivamente.

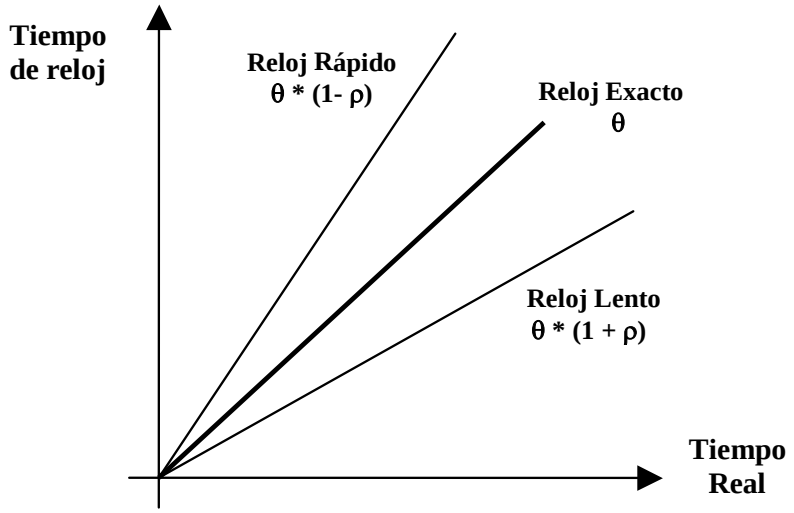


Figura 3.10. Desviación de los relojes

El funcionamiento descrito en la figura anterior se traduce en una variación de la tasa de consumo de las LDUs de los flujos involucrados en las aplicaciones multimedia respecto a la tasa nominal. Así, si llamamos θ a la tasa nominal de

consumo, y ρ a la desviación sufrida por el reloj que gestiona la reproducción, en el peor caso, la tasa de consumo real θ_{real} del dispositivo vendrá especificada por la expresión:

$$\theta_{real} = \theta \cdot (1 \pm \rho) \qquad \text{[Ec. 3.3]}$$

El parámetro ρ suele tener un valor constante y recibe el nombre de *drift* o *desviación* del reloj y, en general, la desviación real ρ_{real} sufrida por los dispositivos reproductores se distribuirá de forma uniforme entre [-ρ,+ρ], sufriendo variaciones, tanto positivas como negativas, durante el proceso de reproducción.

Debido a que la desviación sufrida por los dispositivos reproductores (ρ) no es nula, un conjunto de flujos de información regidos por el funcionamiento de dichos procesos reproductores no pueden mantenerse sincronizados sin un proceso de resincronización constante. De hecho, sin un mecanismo de resincronización, pasado un intervalo de tiempo, la asincronía entre los dispositivos puede crecer gradualmente provocando niveles importantes de falta de sincronización, llegándose a superar incluso el valor máximo permitido. A partir de las desviaciones normales que sufren los dispositivos reproductores (debido a la propia implementación de los relojes, cambios de temperatura, etc.) se puede comprobar fácilmente que surgirán situaciones perjudiciales y perceptibles de asincronía. Como es lógico, se producirá una falta de sincronización entre varios receptores si sus tasas de consumo no coinciden y, por lo tanto, también se producirá entre los diferentes flujos de información que ellos reproducen. Además, si la tasa de reproducción del receptor es mayor que la tasa de transmisión del servidor, el *buffer* del receptor puede sufrir una situación de *underflow*, y, por otro lado, si la tasa de reproducción del receptor es menor, puede producirse *overflow*. Las tres situaciones descritas provocarán una interrupción o degradación de la presentación debido a una falta de sincronización y, si no se ejecuta alguna acción correctora, el intervalo de tiempo que puede transcurrir hasta alcanzar el valor máximo de asincronía dependerá del valor de ρ. Por otra parte, la máxima asincronía permitida entre flujos de información dependerá de la criticidad de la aplicación. Por ejemplo, en el caso concreto de la sincronización labial (*lip-sync*), el valor máximo de asincronía viene determinado por la percepción humana y, según [STE96], será de ±80 milisegundos entre los flujos de audio y vídeo.

3.2.3.3. Efectos sobre la Sincronización

Una vez comentadas las principales fuentes de asincronía, a continuación se detallan los principales efectos que éstas producen en cuanto a la sincronización de flujos de información se refiere.

- *Instante de inicio de la reproducción.* Dentro de este aspecto tenemos que distinguir entre el instante de inicio de la reproducción de los flujos (audio, vídeo, etc.) localmente (figura 3.11) y el inicio de la reproducción distribuida y simultánea en todos los receptores de la sesión (figura 3.12). Cuando los receptores forman un grupo que debe reproducir las LDUs de forma sincronizada, si el instante de inicio de la reproducción es diferente en cada uno de los reproductores se produce una pérdida de sincronización inicial. Este problema se agudiza en el caso de grupos *multicast* donde el instante inicial de consumo debe ser el mismo para todos los receptores.

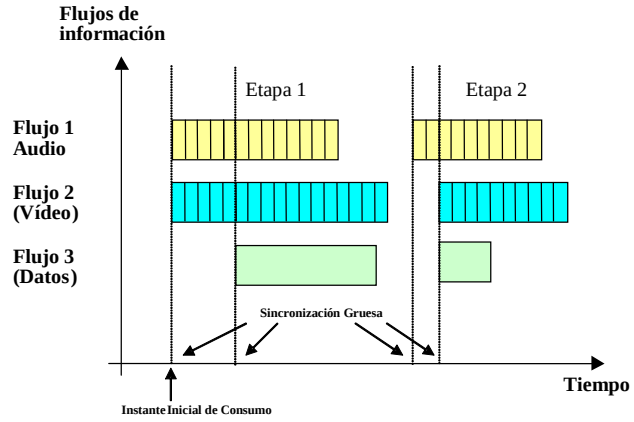


Figura 3.11. Sincronización local del instante inicial y sincronización gruesa

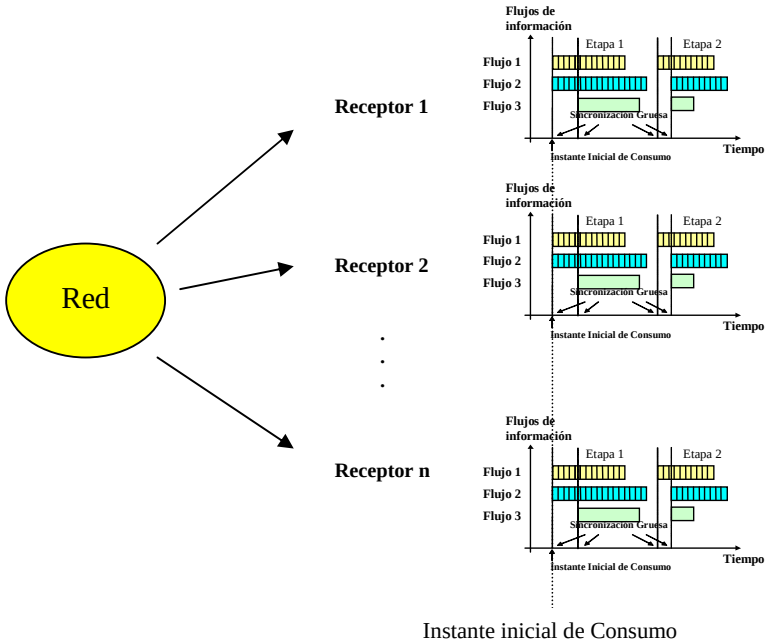


Figura 3.12. Sincronización de grupo. Instante inicial

- *Sincronización Gruesa.* Los puntos de sincronización grues en las presentaciones multimedia se caracterizan generalmente por no encontrase distribuidos periódicamente, no ser muy numerosos y no requerir una sincronización muy exacta, soportando una cierta flexibilidad en la misma. Esta sincronización se refiere a la reproducción de forma sincronizada de flujos de información independientes del tiempo con cualquier tipo de información (por ejemplo, la sincronización en la reproducción de una imagen fija con un vídeo). Por otra lado, la sincronización gruesa también hace referencia a los instantes de inicio de reproducción de las diferentes etapas que forman una presentación (ver figura 3.11). En la figura 3.11, cada etapa se corresponde con la transmisión de tres flujos (audio, vídeo y datos) y se finaliza cuando se deja de transmitir flujos durante un cierto tiempo.

- *Sincronización Fina.* Se suele especificar como sincronización fina a la que existe entre flujos de información continuos (figura 3.13). En general, se caracteriza por una elevada precisión y pequeños márgenes permisibles de asincronía. La principal causa de la asincronía entre los flujos de información se debe a la diferencia en la velocidad de consumo de los reproductores. El ejemplo de sincronización fina más conocido es la sincronización labial o *lip-sync*.

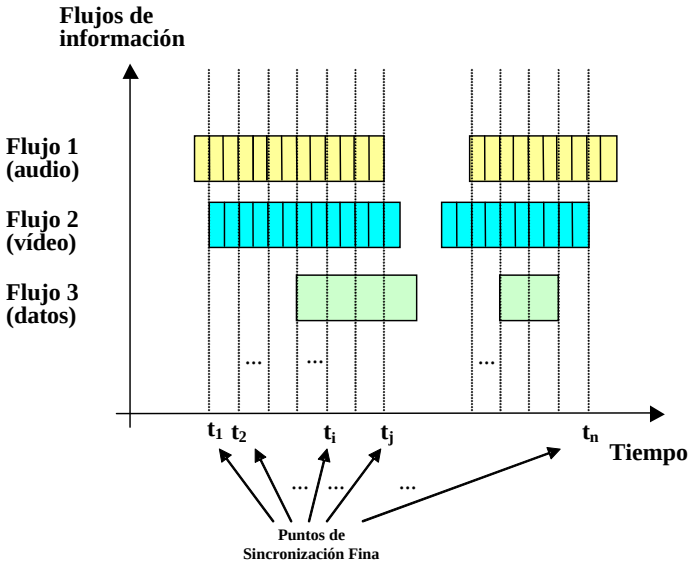


Figura 3.13. Sincronización fina

- *Overflow y/o underflow en los buffers de los receptores*. Debido, sobretodo, al retardo, al *jitter*, a las condiciones cambiantes de la red y a la diferencia de tasas de transmisión y consumo, es posible que, en determinados instantes, los *buffers* dedicados a almacenar las LDUs antes de ser reproducidas, se vacíen (situación de *underflow*) o bien sea vea superada su capacidad (situación de *overflow*), faltando o perdiéndose LDUs, respectivamente. En ambos casos, se produce el efecto inmediato de la pérdida de sincronización entre los flujos de información (falta de *sincronización inter-flujo*), así como la degradación de la presentación (pérdida de *sincronización intra-flujo*).

Por tanto, la totalidad de los aspectos señalados deberán ser corregidos por los protocolos de sincronización multimedia para conseguir minimizar, todo lo posible, la asincronía entre los diferentes flujos de información.

3.2.3.4. Calidad de Servicio de Sincronización

En la siguiente tabla se muestran los valores de asincronía máxima permitida entre dos tipos de flujos, la mayoría de ellos, obtenidos mediante numerosos experimentos y experiencias, referenciados en [STE96] y [BLA96] y utilizados en el resto de la Tesis como referencia.

Flujo		**Modo de Aplicación**	**QoS**
Vídeo	**Animación**	Correlada	±120 ms
	Audio	*Lip-sync*	±80 ms
	Imágenes	Solape	±240 ms
		Sin solape	±500 ms
	Texto	Solape	+/- 240 ms
		Sin solape	± 500 ms
Audio	**Animación**	Correlación de eventos (p.ej. baile)	± 80 ms
	Audio	Muy acoplado (estéreo)	± 11 µseg.
		Poco acoplado (modo diálogo con varios participantes)	± 120 ms
		Poco acoplado (p.ej. música de fondo)	± 500 ms
	Imágenes	Muy acoplado (p.ej. música con sus notas musicales)	± 5 ms
		Poco acoplado (p. ej. diapositivas)	± 500 ms
	Texto	Anotaciones de texto	± 240 ms
	Puntero	Audio relacionado con el ítem que indica el puntero	- 500 ms + 750 ms

Tabla 3.1. Calidad de Servicio para propósitos de Sincronización Inter-flujo

Estos valores suponen una guía para cualquier especificación en cuanto a la sincronización de una presentación multimedia. Sin embargo, según Blakowski y Steinmetz, estos valores no son totalmente estrictos sino que pueden relajarse en función del contenido de la aplicación multimedia en cuestión.

3.3. Protocolos

Los protocolos más importantes utilizados en la Tesis son los siguientes: NTP ([MIL91]) para la obtención del tiempo global y RTP/RTCP ([SCH96]) para la implementación del algoritmo propuesto. Como NTP es un protocolo muy conocido y muy utilizado, se va resumir al máximo su descripción. Sin embargo, ya que se van a proponer modificaciones de los formatos básicos de los paquetes RTP, sí se ha creído necesario incluir una descripción breve de RTP/RTCP para facilitar la comprensión de los capítulos posteriores.

Además, también se ha utilizado un mecanismo de comunicación local entre procesos o aplicaciones denominado *mbus*, basado en el CCCP (*Conference Control Chanel Protocol*, [HAN96]), para el intercambio interno de información entre aplicaciones locales, esencial para poder llevar a cabo la sincronización entre flujos (inter-flujo), parte del algoritmo propuesto.

3.3.1. El protocolo NTP

Existen múltiples razones por las que es interesante mantener sincronizados los relojes de los equipos de una red. La implantación de un servicio de sincronización ofrece obvias ventajas dentro de las siguientes áreas:

- Correo electrónico y listas de distribución, con el fin de obtener fiabilidad en las fechas de envío y recepción de mensajes.

- Sistemas Proxy-caché: Es fundamental que en el intercambio de documentos entre servidores los diversos tiempos asociados al documento (última modificación, tiempo en la caché, etc.) sean precisos para que los documentos puedan considerarse consistentes de acuerdo con la política de refresco y expiración de documentos de la caché.

- Seguridad en red: La detección de problemas de seguridad frecuentemente exige poder comparar registros (*logs*) de acceso de máquinas diferentes, para lo que es imprescindible la coincidencia horaria de las mismas.

- Aplicaciones multimedia donde los requerimientos de sincronización de los relojes de los equipos involucrados influyen en la calidad de las aplicaciones.

En cuanto a métodos para conseguir un tiempo global, en el grupo de investigación *Sistemas Distribuidos en Tiempo Real*, al que pertenece el autor de la Tesis, se han estudiado varias posibilidades:

- Utilizar protocolos de Tiempo Global como, por ejemplo, el de *Sincronización por Olas* y/o el protocolo NTP (*Network Time Protocol*).

- O bien obtener el tiempo del sistema de satélites GPS utilizando receptores adecuados.

El Protocolo de *Sincronización por Olas* está detallado en [EST95] (donde aparece la especificación formal con *Máquinas de Estado Comunicantes*) y en [RAY90]. La versión 3 del protocolo NTP viene definida en la RFC 1305 ([MIL91]). En [MIL92] se puede encontrar la especificación formal de dicha versión. Actualmente ya está disponible la versión 4 de NTP ([MIL94]).

Dentro del Grupo de Investigación, se ha realizado una evaluación de las prestaciones de ambos protocolos presentándose los resultados en [GUE97], concluyéndose que se obtienen mejores prestaciones mediante el protocolo NTP.

En cuanto al uso del sistema GPS, el coste asociado a disponer de un sistema GPS en cada una de las estaciones sería demasiado alto, de momento. En [BOR02b] se presenta una comparativa del grado de sincronización obtenido por receptores GPS comerciales de bajo coste y por NTP en dos servidores con sistemas operativos diferentes como son Windows 2000 y Linux.

En un estudio realizado por D. L. Mills, presentado en [MIL91], tomando como base 94.260 estaciones conectadas a Internet, y utilizando protocolos de transferencia de tiempos propios de los sistemas operativos, se concluyó que la mitad de las estaciones presentaban un error mayor de 2 minutos, y el 10% un error superior a 4 horas. La mayoría de las estaciones disponen de un reloj mantenido por una batería que usa un oscilador de cuarzo con desviaciones típicas de un segundo al día, pudiendo alcanzar incluso desviaciones de semanas después de intervenciones manuales sobre el reloj local. Parece, pues, que muchas aplicaciones en redes distribuidas necesitan sistemas de mantenimiento del tiempo más exactos y seguros. Las principales causas por las que se ha elegido el protocolo NTP como sistema para sincronizar los relojes y gestionar la distribución del tiempo, se enumeran a continuación:

- Se ha propuesto como el protocolo de sincronización a utilizar en Internet. Esta red se puede considerar como una red 'agresiva' en cuanto a retardos y congestión se refiere y, por lo tanto, es un buen banco de pruebas para medir las prestaciones del protocolo. Las medidas y pruebas realizadas con el protocolo NTP en [MIL91] se realizaron sobre más de 100.000 estaciones y más de 1.500 redes de conmutación de circuitos interconectadas mediante pasarelas. Por otra parte, a pesar de la constante evolución de Internet y la tendencia de crecimiento, se ha propuesto a NTP como protocolo estándar de sincronización de tiempos.

- Internet se puede considerar como una red que intenta proporcionar el mejor servicio posible, pero su velocidad y seguridad varía considerablemente a lo largo del trayecto seguido por la información. Sin embargo, se ha comprobado que el protocolo NTP alcanza precisión de milisegundos, incluso en casos de fallo o mal funcionamiento de los relojes, de los servidores de tiempo o de la propia red.

- Utiliza los protocolos de red y transporte IP y TCP/UDP, que forman la pila de protocolos más utilizada por la mayoría de aplicaciones y entornos multimedia.

- Además, como utiliza un único formato, NTP es fácil de implementar y puede ser utilizado con una gran variedad de sistemas operativos y entornos de red. Un efecto inmediato es la existencia de software que implementa el protocolo NTP, con diferentes versiones para diferentes plataformas (estación de trabajo o PCs, con sistemas operativos UNIX, Linux, Windows, etc.).

Se podría destacar el trabajo desarrollado por Mills en [MIL92], que constituye una especificación formal de la versión 3 del protocolo NTP utilizada. Tal y como aparece en la propia RFC 1305, se utiliza para sincronizar un conjunto de clientes y servidores en un entorno distribuido. En este documento se definen, de forma muy extensa y detallada, la arquitectura, algoritmos, entidades, y protocolos utilizados. Por otra parte, en [MIL91] se resumen los requisitos de funcionamiento, se describe el modelo analítico, se estudian los algoritmos de análisis, y se presenta el rendimiento bajo condiciones típicas de Internet. En otro artículo [MIL91b], se describe la referencia de tiempos utilizada por NTP y su relación con otras referencias de tiempo estándar en uso.

Lo que hace único al modelo de NTP es la configuración adaptativa, los mecanismos de interrogación, filtrado, selección y corrección que permiten adaptarse dinámicamente al entorno de Internet o de redes de área amplia en general.

Desde la primera descripción del NTP en la RFC 958 [MIL85b], el protocolo ha evolucionado significativamente en diferentes aspectos. Está construido sobre el protocolo de red Internet IP y el protocolo UDP, que proporciona un mecanismo de transporte no orientado a la conexión. Sin embargo, dada la estructura del protocolo NTP, es fácil adaptarlo a las características de otros protocolos diferentes de IP y UDP. NTP está específicamente diseñado para mantener la exactitud, incluso cuando se utiliza sobre rutas típicas de Internet que incluyen numerosos routers, retardos muy dispersivos y redes poco seguras.

El entorno de funcionamiento se sustenta en el *modelo de implementación* y en el *modelo de servicio*. El *modelo de implementación* se basa en una arquitectura con sistema operativo multi-proceso, aunque también es posible utilizar otras arquitecturas; el *modelo de servicio* se basa en un diseño de retorno de tiempos, que depende únicamente del *offset* estimado entre relojes, pero que no requiere un sistema de reparto de datos excesivamente seguro. La subred de sincronización utiliza una configuración jerárquica *maestro/esclavo* autoconfigurable, con las rutas de sincronización determinadas por un sistema de peso mínimo a lo largo de un árbol de alcanzabilidad. Dentro de esta configuración, denominaremos *servidor de tiempo* a cualquier elemento que realice tareas de distribución de tiempos y los *clientes* únicamente deberán recibir dicho tiempo y actualizar su reloj local.

Respecto a los requerimientos del entorno donde se utiliza NTP, se pueden destacar:

- La *fuente o servidor de referencia primaria* debe estar sincronizado con un estándar nacional, bien a través de una red de cable, vía radio o bien por relojes atómicos calibrados. Dichas fuentes servidoras de tiempo deben coincidir con el tiempo UTC (*Coordinated Universal Time*)[(1)].

- Los *servidores de tiempo* (cualquier elemento que proporcione el tiempo a otros nodos), deben proporcionar un tiempo preciso y exacto, aunque existan grandes variaciones en el retardo entre el servidor y los elementos objeto de sincronización. Este aspecto requiere el diseño de algoritmos de filtrado y combinación, así como mecanismos de sincronización.

- La *subred de sincronización* (red formada por los servidores y clientes) debe ser segura, incluso bajo condiciones de inestabilidad donde la conexión entre servidores y clientes se pueda perder durante días. Este condicionante requiere servidores de tiempo redundantes y caminos alternativos, así como mecanismos para realizar una reconfiguración dinámica cuando se produzcan fallos o un mal funcionamiento.

[1] http://www.aldridge.com/utc.html

- El *protocolo de sincronización* debe trabajar de forma continua y proporcionar información actualizada con suficiente frecuencia para compensar los desvíos que sufren los relojes locales utilizados en los sistemas de ordenadores típicos. El protocolo debe poder trabajar en entornos heterogéneos que incluyen desde estaciones de trabajo hasta supercomputadores, reduciéndose a unos mínimos los requerimientos sobre el sistema operativo. El software de sincronización y especialmente el software de los clientes, debe ser fácil de instalar y configurar.

Respecto a la descripción propiamente dicha del protocolo, en principio, proporciona mecanismos para sincronizar el tiempo con una precisión del orden de nanosegundos, incluyendo mecanismos para estimar el error entre el reloj local y el servidor de tiempo con el cual se debe sincronizar. Como parte del protocolo, también se describen algoritmos para filtrar las muestras de los *offsets* (diferencia de tiempos) del reloj que se han ido recogiendo con respecto a un determinado servidor. Estos algoritmos han sido mejorados como resultado de los experimentos descritos en [MIL85a]. Además, a partir de la experiencia en redes operativas con múltiples servidores que incluían relojes radio en diferentes emplazamientos de EEUU y con clientes en EEUU y Europa, se han desarrollado algoritmos seguros para la elección de relojes fiables de entre un conjunto de relojes donde podían existir relojes imperfectos ([DEC89], [MIL91]).

La precisión que se puede conseguir con NTP depende en gran medida de la precisión del hardware del reloj local. Por lo tanto, se debe tener en cuenta algún mecanismo que ajuste el tiempo y frecuencia del reloj software lógico en respuesta a las correcciones producidas por NTP. En [MIL92] se describe el diseño de un reloj lógico evolucionado de la implementación *Fuzzball*, y detallada en [MIL88]. Este diseño incluye la eliminación del *offset* y compensación de frecuencia, procesos mediante los que se puede obtener una precisión del orden de milisegundos, incluso tras períodos extensos en los cuales se ha perdido la sincronización con la fuente de referencia. Asimismo, se propone un algoritmo opcional, para mejorar la exactitud, mediante la combinación de los *offsets* de un conjunto de relojes. En [MIL92] se detalla el modelo matemático de forma exhaustiva y el análisis de los algoritmos de reloj local de NTP. También se analizan las fuentes y la propagación de errores, presentando una serie de principios correctores relativos al servicio de transferencia de tiempos.

3.3.2. Los Protocolos RTP/RTCP

El protocolo RTP (*Real-time Transport Protocol*) es el protocolo de transporte para flujos multimedia en Internet. Fue diseñado para trabajar con *IP Multicast*, aunque se puede utilizar de forma *unicast*, para proporcionar información temporal y de sincronización de flujos multimedia. Se trata de un

protocolo ligero (*light-weight protocol*) sin mecanismos de control de errores ni de control de flujo. Además, no proporciona ni reserva de recursos ni control de la calidad de servicio. Se trata de un protocolo de transporte independiente de la tecnología de red sobre la que se utilice.

En la figura 3.14 se puede observar cómo puede encajar RTP en la pila de protocolos TCP/IP.

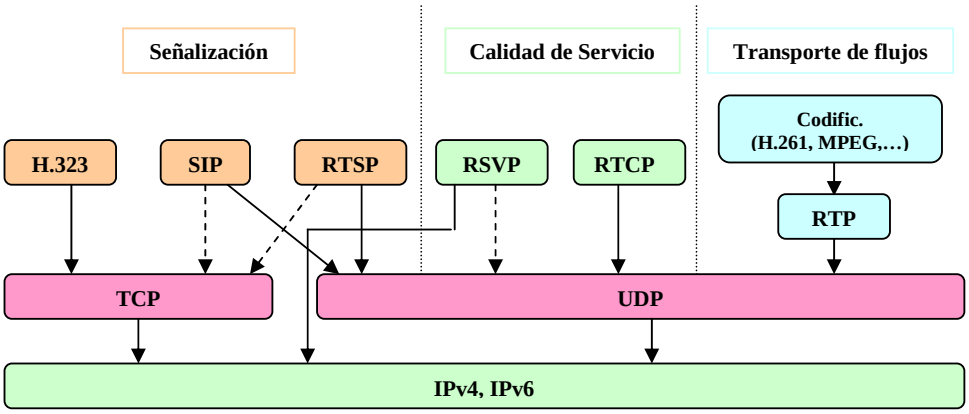


Figura 3.14. RTP en la pila de protocolos

En este apartado se va a describir brevemente el método utilizado por el protocolo RTP para conseguir una sincronización entre diferentes flujos multimedia, como pueden ser de audio, vídeo o la entrega de datos en tiempo real. Para ello, será necesario conocer, en principio, el funcionamiento del protocolo, así como los paquetes utilizados para la transmisión de información de datos y de control. En definitiva, se pretende explicar cómo una estación receptora puede coordinar diferentes flujos de datos procedentes de un mismo emisor, transmitidos de forma separada mediante diferentes sesiones RTP. Para ello, el receptor hará uso de información incluida en la cabecera de los paquetes de datos RTP y del protocolo de control complementario, RTCP, para llevar a cabo la sincronización. En la RFC 1889 ([SCH96]) se incluye una explicación más detallada de RTP y RTCP.

RTP/RTCP proporcionan todas las características para la transmisión de información de flujos multimedia en tiempo real y ofrecen los mecanismos necesarios para que las aplicaciones reproductoras puedan conseguir una *sincronización local* de los flujos recibidos (*Sincronización inter-flujo*).

Una *sesión RTP* consiste en la asociación de un grupo de participantes que se intercambian un mismo flujo mediante RTP en una *sesión multimedia*. Dicho flujo de información transmitido en la sesión RTP quedará definido por una pareja

particular de direcciones de destino a nivel de transporte (una dirección de red más dos puertos, uno para RTP y otro para RTCP). El par de direcciones destino a nivel de transporte podrán ser comunes para todos los participantes, como en el caso de *IP Multicast*, o podrán ser diferentes para cada uno, como en el caso de transmisiones *unicast*. Se define *sesión multimedia* como la transmisión de varios flujos multimedia, en la que cada flujo es transportado en una *sesión RTP* separada con su propia dirección de transporte de destino, lo cual implica flujos separados. El esquema general de funcionamiento se representa en la figura 3.15.

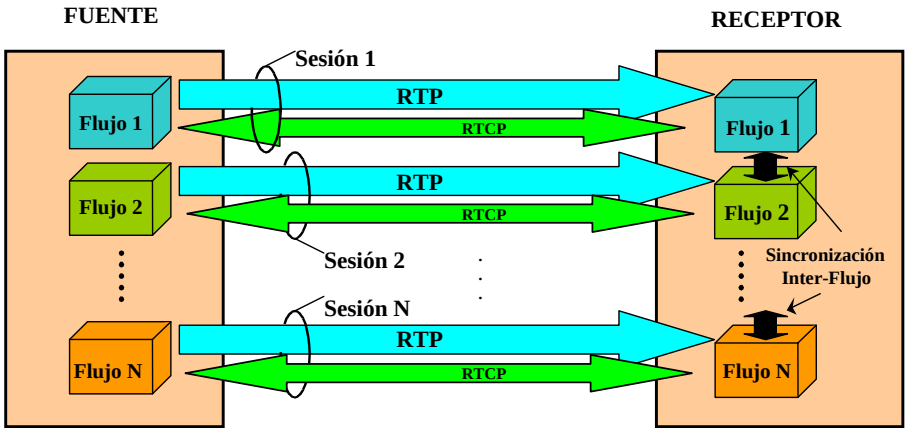


Figura 3.15. Esquema general de una *Sesión Multimedia* utilizando RTP/RTCP

Para identificar la procedencia de un flujo de paquetes RTP se utilizan los denominados *Identificadores de Fuente* o *SSRC* (*Synchronization Source*), únicos en cada sesión, que se incluyen en la cabecera RTP, para que los receptores puedan identificar los paquetes del flujo provenientes de cada fuente involucrada en la sesión multimedia. Todos los paquetes procedentes de la misma fuente formarán parte del mismo espacio de temporización y de números de secuencia. Un ejemplo de fuente RTP puede ser un transmisor de un flujo de paquetes generado a partir de una fuente de señal tal como un micrófono o una cámara de vídeo.

También es posible que en una sesión multimedia se necesiten *medios no RTP*, como pueden ser otros protocolos y mecanismos que necesarios, en adición a RTP, para proporcionar un servicio útil. En nuestro caso, NTP será uno de estos protocolos.

Antes de pasar a la descripción del protocolo, veamos un ejemplo para ilustrar las operaciones básicas de las aplicaciones que utilizan RTP dentro de un escenario típico, como es una conferencia multimedia típica, con flujos de audio y vídeo.

3.3.2.1. Ejemplo: Conferencia de Audio y Vídeo

Si se desea transmitir flujos de audio y vídeo en una conferencia, se transmitirán de forma separada mediante RTP con las direcciones *multicast* (o *unicast*) y puertos correspondientes. Como se ha explicado anteriormente, cada flujo utilizará una dirección y una pareja de puertos (un puerto para la transmisión de datos RTP y el otro para la transmisión de información de control RTCP). En el caso de una videoconferencia con flujos de audio y vídeo, la dirección *multicast* será la misma para ambos flujos. En las aplicaciones, si el puerto elegido para enviar los paquetes RTP del flujo de vídeo es *n*, el de los paquetes RTCP asociados a dicha transmisión, normalmente, será una unidad mayor (es decir, *n+1*). Del mismo modo, si el puerto elegido para enviar los paquetes RTP del flujo de audio es m, el de los paquetes RTCP asociados será *m+1*.

Las aplicaciones de audio y vídeo utilizadas por cada participante enviarán datos en pequeños bloques. Cada bloque, con un tamaño establecido a priori, estará precedido por una cabecera RTP, formando un paquete RTP que será encapsulado en un paquete UDP para ser transmitido a través de una red IP. La cabecera RTP, cuyo formato se describe en apartados posteriores, indicará qué tipo de codificación de audio o vídeo contiene cada paquete para que los receptores lleven a cabo la decodificación correcta (por ejemplo, GSM para el audio y H.261 para el vídeo). De esta forma, los transmisores, si es necesario, podrán, incluso, cambiar la codificación durante la conferencia como les convenga (por ejemplo, para evitar congestión en la red) y el receptor podrá llevar a cabo la decodificación correcta. Además, la cabecera contiene información temporal y un número de secuencia para permitir al receptor solucionar problemas de red, como pueden ser la pérdida, el desorden o los retrasos de los paquetes recibidos.

Durante el transcurso de la conferencia, las aplicaciones transmitirán de forma periódica información al puerto de control RTCP. Está información será de gran utilidad para conocer el estado de la transmisión así como la identidad de los participantes (nombre, dirección de correo electrónico, teléfono, etc.).

Entre las sesiones de audio y vídeo no habrá una correspondencia directa a nivel RTP, pero sí a nivel RTCP mediante un distintivo establecido como CNAME (*Canonical Name* o nombre canónico, descrito posteriormente), de forma que las dos sesiones se puedan asociar de alguna forma en el receptor.

La explicación de la separación entre los distintos flujos reside en la posibilidad de permitir, a algunos participantes de la conferencia, recibir sólo aquellos flujos que les interese o puedan recibir. De esta manera, si en una conferencia se está transmitiendo audio y vídeo con una tasa mayor a la permitida por la conexión de alguno de los posibles receptores, éste tendrá la opción de recibir solamente uno de los flujos. Como ejemplo, se podría pensar en un receptor

con problemas de ancho de banda que podría optar por seleccionar solamente el flujo de audio para, por lo menos, poder escuchar la conferencia.

Además, a pesar de la separación de los dos flujos en sesiones RTP distintas, gracias a la información temporal que contienen los paquetes, tanto RTP como RTCP, en recepción, se podrá sincronizar la reproducción de ambos flujos.

3.3.2.2. Descripción

Como se ha comentado, el protocolo RTP proporciona funciones de transporte de red extremo a extremo de forma *multicast* o *unicast*, y es el indicado para aplicaciones de transmisión de información en tiempo real, como son el audio y el vídeo. De forma independiente, este transporte de datos se complementa con el protocolo de control RTCP, que permite monitorizar la entrega de datos y proporcionar un adecuado control entre los participantes.

Las funciones proporcionadas por RTP incluyen, entre otras, la identificación del tipo de información, el número de secuencia y una marca de tiempo del instante de generación (*timestamp*). Todo esto, junto con RTCP que se encarga de la transmisión 'periódica' de paquetes de control por parte de todos los participantes, hace de RTP un protocolo idóneo para obtener la sincronización en tiempo real entre diferentes flujos de información multimedia.

Cabecera del paquete RTP

La cabecera de un paquete RTP, tiene el formato mostrado en la figura 3.16. Los primeros veinte octetos se presentan en todos los paquetes RTP, mientras que la lista de los identificadores CSRC se presentan sólo cuando los inserta un dispositivo *mezclador* (definido en la RFC 1889), que es un sistema intermedio que recibe paquetes RTP de una o más fuentes y los combina (incluso cambiando el formato de los datos si fuera necesario) y reenvía un nuevo paquete RTP.

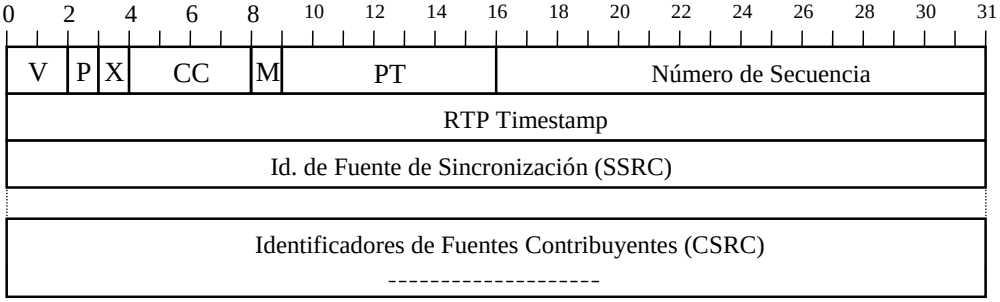


Figura 3.16. Cabecera del paquete RTP

Los campos de dicha cabecera tienen el siguiente significado:

- *Versión (V, Version):* Este campo de 2 bits identifica la versión de RTP. La versión definida para esta especificación es la 2. (El valor 1 se utilizó por la primera versión borrador de RTP, y el 0 por el protocolo inicialmente implementado en la herramienta Mbone *vat*, de audio).
- *Bit de Relleno (P, Padding):* Si se activa el bit de relleno, el paquete contendrá uno o más octetos adicionales de relleno, al final de la cabecera, que no forman parte de la carga útil. En el último octeto del relleno se especifica cuántos octetos deben ser ignorados.
- *Bit de Extensión (X, eXtension):* Si se activa el bit de extensión, la cabecera irá seguida por sólo una cabecera de extensión, con el formato definido en la RFC 1889.
- *Contador de CSRC (CC, Count of CSRC):* El contador CSRC de 4 bits contiene el número del identificadores CSRC que van a continuación de la cabecera fija.
- *Bit de Marcador (M, Marker):* La interpretación del marcador se definirá en cada perfil. Está pensado para señalizar eventos significativos, como por ejemplo, el primer paquete que se envía en un determinado flujo de paquetes.
- *Tipo de Carga Útil (PT, Payload Type):* Este campo de 7 bits identifica el formato de la carga útil RTP y determina su interpretación por parte de las aplicaciones.
- *Número de Secuencia (Sequence Number):* El número de secuencia (16 bits) se ve incrementado por cada uno de los paquetes de datos RTP enviados, y puede ser utilizado por el receptor para detectar pérdida de paquetes y restaurar la secuencia de paquetes. El valor inicial de los números de secuencia es aleatorio.
- *Marca Temporal (Timestamp):* La marca temporal de 16 bits refleja el instante de muestreo del primer octeto del paquete de datos RTP. El instante de muestreo debe provenir de un reloj que incremente monótona y linealmente el tiempo para permitir la sincronización y el cálculo del *jitter*. La resolución del reloj debe ser lo suficientemente grande para producir el nivel de sincronización deseado y para medir el *jitter* de llegada del paquete. La frecuencia de reloj depende del formato de los datos contenidos en la carga útil de los paquetes RTP. Si los paquetes RTP se generan periódicamente, se utilizará el instante de muestreo que viene determinado por el dispositivo de reproducción utilizado, no por una lectura del reloj del sistema. Como ejemplo, para audio con tasa fija (*fixed-rate audio*) el reloj utilizado para obtener las marcas temporales, debería incrementarse, probablemente, en uno cada periodo de muestreo. Si una aplicación de audio lee bloques que cubren unos 160 periodos de muestreo del dispositivo de entrada, las marcas temporales deberían incrementarse en 160 para cada uno de los bloques, tanto si los bloques se transmiten en

un paquete como si se desechan como silencios. El valor inicial de las marcas temporales se toma también de forma aleatoria.

- *SSRC (Synchronization SouRCe):* El campo SSRC de 32 bits identifica la fuente. Este identificador se elige aleatoriamente, con la intención de que dos fuentes de sincronización dentro de una misma sesión RTP no tengan los mismos identificadores SSRC.
- *Lista CSRC:* de 0 a 15 ítems, con 32 bits cada uno. La lista CSRC identifica las fuentes contribuyentes en la carga útil contenida en este paquete. El numero de identificadores lo da el campo CC. El campo CSRC será insertado por mezcladores, utilizando los identificadores SSRC de las fuentes contribuyentes.

Protocolo de Control RTCP

El Protocolo RTCP se basa en la transmisión periódica de paquetes de control a todos los participantes dentro de una sesión, utilizando los mismos mecanismos de distribución que para los paquetes RTP de datos. Será necesario que los protocolos de niveles inferiores proporcionen multiplexación de los paquetes de datos y de control como, por ejemplo, en el caso de UDP que utiliza puertos separados. Este protocolo desempeña cuatro funciones:

1. La principal función es proporcionar realimentación sobre la distribución de los datos. Esta función es posible gracias a los informes de control RTCP, enviados por los emisores y receptores.

2. Transportar un identificador global y único a nivel de transporte, para una fuente RTP, denominado nombre canónico o CNAME (*Canonical Name*). Puesto que el identificador SSRC puede cambiar si se descubre un conflicto o se reinicia una aplicación, los receptores necesitarán conocer el CNAME para identificar a cada participante. Los receptores también necesitarán el CNAME para asociar múltiples flujos de datos que provengan de un mismo participante en un conjunto de sesiones RTP relacionadas.

3. Controlar la tasa de envío de información de información de control para que se permita escalar a un mayor número de participantes.

4. Transportar una información mínima de control de la sesión, por ejemplo la identificación de los participantes que se suele mostrar en la *interface gráfica* de las aplicaciones de usuario.

Tipos de Paquetes RTCP

La especificación de RTCP define varios tipos de paquetes que transportan información de control:

- *SR o Sender Report* (informe del transmisor): Paquete con datos útiles para mantener estadísticas de transmisión y recepción de los participantes que están activos como transmisores.
- *RR o Receiver Report* (informe del receptor): Paquete con datos útiles para mantener estadísticas de recepción de los participantes que no están activos como transmisores sino que son únicamente receptores.
- *SDES o Source DEScription* (descripción de la fuente): Paquete con datos de la fuente transmisora, como, por ejemplo, el CNAME (*Canonical Name*).
- *BYE* (adiós): Paquete para indicar el final de la participación o abandono de la sesión RTP.
- *APP* (de aplicación): Paquete con funciones específicas definido para una aplicación en particular.

Formato de los Paquetes RTCP

Cada paquete RTCP empieza con una parte fija similar a la de los paquetes de datos RTP, seguido por elementos estructurados que pueden ser de longitud variable dependiendo del tipo de paquete, pero que siempre finalizan con una marca de 32 bits.

Múltiples paquetes RTCP pueden ser concatenados formando un paquete compuesto, sin necesidad de separadores, que es transmitido en una única PDU del protocolo de nivel inferior (por ejemplo, UDP). Cada paquete individual RTCP dentro del paquete compuesto se puede procesar independientemente sin ningún requerimiento en cuanto a un orden o combinación de paquetes. Sin embargo, para poder ejecutar las funciones del protocolo, se imponen las siguientes restricciones:

- Las estadísticas de recepción (en SR o RR) deberán ser enviadas con la periodicidad permitida por las restricciones de ancho de banda, para maximizar la resolución de las estadísticas, por lo que cada paquete compuesto RTCP transmitido periódicamente deberá incluir un paquete de informe.

- Los nuevos receptores necesitan recibir el CNAME desde una fuente tan pronto como sea posible para identificar la fuente y empezar a asociar los flujos con propósitos tales como, por ejemplo, conseguir sincronización

entre ellos (sincronización labial), de forma que cada paquete compuesto deberá incluir también el paquete SDES con el CNAME.

- El número de tipos de paquetes que pueden aparecer, en primer lugar, dentro del paquete compuesto deberá estar limitado para aumentar el número de bits constantes en la primera palabra y la probabilidad de validar con éxito los paquetes RTCP frente a paquetes de datos con direcciones erróneas u otros paquetes no relacionados.

Así, todos los paquetes RTCP deben ser transmitidos en un paquete compuesto por, al menos, dos paquetes individuales, siguiendo el siguiente formato:

- Si el paquete está encriptado, llevará un prefijo de encriptación, de 32 bits.

- El primer paquete del paquete compuesto será siempre un paquete de informe (SR o RR) para facilitar la validación de la cabecera que debe realizarse tal y como se define en la RFC 1889.

- Paquetes RR adicionales.

- Un paquete SDES incluyendo el CNAME deberá ser incluido en cada paquete compuesto.

- Otros tipos de paquetes, como BYE o APP, pueden seguir en cualquier orden, incluso varios de cualquier tipo. Los paquetes BYE deben ser los últimos.

Paquetes de Informe del Transmisor y del Receptor

En una sesión RTP, los receptores RTP proporcionan información de realimentación (*feedback*) sobre la calidad de la recepción utilizando *paquetes de informes RTCP*, que pueden tomar una de las dos formas indicadas anteriormente, según si el receptor es también emisor o no: informes de emisor si lo es (SR o *Sender Reports*) y de receptor si no lo es (RR o *Receiver Reports*). La única diferencia entre la forma del informe del transmisor (SR) y del receptor (RR), además del código del paquete, es que el informe del transmisor incluye una sección de información del emisor, de 20 bytes, para el uso de los emisores activos.

A continuación, se describe el formato de ambos tipos de paquetes de informe, cómo se pueden extender de una manera específica por el perfil en caso de que una aplicación necesite información de realimentación adicional, y cómo pueden ser utilizardos.

- SR: Paquete RTCP de informe del transmisor

El paquete de informe del transmisor (SR) tiene el formato mostrado en la figura 3.17 y consta de tres secciones, posiblemente seguidas por una cuarta sección de extensión definida por la especificación de un perfil.

La primera sección, la cabecera, tiene una longitud de 8 octetos, y sus campos tienen el siguiente significado:

- *Versión y Bit de Relleno (P, padding):* Mismo significado que para los paquetes RTP.
- *Contador de Informe Receptor (RC, Reception report Count):* 5 bits. Contiene el número de bloques de informes de recepción incluidos en el paquete.
- *Tipo de Paquete (PT, Packet Type):* Ocho bits que codifican la constante '200' para identificar que se trata de un paquete SR de RTCP.
- *Longitud:* 16 bits que indican la longitud del paquete RTCP en palabras de 32 bits menos uno, incluyendo la cabecera y cualquier relleno.
- *SSRC:* Identificador de la fuente que genera el paquete SR.

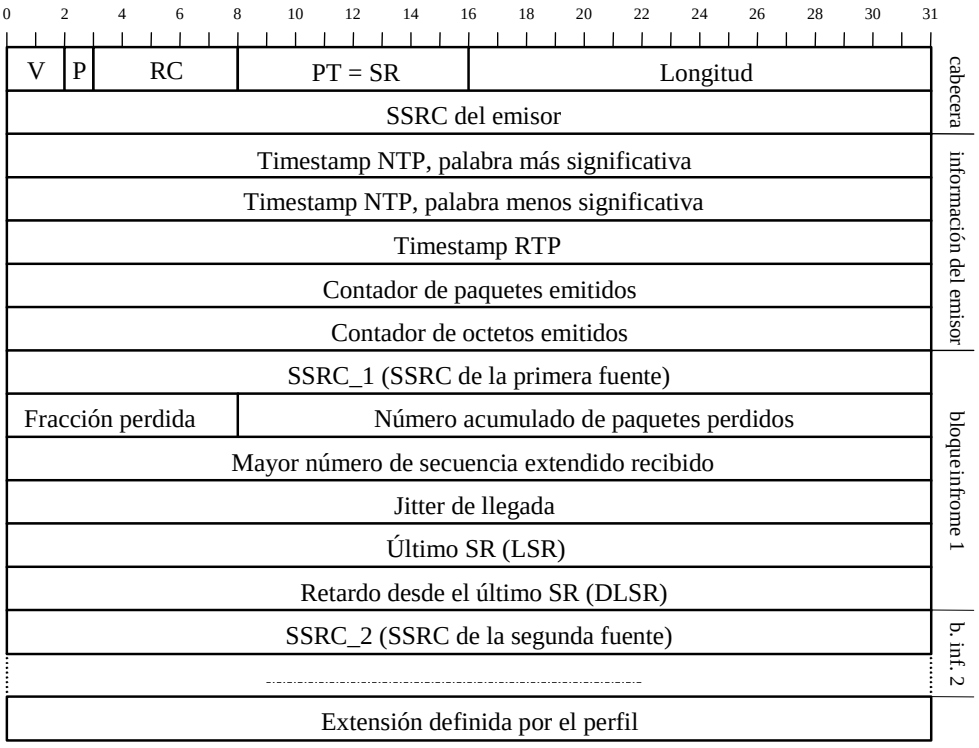


Figura 3.17. Paquete SR de informe del transmisor

En la segunda sección está la información del propio transmisor, con una longitud de 20 octetos, y está presente en cada paquete SR. Los campos incluidos en ella tienen el siguiente significado:

- *Marcas Temporales NTP (NTP timestamp):* 64 bits donde se codifica el tiempo NTP del envío del informe, que puede ser utilizado en combinación con las marcas temporales devueltas en los informes de recepción recibidos de otros receptores para medir el retardo de ida y vuelta a dichos receptores.
- *Marcas Temporales RTP (RTP timestamp):* 32 bits que codifican el mismo instante que las marcas temporales NTP anteriores, pero con las mismas unidades y con el mismo *offset* aleatorio, que las marcas temporales de los paquetes de datos RTP. *Esta correspondencia puede ser utilizada para la sincronización intra e inter flujo para fuentes cuyas marcas temporales NTP estén sincronizadas*, y puede ser utilizada por receptores de flujos independientes para estimar la frecuencia nominal del reloj RTP.
- *Contador de Paquetes emitidos:* Indica el número total de paquete de datos RTP transmitidos por el emisor desde el comienzo de la transmisión hasta el instante en que se generó el paquete SR.
- *Contador de Octetos Emitidos:* Indica el número total de octetos de carga útil (es decir, sin incluir cabecera ni relleno) transmitidos en paquetes de datos RTP por el emisor desde el comienzo de la transmisión hasta el instante en que se generó el paquete SR.

Por último, la tercera sección contiene cero o más bloques de informes de recepción dependiendo del número de otras fuentes 'escuchadas' por este emisor desde el último informe. Cada bloque de informe de recepción contiene estadísticas de la recepción de paquetes RTP procedentes de una única fuente. Estas estadísticas son:

- *SSRC_n (identificador de fuente):* El identificador SSRC de la fuente a la que hace referencia la información del informe.
- *Fracción Perdida:* Fracción de paquetes de datos RTP perdidos de la fuente, desde el último envío de paquetes SR o RR. Es el resultado de dividir el número de paquetes perdidos por el número total de paquetes esperados.
- *Número Acumulado de Paquetes Perdidos:* Número total de paquetes de datos RTP perdidos de la fuente desde el inicio de la recepción. Es la diferencia entre el número de paquetes esperado y el número de paquetes recibido en el instante de generación del informe.
- *Mayor Número de Secuencia Extendido Recibido:* Los 16 bits más bajos contienen el mayor número de secuencia recibido en un paquete de datos RTP de la fuente SSRC_n, y los 16 bits más significativos extienden ese

número de secuencia con el correspondiente contador de los ciclos de números de secuencia.

- *Jitter de Llegada:* 32 bits que contienen una estimación de la variación estadística de los instantes de llegada de los paquetes de datos RTP, medido en unidades de marcas temporales (*timestamp*) y expresado como un entero sin signo.
- *Último SR (LSR, Last SR timestamp):* Contiene los 32 bits del centro de los 64 bits de la marca temporal NTP recibida en el paquete SR más reciente, de la fuente SSRC_n.
- *Retardo desde el Último SR (DLSR, Delay since Last SR):* 32 bits que indican el retardo, expresado en unidades de 1/65536 segundos, entre el último paquete SR recibido de la fuente SSRC_n y el envío del informe.

Como se puede apreciar, en el paquete existe una extensión que puede ser utilizada para añadir información adicional necesaria para el buen funcionamiento de un perfil o aplicación específicos.

- RR: Paquete RTCP de informe del receptor

El formato para el paquete de informe RR, será el mostrado en la figura 3.18.

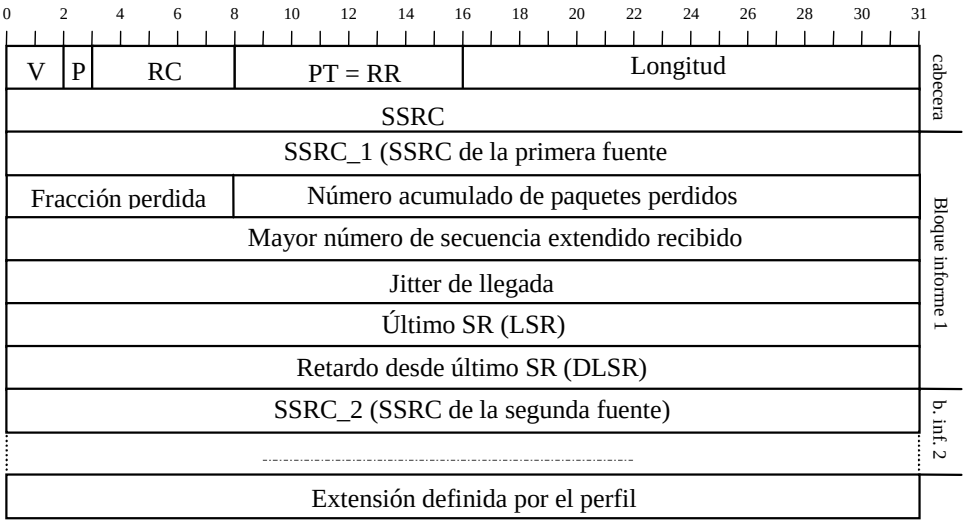


Figura 3.18. Paquete RR de informe del receptor

El formato del paquete RR es el mismo que el del paquete SR con dos diferencias: el valor del campo tipo de paquete (PT) contiene, en este caso, la constante ‘201’ y las cinco palabras de la información del emisor son omitidas (las marcas temporales NTP y RTP y los contadores de paquetes y octetos enviados).

Los demás campos tienen el mismo significado que en el paquete SR. En este caso, también existe una extensión que puede ser utilizada para añadir información adicional necesaria para el buen funcionamiento de un perfil o aplicación específicos.

Este paquete es considerado de gran importancia en la Tesis, ya que esta extensión será utilizada para incluir determinados campos que informarán a la fuente transmisora del estado de los procesos de reproducción de los receptores.

Según la RFC 1889, un perfil o una aplicación pueden definir extensiones específicas a los paquetes de informe si existe información adicional que deba ser enviada periódicamente sobre el emisor o los receptores. Este método debería ser usado con preferencia antes de definir otro tipo de paquetes RTCP ya que requiere menos sobrecarga u *overhead*, al contener menos octetos el paquete (no habrá cabecera ni campo SSRC) y el análisis de los paquetes en las aplicaciones que conozcan dicho perfil será más simple y rápido.

Si se requiere información adicional sobre el emisor, debería ser incluida en los paquetes SR, pero no estaría presente en los paquetes RR. Si se necesita más información sobre los receptores, dichos datos podrán ser incluidos como un array de bloques paralelos al array existente de bloques de informes de recepción.

En la presente Tesis se propone una modificación del formato del paquete RR en la cual se incluyen determinados datos necesarios para la correcta ejecución del algoritmo presentado. Estos datos se han incluido, como se ha comentado, en el campo 'extensión definida por el perfil o por la aplicación'. El formato del paquete RR modificado se describirá en el capítulo siguiente, y seguirá las premisas anteriores, indicadas en la RFC 1889.

- SDES: Paquete RTCP de descripción de la fuente

Tal y como se aprecia en la figura 3.19, el paquete SDES tiene una estructura de tres niveles, y está compuesto por una cabecera y cero o varios bloques, cada uno de los cuales se compone de una serie de ítems que describen la fuente identificada en dicho bloque.

El significado de los campos es el siguiente:

- *Versión (V), Relleno (P, padding), longitud:* msmo significado que en paquete SR.
- *Tipo de Paquete (PT, packet type):* En este caso, contiene la constante '202', que identifica que este es un paquete RTCP SDES.
- *Contador de Fuentes (SC, Source Count): 5 bits.* Indica el número de bloques SSRC/CSRC contenidos en el paquete.

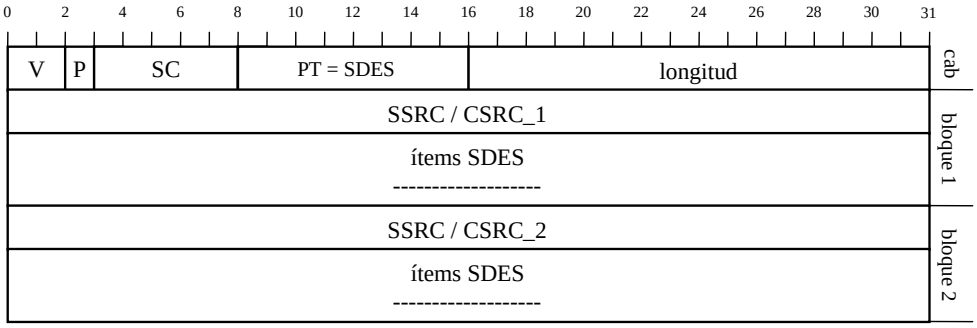


Figura 3.19. Paquete SDES de descripción de la fuente

Cada bloque consiste en un identificador SSRC/CSRC seguido por una lista de cero o varios ítems, que contienen información sobre el SSRC/CSRC. Cada uno comienza con una demarcación de 32-bits, y cada ítem consta de un campo de 8 bits para el tipo de ítem, un contador de 8 bits que indica la longitud del texto (sin incluir estos dos octetos) y el propio texto (de hasta 255 octetos), codificado en UTF-8 ([YER98]).

Entre los ítems incluidos en el paquete SDES, podemos destacar, principalmente, el CNAME y otros ítems utilizados por las aplicaciones que sirven para proporcionar información adicional del participante dentro de una sesión, como, por ejemplo, el nombre, teléfono, dirección de correo electrónico, dirección o localización geográfica y herramienta utilizada. Éstos se corresponden con información introducida por el usuario, normalmente, en las aplicaciones, para dar a conocer sus propios datos. De entre todos ellos, sólo el ítem CNAME se considera obligatorio, por lo que será el único que se explicará en este capítulo. La información del resto de ítems se puede encontrar en la RFC 1889.

El identificador CNAME tiene el formato de la figura 3.20 y tiene las siguientes propiedades:

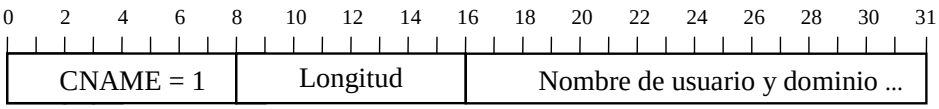


Figura 3.20. Identificador CNAME

- Debido a que la asignación aleatoria del identificador SSRC puede cambiar si se descubre un conflicto o si se reinicia la aplicación transmisora, el ítem CNAME es necesario para proporcionar la relación del identificador SSRC con un identificador de la fuente que permanece constante.

- Al igual que el identificador SSRC, el identificador CNAME también debe ser único entre todos los participantes dentro de una sesión RTP.

- El CNAME de cada participante se fijará para, así, proporcionar un vínculo de unión entre múltiples herramientas utilizadas por un participante en un grupo de sesiones RTP relacionadas (por ejemplo, las herramientas de audio y vídeo en una videoconferencia).

- Para facilitar una monitorización externa, el CNAME debería ser el adecuado para que un programa o persona pudiera localizar la fuente.

Por lo tanto, el CNAME debe ser deducido algorítmicamente y no introducido manualmente, cuando sea posible. Para cumplir estos requerimientos, se deberá utilizar el siguiente formato a menos que un perfil especifique una sintaxis o semántica alternativa. El CNAME deberá tener el formato '*usuario@host'*, o '*host'* si el nombre de usuario no está disponible como en los sistemas monousuario. Para ambos formatos, '*host'* es el nombre de dominio completo del equipo desde el que se originan los datos en tiempo real, según el formato expresado en las RFCs 1034, 1035 y la RFC 1123, o la representación en ASCII de la dirección numérica del host en el interfaz de red utilizado para las comunicaciones RTP. Como ejemplo, se podrían obtener CNAMEs como los siguientes: *fboronat@TLM01* o *fboronat@158.42.149.43*.

- BYE: Paquete RTCP de despedida

El paquete de despedida tiene el formato que aparece en la figura 3.21 y se utiliza para indicar que una o más fuentes no seguirán estando activas en una sesión.

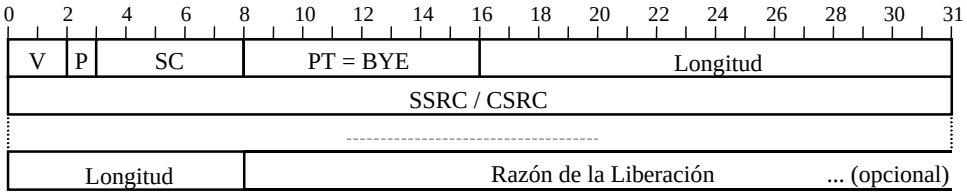


Figura 3.21. Paquete BYE de despedida

El significado de sus campos es:

- *Versión (V), Relleno (P, padding), longitud:* tienen el mismo significado que en los paquetes anteriores.
- *Tipo de Paquete (PT, packet type):* Contiene la constante '203' para identificar que es un paquete RTCP BYE.

- *Contador de fuente (SC):* El número del identificadores SSRC/CSRC incluidos en este paquete BYE.

Opcionalmente, el paquete BYE puede incluir un campo de texto indicando la razón del abandono como, por ejemplo, '*mal funcionamiento de la cámara'* o '*lazo RTP detectado'*, y todo ello precedido por un octeto donde se indica la longitud (en octetos) de dicha razón.

- APP: Paquete RTCP definido por la aplicación

El paquete APP está pensado para un uso experimental mientras nuevas aplicaciones y nuevas características son desarrolladas, sin la necesidad de registrar nuevos valores de tipos de paquetes. Su formato es el mostrado en la figura 3.22.

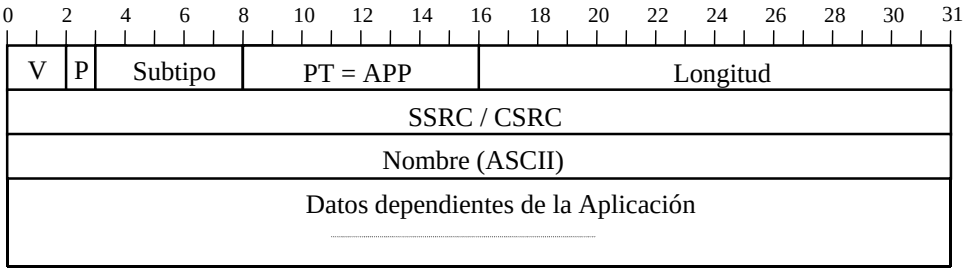


Figura 3.22. Paquete APP, definido por la aplicación

Sus campos tienen el siguiente significado:

- *Versión (V), Relleno (P, padding), longitud:* mismo significado que los anteriores.
- *Subtipo:* Se puede utilizar como un subtipo para permitir que un conjunto de paquetes APP se definan bajo un único nombre, o por algún dato dependiente de la aplicación.
- *Tipo de Paquete (PT, packet type):* 8 bits. Contiene la constante '204' para identificarlo como un paquete RTCP APP.
- *Nombre:* 4 octetos. Un nombre elegido por la persona que define al conjunto de paquetes APP, y que debe ser único con respecto a otros paquetes APP que las aplicaciones pudieran recibir. El creador de la aplicación puede elegir la utilización del nombre de la aplicación y, así, coordinar la asignación de los valores subtipo con otros autores que quieran definir nuevos tipos de paquetes para dicha aplicación. Como alternativa, se recomienda que los demás elijan un nombre basado en la entidad que ellos representen y coordinar el uso del nombre dentro de dicha entidad. El nombre es interpretado como una secuencia de cuatro caracteres ASCII, con distinción entre mayúsculas y minúsculas. Los

paquetes APP con nombres desconocidos deberán ser ignorados por las aplicaciones.

- *Datos dependientes de la Aplicación* (de longitud variable, múltiplo de 32 bits): Los datos que dependen de la aplicación pueden o no aparecer en un paquete APP. Éste es interpretado por la aplicación y no por RTP.

Después de comprobar si el uso extendido de los nuevos paquetes está justificado, la RFC 1889 recomienda que cada paquete APP sea redefinido sin los campos de subtipo y nombre y sea registrado por la IANA utilizando un tipo de paquete RTCP.

Este tipo de paquete ha sido utilizado en la Tesis para definir nuevos paquetes necesarios para el intercambio de información entre fuente y receptores. Concretamente se han definido tres nuevos paquetes que siguen el formato del paquete APP de RTCP: uno, denominado APP TIN, que es utilizado por la fuente para informar a todos los receptores del tiempo de inicio de reproducción; otro, denominado APP RET, que es enviado por los receptores a la fuente con información temporal, de modo que ésta pueda calcular el retardo máximo correspondiente a cada receptor; y, por último, otro, denominado APP ACT, utilizado por la fuente para que indicar a los receptores que ajusten su estado de reproducción durante la sesión. La descripción de estos paquetes se explica detalladamente en el capítulo siguiente, así como la utilidad de cada uno.

3.3.3. *Mbus*: Bus de Conferencia Local

Ya que el algoritmo propuesto se va a implementar en aplicaciones multimedia independientes para cada flujo, ha sido necesario el uso de un mecanismo que facilitara la coordinación entre ellas, sobre todo para conseguir la sincronización de flujos deseada. En nuestro caso se ha utilizado el mecanismo denominado *mbus* (*Message Bus*)[(2)] que, en principio, es una tecnología pensada para facilitar el diseño modular de aplicaciones, de la que existen implementaciones para C, C++ y Java.

El mecanismo es muy simple: cada aplicación puede enviar un mensaje broadcast de un cierto tipo por el bus y todas las aplicaciones que estén registradas para recibir dicho tipo de mensajes, recibirán una copia del mismo.

Se trata de un bus de conferencia que transmite información de forma *multicast* con un TTL igual a 0, que mantiene todos los beneficios del *multicast* y la pertenencia dinámica a grupos, pero restringiendo el tráfico al entorno local, es decir, dentro de la propia estación de trabajo.

[2] http://www.mbus.org

Mbus proporciona un canal de intercomunicación entre procesos que se diseñó para realizar una coordinación local en sistemas de conferencia multimedia. Facilita el intercambio de información entre las distintas aplicaciones de los sistemas finales componentes de una aplicación global multimedia compuesta. Permite su coordinación para, por ejemplo, facilitar la sincronización de la representación de los flujos manejados por cada una de las aplicaciones, u ofrecer la posibilidad de configurar las mismas sin la necesidad de la interacción con el usuario.

Un escenario típico en Mbone fue la utilización en una misma estación de trabajo de varias aplicaciones, como, por ejemplo, de audio, vídeo y pizarra y/o editor de textos compartidos para la participación en una conferencia. Estas aplicaciones suelen ser herramientas separadas (aunque existen algunas herramientas que pueden combinarlas) que transmiten flujos de información por separado sobre RTP y necesitan ser coordinadas de alguna manera para ofrecer un resultado global como si de una única aplicación se tratara.

En el caso que nos ocupa, se desea una intercomunicación entre diferentes herramientas para que haya una coordinación lo suficientemente perfecta, como para conseguir una sincronización fina entre los diferentes flujos. El ejemplo más directo es la consecución de sincronización labial o *lip-synch*.

Para conseguir una sincronización lo más exacta posible entre los diferentes flujos que llegan a un receptor, es necesario que las herramientas encargadas de reproducir dichos flujos tengan conocimiento durante todo momento del punto de reproducción (*playout point*) en el que se encuentran las demás (o sólo una, dependiendo de si se elige un algoritmo de sincronización de tipo *maestro/esclavo*). Los mensajes *mbus* son un buen método para conseguir intercambiar información entre ellas para conseguir dicho propósito. Cada aplicación de tiempo real suele introducir un retardo de almacenamiento (*buffering delay*), para adaptar las variaciones del retardo o *jitter* en la red que influyen en el cálculo del punto de reproducción. En cada herramienta, este punto puede ser ajustado para conseguir la sincronización entre flujos. Para ello, las herramientas enviarán mensajes de sincronización de forma broadcast haciendo uso del bus común, para que todas puedan calcular su punto de reproducción sincronizado con las demás. Normalmente, todas calcularán el máximo de entre todos los retardos enviados y ejecutarán un cierto algoritmo de adaptación para conseguir la calidad de sincronización deseada.

El *bus de conferencia* es implementado, normalmente, como *sockets* de datagramas *multicast* (*multicast datagram sockets*) limitados al interface local (*loopback*). La utilización de *IP Multicast* localmente, dentro de una misma máquina, proporciona a un proceso una manera eficiente de enviar información a

un conjunto arbitrario de procesos sin necesidad de que los destinatarios tengan que estar conectados directamente o en red (*wired in*).

Ya que un mismo usuario puede estar participando en varias conferencias simultáneamente, se utiliza la dirección de transporte (puerto de destino UDP) para crear un bus separado para cada conferencia. Cada herramienta sabrá que todo lo que envíe y reciba por dicho bus será relativo a la conferencia en la que está participando. Además, este mecanismo puede mejorar el funcionamiento de las aplicaciones ya que permite que se comuniquen unas con otras cuando sea necesario y que 'despierten' sólo cuando haya actividad en la conferencia.

3.3.3.1. Origen de *mbus* (CCCP)

Mbus está basado casi en su totalidad en el *Conference Control Chanel Protocol* (*CCCP*) surgido como parte del proyecto MICE (*Multimedia Integrated Conferencing for Europe,* [HAN95a]).

El *CCCP* ofrece un canal de control de la conferencia (*CCC*) que se encarga de unir los elementos que componen la conferencia, de forma semejante a la comunicación interna utilizada por el sistema operativo. Este canal de comunicación común ofrece facilidades y servicios a las aplicaciones que necesitan comunicarse con otras consiguiendo una estructura compacta y centralizada. Soporta comunicación *multicast* (siempre en una misma máquina) pudiendo enviar un mismo mensaje a varias aplicaciones participantes en la conferencia. Además, ofrece varios niveles de confirmación y fiabilidad en la entrega de dichos mensajes.

La figura 3.23 se muestra el esquema de una conferencia mediante el CCCP entre tres estaciones de trabajo que transmiten audio, vídeo y datos, de forma separada y desde tres puntos diferentes de una red de comunicaciones.

Los bloques Control de Sesión y Floor Control corresponden a las aplicaciones de control, que son las encargadas de administrar y controlar, de forma centralizada, la conferencia. Éstas, por ejemplo, pueden ser capaces de seleccionar los medios a utilizar (audio, vídeo o texto) e incluso decidir la tasa de transmisión de datos de cada uno, así como su comportamiento ante el usuario.

El canal de control de la conferencia (CCC) es la vía de comunicación (bus local) utilizada por las aplicaciones multimedia y de control. Es la encargada de retransmitir los mensajes a las direcciones de destino y de proporcionar una coordinación entre todas las aplicaciones utilizadas en la conferencia. Su funcionamiento fue perfeccionado para dar origen al bus de conferencia local *mbus*, explicado con un poco más de detalle en los apartados siguientes.

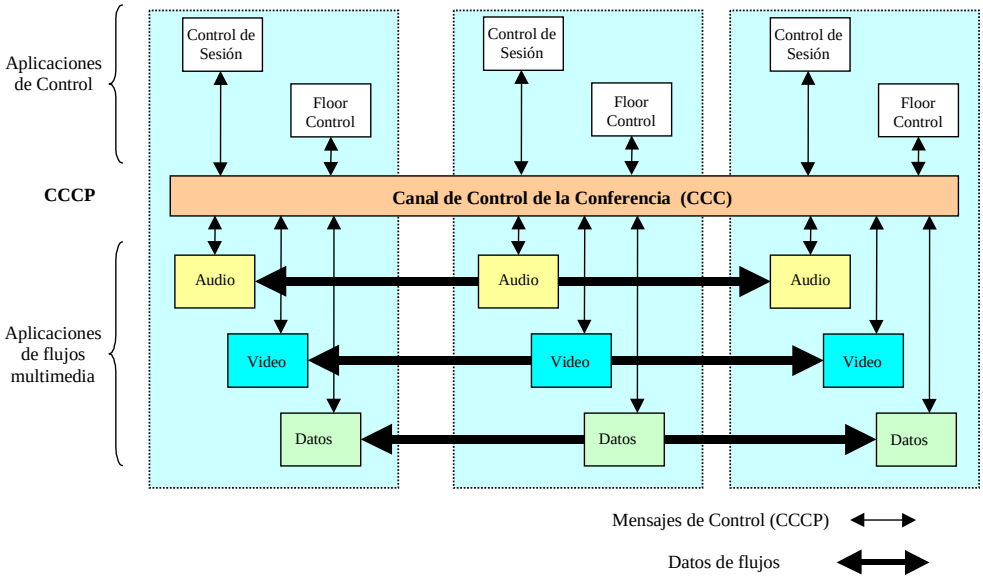


Figura 3.23. Esquema de una conferencia mediante CCCP

3.3.3.2. Biblioteca *mbus*

Existe un esquema flexible para direccional los procesos en *mbus*, así como facilidades en cuanto a la entrega de mensajes de forma fiable. *Mbus* sólo proporciona el mecanismo para poder coordinar una conferencia, pero no define cómo realizar la coordinación ni el contenido de sus mensajes.

Existe una librería programación de *mbus* de dominio público, cuya utilización se describe en [PER98] así como dónde obtener documentación relacionada(3).

Direccionamiento

Cada entidad o aplicación conectada al bus de conferencia local, deberá responder a los mensajes enviados a una o más direcciones. Las direcciones utilizadas son cadenas de caracteres que siguen el siguiente formato:

(Media:Type Module:Type App:Name Instance:Id)

[3] http://www.mbus.org/

, donde uno o más campos pueden omitirse mediante un espacio en blanco, de forma que el mensaje sería retransmitido a todos los procesos que no estén especificados en dicho formato.

- El elemento *Media:Type* identifica al tipo de flujo procesado por una aplicación. Los valores actualmente definidos por la propia especificación son '*audio'*, '*vídeo'*, '*workspace'* y '*control'*.

- El elemento *Module:Type* define una parte lógica de una aplicación. El valor '*ui'* designa al interface de usuario de una aplicación, el valor '*engine'* define al programa principal y '*transcoder'* a un medio transcodificador. La lista de posibles valores es ampliable.

- El elemento *App:Name* identifica la aplicación que se está utilizando (por ejemplo, *rat*, *vic*, etc).

- El elemento *Instance:Id* se utiliza para distinguir varias instancias de una misma aplicación (por si se ejecutan en la misma máquina). Se trata de un identificador único para cada instancia. Muchas aplicaciones Unix utilizan para este valor el número de identificación del proceso (*pid*), aunque esto no es obligatorio.

Como ejemplos de direcciones *mbus* podemos tener las cadenas de caracteres mostradas en la siguiente tabla:

EJEMPLO DE DIRECCIÓN	COMENTARIO
(media:audio module:ui app:rat Instance:124)	La *interface* de usuario del proceso *rat* con identificativo*124.*
(media:workspace module:ui)	La *interface* de usuario de todas las aplicaciones *workspace.*
(media:audio)	Todas las aplicaciones de *audio.*
(app:rat)	Todas los instancias de la aplicación *rat* que se estén ejecutando en la máquina.

Tabla 3.2. Ejemplos de direcciones *mbus*

Formato de los mensajes

Un *mensaje mbus* contiene dos partes: una cabecera y un cuerpo. La cabecera es utilizada para indicar cómo y dónde debe ser entregado el mensaje. El cuerpo proporciona información y comandos a la entidad o aplicación de destino.

- *Cabecera*

En la cabecera, entre otros datos, aparece la versión de *mbus* utilizada (en la Tesis es la '*mbus/1.0'*), un número de secuencia, una marca de tiempo o *timestamp* (número de segundos transcurridos desde las 00:00 horas del 1 de enero de 1970), el tipo de mensaje (fiable o no fiable), las direcciones de origen y de destino y, finalmente, un campo de reconocimiento, donde aparecen todos los números de secuencia de los mensajes que se desee reconocer con el mensaje.

- *Cuerpo del mensaje*

Los mensajes o comandos deben ser construidos utilizando nombres jerárquicos para agrupar comandos conceptualmente relacionados bajo una misma jerarquía. Se utiliza un punto '.' para la delimitación entre el prefijo de una jerarquía y su comando. Ya que el sistema de direccionamiento utilizado en *mbus* permite especificar direcciones incompletas omitiendo ciertos elementos de las mismas, para enviar comandos a grupos de aplicaciones, los nombres de los comandos no deben ser ambiguos de tal manera que sea posible interpretarlos o desecharlos sin considerar la dirección del mensaje. A continuación, se lista una serie de prefijos de comandos predefinidos, correspondientes a las principales jerarquías definidas[4] :

Prefijo	Descripción de la clase
mbus.	Comandos básicos generales de *mbus*
conf.	Comandos relacionados con el control de conferencias
rtp.	Comandos relacionados con RTP
audio.	Comandos específicos para las herramientas de audio
video.	Comandos específicos para las herramientas de vídeo
security.	Comandos relacionados con la seguridad
status.	Comandos para comunicar la información de estado, las condiciones de error, etc.

Tabla 3.3. Prefijos de comandos *mbus*

Además, también se pueden definir *comandos específicos de las aplicaciones* para permitir que cada aplicación conectada al *mbus* pueda definir un número de comandos privados. Dichos comandos empezarán con la secuencia *tool*, con el formato típico *<tool-name>*, como, por ejemplo, *tool.rat*.

- *Comandos básicos mbus*

En la tabla 3.4 se presentan una serie de comandos independientes de las aplicaciones.

[4] En [OTT99a], aparecen explicados los comandos más comunes de dichas jerarquías.

SINTAXIS	COMENTARIO
mbus.hello( )	Se envia de forma periódica al bus a partir del inicio para notificar la presencia en el mismo.
mbus.bye( )	Para anunciar que se va a abandonar el bus.
Mbus.quit( )	Para pedir a otras entidades que terminen su proceso y que se desconecten del *mbus*. Si lo hacen o no ya depende de las propias aplicaciones.

Tabla 3.4. Comandos independientes de las aplicaciones

- *Comandos relacionados con RTP*

En la tabla 3.5 se muestran los comandos utilizados para proporcionar información sobre una fuente en RTP.

SINTAXIS	COMENTARIO
rtp.ssrc( )	Envía el identificador SSRC que se utilizará en el resto de la sesión
rtp.source.exists (ssrc validityTime)	Enviado para asegurar que una fuente determinada está presente en la sesión. *ValidityTime* representa el tiempo durante el cual dicha fuente deberá ser considerada activa o presente en la sesión.
rtp.source.remove (ssrc)	Para indicar que la fuente ha abandonado la sesión.
rtp.source.name (ssrc name) *rtp.source.email (ssrc email)* *rtp.source.phone (ssrc phone)* *rtp.source.loc (ssrc loc)* *rtp.source.tool (ssrc tool)* *rtp.source.cname (ssrc cname)*	Comandos para enviar información contenida en el paquete RTCP SDES desde el programa principal al interface gráfico, cuando se trata de dos programas independientes, comunicados vía *mbus*.
rtp.source.reception (ssrc packetsRecv packetsLost packetsMisordered jitter validityTime)	Envia la información contenida en el informe RR, desde un programa a su interface gráfica. Parámetros: número total de paquetes recibidos, perdidos y desordenados, *jitter* y el *validityTime*.
rtp.source.packet.loss (dest_ssrc src_ssrc loss% validityTime)	Indica la pérdida instantánea de paquetes observada entre dos fuentes.
rtp.source.active (ssrc validityTime)	Indica que una fuente está transmitiendo datos dentro de la sesión.
rtp.source.inactive (ssrc)	Indica que la fuente ha dejado de transmitir.
rtp.source.mute (ssrc muteState)	Indica, mediante el parámetro *muteState* si la fuente está en modo '*mute*' (valor '1') o no ('0').
rtp.source.packet.duration (ssrc packetDuration):	Indica la duración, en milisegundos, de los paquetes recibidos desde una fuente.
rtp.source.codec (ssrc codec)	Indica codificación de datos que utiliza la fuente.
rtp.source.playout (ssrc playoutDelay)	Indica el retardo de reproducción (*playout delay*), para una determinada fuente.

Tabla 3.5. Comandos relacionados con RTP

El último comando de la tabla, *rtp.source.playout (ssrc playoutDelay), es el que se ha utilizado en la presente Tesis para intercambiar los retardos de reproducción entre las aplicaciones* y, así, conseguir la sincronización en la reproducción de los diferentes flujos en un mismo receptor (inter-flujo). El proceso se explica en el capítulo siguiente.

Transporte

Todos los mensajes *mbus* se transmiten como mensajes UDP, pudiendo ser enviados de dos formas distintas:

1.- De forma *multicast* o broadcast localmente, con una dirección y puerto asignados por la IANA dentro de los rangos propuestos por la RFC 2365 ([MEY98]). Esta dirección y puerto serán utilizados por todas las entidades que usen el *mbus*. Mientras la IANA no elija un rango de direcciones y un puerto, se deberá utilizar la dirección *multicast* 239.255.255.247 y el puerto 47000 (en base decimal).

2.- De forma *unicast* (via UDP) al puerto de una entidad de una determinada aplicación. Será utilizada por implementaciones en las que se quiera entregar mensajes *mbus* directamente a una única entidad de una aplicación y no a todas las aplicaciones conectadas al bus.

Tal como se ha comentado anteriormente, para que los mensajes no lleguen a otros equipos de la red, en los dos casos se utilizará un valor de TTL igual a 0, para que los mensajes *mbus* nunca salgan a la red y siempre serán procesados únicamente por las aplicaciones que estén conectadas al bus dentro de la propia estación.

Fiabilidad y privacidad

Los mensajes intercambiados entre aplicaciones dentro de una conferencia, pueden tener diferentes requerimientos en cuanto a fiabilidad. A través de *mbus* se puede enviar tanto mensajes fiables como no fiables. Algunos mensajes tendrán un carácter más bien informativo, conteniendo información de estado de las aplicaciones (como, por ejemplo, el estado del volumen del audio), que será actualizada constantemente. Otros mensajes (como peticiones de ciertos parámetros o peticiones a servidores) pueden requerir una respuesta del proceso de destino proporcionando un reconocimiento explícito. También existirán mensajes que modifiquen el estado de las aplicaciones o de la propia conferencia y que es crucial que no se pierdan. Este último tipo de mensajes tienen que ser enviados de forma fiable a los receptores, mientras que los mensajes del primer tipo no requieren de ningún tipo de mecanismo de fiabilidad en el nivel de transporte de *mbus*.

Por otro lado, *mbus* permite la utilización de técnicas de encriptación para asegurar la privacidad en el intercambio de mensajes y, así, evitar problemas con usuarios maliciosos o incluso con otros usuarios que también estén utilizando herramientas de conferencia incluso en la misma máquina.

Conocimiento de la existencia de otras aplicaciones

Para que las aplicaciones conectadas utilizando *mbus* puedan conectarse unas a otras, previamente será necesario que cada una sepa qué otras aplicaciones están presentes. Del mismo modo, cuando se conecte una aplicación, ésta deberá saber qué aplicaciones están conectadas y éstas deberán detectar su presencia en el bus. Además, cada aplicación debe saber en todo momento si otra aplicación sigue en el bus o lo ha abandonado (por cualquier motivo: decisión propia, fallo, etc.).

Cada aplicación debe anunciar su presencia en el bus, después de arrancar y deberá estar constantemente comunicando su presencia, independientemente del orden de conexión al bus. Del mismo modo, cuando abandone el bus, deberá comunicarlo antes mediante el mensaje correspondiente.

Los mensajes utilizados para anunciar su presencia y para anunciar su abandono son *mbus.hello* y *mbus.bye*, respectivamente. Cada aplicación enviará mensajes *mbus.hello* periódicamente para avisar a las demás aplicaciones de su presencia en el bus, y enviará el mensaje *mbus.bye* cuando decida abandonar el bus. El intervalo de transmisión periódica de mensajes *mbus.hello*, es dinámico dependiendo del número de aplicaciones conectadas al bus (para evitar congestión en el mismo y favorecer la escalabilidad) y se calcula de acuerdo a lo especificado en [OTT00]. Si una aplicación deja de recibir mensajes *mbus.hello* de otra aplicación durante un cierto tiempo puede considerar que dicha aplicación ha abandonado el bus por cualquier motivo.

3.4. Aplicaciones Multimedia

Como se ha comentado en el capítulo anterior, existen multitud de herramientas que utilizan RTP. En la Tesis se han empleado las herramientas *vic*, para la transmisión de vídeo, y *rat*, para la transmisión de audio, debido, principalmente, a que se ha podido obtener el código fuente a través de Internet, y a que, tras su estudio y comprensión, ha podido ser modificado para implementar el algoritmo propuesto. Otros investigadores también han utilizado estas herramientas para implementar sus propuestas, como es el caso de [BIE96], [KOU96], [MAR96], [MAR96a], [CHE96], [RAM97], [BOL98], [SHI98], [SHI98a], [TAS98b], [DIO99], [SHI99], [GON00], [TAS00], [KUO01], [ISH02]. Todos ellos han utilizado RTP/RTCP para la transmisión de audio y vídeo en tiempo real. Los algoritmos RTP no se han implementado como un nivel separado sino que forman

parte del propio código de las aplicaciones. El código fuente de estas herramientas está disponible en Internet para diferentes sistemas operativos. Además, ambas fueron modificadas en algunas de sus versiones para incluirlas en el proyecto *ReLaTe* (*REmote Language Teaching*) para conseguir sincronización fina entre ambos flujos de datos.

En nuestro caso, como no se disponía de ninguna aplicación de transmisión de datos sobre RTP/RTCP, como parte del trabajo de la presente Tesis, se ha desarrollado una aplicación de transmisión de texto sobre RTP, que será descrita en el capítulo 5.

A continuación se describen brevemente las dos aplicaciones utilizadas, *vic* y *rat*. Para más información se puede consultar en Internet la documentación y guías de usuario asociadas a las mismas[(5)].

3.4.1. Vic[(6)]

Vic ([CAN95]) es una herramienta para transmisión de vídeo sobre RTP/RTCP, que mejoraba las herramientas *nv* e *ivs*. La interface gráfica está desarrollada en TCL/TK, sin embargo, el resto de la aplicación, es decir, el núcleo de la aplicación, está desarrollado en C++. En la figura 3.24 se muestra la interface gráfica de la versión 2.8, disponible en Internet.

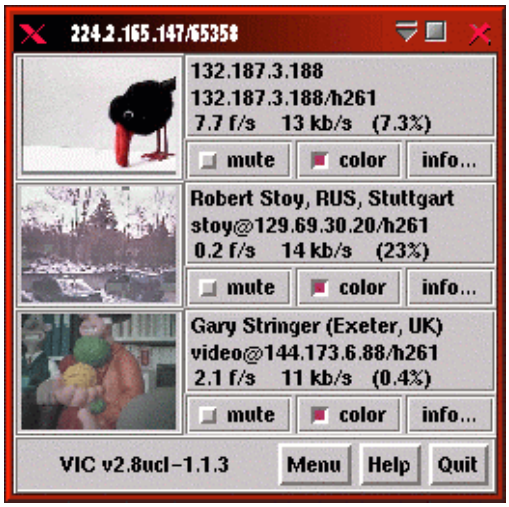


Figura 3.24. Ejemplo de pantalla de la herramienta *vic*

[5] http://www-mice.cs.ucl.ac.uk/multimedia/software/documentation/
[6] http://www-mice.cs.ucl.ac.uk/multimedia/software/vic/

Las principales características de esta herramienta son la independencia del nivel de red empleado, el soporte de *codecs* basados en hardware (*hardware-based codecs*), un modelo de conferencia coordinado, un interfaz de usuario extensible y el soporte de diversos algoritmos de compresión. Esta herramienta fue concebida inicialmente como una aplicación para evaluar los protocolos TENET de red de tiempo real y soportar simultáneamente la arquitectura de sesiones utilizada en la red Mbone.

Una de las características que ha influido en la elección de esta herramienta, para implementar el algoritmo propuesto, ha sido que ha ido adaptándose a la evolución del protocolo RTP, incluyendo cambios y mejoras según se iban desarrollando.

En las primeras versiones, la herramienta no disponía de un algoritmo de adaptación del punto de reproducción y las tramas de vídeo eran reproducidas tan pronto como se recibía un paquete con una marca de final de trama. Ya que se toleran pequeñas variaciones del *jitter*, este problema no era severo pero descartaba la posibilidad de implementar sincronización inter-flujo.

Vic se puede utilizar, tanto de forma *multicast* como *unicast*. Para una red local, por ejemplo, se podrá utilizar cualquier dirección *multicast* de la clase D que no esté reservada. El formato para ejecutar *vic* desde la línea de comandos es el siguiente:

vic [opciones] dirección/puerto

El número del puerto no podrá ser menor de 1024. Se utilizarán dos puertos, el especificado y otro posterior (para RTP y RTCP, respectivamente). Todos los usuarios de la sesión *multicast* deberán utilizar la misma dirección *multicast* y puerto para poderse comunicar mutuamente. En la siguiente figura se muestra un ejemplo de ejecución de la aplicación *vic* con un TTL de 16 a una dirección *multicast* 224.2.2.60 con el puerto 5008.

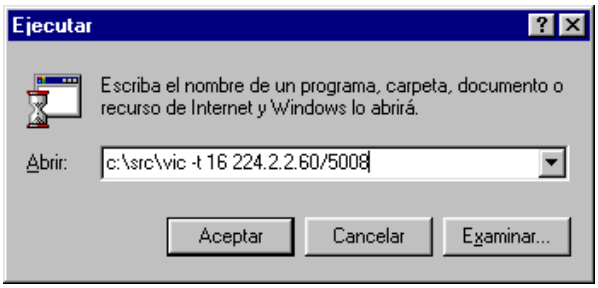


Figura 3.25. Ejemplo de ejecución de *vic*

En el ejemplo anterior, se utiliza la opción '-t' para especificar el TTL del datagrama IP. Por defecto, *vic* utiliza un TTL de 16 y el valor máximo permitido es de 255.

La aplicación permite, entre otras muchas cosas, seleccionar diferentes formatos de codificación para la transmisión de vídeo (*nv*, JPEG, H.261, H.263, etc.), la tasa de envío (tramas por segundo), el nivel de calidad, el dispositivo hardware utilizado y el puerto.

3.4.2. Rat[7]

La herramienta *rat* (*Robust Audio Tool*) ([HAR95], [HAR96] y [KOU96]) es una aplicación de conferencia de audio de código abierto, que permite a los usuarios participar en conferencias de audio a través de Internet. Fue desarrollada y está siendo continuamente mejorada por el *Computer Science Department*, del *UCL*. Nació como resultado de una serie de experiencias obtenidas de aplicaciones para actividades piloto realizadas en proyectos de investigación como MICE y RELATE, comentados en el capítulo anterior. Está basada en herramientas de audio, tales como *vat* ([JAC92]), pero incluye nuevas técnicas desarrolladas en UCL que mejoran la calidad del flujo de audio. Está basada en estándares del IETF, utilizando RTP sobre UDP, como protocolo de transporte, y sigue el perfil definido en la RFC 1890 ([SCH96a]). La versión 4.2 está disponible para un gran abanico de plataformas: FreeBSD, IRIX, Linux, Solaris y Windows. Ofrece muchas ventajas frente a otras herramientas de audio como las detalladas en el capítulo anterior, ya que proporciona protección frente a pérdidas de paquetes y un mecanismo que se ocupa de la falta de facilidades de tiempo real.

Se puede utilizar para conferencias de audio, tanto para conferencias punto a punto (*unicast*), uniendo tan solo a dos equipos de una misma red, como para conferencias multipunto (*multicast*), donde muchos participantes de diversos lugares pueden unirse a la conferencia mediante una red capaz de trabajar de forma *multicast*.

Contiene un gran abanico de codecs de diferentes tasas y calidades, recuperación frente a pérdidas de paquetes basado en el receptor y codificación de canal basada en el emisor con la posibilidad de incluir redundancia. Ofrece una buena calidad de sonido dependiendo de las condiciones de la red. Además, permite encriptación para conversaciones privadas. Las codificaciones de audio permitidas son: G.711 PCM μ-Law (64kb/s), G.711 PCM A-Law (64kb/s), Wide-Band ADPCM (64kb/s), G.726 ADPCM (16-40kb/s), DVI ADPCM (32kb/s), Variate Rate DVI ADPCM (~32kb/s), Full Rate GSM (13kb/s) y LPC (5.6kb/s).

[7] http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/

En un principio fue desarrollada para abordar problemas de inteligibilidad del habla, que provienen del uso de redes compartidas de bajo coste (*shared low-cost networks*) y de hardware de propósito general. Posteriormente fue mejorada para soportar efectos 3D requiriendo mayor ancho de banda para transmitir el audio.

Esta herramienta contiene un algoritmo denominado *cushion* que se ocupa de la falta de propiedades de tiempo real de los equipos empleados en las audioconferencias sobre Internet. El algoritmo estima la carga de la estación en cada momento y utiliza *buffers* del driver del dispositivo para minimizar los huecos o *gaps* producidos en el audio, minimizando el retardo. Los resultados del algoritmo muestran un descenso en el número y longitud de los *gaps* y un descenso en la media y varianza del retardo extremo a extremo percibido con respecto a otras herramientas de audio existentes ([HAR96]).

Hay que destacar que *rat* es sólo una aplicación de audio, no realiza servicios de llamada ni escucha a los anuncios de sesiones *multicast* en red. Para ello, se debe utilizar en combinación con herramientas como el *SDR*, comentrada en el capítulo anterior.

Como en el caso del *vic*, la interface gráfica también está desarrollada en TCL/TK, sin embargo, el resto de la aplicación, es decir, el núcleo de la aplicación está desarrollado en 'C'.

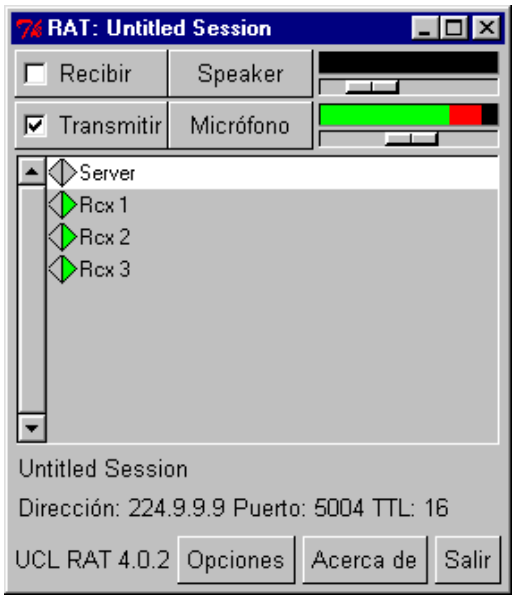


Figura 3.26. Ejemplo de pantalla de la herramienta *rat*

El formato para ejecutar *rat* desde la línea de comandos es el mismo que para *vic* (*rat [opciones] dirección/puerto*). En el siguiente ejemplo, se ejecuta *rat* con un TTL de 16 a la dirección *multicast* 224.2.2.60 y el puerto 5004.

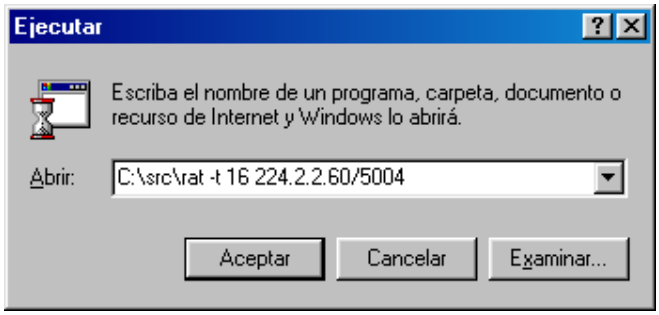


Figura 3.27. Ejemplo de ejecución de *rat*

3.5. Conclusiones

En este capítulo se ha presentado en su primer punto un conjunto de definiciones y clasificaciones relacionadas con los *Sistemas Multimedia* y los *Protocolos de Sincronización*. Dichos protocolos deben garantizar la *sincronización intra-flujo* (referida a la reproducción de cada unidad de datos o LDU que forma cada flujo de información en los instantes de tiempo correspondientes indicados en sus relaciones temporales como, por ejemplo, en el flujo de audio o el de vídeo) y una *sincronización inter-flujo* (es decir, una sincronización entre los diferentes flujos de información, como por ejemplo, la sincronización labial, o *lip-sync*, entre el audio y el movimiento de los labios de la persona que esté hablando en una videoconferencia). Además, se ha presentado el concepto de *sincronización de grupo de flujos* en diferentes receptores, también conocida como *sincronización inter-destinatario*, que supone la representación de uno o varios de los flujos involucrados en una aplicación multimedia, al mismo tiempo, es decir, de forma simultánea, en todos los receptores de dicha aplicación.

En el siguiente punto, se han señalado los *principales requerimientos de sincronización* en las aplicaciones multimedia en cuanto a *calidad de servicio*, señalando al *retardo*, el *jitter* y la *desviación en las tasas de consumo* de cada reproductor como las principales causas que generan asincronía en un sistema distribuido. Considerando que cada flujo de información utiliza una conexión diferente, el retardo sufrido por cada flujo será distinto al del resto de flujos. Además, si la red no garantiza un valor máximo del *jitter* se podrán producir situaciones de *overflow* y *underflow* en los *buffers* de los receptores. Por otro lado, la inevitable desviación entre los procesos de reproducción provocará la aparición de asincronía entre los diferentes flujos de información dependientes del tiempo

que deberá corregirse para que no sea perceptible por el usuario. Se finaliza el punto indicando una serie de valores de asincronía máxima permitida entre diferentes tipos de flujos, extraídas de experimentos realizados por Steinmetz y Blakowski, recogidas en [STE96] y [BLA96].

Posteriormente, se han descrito los protocolos y aplicaciones utilizadas en la Tesis. En el tercer punto se han presentado los protocolos utilizados en la misma para la implementación del algoritmo propuesto que permite garantizar la sincronización de grupo de flujos multimedia, como son NTP y RTP/RTCP. El primero es el protocolo propuesto para la sincronización de relojes en Internet, mientras que RTP/RTCP han sido propuestos para la transmisión de flujos en tiempo real. Se ha incluido una descripción breve de estos últimos, ya que, en la propuesta realizada en la Tesis, se han modificado las estructuras o formato de algunos de sus paquetes para la implementación del algoritmo y, además, se aprovecha de las marcas temporales utilizadas por ambos y de la información de realimentación proporcionada por los mensajes de informe RTCP.

También se ha explicado el bus de conferencia local denominado *mbus* que se ha utilizado para el intercambio de mensajes entre aplicaciones en los receptores para conseguir la sincronización inter-flujo.

Por último, se han descrito las dos herramientas MBone, *rat* y *vic*, que se basan en la utilización de RTP/RTCP para la transmisión de audio y vídeo, respectivamente, cuyo código abierto ha sido modificado para incluir el algoritmo desarrollado en la Tesis.

Capítulo 4

Algoritmo de Sincronización de Grupo basado en los Protocolos RTP/RTCP y NTP

4.1. Introducción

En el presente trabajo, como contribución más importante, se ha desarrollado un *Algoritmo de Sincronización de Grupo de Flujos Multimedia, basado en los protocolos RTP/RTCP ([SCH96]) y NTP ([MIL91]), que proporciona sincronización entre flujos de información, tanto dependientes como independientes del tiempo, en un sistema distribuido, en redes deterministas y/o estadísticas (tomando como referencia los protocolos Feedback ([RAN92], [RAM92], [RAM93], [RAM93b], [RAN95a] y [RAN95b]) y Feedback Global ([GUE97])).*

Los objetivos del mismo han sido los siguientes:

- Garantizar la sincronización de los *instantes de inicio de la reproducción*, la *sincronización gruesa* y la *sincronización fina*, entre

flujos de información *dependientes* e *independientes del tiempo*, dentro de los límites establecidos en [STE96].

- Utilizar los protocolos RTP/RTCP para implementar el algoritmo propuesto.

- Utilizar las ideas, en cuanto a la implementación de las acciones de resincronización, empleadas por la mayoría de los protocolos de sincronización multimedia, basadas en operaciones sencillas de '*salto'* y '*espera'*.

- Mantener una baja carga de información de control y mensajes dedicados a la sincronización en comparación con el número total de LDUs transmitidas.

En cuanto a la arquitectura del entorno, donde típicamente se puede aplicar el nuevo algoritmo propuesto, para obtener una sincronización entre flujos de información, se tratará de cualquier sistema multimedia distribuido geográficamente, basado en una red con una o varias fuentes o servidores de flujos multimedia y uno o varios receptores reproductores. La estructura física del sistema se muestra en la figura 4.1.

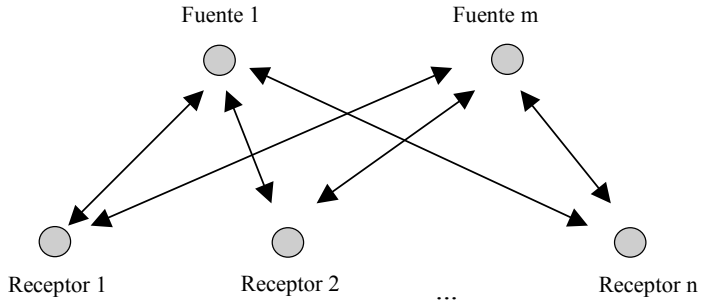


Figura 4.1. Arquitectura de funcionamiento

En cuanto a la red que permite la interconexión entre las fuentes y los receptores, puede estar constituida por una red LAN, LAN/MAN o WAN. En cualquiera de los casos, entre las características que las definen, nos interesará conocer el retardo que se introduce en la transmisión de los paquetes o, por lo menos, los límites máximo y mínimo de dicho retardo, y el *jitter* asociado. Será importante conocer si la red garantiza un retardo máximo y que, en cualquier caso, dicho retardo no supere valores del orden de cientos de milisegundos para que la red sea capaz de soportar aplicaciones multimedia en tiempo real, bien a través de las propias características de la red (como, por ejemplo, en una red ATM), bien mediante la reserva de recursos (por ejemplo, a través de protocolos como RSVP).

La función de las fuentes consiste en capturar, en tiempo real, la información de cada flujo multimedia mediante diferentes dispositivos (por ejemplo, un micrófono con la correspondiente tarjeta de audio, una cámara con su tarjeta capturadora de video, el teclado, etc.) y transmitir esa información, según diferentes estándares de compresión y codificación, encapsulada en paquetes RTP. Las fuentes, en general, pueden estar implementadas por un PC de elevadas prestaciones o estación de trabajo con las configuraciones adecuadas de entrada/salida.

Por otra parte, tal como muestra la figura 4.2, los receptores estarán conectados a la red y, mediante el algoritmo de sincronización propuesto, se conseguirá que reproduzcan la información multimedia (vídeo, audio, imágenes, texto, etc.) de forma sincronizada no solo localmente (*sincronización inter-flujo*) sino también entre receptores (*sincronización de grupo*). Es conveniente que las tareas que realicen estos dispositivos sean lo más simples y sencillas posibles para no tener que incrementar las prestaciones y la complejidad asociada.

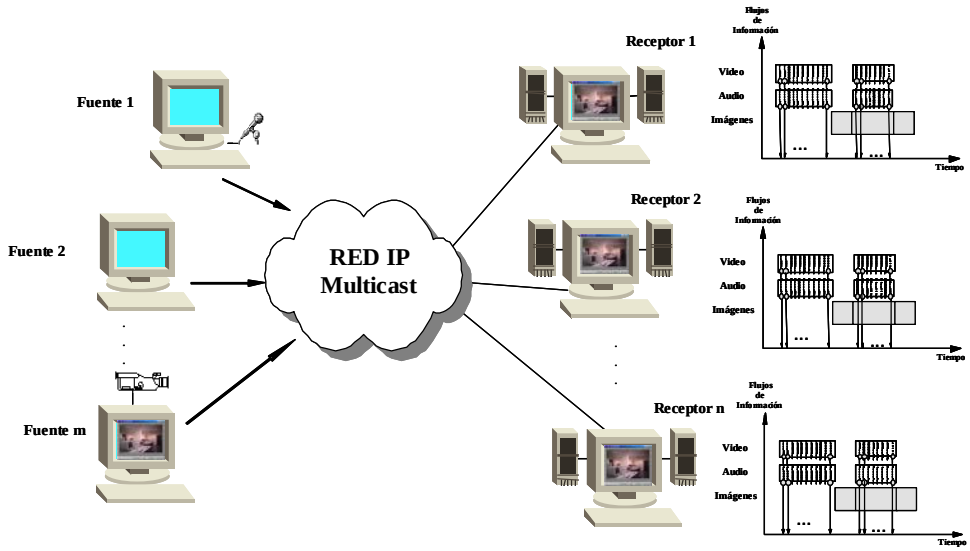


Figura 4.2. Arquitectura de funcionamiento

El protocolo *Feedback* original ([RAN92], [RAM92], [RAM93], [RAM93b], [RAN95a] y [RAN95b]) permitía utilizar dispositivos receptores sencillos (llamados mediáfonos). Sin embargo, el algoritmo propuesto, requiere que los dispositivos reproductores realicen alguna tarea adicional pero, en cualquier caso, también se puede implementar con dispositivos sencillos, con el hardware necesario (por ejemplo, tarjeta de sonido con los correspondientes

altavoces, el monitor o pantalla, etc.), manteniendo la simplicidad del protocolo original.

Según el modelo de referencia de [BLA96], el algoritmo de sincronización propuesto resulta apropiado para un sistema multimedia distribuido, donde se realiza una transmisión *multicast* (si la red lo permite) de flujos individuales no multiplexados desde varias fuentes independientes hacia diferentes receptores, a través de una red determinista o con una cierta calidad de servicio garantizada, donde los retardos máximos sean limitados y conocidos a priori. La especificación de la sincronización se hace mediante marcas de tiempo en los paquetes RTP en el instante de la captura y el cálculo de la planificación de la reproducción sincronizada se realiza en tiempo de ejecución. Por otro lado, la sincronización está basada en el eje de tiempos NTP (*especificación basada en ejes*) y para realizar la sincronización inter-flujo y de grupo se utiliza un doble mecanismo *maestro/esclavo*.

4.2. Descripción del algoritmo propuesto

A continuación, se presenta, de forma descriptiva, el algoritmo desarrollado en la Tesis para garantizar la sincronización deseada entre los diferentes flujos de información.

Tal como se ha comentado en la introducción, este algoritmo se implementará haciendo uso de los protocolos RTP y RTCP, definidos en la RFC 1889 ([SCH96]).

El algoritmo propuesto incluye tres procesos de sincronización:

1. *Sincronización Intra-flujo*: el algoritmo supone una sincronización intra-flujo, que garantiza un proceso de reproducción continua y correcta de cada uno de los flujos multimedia, sea cual sea su naturaleza.

2. *Sincronización de Grupo*: este proceso supone dos partes: una *parte inicial*, en la que todos los receptores deberán iniciar la reproducción del *flujo maestro* en el mismo instante (*Instante Inicial de Consumo*); y, a partir de dicho instante, se comienza una *segunda parte* en la que todos ellos deberán reproducir dicho flujo, al mismo tiempo (es decir, de forma sincronizada) y con la mayor continuidad posible.

3. *Sincronización Inter-Flujo*: incluye también un proceso de *Sincronización Local* en cada receptor entre todos los procesos de reproducción de aquellos flujos que reciba dicho receptor. Este proceso también es conocido como *Sincronización fina entre flujos multimedia* en cada uno de

los receptores. Los procesos reproductores de los flujos considerados *esclavos* se sincronizarán al proceso reproductor del flujo considerado como *maestro* (involucrado en el proceso de sincronización de grupo). Este proceso se encargará de mantener las relaciones temporales (que existían entre los flujos en el momento de su captura) en el momento de su reproducción en cada receptor.

El funcionamiento se muestra en la figura 4.3, pero como resulta un tanto compleja de entender, se ha desglosado en varias figuras (figuras 4.4 a 4.7), que proporcionarán un mejor entendimiento del mismo.

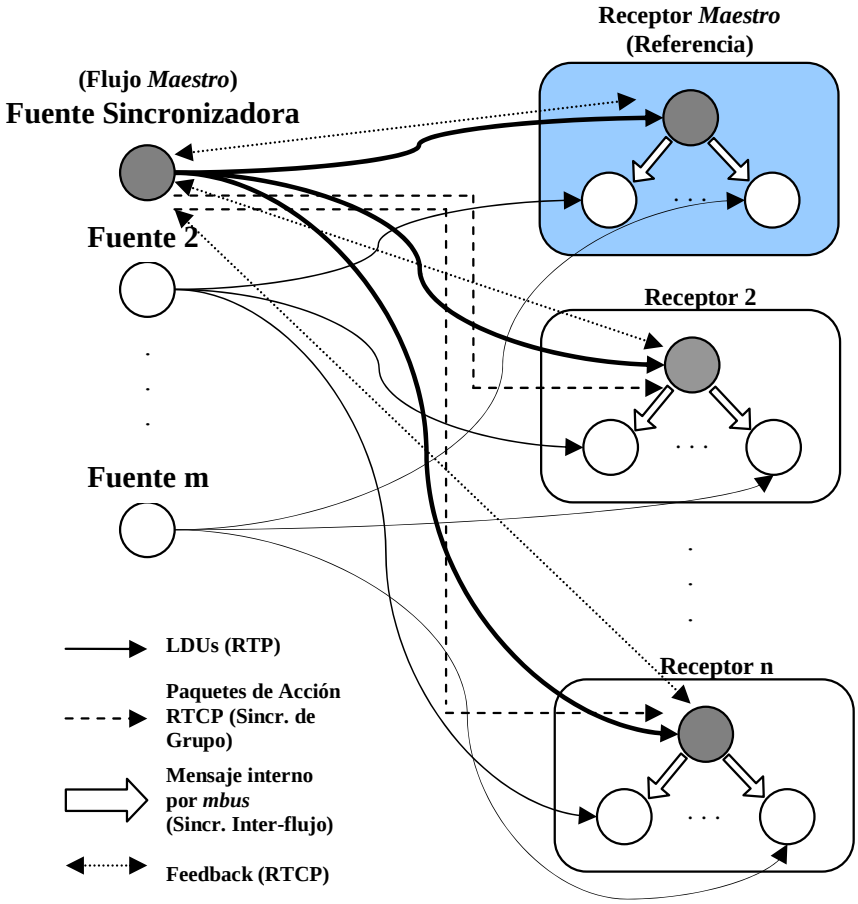


Figura 4.3. Flujos de información de datos y de control del algoritmo

En la figura 4.4, se puede apreciar la existencia de una transmisión, que puede ser *multicast* o *unicast*, de flujos multimedia mediante RTP desde una o varias fuentes transmisoras a uno o varios receptores. Uno de los flujos multimedia

es tomado como flujo *maestro* (líneas y flechas de mayor grosor) y, además, de entre todos los receptores se selecciona uno de ellos como *receptor maestro* (azul en la figura), cuyo estado de reproducción del flujo *maestro* será tomado como referencia para determinar el estado de reproducción de cada uno de los demás receptores (*esclavos*). Este *receptor maestro* podrá ser elegido de varias maneras, según determinados criterios, en el algoritmo propuesto. Se utilizará RTCP para enviar mensajes de control durante la sesión.

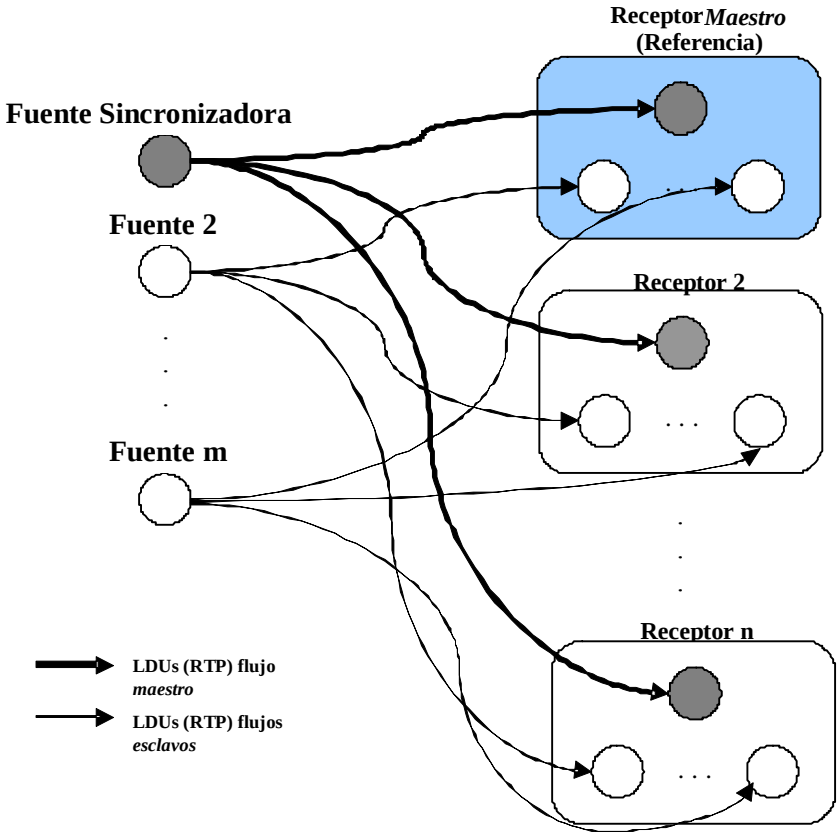


Figura 4.4. Transmisión de flujos mediante RTP

Aprovechando los mensajes de realimentación o *feedback* del protocolo RTCP, el algoritmo propuesto, que se ejecutará en la fuente transmisora del flujo *maestro* (fuente sincronizadora), hará uso de los paquetes RTCP RR (modificados para incluir datos necesarios para ejecutar el algoritmo), para determinar el estado de reproducción del flujo *maestro* en todos los receptores. De esta manera la fuente sincronizadora podrá conocer el estado de los procesos de reproducción de los mismos en todo momento. Este proceso se muestra en la figura 4.5.

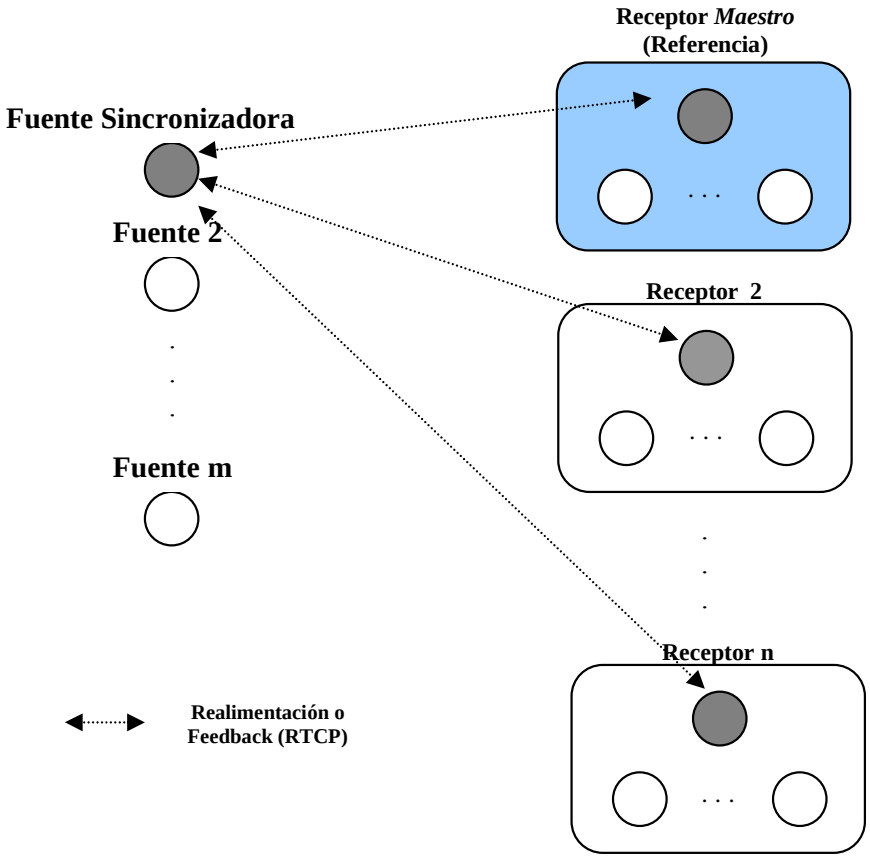


Figura 4.5. Mensajes de realimentación correspondientes al flujo *maestro* en cada receptor

La fuente sincronizadora, en caso de detectar desviaciones (receptores adelantados o retrasados en la reproducción del flujo *maestro*, con respecto al receptor *maestro*), por encima de un valor umbral, generará mensajes de '*acción*' para que los procesos de los receptores ajusten la reproducción del flujo *maestro* mediante '*saltos'* o '*pausas'*. De esta manera se consigue que todos los receptores estén reproduciendo el flujo *maestro* de forma sincronizada (*Sincronización de Grupo entre receptores*). El proceso se representa en la figura 4.6.

Una vez conseguida la sincronización de grupo entre receptores para el flujo *maestro*, también se persigue la obtención de una *Sincronización Inter-flujo Local* (como, por ejemplo, la sincronización labial o *lip-sync*). Para ello, se hará uso del canal de comunicaciones interno proporcionado por *mbus*. En cada receptor, y de forma local, el proceso reproductor del flujo *maestro* (que estará sincronizado en todos los receptores mediante el proceso anterior) enviará información sobre su estado de reproducción en cada momento a los procesos

reproductores locales de los demás flujos *esclavos*, para que éstos sincronicen su reproducción mediante *'saltos'* y *'pausas'*. Este proceso se representa en la figura 4.7.

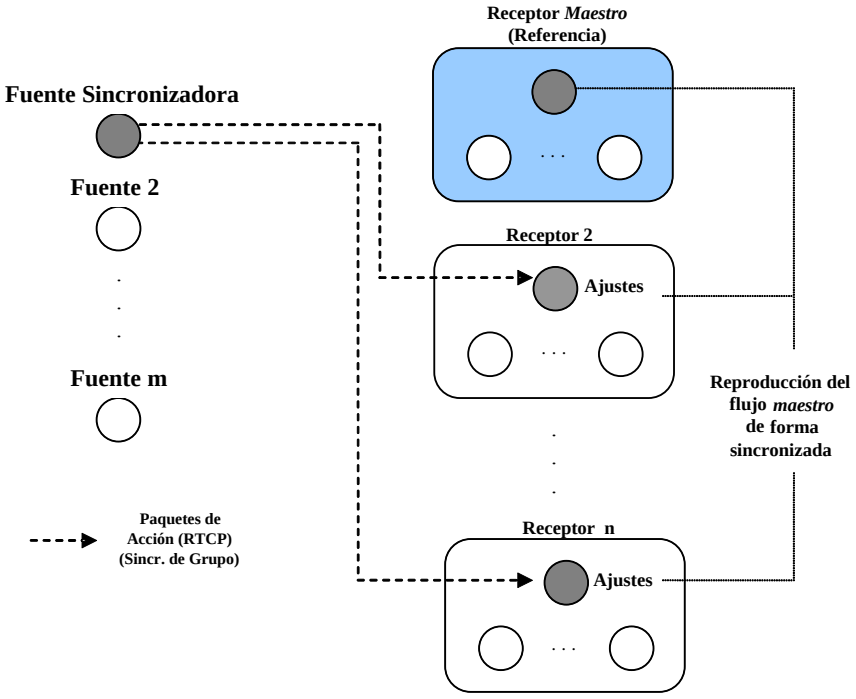


Figura 4.6. Mensajes RTCP de acción enviados por el transmisor del flujo *maestro* a los *receptores esclavos*

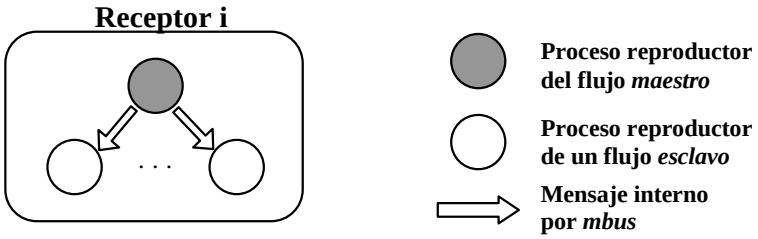


Figura 4.7. Sincronización Local inter-flujo a través del bus interno *mbus*

A continuación se detallan los tres procesos de sincronización que forman parte del algoritmo propuesto.

4.2.1. Sincronización Intra-flujo (local, un flujo)

Para conseguir una reproducción continua en los receptores se deberá garantizar que no se produzca *underflow* ni *overflow* de los *buffers* de recepción, que son las principales causas de las discontinuidades en los procesos de reproducción. A continuación se va a definir, haciendo uso de dos teoremas, el tamaño mínimo y máximo del *buffer* de recepción para evitar ambas situaciones, teniendo en cuenta que se parte de un conocimiento previo del retardo máximo y mínimo que van a sufrir las LDUs en su trayecto desde la fuente hasta los receptores.

Para evitar que se produzcan *underflow* en los *buffers* de recepción, en nuestro entorno de trabajo supondremos que se cumplirá el siguiente teorema, cuya demostración se puede encontrar en [RAM95b]

Teorema1: *"Para garantizar la reproducción continua de un flujo multimedia en un receptor, la fuente multimedia debe transmitir la LDU n-ésima del flujo hacia el receptor en un instante de tiempo anterior a (n-1) * θ * (1 - ρ) * (Ret_{max} – Ret_{min}) después de la transmisión de la primera LDU hacia ese receptor"*, donde θ es el periodo nominal de generación (en la fuente) o reproducción (en el receptor) de la LDU, ρ es la desviación o *drift* del periodo de reproducción de una LDU, y Ret_{max} y Ret_{min} son límites máximo y mínimo del retardo que puede experimentar la LDU desde la fuente al receptor.

Se puede asegurar una reproducción continua, sin que se produzca *underflow* del *buffer*, si el *jitter* de la red no excede el periodo de reproducción menor (es decir, [θ * (1-ρ)]), o bien si antes de iniciar la reproducción se han almacenado un número *P* de LDUs cuyo tiempo de reproducción sea mayor o igual al *jitter* de la red.

El mínimo número de LDUs prealmacenadas (*P*) en los *buffers* de recepción de cada receptor necesario para garantizar una reproducción continua será de:

$$P = \left\lceil \frac{(\mathrm{Re}t_{\max} - \mathrm{Re}t_{\min})}{\theta * (1-\rho)} \right\rceil \quad LDUs \qquad \text{[Ec. 4.1]}$$

En nuestro caso, se forzará en los receptores un instante inicial de consumo que asegure que se almacenen suficientes LDUs en el *buffer* de recepción para que no haya discontinuidades en la reproducción.

En todos estos cálculos se ha asumido el peor caso, en el que el retardo experimentado por cada LDU transmitida después de las prealmacenadas será tan

grande como Ret_{max}, y que el periodo de reproducción de cada LDU puede ser tan pequeño como $\theta * (1-\rho)$, ya que éstas representan las restricciones más críticas para garantizar continuidad. En la práctica, los retardos en la red sufridos por las LDUs pueden ser mucho más pequeños que Ret_{max} y, además, la variación en las tasas de reproducción pueden producir que los periodos de reproducción de las LDUs sean mayores que $\theta * (1-\rho)$. Ambas anomalías llevarán a la acumulación de LDUs en los receptores y, por tanto, también serán necesarias técnicas de almacenamiento o *buffering* que sirvan para contrarrestar las variaciones en el retardo y las tasas de reproducción.

Mientras que el mínimo número de LDUs que se necesita que estén almacenadas en el *buffer* es igual a las calculadas como prealmacenadas (*P*, calculado en la ecuación 4.1), el máximo número de LDUs almacenadas requerido para preservar la continuidad será el calculado por el siguiente teorema ([RAM95b]):

Teorema 2: "*La capacidad del buffer requerida en un receptor para garantizar la continuidad de la reproducción de un flujo de datos de n LDUs viene dada por la expresión mostrada en la ecuación 4.2"*

$$B = \left\lceil \frac{2*\left(\mathrm{Re}t_{\max} - \mathrm{Re}t_{\min}\right) + 2*\theta*\rho*(n-1)}{\theta*(1+\rho)} \right\rceil \quad LDUs \qquad \text{[Ec. 4.2]}$$

B representa la capacidad del *buffer* (en número de LDUs) requerida para que al llegar la LDU n-ésima al receptor no se pierda debido a que éste esté lleno (*overflow* del mismo). Si la capacidad disponible en el receptor es inferior a *B*, el *buffer* puede sufrir *overflow* llevando a una consiguiente pérdida de LDUs y a una discontinuidad en la reproducción.

En la ecuación 4.2, el primer término ([$2*(Ret_{max} - Ret_{min}) / (\theta * (1+\rho)$]) representa la capacidad del *buffer* requerida para evitar el *jitter* en los retardos de red y es independiente del tamaño del flujo multimedia (n número de LDUs). El segundo término ([$2 * \theta * \rho * (n-1) / (\theta * (1+\rho)$]) representa la capacidad necesaria para evitar las variaciones en la tasa de consumo en el receptor y, como se puede apreciar, depende del tamaño del flujo de datos (de *n*), lo cual es indeseable en la práctica.

De lo anterior se deduce que, para garantizar la continuidad de la reproducción de los flujos de datos, se tendrá que utilizar algún mecanismo para que no se produzcan *overflows* o *underflows* de los *buffers* de recepción de los reproductores. En [RAM95b] se presenta una técnica mediante la cual la fuente multimedia, a partir de la recepción de mensajes *feedback* periódicos que los receptores le envían, estima los tiempos de reproducción de las LDUs en los

receptores y adapta su tasa de transmisión (eliminando o duplicando LDUs) evitando, así, *overflows* y/o *underflows* de los *buffers* en cada receptor. En nuestro caso, las acciones las va a realizar el receptor a partir de información enviada por la fuente sincronizadora, para obtener la sincronización.

En adelante, supondremos que la sincronización intra-flujo y, por tanto, la continuidad en la reproducción de cada flujo, está asegurada mediante técnicas de *buffering* como las propuestas en [RAM95b]. El algoritmo propuesto se ocupará sólo de asegurar, por un lado, la *sincronización de grupo* entre receptores y, por otro lado, la *sincronización inter-flujo* en cada receptor.

4.2.2. Sincronización de Grupo (distribuida)

Para conseguir que los diferentes flujos multimedia se reproduzcan a la vez, es decir, sincronizados en los diferentes receptores, es importante que exista algún mecanismo que corrija las desviaciones de unos con respecto a otros, debidas a las diferentes causas mencionadas en el capítulo anterior. Cuando las tasas de reproducción de los receptores están perfectamente ajustadas y los retardos de la red sufridos por las LDUs entre las fuentes y los receptores son deterministicos, la sincronización multimedia entre receptores puede ser garantizada si, primero, la fuente les indica cuándo deben iniciar la reproducción y, a continuación, les envía las LDUs a una tasa constante o con marcas de tiempo (*timestamps*) que indiquen cuándo se deberán reproducir. Sin embargo, debido, principalmente, al *jitter* y a los desajustes en las tasas de reproducción, los reproductores pueden perder la sincronía momentos después de iniciar la reproducción.

Si suponemos que Ret_{min} y Ret_{max} son los valores mínimo y máximo del retardo que puede sufrir una LDU desde la fuente hasta el receptor, respectivamente, y suponemos que la fuente inicia la transmisión en el instante T_0, el *jitter* puede producir que el receptor empiece su reproducción tan pronto como $T_0 + Ret_{min}$, o tan tarde como $T_0 + Ret_{max}$. Por tanto, se puede encontrar una asincronía inicial de, al menos, ($Ret_{max} - Ret_{min}$) entre cada par de receptores. Además, también se producirá una falta de sincronización debida a los desajustes en la tasa de consumo de cada receptor. Supongamos que los receptores poseen una tasa de consumo de θ LDUs por segundo y que la desviación o *drift* en el periodo de reproducción de los receptores está acotado por $\pm\rho$. Según estos parámetros, algún receptor podrá reproducir a la velocidad más rápida (con un período de $\theta * (1-\rho)$), mientras que otro receptor podrá reproducir a la tasa más lenta (con un período de $\theta * (1+\rho)$). La asincronía máxima, en unidades de datos o LDUs, entre dos reproductores que reproducen a la máxima y a la mínima velocidad, respectivamente, se verá incrementada a medida que progrese la reproducción, alcanzando un máximo de:

$$A = \left\lceil \frac{\left(\mathrm{Re}t_{\max} - \mathrm{Re}t_{\min}\right) + 2 * \theta * \rho * n}{\theta * (1 - \rho)} \right\rceil \qquad LDUs \qquad \text{[Ec. 4.3]}$$

, donde *A* representa la máxima *asincronía* entre los dos reproductores, en el instante en que el receptor más lento está reproduciendo la LDU con el número de secuencia *n.*

En la ecuación 4.3, el término $(Ret_{max} - Ret_{min}) / (\theta * (1-\rho))$ es la contribución del *jitter* de la red a la *asincronía*, mientras que el factor $(2 * \theta * \rho * n) / (\theta * (1-\rho))$ representa el efecto de las diferentes tasas de consumo entre los diferentes receptores. Se puede observar que *A* se incrementa linealmente con la progresión de la reproducción, es decir, a medida que aumenta el número de LDUs consumidas, aspecto que, lógicamente, es inaceptable en la práctica y que deberá ser corregido mediante la utilización de técnicas de sincronización que produzcan resincronizaciones cada cierto tiempo.

Para la sincronización distribuida del grupo de receptores involucrados en la aplicación multimedia, será necesario que se cumplan dos factores: en primer lugar, que todos los receptores empiecen a reproducir los flujos en el mismo instante (denominado *Instante Inicial de Consumo*) y, en segundo lugar, que a lo largo de toda la aplicación, *todos los receptores reproduzcan sus flujos de forma sincronizada*, es decir, que se tendrá que implementar algún mecanismo que corrija las desviaciones en la reproducción de los receptores (*sincronización fina entre receptores*).

4.2.2.1. Instante Inicial de Consumo y Sincronización Gruesa

Como se ha comentado en el apartado anterior, nuestro algoritmo de sincronización en una presentación multimedia, se encuentra como primer problema a resolver, la *sincronización del instante inicial de consumo*, es decir el momento de inicio de la reproducción conjunta. En nuestro caso, se resolverá dicho problema asegurando que todos los receptores ajusten dicho punto para el flujo tomado como *maestro*, ya que los procesos de reproducción del resto de flujos seguirán al proceso del flujo *maestro* gracias al proceso de sincronización inter-flujo. Esta sincronización inicial será importante ya que todos los receptores deberán empezar a reproducir los correspondientes flujos en el instante inicial indicado.

Además del punto inicial, en una presentación multimedia se puede realizar una división de la misma en etapas, tal y como se muestra en la figura 4.8, cada una de las cuales tendrá un instante de inicio que también deberá ser sincronizado. A

partir de dicha división en etapas, se deberá garantizar lo que denominamos una *Sincronización Gruesa* que se reducirá, en cada etapa, a garantizar la sincronización del punto inicial del consumo del flujo *maestro*. En adelante, se entenderá que una etapa finaliza cuando se haya dejado de transmitir, durante un determinado intervalo de tiempo, el flujo *maestro*, y se entiende que comienza de nuevo al reanudar éste la transmisión de dicho flujo. Ambos aspectos se muestran en la figura.

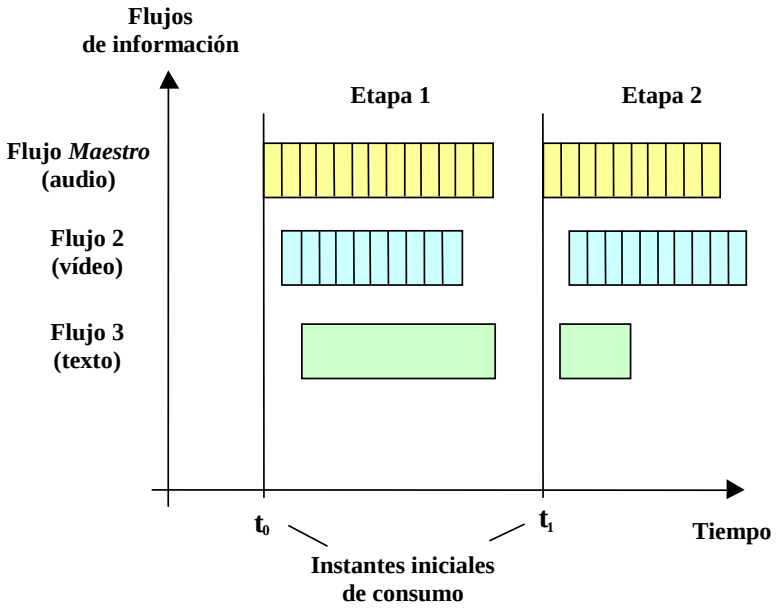


Figura 4.8. Mecanismo de sincronización del instante inicial de consumo y Sincronización Gruesa

Para resolver los problemas de sincronización del instante inicial y de la sincronización gruesa en cada etapa, se necesitará garantizar la existencia de un *tiempo global a todos los receptores* (y a todas las fuentes). Evidentemente, la precisión en la sincronización del instante inicial de consumo y de la sincronización gruesa dependerá directamente de la precisión del protocolo de sincronización que ofrezca el servicio de tiempo global y, a su vez, la precisión de éste dependerá del entorno donde se ejecute, y de la estructura del propio algoritmo. En el algoritmo propuesto, este tiempo global se obtendrá por medio del *protocolo NTP*.

La solución al primer problema (sincronización del instante inicial de consumo) es sencilla y se reduce a especificar el instante de consumo, de forma *multicast* (si la red lo permite), a todos los reproductores, aprovechando que existe un tiempo global y común a todos ellos. Se utilizará un paquete RTCP de control,

en particular, el paquete APP definido en la RFC 1889 ([SCH96]), con nuevas extensiones definidas para la implementación del algoritmo. El formato del paquete propuesto será mostrado en el apartado siguiente, y contendrá un campo con la estampación NTP del instante en el que el grupo de receptores deberá comenzar la reproducción del flujo *maestro* (en la figura 4.8, sería el audio), con el cual los demás flujos *esclavos* (en la figura 4.8, el vídeo y los datos) se sincronizarán inmediatamente haciendo uso del mecanismo de sincronización inter-flujo del algoritmo propuesto, descrito posteriormente.

El instante NTP enviado se corresponderá con un valor de tiempo (expresado, por ejemplo, en milisegundos), que utilizará cada estación receptora para determinar el instante inicial de consumo propio. Lógicamente, el inicio del consumo se producirá cuando el tiempo del reloj local de cada estación reproductora coincida con el instante de tiempo indicado en dicho paquete. Como todos los receptores deberán estar sincronizados vía NTP, el instante '*real*' de inicio de la reproducción del flujo *maestro* coincidirá en todos ellos.

Por otro lado, para solucionar el problema de garantizar la sincronización gruesa de cada etapa se propone realizar esta misma operación.

Cálculo del Instante Inicial de Consumo

Para el cálculo del instante inicial de consumo, en principio, se deberá conocer el retardo medio que sufren los paquetes del flujo *maestro* entre la fuente y los receptores (figura 4.9).

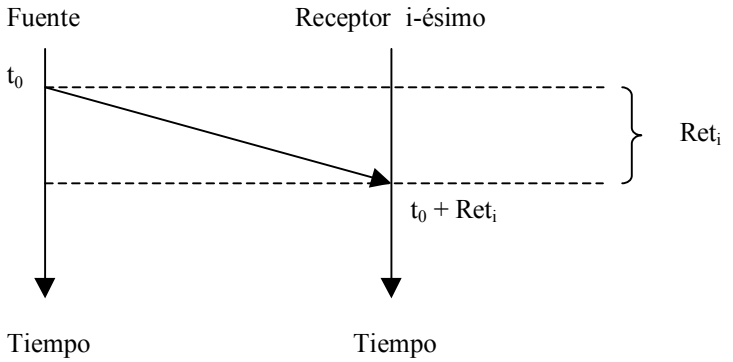


Figura 4.9. Retardo hasta el Receptor i-ésimo.

Una vez obtenido el retardo de cada uno de los receptores (Ret_i), se tomará el máximo de ellos y se estimará que el instante inicial de consumo será:

$$T_{in}(m) = t_0(m) + Ret_{max}(m) + \Delta \qquad \text{[Ec. 4.4]}$$

, siendo $t_0(m)$ el tiempo de inicio de la transmisión del flujo *maestro* (m), $Ret_{max}(m)$ el máximo retardo sufrido desde la fuente del flujo *maestro* hasta cada reproductor

$$Ret_{max}(m) = max\ \{ Ret_i(m)\ ,\ \forall i\} \qquad \text{[Ec. 4.5]}$$

, y Δ será un tiempo adicional añadido para permitir que los datos de cada flujo lleguen a los receptores y se almacenen en los *buffers* de recepción. Se corresponderá con el tiempo necesario para almacenar un número suficiente de LDUs en el receptor con mayor retardo para que no se produzca situaciones de *underflow* cuando inicie su reproducción.

$$\Delta = \frac{\left[\mathrm{Re}t_{\max} - \mathrm{Re}t_{\min}\right]}{(1-\rho)} \qquad \text{[Ec. 4.6]}$$

, donde Ret_{max} y Ret_{min} son los límites, máximo y mínimo, del retardo introducido por la red, que serán conocidos a priori.

De esta manera se asegura que todos los receptores tendrán un número suficiente de LDUs en el *buffer* de recepción antes de iniciar la reproducción, que asegure una reproducción continua, a pesar del *jitter*.

A continuación se detallan dos opciones diferentes para calcular el valor de Ret_i:

1.- Mediante el envío de varios paquetes de eco y respuesta (*Echo Request y Echo Reply*) definidos en IP (RFC 791), y el cálculo de un promedio de los retardos de ida calculados.

2.- Otra forma de calcular dicho retardo, que es la utilizada en la presente Tesis, será la de enviar, al iniciar las aplicaciones (antes de iniciar la reproducción), varios paquetes de control por parte de todos los receptores hacia la fuente. Aprovechando el servicio de tiempo global NTP, este paquete de control contendrá el instante de tiempo con el que los receptores lo envíen (en formato NTP), de tal forma que la fuente, al recibirlo, sabrá el retardo sufrido por ese paquete desde el receptor hasta llegar a la fuente[(1)]. Se enviarán varios paquetes para obtener el retardo promedio entre la fuente y

[1] En éste cálculo se ha supuesto que el retardo sufrido por los paquetes en el trayecto de ida desde la fuente hasta el receptor es el mismo que el sufrido en el trayecto de vuelta, del receptor a la fuente, tal y como lo supone ICMP o el propio RTP/RTCP.

cada uno de los receptores. En concreto, el paquete de control utilizado en la segunda opción, será también un paquete RTCP APP definido para ser utilizado con este fin y dependiente de la aplicación, que denominamos *paquete APP RET*, para el cálculo del retardo, y se describirá en el siguiente apartado.

Por otro lado, el parámetro Δ anterior, puede ser obtenido, de una forma más ajustada, tanto en la fase inicial (antes de iniciar la reproducción) como al principio de cada etapa, a partir de los retardos medidos para todos los paquetes APP RET (en la primera etapa) o de los paquetes RTCP RR durante la sesión multimedia, recibidos de los receptores.

$$\Delta = \frac{\left\lceil \max\{\mathrm{Re}\,t_i(m), \forall i\} - \min\{\mathrm{Re}\,t_i(m), \forall i\} \right\rceil}{(1-\rho)} \qquad \text{[Ec. 4.7]}$$

En todo caso, en las redes para las que se propone nuestro algoritmo, es decir, deterministas o redes que garanticen una cierta calidad de servicio, los límites máximo y mínimo del retardo que introduce la red serán conocidos a priori y se podría utilizar directamente en la fórmula anterior para calcular T_{in}.

- Paquete APP RET para el cálculo del retardo

El nuevo paquete definido, denominado *paquete APP RET*, tendrá el siguiente formato:

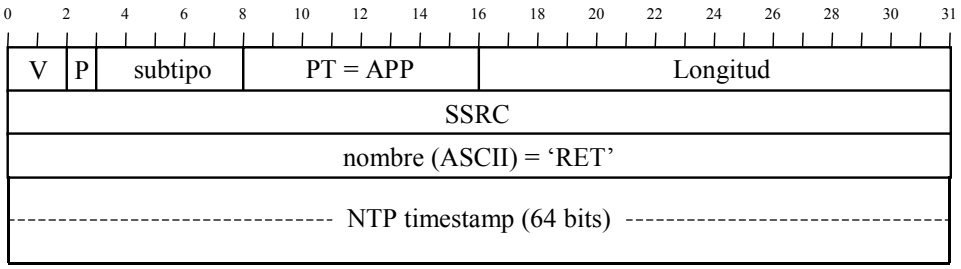


Figura 4.10. Paquete APP RET para el cálculo del retardo

, donde el significado de los campos es el siguiente:

- *Versión* y *Bit de relleno* (*P, padding*): Mismo significado que para los paquetes RTP explicados en el capítulo anterior. En este caso no habrá relleno (P='0') puesto que el paquete tiene un tamaño exacto de 5 palabras de 32 bits.

- *Subtipo*: 5 bits que se utilizan para agrupar un conjunto de paquetes APP bajo el mismo nombre. En este caso no se utiliza.
- *PT o tipo de paquete*: 8 bits que indicarán el tipo RTCP-APP y, por tanto, tomará el valor decimal constante '204', según la RFC 1889 ([SCH96]).
- *Longitud*: 16 bits que indican la longitud del paquete menos una, en palabras de 32 bits. En este caso la longitud del paquete es de 5 palabras de 32 bits por lo que este campo tomará el valor de 4.
- *SSRC* o *Synchronization source* (32 bits), identifica a la fuente del flujo multimedia, emisora del paquete.
- *Nombre*: 32 bits que contienen 4 caracteres ASCII de 8 bits con el nombre del paquete, que en este caso será 'RET'
- *NTP Timesatmp*: 64 bits que contendrán, en formato NTP, el instante de tiempo de envío del paquete.

Una vez recibidos todos paquetes correspondientes a todos los receptores, la fuente podrá estimar el instante inicial de consumo (T_{in}), tal y como se ha explicado anteriormente.

La utilización del envío de este paquete tiene la ventaja, con respecto a la utilización de paquetes de eco y respuesta (*Echo Request* y *Echo Reply*), de que la fuente podrá conocer más rápidamente el retardo máximo de entre todos los receptores ya que son ellos los que se encargan de enviar los paquetes APP RET al iniciar las aplicaciones. Como se trata de un paquete RTCP, no se enviarán todos los paquetes APP RET de todos los receptores al mismo tiempo sino que se calculará el tiempo de transmisión según indica la RFC 1889, de tal manera que afecte lo mínimo posible a la escalabilidad del algoritmo.

- Paquete APP TIN de Inicio de Consumo.

Para comunicar a los receptores el instante inicial de consumo, también ha sido necesario definir un nuevo paquete dependiente de la aplicación (tipo RTCP APP), que se ha denominado *paquete APP TIN*, y que sigue el siguiente formato:

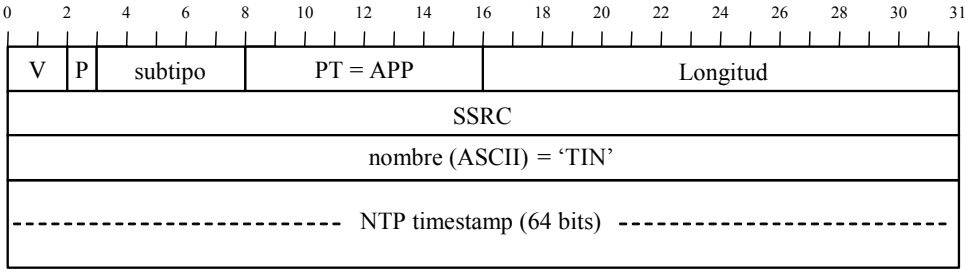


Figura 4.11. Paquete APP TIN para indicar el instante de inicio del consumo

, donde el significado de los campos es prácticamente el mismo que en el paquete anterior, exceptuando los siguientes:

- *Nombre*: 32 bits que contienen 4 caracteres ASCII de 8 bits con el nombre del paquete, que en este caso será 'TIN'.
- *NTP Timestamp:* 64 bits que contienen, en formato NTP, el instante en que deberán iniciar el consumo todos los reproductores de dicho flujo, calculado por la fuente sincronizadora, de la manera indicada en el siguiente apartado.

Una vez transmitido el paquete APP TIN (de inicio de consumo), a continuación, se enviarán los paquetes RTP conteniendo las muestras del flujo correspondiente, que serán almacenadas en los receptores hasta que llegue el momento de la reproducción (instante inicial de consumo).

Como se ha comentado anteriormente, lógicamente, la asincronía producida en el inicio de la reproducción de los diferentes flujos únicamente dependerá de la imprecisión impuesta por la sincronización de relojes obtenida por NTP y también de que los paquetes APP TIN lleguen a tiempo.

En cuanto a la sincronización gruesa de cada etapa, como también se ha comentado, ya no será necesario el envío de paquetes APP RET ya que se puede obtener el retardo entre la fuente y los receptores a partir de los informes que éstos envían a la fuente (paquetes RTCP RR). Utilizando el mismo procedimiento calculado anteriormente, la fuente sincronizadora será capaz de calcular el instante de inicio de la etapa y enviar de nuevo un paquete APP TIN con el valor de tiempo NTP correspondiente.

Si nos fijamos en el ejemplo presentado en la figura 4.8, los procesos dedicados a la reproducción del flujo *maestro* (audio), recibirán dos indicaciones de sincronización gruesa a lo largo de la reproducción de la presentación, uno con el valor NTP del T_{in} correspondiente a t_0 y otro con el valor T_{in} correspondiente a t_1. Los otros dos flujos (vídeo y texto) tendrán constancia de estas indicaciones mediante una comunicación por el bus interno *mbus* con el proceso de reproducción del flujo *maestro* y que será de gran importancia a la hora de la reproducción final sincronizada de los tres flujos. Nótese que, en el ejemplo, al comienzo de las etapas representadas no hay transmisión de datos de los flujos 2 y 3, por lo que se podría suponer que en el momento de iniciar su reproducción los procesos reproductores correspondientes a dichos flujos no van a estar sincronizado con el proceso reproductor del flujo *maestro*. Esto no va a ser así, ya que los flujos 2 y 3, cuando tengan que iniciar su reproducción tendrán constancia del estado de reproducción del flujo *maestro* gracias al método de comunicación interno entre procesos a través de *mbus*.

4.2.2.2. Sincronización Fina entre Receptores

Una vez resueltos los problemas de la sincronización de los instantes iniciales de consumo y de la sincronización gruesa al inicio de cada etapa, el algoritmo de sincronización debe garantizar que también se mantenga una *sincronización fina entre los procesos reproductores de los diferentes receptores*, lo cual se conseguirá si todos los receptores reproducen de forma continua y sincronizada el flujo *maestro.*

Lo denominamos '*sincronización fina entre receptores'* porque no se trata de una sincronización local entre flujos (como la sincronización labial o *lip-sync)*, sino que es una sincronización del estado de reproducción del flujo *maestro* (por ejemplo, el de audio) en cada uno de los receptores que participan en la aplicación multimedia distribuida (*sincronización distribuida*), tal como se puede apreciar en la figura 4.12. Los demás flujos se sincronizarán posteriormente al flujo *maestro* según el método de sincronización local inter-flujo propuesto, que será explicado posteriormente.

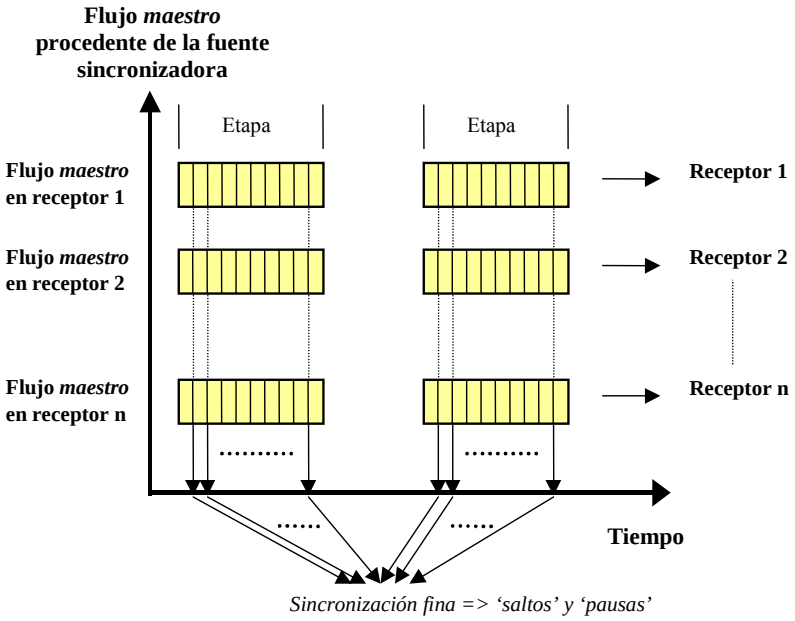


Figura 4.12. Mecanismo de sincronización fina entre receptores

El algoritmo propuesto es perfectamente válido para el caso de transmisión de información almacenada remotamente o *en directo*. La información a transmitir no estará necesariamente almacenada en la fuente sino que podrá ir generándose en tiempo real (fuente *en directo*) como, por ejemplo, las muestras de audio recogidas por un micrófono.

Con el objetivo de resolver el problema de la sincronización fina entre receptores, el algoritmo propuesto aprovecha, para sincronizar el flujo *maestro*s durante la reproducción del mismo en cada receptor, la existencia de un tiempo global y la sencillez de implementación del mecanismo de realimentación (*feedback*) de RTP/RTCP. Como ya se ha explicado en el capítulo anterior, los paquetes RTP llevan en sus cabeceras (Figura 4.13) marcas de tiempo que ayudarán a la sincronización fina entre todos los procesos reproductores del flujo *maestro* de los receptores.

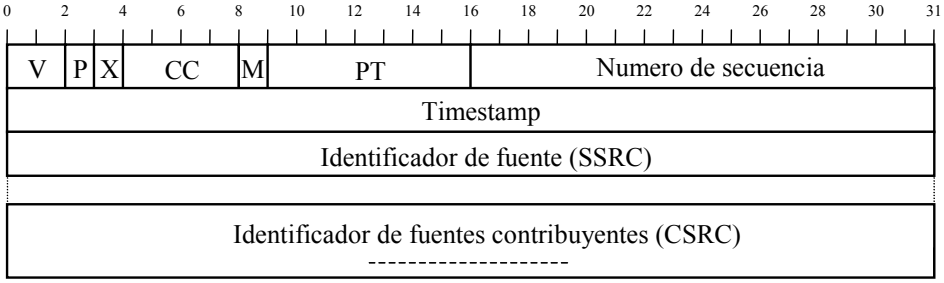


Figura 4.13. Cabecera del paquete RTP

El proceso de sincronización fina entre receptores propuesto consta, tal y como se puede apreciar en la figura 4.14, de dos fases:

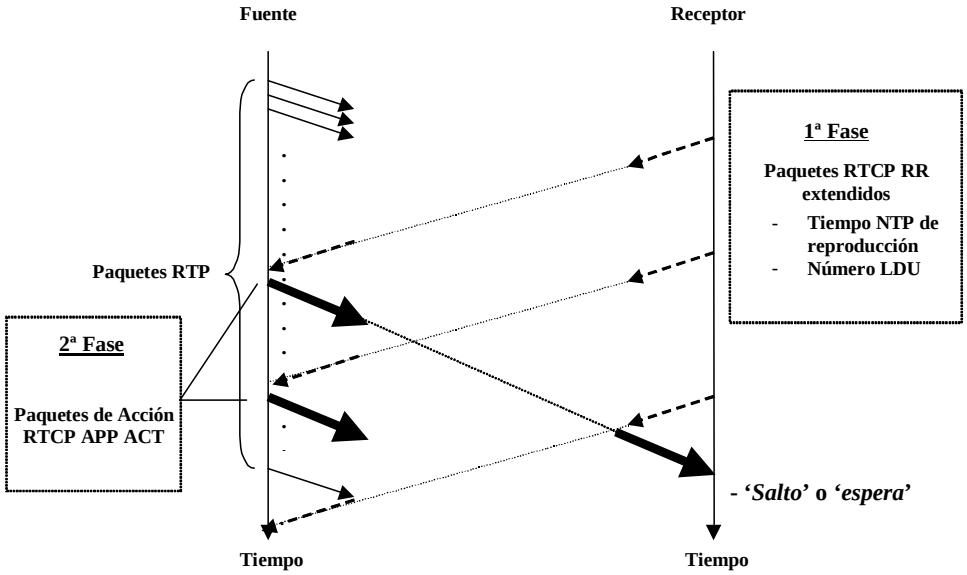


Figura 4.14. Paquetes de control para la sincronización fina entre receptores

Primera fase

En la primera fase, los procesos reproductores informan de forma 'periódica' (según se especifica en la RFC 1889, dependiendo del número de participantes activos en la sesión) a la fuente sobre el estado de la reproducción, aprovechando los *paquetes de control RTCP RR* (mensajes de realimentación o *feedback*). Estos mensajes tendrán la misma función que los mensajes *feedback* del protocolo *Feedback-Global* ([GUE97]).

Siguiendo la implementación de dicho protocolo, estos mensajes de realimentación o *feedback* contendrán el número de la última LDU consumida procedente de la fuente sincronizadora y el instante NTP de su reproducción. Para ello, se ha añadido una extensión al paquete RR de RTCP, conteniendo, entre otros datos, dicha información. A dicho paquete se le ha denominado *paquete RR extendido*, y su formato queda tal y como muestra la figura 4.15.

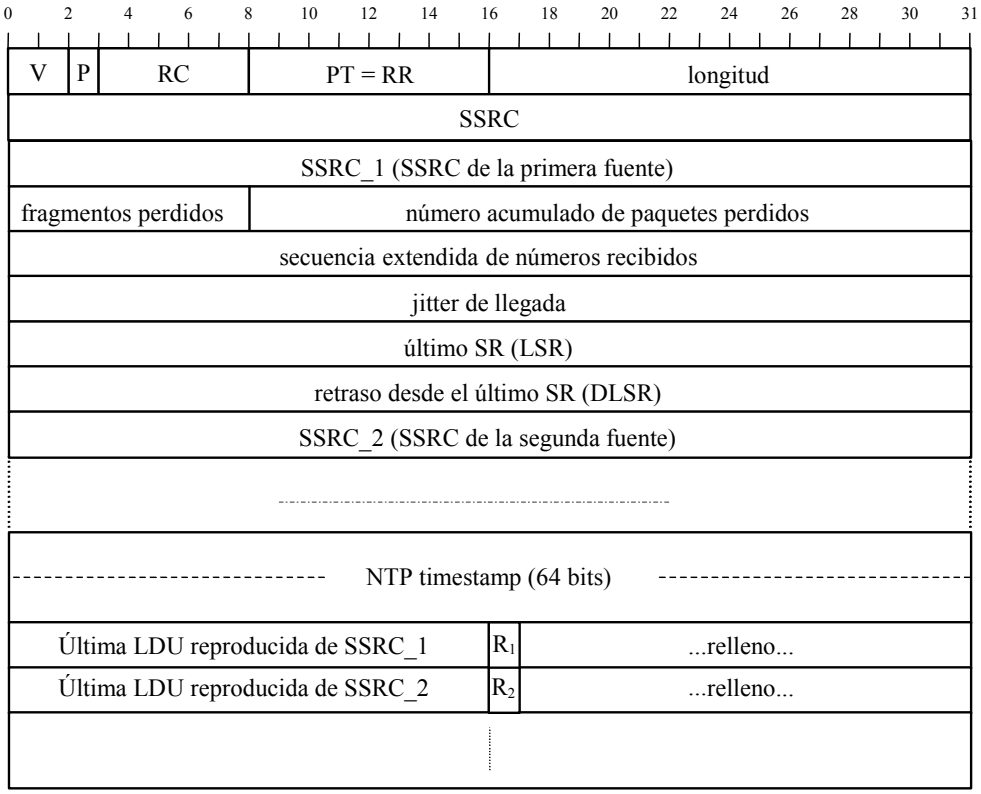


Figura 4.15. Paquete RTCP RR extendido

Al formato original definido en la RFC 1889, tal y como se puede apreciar en la figura, se han añadido los siguientes campos:

- *NTP Timesatmp*: 64 bits que contendrán, en formato NTP, el instante de tiempo con el que los receptores generan el paquete RR.
- *Última LDU reproducida de SSRC_1*: Indica el número de secuencia de la última LDU consumida en el momento de generar el paquete, procedente de la fuente cuya identificativo es SSRC_1. Como se puede apreciar, existirá un campo con la última LDU consumida de cada una de las fuentes de las cuales el receptor reciba datos.
- *Bit* R_i*:* Bit que tomará valores alternados ('1' y '0') y será utilizado por la fuente sincronizadora para comprobar que la información recibida no esté anticuada a la hora de confeccionar los paquetes de acción.

La modificación realizada al paquete RTCP RR original cumple con las especificaciones indicadas en la RFC 1889 ([SCH96]) en cuanto a las premisas que se deben cumplir a la hora de utilizar la extensión de los informes de emisión y de recepción (RTCP SR y RTCP RR).

En el algoritmo propuesto no se sigue lo especificado en otros algoritmos (como, por ejemplo, [RAN92] o [GUE97]) en los que se calcula la frecuencia o tasa óptima de envío de mensajes de realimentación o *feedback*, ya que, en este caso, se hace uso de los mensajes RR que se envían de forma periódica, con una tasa calculada según se define en la RFC 1889 ([SCH96]).

Segunda Fase

Una vez la fuente haya recibido todos los mensajes de todos los receptores conteniendo la información señalada anteriormente, se ejecutará un *algoritmo* para saber si los receptores van adelantados o retrasados en su reproducción (según un criterio determinado, con respecto a un punto de referencia) y actuar sobre ellos enviándoles un paquete especial, que se ha denominado *paquete APP ACT* o *paquete de acción* que llevará implícita la consiguiente orden de 'saltar' o 'esperar' en su proceso de reproducción.

En nuestro algoritmo, a la hora de buscar un receptor como referencia para la sincronización, se han considerado tres *Algoritmos de Selección de la Referencia Maestra*, denominados:

1.- Algoritmo de sincronización al estado de reproducción del receptor considerado como el más '*lento'* en su reproducción.
2.- Algoritmo de sincronización al estado de reproducción del receptor considerado como el más '*rápido'* en su reproducción.
3.- Algoritmo de sincronización al punto de reproducción '*medio'* de todos los receptores.

Funcionamiento General

El funcionamiento general del algoritmo propuesto es el mostrado en al figura 4.16, donde se representa la fuente sincronizadora (transmisora del flujo *maestro*) y los receptores *i* y *j* de la sesión.

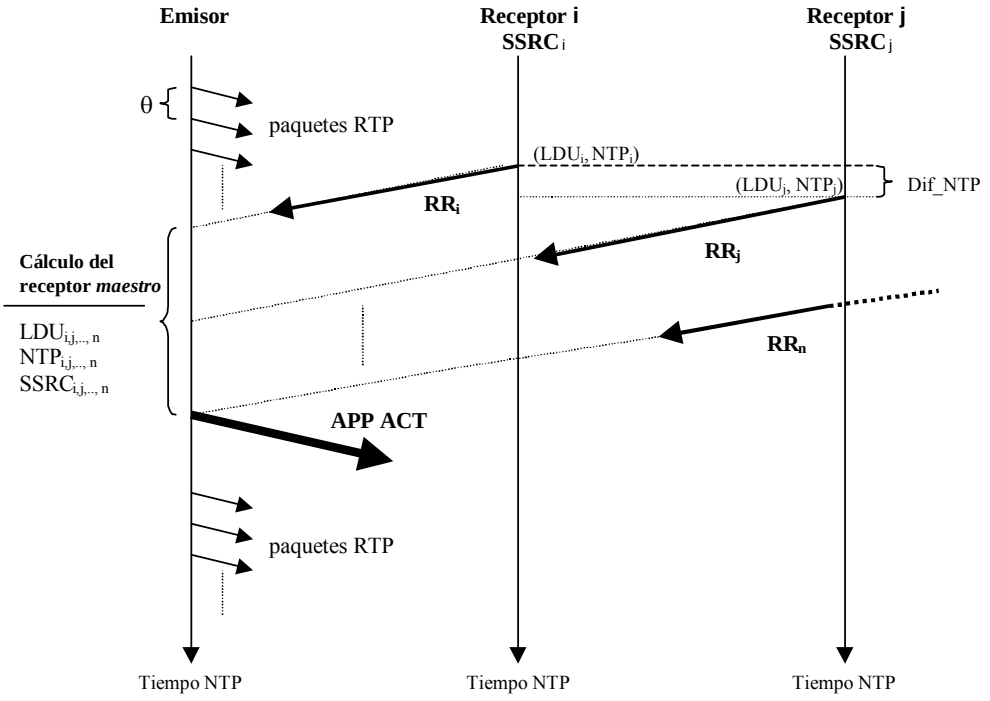


Figura 4.16. Funcionamiento General

Durante la sesión, la fuente sincronizadora irá recibiendo, de uno en uno, los paquetes RR extendidos pertenecientes a todos los receptores que estén reproduciendo el flujo *maestro* transmitido por ella. De dichos paquetes extraerá la información relacionada con el identificador del receptor (*SSRC*), la última LDU reproducida por el mismo y el instante NTP en que dicho receptor reprodujo dicha LDU. Esta información se irá guardando en una tabla creada por la propia fuente sincronizadora con un número de registros igual al número de receptores participantes en la sesión (*n*), con la estructura mostrada en la tabla 4.1.

En los casos en que la fuente reciba un segundo paquete RR extendido procedente de un mismo receptor antes de completar toda la tabla, actualizará la información, con el fin de mantener la tabla con valores más recientes.

SSRC	Última LDU	NTP timestamp	Bit de Reproducción R_i
$SSRC_1$	LDU_1	NTP_1	bit_1
$SSRC_2$	LDU_2	NTP_2	bit_2
$SSRC_n$	LDU_n	NTP_n	bit_n

Tabla 4.1. Información necesaria para cálculo de *receptor maestro*

La columna '*bit de reproducción*' indica si el receptor está o no activo y se utiliza para saber si el receptor incluido en la sesión está o no reproduciendo el flujo *maestro* y, por tanto, su información (LDU_i y NTP_i) deberá ser tomada en cuenta (bit a '1') o no (bit a '0') para realizar el cálculo del punto de reproducción de referencia. Este bit será necesario para poder considerar los abandonos de los receptores durante la sesión y evitar que los datos referentes a receptores no presentes en la sesión afecten al resto en un momento dado.

Una vez completada la tabla con los nuevos datos procedentes de todos los receptores activos, se estará en disposición de ejecutar cualquiera de los tres algoritmos propuestos.

Lo ideal, a la hora de calcular la referencia o receptor *maestro* con el cual se sincronizarán todos los demás receptores, sería que todos los receptores mandasen un paquete de control con dicha información a la vez, es decir, con la misma referencia temporal o instante NTP. Esto, lógicamente, en sesiones con un elevado número de usuarios podría suponer un envío masivo de paquetes de todos los receptores a la fuente en ciertos instantes, lo cual podría colapsarlo, afectando a la escalabilidad del algoritmo. Este ha sido uno de los motivos por los que se ha elegido el paquete RTCP RR para enviar la información necesaria para el algoritmo propuesto. Tal y como describe la RFC1889, cada receptor mandará su paquete de informe RR extendido de forma aleatoria. Por lo tanto, el momento NTP con el que los receptores enviarán sus paquetes RR extendidos no será el mismo. Debido a esta aleatoriedad en el envío, la fuente se verá obligada a buscar una relación entre la última LDU consumida y el tiempo global y 'real' NTP. Esta operación será explicada para cada una de las opciones propuestas de selección de la *referencia maestra*.

Algoritmos de selección de la referencia *maestra*

A continuación se van a exponer las tres técnicas o algoritmos de selección de un punto de referencia al que adaptar los procesos de reproducción de los flujos multimedia para obtener la sincronización de grupo deseada.

- *Algoritmo de sincronización al estado de reproducción del receptor considerado como el más 'lento' en su reproducción*

Este algoritmo calcula cuál de todos los receptores es el más lento en la reproducción, y lo considera como receptor *maestro*, es decir, su estado de reproducción será considerado como la *referencia* a la que el resto de receptores deberán sincronizarse. Este receptor *maestro* no será siempre el mismo, debido a que la fuente volverá a ejecutar el algoritmo para cada ciclo de recepción de todos los paquetes RR extendidos y el estado de la reproducción de cada uno puede haber variado con el tiempo. Por tanto, en cada cálculo, los motivos para que un receptor sea elegido como *maestro*, vendrán determinados por diferentes parámetros como el retardo de la red, la desviación de los relojes, el efecto del sistema operativo, la máxima asincronía permitida, etc. Estos factores afectarán de forma directa y diferente a cada receptor y, por tanto, cualquier receptor podrá ser el *maestro* en cualquier instante de la transmisión.

Una vez recibidos todos los paquetes RR extendidos de todos los participantes y guardados en la tabla descrita (tabla 4.1), la fuente ya estará en disposición de calcular cuál de todos es el más lento en su reproducción. Éste comparará los valores de la LDU y del instante NTP de reproducción del primer RR extendido recibido de un receptor con respecto del siguiente RR extendido guardado en la tabla. Los valores del receptor que se considere más lento en su reproducción se guardarán en variables y se comparará de nuevo con los valores del siguiente RR extendido guardado, y seguirá así hasta que se procesen todos los RR extendido guardados pertenecientes a todos los participantes (el número activo de participantes reproductores del flujo *maestro* transmitido deberá ser conocido en todo momento por la fuente). Al final del proceso, se sabrá cuál de todos los receptores es el más lento y por lo tanto el que será considerado como referencia o receptor *maestro*. Los valores de la última LDU consumida por dicho reproductor y el instante NTP de su reproducción serán almacenados como referencia.

A partir de la información de la última LDU y el instante NTP de reproducción del receptor *maestro*, con el cual todos los demás receptores deberán sincronizarse, la fuente sincronizadora compondrá el paquete APP ACT de acción , que enviará de forma *multicast*, si es posible, a todos los receptores para que sincronicen sus procesos reproductores con el del receptor de referencia o *maestro*.

Para el cálculo descrito, si nos fijamos en la Figura 4.16, los paquetes RTP del flujo *maestro* se transmiten de forma continua, debiéndose reproducir de la misma forma (idealmente), pero retardada con respecto al instante de envío del transmisor, en todos los receptores. Por lo tanto, si se toma la tasa de envío de la fuente (θ o período de duración de las LDU) como referencia, podremos calcular el tiempo transcurrido entre la reproducción de un paquete y otro mediante sus números de secuencia (obtenidos de la cabecera del paquete RTP). Este valor

tiempo obtenido será el transcurrido entre la reproducción de LDUs con diferentes números de secuencia y, por tanto, se podrá comparar con valores de tiempo tomados a partir del eje de tiempos global (NTP) utilizado por todos los receptores.

Tomando esta información como punto de partida, para poder conocer el receptor con reproducción temporal más lenta, la fuente tomará, en primer lugar, los datos pertenecientes a los dos primeros receptores guardados en la tabla y realizará los siguientes pasos, siempre y cuando el tiempo NTP y la última LDU consumida del segundo receptor sean mayores que las del primero:

1. Primero, la fuente calculará la diferencia de las últimas LDUs consumidas por parte de sendos receptores y la multiplicará por la duración de las LDUs para obtener el tiempo transcurrido entre una y otra.

$$\begin{aligned} Dif\_LDU &= LDU_2 - LDU_1 \\ Dif\_LDU(ms) &= Dif\_LDU * \theta(ms) \end{aligned} \qquad \text{[Ec. 4.8]}$$

2. Este resultado lo comparará con la diferencia de los tiempos NTP de reproducción recibidos.

$$Dif\_NTP = NTP_2 - NTP_1 \qquad \text{[Ec. 4.9]}$$

Para poder comparar ambos resultados, éstos deberán estar en las mismas unidades, por ejemplo en milisegundos (*ms*). Para ello, se pasará el resultado *Dif_NTP* a milisegundos.

Si el valor *Dif_LDU* (en milisegundos) resulta mayor que el valor de *Dif_NTP* (en milisegundos), resultará que el *Receptor 2* habrá reproducido en el mismo instante una LDU mayor y, por lo tanto, el *Receptor 1* se tomará como el receptor más lento de los dos comparados, de modo que sus valores se guardarán para ser comparados con los obtenidos del siguiente receptor. Si, por el contrario, el valor de *Dif_NTP* (en milisegundos) resulta ser el mayor, evidentemente se tomaría al *Receptor 2* como el más lento y sus valores serían los guardados para compararlos con los obtenidos del siguiente receptor guardados en la tabla.

Esto mismo expresado en pseudocódigo:

IF (Dif_LDU(ms) > Dif_NTP(ms))
THEN $SSRC_2$ será el más lento y sus valores se guardan
ELSE
$SSRC_1$ será el más lento y sus valores se guardan
END IF

Suponiendo que sólo hubiese dos receptores reproduciendo el flujo transmitido, el algoritmo habría terminado y, a continuación, se compondría el paquete de acción APP ACT, a partir de los valores de número de LDU y tiempo NTP, obtenidos de la tabla, correspondientes al receptor más lento. Pero como se ha comentado antes, esto sólo se producirá cuando la última LDU y el tiempo NTP de reproducción del segundo receptor sean mayores que las del primero.

Esto no ocurrirá siempre así, ya que la tabla puede ser actualizada, antes de ser completada, con la información de algún receptor del que ya se disponía de información guardada, o bien los informes pueden llegar más tarde por diferentes razones. A continuación se detallan, también en pseudocódigo, todos los casos posibles a la hora de calcular el receptor *maestro* considerado como el más lento:

En un primer momento, se tomará al primer receptor de la tabla como el más lento de forma temporal.

$LDU_{maestro} = LDU_1$
$NTP_{maestro} = NTP_1$
$SSRC_{maestro} = SSRC_1$

A continuación, se comparará con todos los demás receptores participantes, para quedarse, finalmente, con el más lento en su reproducción.

```
FOR (i = 2, i <=N, i++)
    IF (LDU_i > LDU_maestro) THEN
        Dif_LDU = LDU_i – LDU_maestro
        Dif_LDU(ms) = Dif_LDU * θ (ms)

        IF (NTP_i > NTP_maestro) THEN
            Dif_NTP = NTP_i – NTP_maestro
            Dif_NTP(ms) = conversión_ms(Dif_NTP)
            IF (Dif_NTP(ms) > Dif_LDU(ms)) THEN
                LDU_maestro = LDU_i
                NTP_maestro = NTP_i
                SSRC_maestro = SSRC_i
            END IF
        END IF
    ELSE
        Dif_LDU = LDU_maestro - LDU_i
        Dif_LDU(ms) = Dif_LDU * θ (ms)

        IF (NTP_maestro <= NTP_i) THEN
            LDU_maestro = LDU_i
            NTP_maestro = NTP_i
```

$SSRC_{maestro} = SSRC_i$
ELSE
$Dif\_NTP = NTP_{maestro} - NTP_i$
$Dif\_NTP(ms) = conversión\_ms(Dif\_NTP)$

IF $(Dif\_NTP(ms) <= Dif\_LDU(ms))$ *THEN*
$LDU_{maestro} = LDU_i$
$NTP_{maestro} = NTP_i$
$SSRC_{maestro} = SSRC_i$
END IF
END IF
END IF
END FOR

, siendo *i* el número del receptor analizado de la tabla y *N* el número total de receptores.

Una vez conocido el receptor más lento, se borrará la tabla con la información de los receptores y se tomarán los valores de última LDU y tiempo NTP de reproducción correspondientes al receptor *maestro* para componer el paquete APP ACT tal y como se explicará posteriormente.

Ejemplo1. Sincronización al estado de reproducción del receptor más lento en su reproducción

A continuación, se presenta un ejemplo numérico para entender mejor todo el proceso anterior. Supongamos que nos encontramos en una transmisión de un único flujo (por ejemplo, audio) en la que participan una fuente transmisora (sincronizadora) y *n* receptores, tal y como se muestra en la figura 4.17.

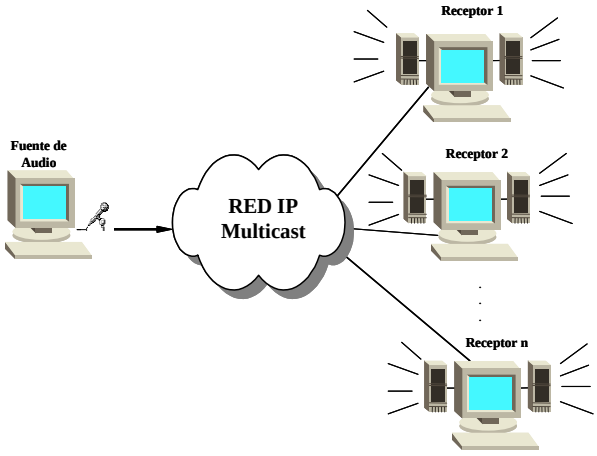


Figura 4.17. Escenario del ejemplo 1

Supongamos que la fuente acaba de recibir todos los paquetes RR extendidos pertenecientes a los *n* participantes de la sesión y ha completado su tabla con los siguientes valores (tomados de una prueba realizada en laboratorio):

SSRC	última LDU	Instante NTP[(2)]	Bit de reproducción R_i
$SSRC_1$	48674	1491954565	1
$SSRC_2$	48680	1491958348	1
$SSRC_n$	48676	1491957841	1

Tabla 4.2. Datos del Ejemplo 1

El diagrama de tiempos respecto a la última LDU visto por la fuente sincronizadora sería el siguiente:

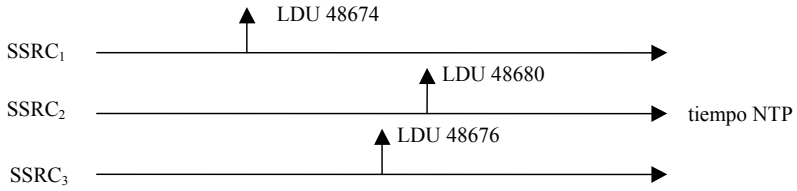


Figura 4.18. Diagrama de tiempos del ejemplo

La fuente tomará los valores del primer receptor guardado en la tabla (siempre y cuando esté reproduciendo su flujo, es decir, que su bit de reproducción esté a '1') y lo comparará con el siguiente para saber cuál de los dos es el más lento. Teniendo en cuenta que la fuente está transmitiendo paquetes de una duración de 20 milisegundos (θ = 20) , se procederá tal y como se ha explicado anteriormente:

$LDU_{maestro} = 48674$
$NTP_{maestro} = 1491954565$
$SSRC_{maestro} = SSRC_1$

$LDU_2 > LDU_1$ *THEN*
$Dif\_LDU = LDU_2 - LDU_1 = 48680 - 48674 = 6$
$Dif\_LDU(ms) = Dif\_LDU * \theta(ms) = 6 * 20 = 120\ ms$

$NTP_2 > NTP_1$ *THEN*

[2] El tiempo NTP guardado está en formato de 32 bits, pasado a decimal, y corresponde, aproximadamente, a las 16 horas del 2 de Marzo del 2002.

$Dif\_NTP = NTP_2 - NTP_1 = 1491958348 – 1491954565 = 507$
$Dif\_NTP(ms) = conversión\_ms(Dif\_NTP) = 57{,}72\ ms$

$Dif\_NTP(ms) <= Dif\_LDU(ms)\ THEN$
$SSRC_1$ *continúa como el más lento.*

El Receptor 1 continuaría siendo temporalmente el más lento en su reproducción y sería comparado con todos los demás hasta llegar al último (suponiendo que ninguno fuese más lento que él).

$LDU_N > LDU_1\ THEN$
$Dif\_LDU = LDU_N – LDU_1 = 48676 – 48674 = 2$
$Dif\_LDU(ms) = Dif\_LDU * \theta\,(ms) = 2 *20 = 40\ ms$

$NTP_N > NTP_1\quad THEN$
$Dif\_NTP = NTP_N – NTP_1 = 1491957841 – 1491954565 = 3276$
$Dif\_NTP(ms) = conversión\_ms(Dif\_NTP) = 49{,}98\ ms$

$Dif\_NTP(ms) > Dif\_LDU(ms)\ THEN$
$LDU_{maestro} = LDU_N$
$NTP_{maestro} = NTP_N$
$SSRC_{maestro} = SSRC_N$

Al final, en el ejemplo, el receptor *N* sería el más lento de todos y, por lo tanto, sus dos valores se tomarían como referencia para completar el paquete APP ACT de acción.

En caso de escoger este algoritmo, la fuente tendrá la opción de elegir un valor de *Máxima Asincronía* entre todos los receptores. Este valor será la máxima desviación permitida entre el receptor más rápido en su proceso de reproducción y el receptor más lento. De esta forma, en el caso de no detectar algún par de receptores que se desviase por encima de ese valor, la fuente no enviaría ningún paquete APP de acción y, por tanto, se reduciría el envío de paquetes de control para evitar posibles sobrecarga y situaciones de congestión de la red.

Ventajas e inconvenientes del algoritmo:

La principal *ventaja* del algoritmo es que existe un reproductor, el *maestro*, cuyo proceso de reproducción no sufrirá alteraciones (‘*saltos’* o *‘pausas’*) en su reproducción. Otra *ventaja* sería que todos los receptores incluidos en la sesión, terminarán reproduciendo a la misma velocidad de consumo que ‘el receptor más lento’ en un determinado momento, sin tener que obligar a ninguno a reproducir un paquete que, posiblemente, no haya recibido todavía y de esa forma tenga una

reproducción no fluida (ininteligible), debido a las *'pausas'* que ello produciría en su reproducción.

El principal *inconveniente* detectado está en el aumento progresivo del tamaño necesario de los *buffers* de reproducción de los receptores 'rápidos' que, además de poder llegar a producir una saturación u *overflow* de los mismos, produce una perdida de sensación de tiempo real entre la fuente y los receptores.

- *Algoritmo de sincronización al estado de reproducción del receptor considerado como el más 'rápido' en su reproducción*

Este algoritmo realizará las mismas funciones que el anterior, pero con la diferencia de que el receptor que se toma como referencia o *maestro* será ahora el más rápido, es decir, que todos los procesos reproductores del flujo *maestro* de los demás receptores se sincronizarán al proceso correspondiente al receptor con un estado de reproducción más avanzado.

Está pensado para los casos en que las características de la red de todos los participantes sean parecidas y entonces convenga que el retardo de los receptores no se incremente con el tiempo, como pasaba en el caso anterior, debido a una sincronización constante con el más lento.

Una vez rellenada la tabla (tabla 4.1) con la información recogida de los paquetes RR extendidos provenientes de todos los receptores, la fuente se encargará de calcular el receptor considerado según dichos datos como el 'más rápido' que se convertirá en el receptor *maestro*. A continuación se explica en pseudocódigo el proceso realizado por la fuente para calcular dicho receptor:

En principio se toma al primer receptor de la tabla como el más rápido de forma temporal.

$$LDU_{maestro} = LDU_1$$
$$NTP_{maestro} = NTP_1$$
$$SSRC_{maestro} = SSRC_1$$

A continuación, se compara con todos los demás receptores participantes, para quedarse al final con el más rápido en su reproducción.

FOR (i = 2, i <= N, i++)
  IF ($LDU_i > LDU_{maestro}$) THEN
    $Dif\_LDU = LDU_i - LDU_{maestro}$
    $Dif\_LDU(ms) = Dif\_LDU * \theta\,(ms)$

```
            IF( NTP_i <= NTP_maestro) THEN
                    LDU_maestro = LDU_i
                    NTP_maestro = NTP_i
                    SSRC_maestro = SSRC_i
            ELSE
                    Dif_NTP = NTP_i – NTP_maestro
                    Dif_NTP(ms) = conversión_ms(Dif_NTP)
                    IF (Dif_NTP(ms) <= Dif_LDU(ms)) THEN
                            LDU_maestro = LDU_i
                            NTP_maestro = NTP_i
                            SSRC_maestro = SSRC_i
                    ENDI IF
            END IF
    ELSE
            Dif_LDU = LDU_maestro - LDU_i
            Dif_LDU(ms) = Dif_LDU * θ (ms)

            IF (NTP_maestro > NTP_i) THEN
                    Dif_NTP = NTP_maestro - NTP_i
                    Dif_NTP(ms) = conversión_ms(Dif_NTP)
                    IF (Dif_NTP(ms) > Dif_LDU(ms)) THEN
                            LDU_maestro = LDU_i
                            NTP_maestro = NTP_i
                            SSRC_maestro = SSRC_i
                    END IF
            END IF
      END IF
END FOR
```

, siendo *i* el número del receptor analizado de la tabla y *N* el número total de receptores.

Como en el caso anterior, una vez conocido el receptor más rápido en su reproducción, se borrará la tabla con la información de los receptores y se tomarán los valores de la última LDU y tiempo NTP correspondientes al receptor *maestro* para componer el paquete APP ACT.

En caso de escoger este algoritmo, también dispone la fuente o emisor la posibilidad de elegir un valor de *Máxima Asincronía* entre todos los receptores. Este valor también, como en el caso anterior, será la máxima desviación permitida entre el receptor más rápido en su proceso de reproducción y el receptor más lento. De esta forma, en el caso de no detectar algún par de receptores que se desviase por encima de ese valor, la fuente no enviaría ningún paquete APP de acción y, por

tanto, se reduciría el envío de paquetes de control para evitar posibles sobrecarga y situaciones de congestión de la red.

Ejemplo 2. Sincronización al estado de reproducción del receptor más rápido en su reproducción

Vamos a tomar el mismo ejemplo anterior (figura 4.17) con los mismos datos (tabla 4.2) para comprender mejor el proceso de cálculo del receptor con reproducción más rápida. Tomando los valores de la tabla y teniendo en cuenta que la fuente está transmitiendo paquetes de una duración de 20 milisegundos ($\theta = 20$), se procederá de la siguiente manera:

$LDU_{maestro} = 48674$
$NTP_{maestro} = 1491954565$
$SSRC_{maestro} = SSRC_1$

$LDU_2 > LDU_1$ *THEN*
$Dif\_LDU = LDU_2 - LDU_1 = 48680 - 48674$
$Dif\_LDU(ms) = Dif\_LDU * \theta\ (ms) = 6 * 20 = 120\ ms$

$NTP_2 > NTP_1$ *THEN*
$Dif\_NTP = NTP_2 - NTP_1 = 1491958348 - 1491954565 = 507$
$Dif\_NTP(ms) = conversión\_ms(Dif\_NTP) = 57{,}72\ ms$

$Dif\_NTP(ms) <= Dif\_LDU(ms)$ *THEN*
$LDU_{maestro} = LDU_2$
$NTP_{maestro} = NTP_2$
$SSRC_{maestro} = SSRC_2$

El Receptor 2 sería ahora temporalmente el más rápido en su reproducción y sería comparado con todos los demás hasta llegar al último (suponiendo que ninguno fuese más rápido que él).

$LDU_N < LDU_{maestro}$ *THEN*
$Dif\_LDU = LDU_{maestro} - LDU_N = 48680 - 48676 = 4$
$Dif\_LDU(ms) = Dif\_LDU * \theta\ (ms) = 4 * 20 = 80\ ms$

$NTP_2 > NTP_N$ *THEN*
$Dif\_NTP = NTP_{maestro} - NTP_N = 1491958348 - 1491957841 = 507$
$Dif\_NTP(ms) = conversión\_ms(Dif\_NTP) = 7{,}73\ ms$

$Dif\_NTP(ms) < Dif\_LDU(ms)$ *THEN*
$SSRC_2$ *continúa como el más rápido.*

Al final, el receptor 2 sería el más rápido de todos y por lo tanto sus valores se tomarían como referencia para completar el paquete APP ACT de acción.

Ventajas e inconvenientes del algoritmo:

También se tiene, como *ventaja*, igual que en el caso anterior, la existencia de un reproductor, el *maestro*, cuyo proceso de reproducción no sufrirá alteraciones (*'saltos'* o *'pausas'*) en su reproducción. Además, este algoritmo tiene la *ventaja* de que si todos los receptores incluidos en la sesión pertenecen a un entorno con características parecidas (retardo de red, *jitter*, etc.), realizarán una reproducción fluida y constante sin necesidad de que aumente el tamaño de sus *buffers* de reproducción.

El *inconveniente*, en este caso, está en que si existe algún receptor en la sesión con un entorno desfavorable, su proceso de reproducción se verá obligado a dar *'saltos'* constantemente en su consumo y, con ello, perderá fluidez en su reproducción llegando, probablemente, a resultar ininteligible. Debido a este inconveniente, este algoritmo está pensado para entornos '*densos*', donde los receptores no estén muy dispersos geográficamente y que todos reproduzcan a un ritmo similar, sin que se produzcan muchos desajustes entre ellos.

- *Algoritmo de sincronización al punto de reproducción 'medio' de todos los receptores*

Este algoritmo se plantea como otra opción al primer algoritmo presentado, no tan crítica como el caso anterior, para poder solucionar posibles aumentos, con el paso del tiempo, del número de LDUs almacenadas en el *buffer* de reproducción de los receptores al estar sincronizándose constantemente al más lento. Será apropiado para aplicaciones donde los receptores estén conectados a una red con características parecidas y, donde el retardo de reproducción no se quiera ver incrementado con el tiempo.

El algoritmo calcula el estado medio en la reproducción de todos los receptores, de forma que, en este caso, no existe ningún proceso de reproducción de un receptor tomado como *maestro* al cual se sincronicen los demás. Se crea un estado de reproducción ficticio o '*virtual*', tomado como referencia, obtenido como el punto medio de reproducción de todos los procesos reproductores de todos los receptores. Todos éstos, sin excepción, sincronizarán el proceso de reproducción del flujo *maestro* al punto indicado en el paquete APP ACT, cuyo contenido será obtenido a partir de dicho punto medio de referencia.

Una vez recibidos todos los paquetes RR extendidos y completada la tabla (tabla 4.1) con la información del estado de reproducción de todos los receptores,

se calculará la media de las últimas LDUs reproducidas y de los instantes NTP de su reproducción correspondientes, pero sólo de aquellos receptores activos en la sesión.

$$LDU_{media} = \frac{\sum_{i=1}^{N} LDU_i}{N} \qquad \text{[Ec. 4.10]}$$

$$NTP_{media} = \frac{\sum_{i=1}^{N} NTP_i}{N} \qquad \text{[Ec. 4.11]}$$

, donde *N* es el número total de receptores activos incluidos ese momento en la sesión

A continuación, se vacía la tabla con la información de los receptores y se compone el paquete APP ACT de acción a partir de los datos de referencia obtenidos.

También, en este caso, la fuente dispone de la opción de configurar un valor de *Máxima Asincronía* entre todo par de receptores. Como en los dos casos anteriores, este valor limitará la máxima desviación permitida entre el receptor más rápido en su proceso de reproducción y el receptor más lento. De esta forma, en el caso de no detectar algún par de receptores que se desviase por encima de ese valor, la fuente no enviaría ningún paquete APP de acción y, por tanto, se reduciría el envío de paquetes de control para evitar una posible sobrecarga y situaciones de congestión de la red.

Ejemplo3. Sincronización al punto de reproducción medio

Tomando el mismo ejemplo que en los dos casos anteriores pero suponiendo, para simplificar, que sólo son tres receptores, aquí la fuente procederá de la siguiente forma:

Se calcula la media de las últimas LDUs reproducidas y de los instantes NTP.

$$LDU_{media} = \frac{\sum_{i=1}^{N} LDU_i}{N} = \frac{48674 + 48680 + 48676}{3} = 48677$$

$$NTP_{media} = \frac{\sum_{i=1}^{N} NTP_i}{N} = \frac{1491954565 + 1491958348 + 1491957841}{3} = 1491956918$$

Estos dos valores se tomarían como referencia para completar el paquete APP ACT de acción.

Ventajas e inconvenientes del algoritmo:

La *ventaja* principal de este algoritmo es que al sincronizarse con el estado medio de reproducción de todos los receptores, habrá menos ajustes y no serán tan pronunciados, siendo más suaves y notándose menos los efectos de los mismos en la reproducción.

Un *inconveniente* asociado a este algoritmo, consiste en que, al ajustarse *todos* los reproductores con respecto a un punto de reproducción medio, no existirá un reproductor *maestro* que no sufra alteraciones (*'saltos'* o *'pausas'*) en su reproducción, como pasaba en los algoritmos anteriores, sino que todos sufrirán adaptaciones en sus procesos de reproducción con las consiguientes discontinuidades provocadas por las mismas.

Por otro lado, al definir una asincronía máxima permitida entre el receptor más lento y el más rápido, podría darse el caso, poco probable pero posible, de que si todo el grupo de receptores tienen un proceso de reproducción que se va retrasando progresivamente, la diferencia entre los puntos de reproducción del receptor más rápido y el más lento puede que no exceda ese valor máximo permitido y sin embargo, el tamaño de los *buffers* necesarios podría ir creciendo de forma permanente, llegando a producir *overflow* de los mismos. También podría darse el caso contrario, poco probable pero también posible, en el que si los procesos de reproducción de todos los receptores se fueran adelantando progresivamente, se podrían producir casos de *underflow* de los *buffers* de algunos receptores.

- Paquete APP ACT de 'acción'

Como resultado de cualquiera de los algoritmos anteriores, siempre que se haya superado la máxima asincronía permitida entre receptores, se generarán mensajes, indicando a los procesos reproductores que realicen las acciones necesarias, como '*saltar*' o '*esperar*' un número determinado de LDUs en el proceso de consumo o reproducción, para conseguir la sincronización.

En nuestro protocolo, los mensajes de *acción*, también serán implementados haciendo uso de mensajes RTCP APP, denominados *paquetes APP ACT*, y que tienen el formato indicado en la figura 4.19.

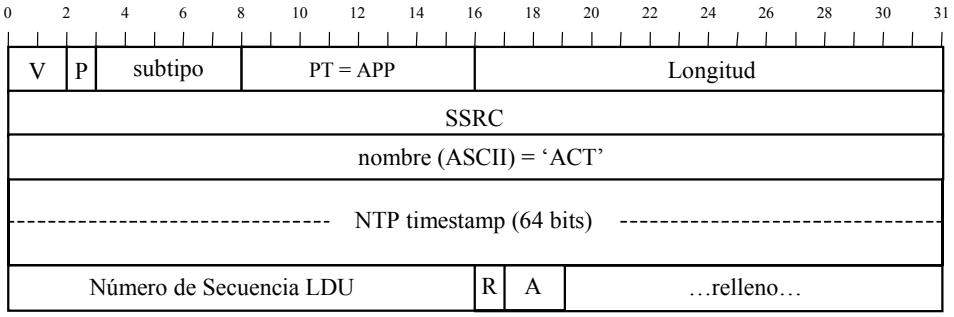


Figura 4.19. Paquete APP ACT de acción

, donde el significado de los campos es el siguiente:

- *Versión* y *Bit de relleno* (*P, padding*): Mismo significado que para los paquetes RTP.
- *Subtipo:* 5 bits que se utilizan para agrupar un conjunto de paquetes APP bajo el mismo nombre. En este caso no se utiliza.
- *PT o Tipo de paquete:* 8 bits que indican que es un paquete RTCP APP y, por tanto, tomarán el valor decimal constante '204', tal como especifica la RFC 1889.
- *Longitud:* 16 bits que indican la longitud menos una, en palabras de 32 bits, del paquete. En este caso, como la longitud del paquete es de 6 palabras de 32 bits, este campo tomará el valor 5.
- *SSRC* o *Synchronization source (32 bits):* identificará la fuente del flujo multimedia, que ha transmitido el paquete.
- *Nombre*: 32 bits que contienen 4 caracteres ASCII de 8 bits con el nombre del paquete, que en este caso será 'ACT'.
- *NTP Timestamp*: 64 bits que contienen, en formato NTP, el instante en el cual los receptores tendrán que reproducir la LDU cuyo número de secuencia vendrá contenida en el siguiente campo.
- *Numero de secuencia LDU*: contiene el número de secuencia de la LDU que los receptores deberán reproducir en el instante NTP contenido en el campo anterior.
- *Bit R:* es el bit de identificación de los paquetes RR extendidos.
- *Bits A*: 2 bits que indican el tipo de algoritmo utilizado por la fuente.
  A = '01', para el algoritmo de sincronización a la media.
  A = '10', para el algoritmo de sincronización al receptor más lento
  A = '11', para el algoritmo de sincronización al receptor más rápido

El bit *R* será necesario para corregir el siguiente *problema* (detectado durante la fase de implementación y pruebas del algoritmo): desde que una fuente envía un paquete APP ACT, de acción, hasta que es recibido por un receptor, y éste realiza los ajustes necesarios, pasa un cierto tiempo en el que dicho receptor es

posible que haya enviado algún paquete RR extendido que llegue a la fuente con información antigua, previa al ajuste realizado, y que, por tanto, ya no debería ser tenido en cuenta por ésta.

Para evitar esta situación se utiliza un bit con valores alternados, es decir tomando valores '1' y '0', alternados en cada ciclo de envío, entendiendo como ciclo el período entre dos mensajes APP ACT consecutivos.

Por tanto, el *bit R* se ha añadido por la necesidad de que las fuentes desechen RR extendidos provenientes de receptores que todavía no habían realizado la acción de '*saltar*' o '*esperar*', como respuesta al último paquete APP ACT enviado por éstas. El funcionamiento es el mostrado en la figura 4.20.

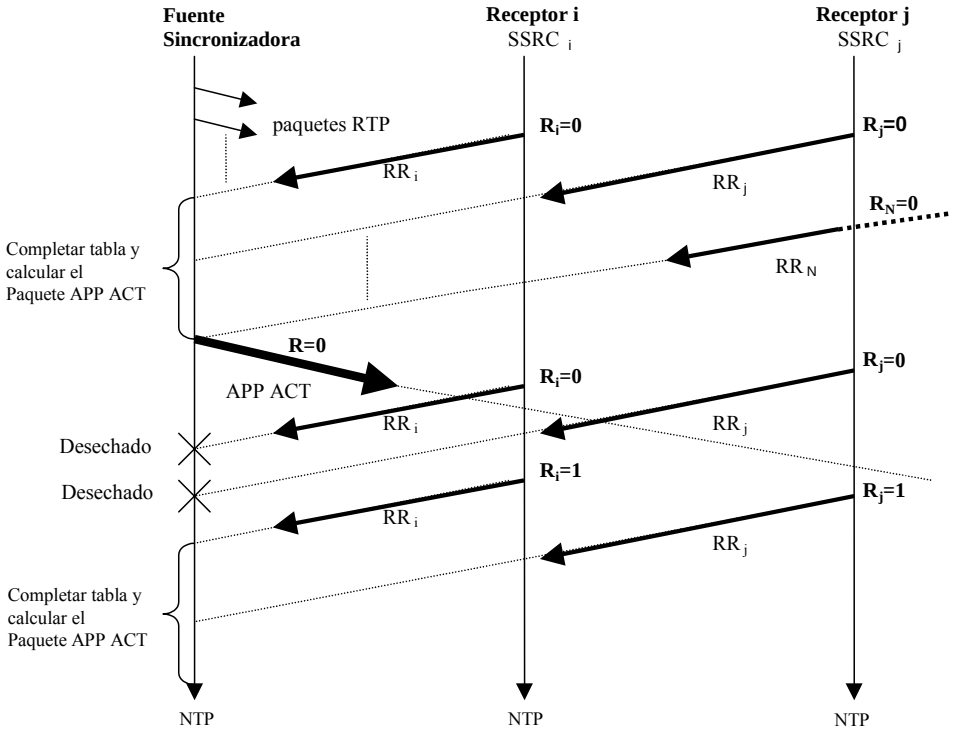


Figura 4.20. Utilidad del bit *R* del paquete APP ACT

En los primeros paquetes RR extendidos que los receptores envían a las fuentes, todos los *bits* R_i se pondrán a '0' (siendo *i* el número de cada fuente, tal y como se puede apreciar en el formato del paquete RR extendido mostrado en la figura 4.15). Cuando cada una de las fuentes haya completado sus tablas (como la tabla 4.1.) con la información proveniente de todos los receptores, ya podrán calcular los campos de los paquetes APP ACT, que, en un primer momento,

enviarán con el *bit R* puesto a '0'. A partir de este momento, dejarán de aceptar paquetes RR extendidos con el *bit* R_i puesto a '0' y sólo tomarán la información de aquellos paquetes RR extendidos que lo lleven activado ('1').

Los receptores, al recibir un paquete APP ACT proveniente de una fuente, con un valor de '1' ó '0' en el *bit R* (en el primer ciclo sería un '0'), éstos mandarán, a partir de ese momento, los siguientes paquetes RR extendidos con el *bit* R_i (siendo *i* el número de dicha fuente) conteniendo el valor contrario al del *bit R* recibido en ese último paquete APP ACT (en la figura, sería un '1') de dicha fuente. En los paquetes APP ACT futuros que dicha fuente envíe a los receptores, el *bit R* irá tomando valores alternados ('1' ó '0') en cada ciclo, para resolver el problema anterior.

- Políticas de Sincronización

Para comprobar el funcionamiento del algoritmo propuesto se proponen dos *políticas de sincronización: Conservadora y Agresiva*.

Siguiendo una *Política Conservadora*, para reducir la cantidad de mensajes que la fuente envía de forma *multicast* a los receptores, el algoritmo permite la opción de elegir o configurar un valor de *Máxima Asincronía Permitida* (normalmente, en milisegundos) entre los receptores según la cual si la fuente detecta una desviación en la sincronización de todos los receptores (con respecto a la referencia *maestra* obtenida) por debajo de dicho valor, no enviará ningún paquete APP ACT de acción. Es decir, que la fuente sólo enviará un paquete APP ACT cuando, por lo menos, alguno de los procesos reproductores de los receptores esté desincronizado con respecto a la referencia *maestra* de la sesión (que dependerá del tipo de algoritmo escogido) un valor superior al elegido (que podrá ser configurado por la propia fuente o por la aplicación que ésta ejecute). Con esto se puede evitar el envío de paquetes de control innecesarios que produzcan una posible sobrecarga o situaciones de congestión en la red, malgastando ancho de banda, reduciendo así la sobrecarga producida por el algoritmo.

Según esta política, se considerará que existe sincronía cuando la reproducción de los receptores se produce dentro de una ventana de sincronización de *W* LDUs. El tamaño de esta ventana dependerá de los requisitos de sincronización que se deseen alcanzar y, en definitiva, de la calidad de la sincronización requerida por la aplicación. A mayor ventana de sincronización menos exactitud en la sincronización estaremos exigiendo, pero menor número de paquetes APP ACT se enviarán y, por tanto, menos *'saltos'* o *'pausas'* se producirán en los procesos de reproducción de los receptores. Sin embargo, cuando estos ajustes se realicen serán más pronunciados (mayores discontinuidades) y se notarán más en el proceso de reproducción.

La utilización de una política conservadora, parece aconsejable en entornos de trabajo estables en cuanto a retardos en red y desajustes en los procesos de sincronización. Sin embargo, bajo ciertas condiciones de red y requisitos de las aplicaciones, será conveniente realizar una resincronización cuando se detecte una ligera posibilidad de asincronía y, así, evitar que ésta aumente y supere el valor máximo permitido. La idea subyacente de la *Política Agresiva* consiste en enviar paquetes APP ACT, incluso si no existe asincronía en el momento de la resincronización, para corregir una posible asincronía posterior no esperada. Dado el funcionamiento de esta política, a priori, el algoritmo puede ser válido en un entorno no estable donde las causas de asincronía son numerosas y sea necesario un elevado número de resincronizaciones. En el caso de un entorno estable, por el contrario, no sólo no introducirá ventajas, sino que existirá el peligro de que se resincronice en determinados casos donde incluso no sea necesario.

- Cálculo de los valores LDU_{act} y NTP_{act}

Una vez finalizados los procesos, sea cual sea el algoritmo escogido de los tres anteriores propuestos, se dispondrá de un valor de LDU y otro de tiempo NTP que servirá de referencia a la fuente para el cálculo del valor del número de secuencia de la LDU y el valor del campo NTP del paquete APP ACT a enviar a los receptores y que tendrá el efecto de producir un '*salto*' o una '*pausa*' en la reproducción de los mismos, en función de si están retrasados o adelantados con respecto a dicha referencia, respectivamente.

Para el valor del número de secuencia de la LDU se tomará el número de secuencia correspondiente al próximo paquete RTP que la fuente tiene preparado para transmitir. Así el valor del campo LDU del paquete APP ACT, que denominamos LDU_{act} será:

LDU_{act} = *Número de secuencia del próximo paquete a transmitir por la fuente*

El valor introducido en el campo NTP del paquete APP ACT, que denominamos NTP_{act}, se calculará a partir de los valores obtenidos como resultado de la ejecución del algoritmo escogido anteriormente, que, por analogía, denominaremos $LDU_{maestro}$ y $NTP_{maestro}$ (en el caso de sincronización a la media, estos valores se corresponderán con los valores obtenidos LDU_{media} y NTP_{media}).

El procedimiento a seguir será el siguiente: primero, se calcula la diferencia entre los números de secuencia del paquete que se dispone a enviar la fuente en este instante y el número de secuencia obtenido como $LDU_{maestro}$. A partir de esta diferencia, si se multiplica por la tasa de generación/consumo (θ), en unidades NTP ($\theta(ntp)$), obtendremos el tiempo que habrá transcurrido desde que se transmitió la

LDU que estaba reproduciendo el proceso de reproducción de referencia (tomado como *maestro*) y la LDU que se va a enviar ahora.

$$Dif\_LDU(ntp) = (LDU_{act} - LDU_{maestro}) * \theta(ntp) \quad \text{[Ec. 4.12]}$$

Si las tasas de generación y consumo de LDUs de la fuente y del receptor *maestro* fueran las mismas exactamente, es decir, que no existiera *drift* o variación en la tasa de reproducción con respecto a la de generación de la fuente, sumando este tiempo al tiempo NTP obtenido del receptor *maestro*, obtendríamos el instante de tiempo en unidades NTP en el que dicho receptor deberá reproducir la futura LDU con número de secuencia LDU_{act}, suponiendo que sigue reproduciendo al mismo ritmo y que dicho ritmo no sufre variación alguna. No obstante, como la tasa de envío de la fuente (θ) puede diferir con respecto a la tasa de consumo del receptor *maestro*, existirá un posible intervalo de consumo $I_c[LDU_{act}]$, donde podemos estar seguros que el receptor *maestro* va a reproducir dicha LDU. Dicho intervalo se representa en la figura 4.21.

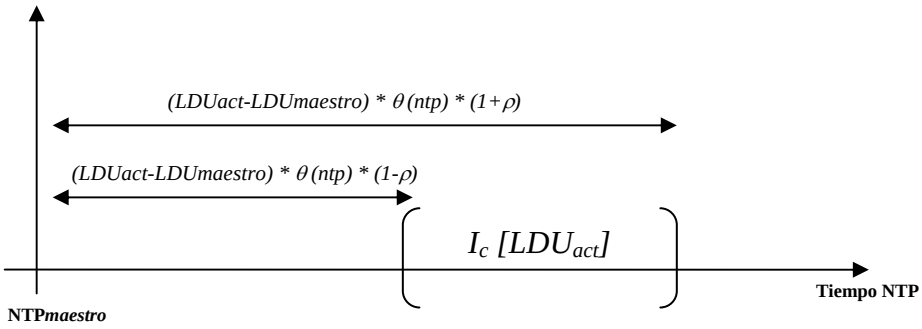


Figura 4.21. Cálculo del instante de ajuste en el algoritmo propuesto

El intervalo $I_c[LDU_{act}]$ estará comprendido entre los dos valores I_{c1} e I_{c2}, calculados según las siguientes ecuaciones:

$$I_c[LDU_{act}] = [I_{c1}, I_{c2}]$$

$$I_{c1} = NTP_{maestro} + (LDU_{act} - LDU_{maestro}) * \theta(ntp) * (1 + \rho) \quad \text{[Ec. 4.13]}$$

$$I_{c2} = NTP_{maestro} + (LDU_{act} - LDU_{maestro}) * \theta(ntp) * (1 - \rho) \quad \text{[Ec. 4.14]}$$

, en las que ρ es el *drift* o variación del período o tasa de reproducción en los receptores.

Como se puede observar, estas ecuaciones no dependen del *jitter* introducido por la red, y, por tanto, el error producido por el *jitter* no afecta al algoritmo. Mientras que en el protocolo *Feedback* ([RAN92], [RAM92], [RAM93], [RAM93b], [RAN95a], [RAN95b]) el *jitter* introducía un error, en el algoritmo propuesto en la presente Tesis no se produce dicho error. Por otra parte, dependiendo del tipo de aplicación, el error producido por el protocolo de tiempo global NTP podrá despreciarse, por ser de unos pocos milisegundos.

Para la implementación del algoritmo en nuestras aplicaciones de prueba se tomará el valor medio del intervalo de tiempo anterior para introducirlo en el campo correspondiente del paquete APP ACT:

$$NTP_{act} = NTP_{maestro} + Dif\_LDU(ntp) * \theta(ntp) \qquad \text{[Ec. 4.15]}$$

, con lo que asumimos un error limitado a $\pm\rho$.

Al recibir el paquete APP ACT con la LDU transmitida y este valor de tiempo, los procesos reproductores de los receptores, considerados como *esclavos*, harán una '*pausa*' si están adelantados en su reproducción o realizarán un 'salto' en su reproducción, si están atrasados, para alcanzar el punto de reproducción indicado en dicho paquete.

Veamos cómo se realizaría el cálculo en cada uno de los tres ejemplos anteriores, suponiendo que el transmisor se dispone a transmitir una LDU con número de secuencia de 48690 (LDU_{act}).

Cálculo de NTP_{act} para el ejemplo 1

En el ejemplo 1, el receptor *maestro* obtenido fue el N, por lo que finalmente los campos LDU_{act} y NTP_{act} que componen el paquete APP ACT serían los siguientes:

$$LDU_{act} = 48690$$

$$Dif\_LDU = LDU_{act} - LDU_{maestro} = 48690 - 48676 = 14$$
$$Dif\_LDU(ms) = Dif\_LDU * \theta\ (ms) = 14 * 20 = 280\ ms$$
$$Dif\_LDU(ntp) = conversión\_ntp(Dif\_LDU(ms)) = 18350{,}08$$
$$NTP_{act} = NTP_{maestro} + Dif\_LDU(ntp) = 1491972915$$

Esto hará que los procesos reproductores de los demás receptores *esclavos* (receptores 1 a N-1) sufran una '*pausa*' en su reproducción para sincronizarse al proceso del receptor 1 que era el más lento.

Cálculo de NTP_{act} para el ejemplo 2

En este caso, el receptor elegido como *maestro* fue el receptor 2, quedando los valores de los campos LDU_{act} y NTP_{act} que componen el paquete APP ACT:

$$LDU_{act} = 48690$$
$$Dif\_LDU = LDU_{act} - LDU_{maestro} = 48690 - 48680 = 10$$
$$Dif\_LDU(ms) = Dif\_LDU * \theta\ (ms) = 10 * 20 = 200\ ms$$
$$Dif\_LDU(ntp) = conversión\_ntp(Dif\_LDU(ms)) = 13107{,}2$$
$$NTP_{act} = NTP_{maestro} + Dif\_LDU(ntp) = 1491971455$$

Esto hará que los procesos reproductores de los demás receptores *esclavos* (receptores 1 y 3 a N-1) sufran un '*salto*' en su reproducción para sincronizarse al proceso del receptor 2 que era el más rápido.

Se puede apreciar, con respecto al resultado del algoritmo anterior, que la fuente indicará a los receptores que reproduzcan el mismo número de secuencia en un instante de tiempo NTP anterior (22,28 milisegundos antes).

$$Diferencia(ms) = conversión\_ms(NTP_{act\ lento} - NTP_{act\ rápido}) = 22{,}28\ ms$$

Cálculo de NTP_{act} para el ejemplo 3

Con los valores medios obtenidos en dicho ejemplo, se pasaría a calcular directamente los valores para componer el paquete APP ACT

$$LDU_{act} = 48690$$
$$Dif\_LDU = LDU_{act} - LDU_{media} = 48690 - 48677{,}33 = 12{,}66$$
$$Dif\_LDU(ms) = Dif\_LDU * \theta\ (ms) = 12{,}66 * 20 = 253{,}2\ ms$$
$$Dif\_LDU(ntp) = conversión\_ntp(Dif\_LDU(ms)) = 16593{,}71$$
$$NTP_{act} = NTP_{media} + Dif\_LDU(ntp) = 1491973511$$

Esto hará que los procesos reproductores de todos los receptores sufran una '*pausa*' o un '*salto*' en su reproducción según estuvieran adelantados o retrasados, respecto al proceso 'virtual' de referencia, respectivamente.

- Acciones en el receptor.

El proceso reproductor de cada receptor, cuando reciba el paquete APP ACT de acción, deducirá si debe realizar un '*salto'* o una *'pausa'*, comparando el número de LDU y el tiempo NTP contenido en dicho paquete con la LDU que esté reproduciendo en ese instante y el tiempo NTP local en ese momento. En definitiva, con estas acciones se consigue *aumentar la velocidad* de algunos

procesos reproductores y *disminuir la velocidad* de otros, mediante discontinuidades en su reproducción, de manera que, al final, todos los receptores reproducirán el flujo *maestro* procedente de la fuente sincronizadora al mismo tiempo y de forma sincronizada.

A continuación, se presenta, en pseudocódigo, el proceso que implementará cada receptor para poder sincronizarse con los valores obtenidos de la recepción de un paquete APP ACT. Primero, se guardarán los valores incluidos en los campos del paquete APP ACT correspondientes al número de secuencia LDU y el instante NTP con el que se reproducirá dicha secuencia.

LDU_{act} = *número de secuencia LDU.*
NTP_{act} = *instante NTP de reproducción de la* LDU_{act}.

Después, el receptor lo comparará con su última secuencia reproducida (LDU_{rep}) y su instante de reproducción (NTP_{rep}) según el siguiente proceso.

```
IF (LDU_act > LDU_rep)    THEN
        Dif_LDU = LDU_act - LDU_rep
        Dif_LDU(ms) = Dif_LDU * θ (ms)

        IF (NTP_act > NTP_rep) THEN
                Dif_NTP = NTP_act - NTP_rep
                Dif_NTP(ms) = conversión_ms(Dif_NTP)
                IF (Dif_NTP(ms) > Dif_LDU(ms)) THEN
                        Ajuste = Dif_NTP(ms) - Dif_LDU(ms)
                        Playout_Delay(ms) = Playout_Delay(ms)+ Ajuste
                ELSE
                        Ajuste = Dif_LDU(ms) - Dif_NTP(ms)
                        Playout_Delay(ms) = Playout_Delay(ms)- Ajuste
                END IF
        ELSE
                Dif_NTP = NTP_rep - NTP_act
                Dif_NTP(ms) = conversión_ms(Dif_NTP)
                Ajuste = Dif_LDU(ms) + Dif_NTP(ms)
                Playout_Delay(ms) = Playout_Delay(ms)- Ajuste
        END IF
END IF
```

En los casos que el receptor tenga una reproducción adelantada, éste aumentará el margen de tiempo entre el instante de recepción y la reproducción de la información contenida en el paquete de datos RTP (variable *Playout_Delay* en el pseudocódigo) que se está procesando para corregir dicho adelanto y sincronizarse con los demás receptores (efecto de '*pausa*'). En caso contrario, disminuirá dicho

margen para reproducir en un menor tiempo los paquetes RTP recibidos (efecto de '*salto'*).

Procediendo de esta forma, no se contribuye a provocar una posible situación de *overflow* o *underflow* de los *buffers* del receptor, como se produce en [RAN92] con la utilización de *buffers* de capacidad limitada en los receptores, ya que la única acción que se realiza es la de retrasar o adelantar la reproducción de la LDU que está llegando en ese instante al receptor, según si el resultado de la comparativa (con la información del paquete APP ACT) indica que el receptor está adelantado o retrasado, respectivamente. Esto se consigue simplemente aumentando o disminuyendo el valor del *playout delay* o retardo de reproducción de dicha LDU, respectivamente.

Por otra parte, la fuente utiliza los *bits A* del paquete APP ACT para indicar a los receptores qué tipo de algoritmo está ejecutando. De esta manera, si se está utilizando algunos de los dos algoritmos de sincronización, tomando como referencia el receptor más lento o al más rápido, los receptores deberán tener las siguientes consideraciones:

- En el caso de estar utilizándose el algoritmo de sincronización tomando como referencia al receptor más lento (bits A = '10'), ningún receptor adelantará su reproducción al recibir el paquete APP ACT, ya que siempre se deberá atrasar para sincronizarse al más lento.

- De la misma forma, en el caso de estar utilizándose el algoritmo tomando como referencia al receptor más rápido (los bits A tomarán los valores '11'), ningún receptor retrasará su reproducción al recibir el paquete APP ACT, ya que siempre se deberá adelantar para sincronizarse al más rápido.

- Ejemplo

Supongamos que nos encontramos en la situación del ejemplo de la Figura 4.17 y que se han obtenido los campos LDU y NTP del paquete APP ACT mediante el algoritmo para el cálculo del estado de reproducción medio. Los valores obtenidos fueron:

$$LDU_{act} = 48690$$
$$NTP_{act} = 1491973511$$

Los receptores, al recibir el paquete APP ACT calcularán inmediatamente el número de secuencia de la última LDU reproducida (LDU_{rep}) y el instante NTP en el cual se ha reproducido (NTP_{rep}). A continuación, lo compararán con los valores

del paquete APP ACT recibido para saber si deben *esperarse* o *adelantarse* en su reproducción con el fin de sincronizarse con los demás receptores.

Por ejemplo, supongamos que el receptor 1, en el momento de recibir el paquete APP ACT está reproduciendo la LDU_{rep} 48682 en el siguiente instante NTP_{rep}:

$$LDU_{rep} = 48682$$
$$NTP_{rep} = 1491965050$$

Primero, se calculará la diferencia de tiempos en la mismas unidades (milisegundos) con respecto a los valores del paquete APP ACT:

$LDU_{act} > LDU_{rep}$ THEN
$\text{Dif\_LDU} = LDU_{act} - LDU_{rep} = 48690 - 48682 = 8$
$Dif\_LDU(ms) = Dif\_LDU * \theta\ (ms) = 8 * 20 = 160\ ms$

$NTP_{act} > NTP_{rep}$ THEN
$Dif\_NTP = NTP_{act} - NTP_{rep} = 8461$
$Dif\_NTP(ms) = conversión\_ms(Dif\_NTP) = 129{,}10\ ms$

$Dif\_LDU(ms) > Dif\_NTP(ms)$ THEN
$Ajuste = Dif\_LDU(ms) - Dif\_NTP(ms) = 30{,}9\ ms$
$Playout\_Delay(ms) = Playout\_Delay(ms) - Ajuste$

Como se puede apreciar, en este receptor, si siguiera su reproducción normal, se reproduciría la secuencia LDU_{act} pasados 160 milisegundos pero, tal como nos indica el valor de *Dif_NTP*, para estar sincronizada la reproducción de dicha secuencia con la referencia *maestra*, deberá ser reproducida al cabo de 129,10 milisegundos, por lo que el proceso reproductor del receptor 1 deberá '*adelantarse'* o '*saltar'* para que la LDU_{act} se reproduzca en el instante indicado. Se adelantará exactamente 30,9 milisegundos restándoselos al valor que había calculado del *Playout_Delay*.

4.2.3. Sincronización Inter-flujo (local, varios flujos)

De forma paralela a la sincronización gruesa y fina entre los receptores respecto al flujo *maestro*, en cada receptor deberá existir un proceso que asegure una sincronización interna entre los diferentes tipos de flujos que reproduzca dicho receptor y que garantice que se mantengan entre ellos las mismas relaciones temporales que existían en el momento de la captura (sincronización inter-flujo).

En este caso, también se ha escogido un *mecanismo de sincronización maestro/esclavo*, donde se tomará uno de los flujos de datos como flujo *maestro* local, al que se sincronizarán el resto de flujos, denominados *esclavos* locales. Normalmente el flujo *maestro* local será aquel que tenga más restricciones en cuanto a calidad de servicio (como suele ser el audio) o bien vendrá determinado por el tipo de aplicación empleada. En cada receptor, las LDUs del resto de flujos *esclavos* locales sufrirán *mecanismos de espera o eliminación* (*delay-or-drop*) para que los procesos reproductores sincronicen su reproducción con la de las LDUs del flujo *maestro* local, por lo que las discontinuidades en la reproducción de los flujos *esclavos* locales serán inevitables.

Como cada proceso reproductor utiliza una referencia temporal diferente, como es el caso de los procesos de los flujos de audio y vídeo, estos deberán hacer uso de la información proporcionada por RTCP y NTP para obtener un formato y un tiempo común para poder negociar sus puntos de reproducción. El esquema propuesto (con una única fuente, para no complicarlo demasiado) se refleja en la figura 4.22. En este apartado se tratará la sincronización local entre flujos (en rojo en la figura).

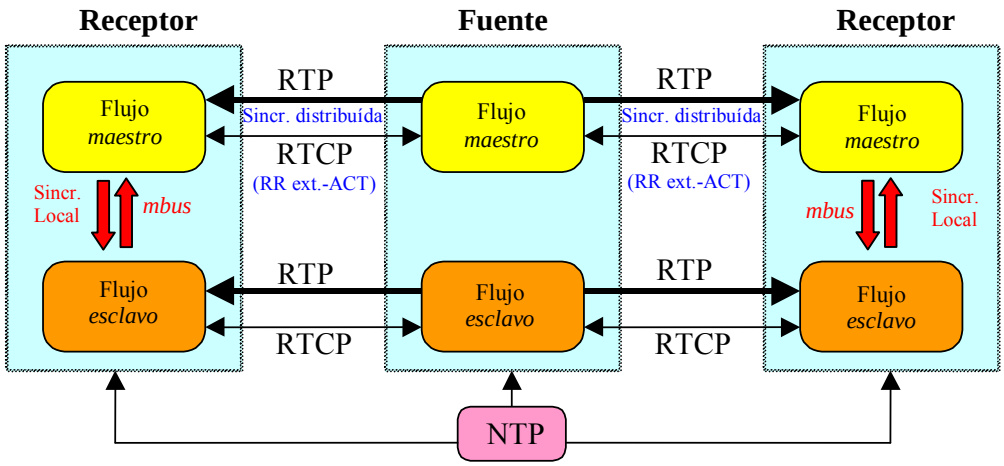


Figura 4.22. *Mbus* local: Sincronización inter-flujo

La negociación se realizará enviándose mutuamente mensajes a través de *mbus* conteniendo la información del retardo de reproducción (*playout delay*) del proceso reproductor de cada flujo, calculado en base al eje de tiempo NTP de referencia común, siguiéndose, a partir de dichos datos, un proceso de sincronización entre procesos (inter-flujo).

Existen diferentes opciones para este proceso de sincronización. Una posible solución (implementada en las herramientas *vic* y *rat* originales) consiste

en sincronizar los puntos de reproducción de todos los reproductores al del proceso reproductor más lento (es decir, aquel que en el momento del intercambio tenga un valor de *playout delay* más elevado), con lo que todos los procesos '*esperarán*' en su reproducción al proceso más lento (el flujo de datos reproducido por este proceso se podría considerar como el flujo *maestro* de la sincronización inter-flujo). Dicho proceso se muestra, de forma genérica para dos flujos, en el diagrama de la figura 4.23.

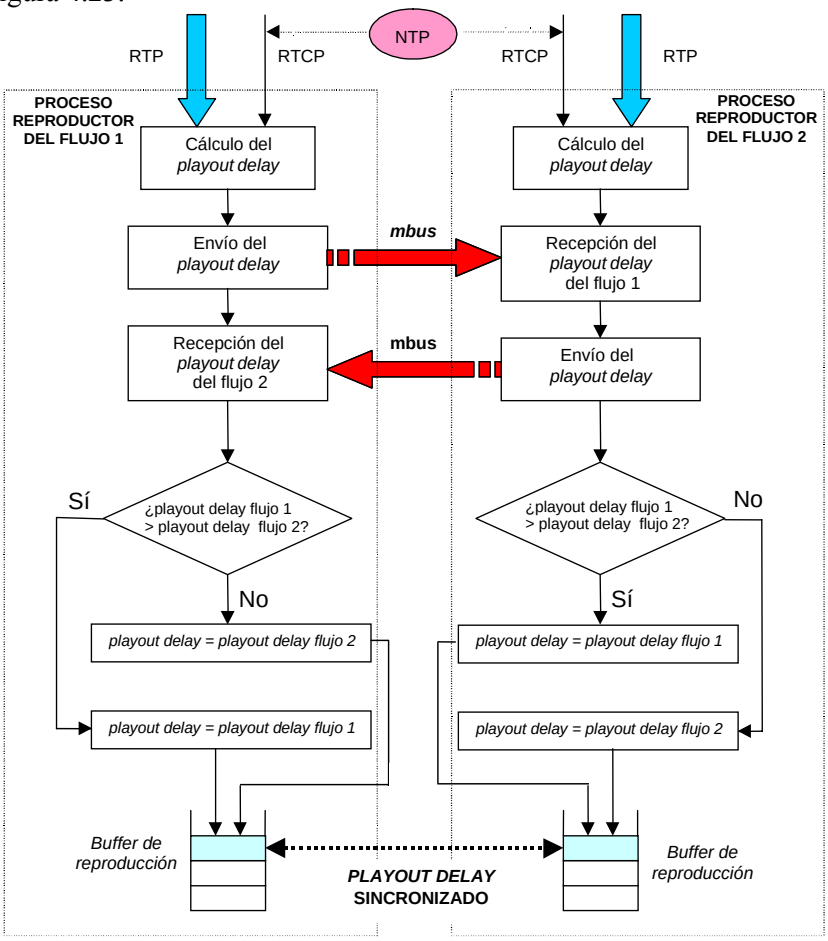


Figura 4.23. Esquema genérico de sincronización inter-flujo

El método mostrado en la figura trata a todos los flujos y sus correspondientes procesos reproductores por igual (no existe un flujo determinado como *maestro*, sino que, en cada instante el proceso reproductor más rápido se espera al del flujo más lento). Sin embargo, tal como se comentó en el capítulo anterior, los flujos multimedia tienen características diferentes. Como ejemplo, podemos citar que un número elevado de ajustes en el proceso reproductor del flujo

de audio puede ser muy perceptible por los usuarios, mientras que no lo son tanto si se producen en el flujo de vídeo.

El esquema anterior deberá ser modificado en nuestro caso ya que al integrar el proceso de sincronización inter-flujo con el de la sincronización de grupo, para que las acciones de sincronización de grupo (que se realicen sobre los procesos reproductores del flujo *maestro*) afecten a los demás flujos *esclavos* locales, interesará que exista un flujo *maestro* local que, además, coincida con el flujo *maestro* utilizado para la sincronización de grupo distribuida, a cuyo proceso reproductor deberán sincronizarse los procesos reproductores de los demás flujos locales considerados como *esclavos* locales.

Al tomar como flujo *maestro* local, el mismo flujo considerado como *maestro* en la sincronización de grupo entre receptores, cada vez que se produzca algún ajuste de sincronización entre receptores (de grupo), automáticamente este cambio se reflejará en la reproducción del flujo *maestro* local y, a su vez, gracias al mecanismo de sincronización local entre flujos (inter-flujo), este cambio también afectará a los procesos reproductores de los flujos *esclavos locales*, produciendo un efecto de sincronización de todos los flujos en todos los receptores a la vez y, por tanto, se obtendrá la sincronización multimedia buscada.

Si para la sincronización inter-flujo hubiéramos elegido como flujo *maestro* local a cualquier otro flujo (por ejemplo el de vídeo) distinto del tomado como *maestro* en la sincronización de grupo entre receptores (por ejemplo, el de audio), el proceso reproductor del flujo *maestro* en la sincronización de grupo (de audio) realizaría ajustes en su proceso de reproducción por dos motivos: por un lado, para sincronizarse a los demás receptores y, por otro lado, para sincronizarse a la reproducción del flujo *maestro* local (de vídeo). Además, al ajustarse al flujo *maestro* local, es posible que se desajustara con respecto a la reproducción sincronizada con los demás receptores (basada en el flujo de audio), lo cual afectaría a dicha sincronización, y así sucesivamente. Esto podría suponer continuos ajustes en el proceso reproductor de dicho flujo lo cual podría llevar a afectar a la inteligibilidad del mismo.

Al tomar el mismo flujo como *maestro* para ambos tipos de sincronización, no será necesario que los flujos *esclavos* se comuniquen con sus correspondientes fuentes mediante mensajes de realimentación (es decir, no será necesario que envíen paquetes RR extendidos) ni tampoco que éstas les envíen paquetes APP de 'acción'. Sólo será necesario este proceso (de sincronización de grupo) entre la fuente del flujo *maestro* (que hemos denominado *fuente sincronizadora*) y los receptores del mismo, tal como se muestra en la figura 4.22, en azul.

En el caso de la sincronización fina entre receptores, cada vez que el proceso reproductor del flujo *maestro* local reciba un paquete APP ACT, de

'acción', es decir, que realice algún '*salto*' o '*espera*' en su proceso de reproducción, informará de dicho cambio a los reproductores de los flujos *esclavos* mediante un mensaje local interno, utilizando el bus local de intercomunicación entre procesos, *mbus* (en rojo, en la figura).

De esta manera, por un lado, el proceso reproductor del flujo *maestro* sólo modificará el valor de su *playout delay* por acciones de resincronización debidas a la sincronización de grupo y, por otro lado, los procesos reproductores del resto de flujos *esclavos* locales lo modificarán por acciones de resincronización debidas a la sincronización inter-flujo (que implícitamente se verán afectadas por los ajustes anteriores). De esta manera, se evita que los dos tipos de sincronización afecten a un mismo flujo con continuos ajustes que podrían llegar a ser, en algunos casos, incluso opuestos.

Por tanto, el proceso de sincronización inter-flujo elegido va a ser un mecanismo *maestro*/*esclavo*, en el que el proceso que comunicará el valor de su *playout delay* será el proceso reproductor del flujo considerado como *maestro* para la sincronización de grupo explicada. Los demás procesos reproductores, al conocer el valor del *playout delay* del proceso *maestro* lo compararán con el suyo propio y, si van atrasados o adelantados (por encima de un valor umbral para evitar continuos ajustes), realizarán las correcciones oportunas para acercarse al estado de reproducción en el que se encuentre el proceso del flujo *maestro*, acortando o aumentando el valor de su *playout delay*, respectivamente.

En la figura 4.24, se muestra el esquema de sincronización inter-flujo propuesto en la Tesis, que seguirán los procesos reproductores del flujo *maestro* y de un flujo *esclavo*, mediante el paso de mensajes vía *mbus*, para conseguir la reproducción sincronizada de ambos.

Cada proceso reproductor, de forma independiente, calcula el retardo de reproducción o *playout delay* de cada LDU recibida según se explicará posteriormente. A continuación, el proceso reproductor del flujo *maestro* comunica el valor de su *playout delay* al proceso *esclavo* utilizando el canal *mbus* interno. De esta forma, el proceso reproductor del flujo *esclavo*, comparará el valor recibido con el valor de su propio *playout delay* y descubrirá si está adelantado o retrasado en su reproducción por encima de un *valor umbral* con respecto al proceso reproductor del flujo *maestro*. Si el adelanto o retraso supera el valor umbral de asincronía, actualizará el valor de su *playout delay* igualándolo al recibido del proceso reproductor del flujo *maestro*. El valor umbral se escogerá según los valores de asincronía máxima permitida entre los procesos reproductores de ambos flujos. De esta manera se reduce el número de ajustes efectuados en los procesos reproductores de los flujos *esclavos* afectando lo mínimo posible dichos procesos.

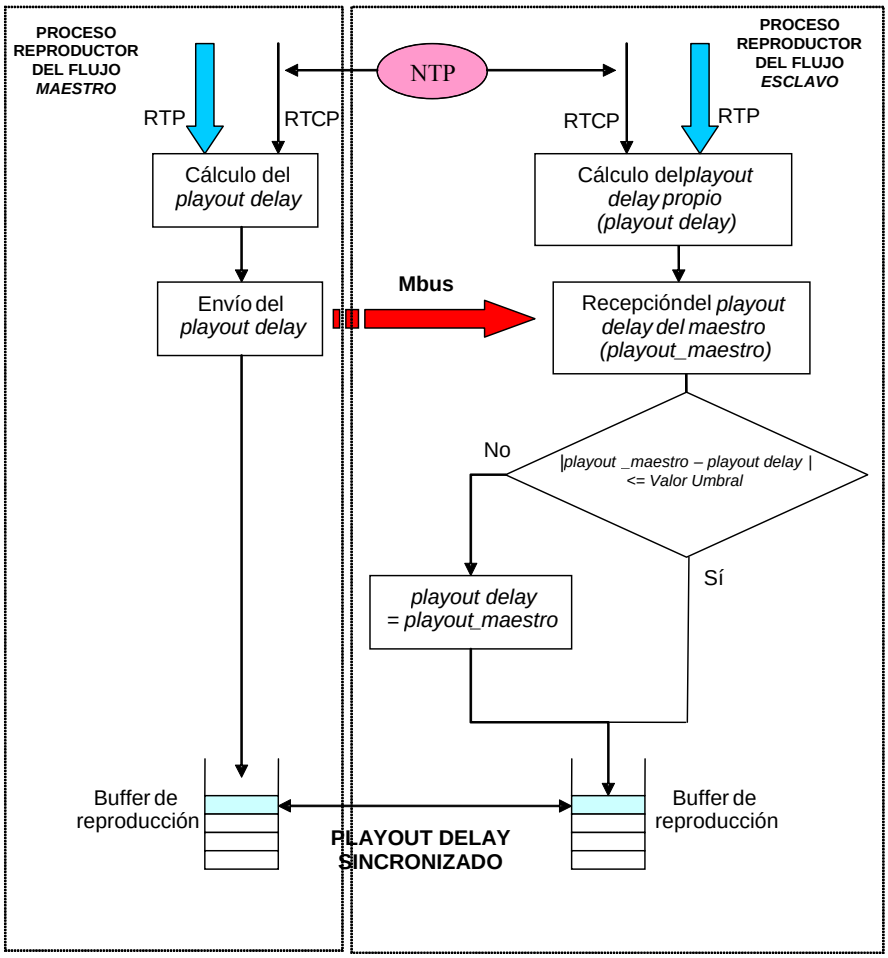


Figura 4.24. Esquema de sincronización inter-flujo propuesto

Según el esquema anterior, en un primer momento, el proceso reproductor del flujo *esclavo*, mientras no reciba un mensaje *mbus*, reproducirá el contenido de los paquetes RTP tal y como los vaya recibiendo. A partir del momento en que se empiece a recibir mensajes *mbus* indicando el valor del *playout delay* del proceso reproductor del flujo *maestro*, empezará a utilizar el *buffer* de reconstrucción, si es necesario, en caso de tener que ralentizar o acelerar su reproducción.

Al final, se consigue que en la reproducción de los paquetes de ambos flujos, generados y enviados al mismo tiempo, por la o las fuentes transmisoras, se almacenen en los *buffers* de reproducción de las aplicaciones correspondientes, con los mismos retardos de reproducción o *playout delay* (o con diferencias inferiores al umbral establecido de antemano) y, por tanto, se reproduzcan sincronizados.

Normalmente, tal y como se ha comentado anteriormente, el flujo de audio se tomará como flujo *maestro* y los procesos reproductores de los demás flujos *esclavos* serán los encargados de sincronizarse con el proceso reproductor del mismo. En la figura 4.25 se muestra como ejemplo el caso de sincronización entre audio y vídeo (*lip-sync*). Tal y como muestra dicha figura, será necesario utilizar un *buffer* de reconstrucción para los flujos *esclavos* (vídeo, en la figura), de forma que su *playout delay* pueda adaptarse al del flujo *maestro* (audio, en este caso), garantizando que las tramas de vídeo recibidas se muestren de manera constante y predecible.

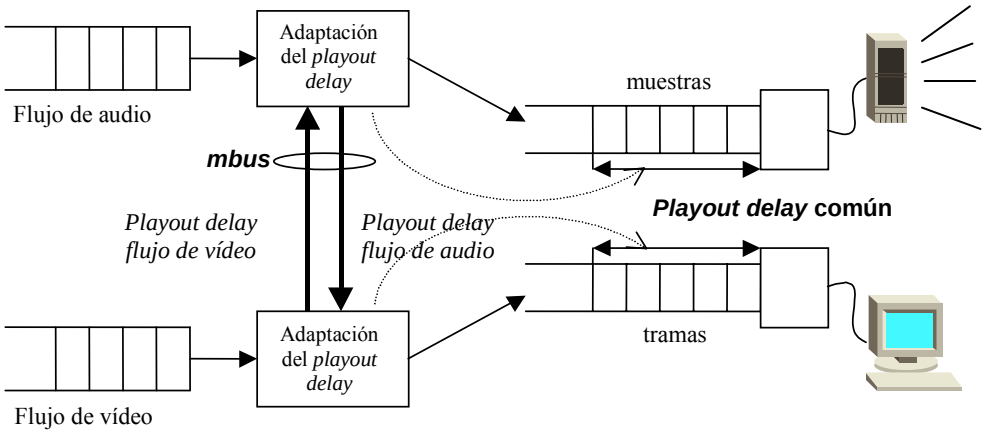


Figura 4.25. Sincronización labial inter-flujo

4.2.3.1. Método utilizado para el cálculo del valor del *playout delay*.

A continuación, en la figura 4.26 se presenta cómo un proceso reproductor de un flujo multimedia calcula el valor del *playout delay* de las LDUs, a medida que las va recibiendo.

La tabla 4.3 contiene el significado de cada uno de los paquetes y variables que aparecen en la figura. En adelante, el término *NTP timestamp* hará referencia a las marcas de tiempo global utilizadas en los paquetes RTCP SR para poder obtener una referencia de reloj común entre diferentes flujos multimedia. Por otro lado, el término *RTP timestamp* hará referencia a las marcas de tiempo cuyos valores dependerán del flujo multimedia y de la frecuencia de muestreo utilizada en cada dispositivo. Estas últimas se utilizan tanto en la cabecera de los paquetes RTP como en la de los paquetes RTCP SR.

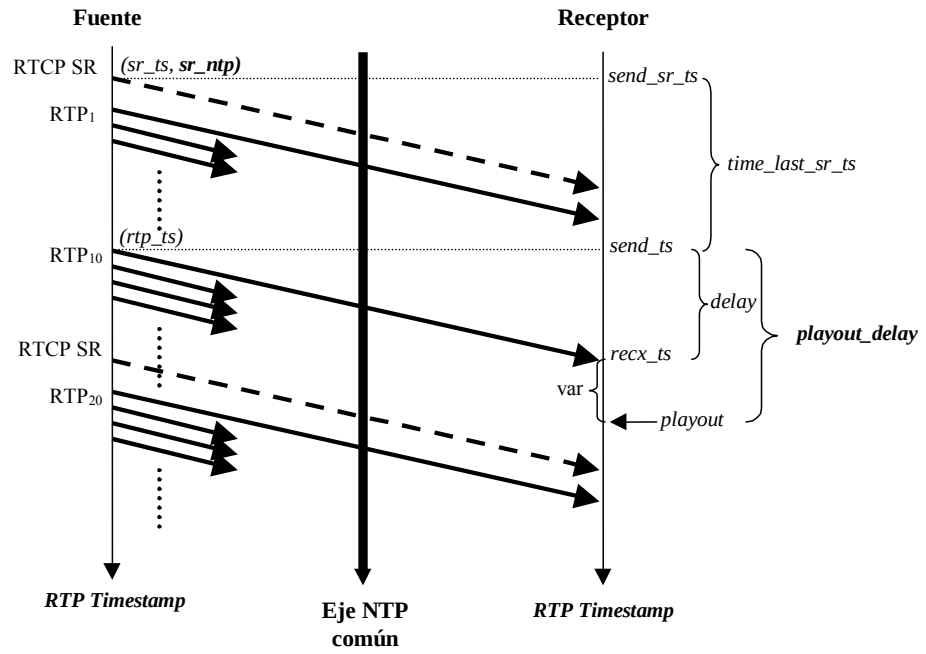


Figura 4.26. Proceso de cálculo del *playout delay*

PAQUETES Y VARIABLES	SIGNIFICADO
En la Fuente	
RTCP SR	Paquete de informe RTCP SR enviado por la fuente
sr_ts	Marca temporal del paquete RTCP SR en unidades *RTP Timestamp* según la referencia de la fuente
sr_ntp	Marca temporal del paquete RTCP SR en unidades *NTP Timestamp*
RTP$_i$	Paquete i-ésimo transmitido
rtp_ts	Marca temporal del paquete RTP en unidades *RTP Timestamp* según la referencia de la fuente
En el Receptor	
send_sr_ts	Instante de transmisión del paquete RTCP SR medido en unidades *RTP Timesptamp* tomando como referencia el eje del receptor
send_ts	Instante de transmisión del paquete RTP medido en unidades *RTP Timesptamp* tomando como referencia el eje del receptor
time_last_sr	Tiempo transcurrido, en unidades *RTP Timesptamp*, entre el envío del paquete y el envío del último informe RTCP SR
recx_ts	Instante de recepción del paquete RTP medido en unidades *RTP Timesptamp* tomando como referencia el eje del receptor
delay	Retardo de tránsito de un paquete RTP. Dependerá, principalmente, del retardo introducido por la red para cada paquete.
playout delay	Retardo de reproducción del paquete, desde el instante en que es transmitido hasta el instante en que se reproduce.
var	Componente del *playout delay* variable que depende en gran medida del *jitter* de la red, calculada por el mecanismo de sincronización intra-flujo utilizado en las aplicaciones.
playout	Instante de reproducción del contenido del paquete RTP

Tabla 4.3. Significado de los paquetes y variables de la figura

En la figura anterior se muestra un ejemplo de una transmisión continua de un flujo multimedia. Se puede apreciar cómo se calcula el valor del *playout delay* para uno de los paquetes de datos recibido (en la figura, el décimo paquete transmitido, denominado RTP_{10}).

Cuando se recibe un paquete, lo primero que se debe calcular es el retardo sufrido en la red (*delay*). Para ello, en el momento de la recepción del paquete se habrá calculado el valor en unidades *RTP timestamp* del instante de recepción del paquete, según la referencia de tiempos RTP del receptor (*recx_ts*). A continuación, se extraerá la marca de tiempos del instante de su transmisión (que viene en la cabecera del propio paquete RTP, en unidades *RTP timestamp*, pero según la referencia de tiempos RTP de la fuente), que se deberá '*mapear'* a la referencia de tiempos del receptor (esta operación será explicada posteriormente). De esta manera se obtiene el instante de envío en unidades *RTP Timestamp* según la referencia de tiempos RTP del receptor (*send_ts*). Esta cantidad se restará al valor del tiempo de recepción del paquete, *recx_ts*, y el valor obtenido será el del retardo sufrido (*delay*) por el paquete RTP, desde su envío hasta su recepción, en unidades *RTP timestamp* según la referencia de tiempos RTP del receptor. Este retardo, como es lógico, dependerá del retardo introducido por la red para cada paquete.

$$delay = (recx\_ts) - (send\_ts) \qquad \text{[Ec. 4.16]}$$

El valor correspondiente al *playout delay* se obtendrá sumando al retardo calculado anteriormente una componente variable (que denominamos *var*), que será, lógicamente, función del *jitter* introducido por la red.

$$playout\_delay = delay + var(jitter) \qquad \text{[Ec. 4.17]}$$

El valor del *jitter* se calculará de la manera indicada en el anexo 8 de la RFC 1889. El valor de la componente *var* en cada momento será el adecuado para garantizar una reproducción continua del flujo (sincronización intra-flujo) y dependerá del algoritmo escogido para obtener dicho tipo de sincronización, cuya especificación no entra dentro del propósito de la presente Tesis. En [LAO02] se presenta una comparativa entre los algoritmos más interesantes propuestos hasta la fecha de su publicación.

Al final, el valor de *playout delay* proporcionará el instante, en unidades *RTP timestamp* según la referencia del receptor, en el que se reproducirá el paquete (en el caso de la figura, RTP_{10}). Si se observa la figura, el punto de reproducción será posterior a la recepción de otros paquetes de datos RTP, por lo que éstos deberán ser almacenados en un *buffer de reconstrucción* para su posterior reproducción. Tal como ya se ha comentado en este capítulo, el tamaño de este

buffer vendrá determinado por el *jitter*, que influye en el cálculo de la componente variable *var*.

Este valor de *playout delay*, en unidades *RTP Timestamp*, no será el intercambiado finalmente por los procesos reproductores de los diferentes flujos multimedia para conseguir la reproducción inter-flujo perseguida, ya que dichos procesos obtienen la referencia temporal de diferentes dispositivos en cuanto a la generación de las marcas de tiempo RTP (por ejemplo, la aplicación *rat* toma la referencia temporal del reloj de la tarjeta de audio, mientras que la aplicación *vic* toma la referencia de tiempos del reloj del PC). Para poder intercambiarse un valor que pueda servir a todos los procesos, el *playout delay* deberá convertirse a unidades *NTP timestamp*.

Playout delay intercambiado = conversión_ntp (playout_delay) [Ec. 4.18]

De esta manera, el valor intercambiado, en unidades *NTP timestamp*, podrá ser comparado con el de otros flujos de datos diferentes transmitidos simultáneamente por la misma fuente o por distintas fuentes sincronizadas con NTP, de forma que, al recibirlos, los diferentes procesos reproductores (como, por ejemplo, los de audio y vídeo) puedan '*negociar'*, de las formas indicadas anteriormente, un valor de *playout delay* común para sincronizar lo más exactamente posible sus procesos de reproducción.

Obtención de la referencia de tiempos de la fuente

Veamos ahora cómo se obtiene la referencia de tiempos de la fuente para '*mapear'* las marcas temporales RTP (*timestamps*) que vienen en los paquetes RTP de datos, a la referencia de tiempo RTP del receptor.

El '*mapeo'* del instante en el que fue enviado el paquete en unidades *RTP Timestamp* según la referencia del receptor (*send_ts*) se puede obtener gracias al uso del protocolo NTP (que permite obtener una referencia del reloj de la fuente en el receptor, gracias a los paquetes RTCP SR) y a partir de la marca temporal en unidades *RTP Timestamp* de la cabecera del propio paquete RTP. El proceso se puede observar, también, en la figura anterior.

Como la cabecera de los paquetes de datos RTP no incluyen información sobre el instante NTP de envío de los mismos, será necesario tomar la referencia del tiempo de la fuente a partir de la información de tiempo NTP de envío contenida en los paquetes de informe RTCP SR enviados por la propia fuente. En éstos, dicha información es proporcionada tanto en formato RTP *timestamp* como en formato *NTP timestamp*. Dicha correspondencia entre tiempo NTP y tiempo RTP en la fuente proporcionará a los receptores la relación que existe entre el

tiempo NTP (común en la fuente y en el receptor) y las marcas *RTP timestamp* de los paquetes de datos transmitidos por la misma.

En el ejemplo de la figura se calcula el valor del *playout delay* del paquete de datos RTP_{10}, una vez se ha recibido un informe RTCP SR anterior (imprescindible para poder realizar el '*mapeo'* de tiempos). Para ello, se siguen los siguientes pasos:

1. Del último informe RTCP SR recibido se toman sus marcas de tiempo *RTP y NTP timestamp* (incluidas en la cabecera) con los que fue transmitido (*sr_ts* y *sr_ntp*, respectivamente). A partir del instante de envío en tiempo NTP (que es igual para los dos equipos) se calculará el instante de envío del informe, en unidades *RTP timestamp* pero ahora según la referencia de tiempos RTP del receptor. Para ello se deberá convertir al formato *RTP timestamp*.

$$send\_sr\_ts = conversión\_ts(sr\_ntp) \qquad \text{[Ec. 4.19]}$$

2. A continuación se compara la marca de tiempo *RTP timestamp* del informe (*sr_ts*) con la marca *RTP timestamp* de envío del paquete de datos RTP_{10} (*rtp_ts*). De esta manera se obtiene el tiempo que transcurrió entre el envío del informe y el envío del paquete de datos (*time_last_sr* en la figura) analizado.

$$time\_last\_sr\_ts = (rtp\_ts) - (sr\_ts) \qquad \text{[Ec. 4.20]}$$

3. Aunque la referencia de tiempos RTP en la fuente y el receptor son distintas, las diferencias relativas se mantienen, pues ambos trabajarán con la misma frecuencia de muestreo. Si este tiempo (en unidades *RTP timestamp*) se suma al instante de envío del informe SR (*send_sr_ts*), se obtiene el tiempo de envío del paquete de datos RTP, en las mismas unidades y ya según la referencia de tiempos RTP del receptor.

$$send\_ts = send\_sr\_ts + time\_last\_sr\_ts \qquad \text{[Ec. 4.21]}$$

4. Una vez conocido el instante RTP de envío, *send_ts*, ya se podrá calcular el retardo sufrido por el paquete en la red (*delay*) de la manera indicada anteriormente (ec. 4.10).

4.3. Comparación con otros Algoritmos de Sincronización

En el siguiente apartado se va a presentar una comparativa del algoritmo de sincronización propuesto y otros algoritmos encontrados en la literatura, presentados en el capítulo 2 de Estado del Arte. Para ello se han consultado los patrones que otros autores han seguido a la hora de comparar algoritmos de sincronización multimedia, como por ejemplo, los presentados en [EHL94], [KÖH94] e [ISH00].

Tal y como se comentó en el capítulo 2, Ehley et al. presentan, en [EHL95], una clasificación de los algoritmos de sincronización multimedia existentes en dicha época agrupándolos en dos esquemas de sincronización: el distribuido (en entorno de red) y el local (dentro de la estación de trabajo).

En [KÖH94], Köhler y Müller presentan otra clasificación en función de varios factores: relojes (relojes globalmente sincronizados y relojes disponibles localmente –*locally available clocks*-), localización del mecanismo de sincronización (en la fuente o en el receptor) y técnicas de control de sincronización (sólo se consideran dos: ajuste de frecuencia de reloj y realización de '*saltos'* y/o '*pausas'*). En este trabajo, y en adelante, la relación entre *algoritmo* y *técnicas* es la siguiente: un algoritmo puede englobar varias técnicas de control.

Ishibashi y Tasaka, en [ISH00], presentan una comparativa de algoritmos de sincronización multimedia en función de las técnicas de sincronización utilizadas en cada uno de ellos. En nuestra comparativa partiremos de esta clasificación e introduciremos otros algoritmos que dichos autores no han considerado, además del propio propuesto en la Tesis.

Las relaciones cualitativas entre los algoritmos encontrados no están suficientemente claras. Una de las principales razones de ello es que las situaciones y entornos para los que han sido propuestos los diferentes algoritmos difieren de unos a otros. Además, no hay unas medidas estándar para evaluar algoritmos de sincronización multimedia. Actualmente, las relaciones cuantitativas entre los algoritmos parece ser bastante caótica.

Para clarificar las relaciones entre los algoritmos de sincronización seguiremos una aproximación paso a paso. A continuación, se van a extraer las principales técnicas utilizadas por los algoritmos estudiados más representativos y se clasificarán en varias categorías. Ya que varias técnicas pueden ser utilizadas juntas en un mismo algoritmo, cada una deberá ser indivisible (atómica), es decir, no tendrá diferentes funciones desde el punto de vista de la sincronización multimedia.

Se finalizará el apartado clasificando a los algoritmos analizados según una serie de factores relacionados con la sincronización multimedia.

4.3.1. Técnicas de Sincronización

Según Ishibashi y Tasaka ([ISH00]), las técnicas de sincronización se pueden clasificar en 4 grupos:

a) *Técnicas de Control Básico*, necesarias en la mayoría de algoritmos indispensables para preservar la estructura temporal.

b) *Técnicas de Control Preventivo*, necesarias para evitar la aparición de asincronía. Son utilizadas antes de que la asincronía ocurra.

c) *Técnicas de Control Reactivo*, necesarias para recuperar la sincronización ante situaciones de asincronía, después de su aparición.

d) *Técnicas de Control Común*, que serían aquellas técnicas que se pueden emplear tanto para prevenir como para corregir asincronías.

Las técnicas de control en el receptor son necesarias debido a la existencia de *jitter* en la red. Además, las técnicas de control básico, podrán ser complementadas, al mismo tiempo, con técnicas de control preventivo y/o reactivo. Las técnicas de control preventivo no pueden evitar completamente la asincronía, por lo que también será necesaria la utilización de técnicas de control reactivo.

Las técnicas de sincronización anteriores pueden ser aplicadas tanto en la fuente como en el receptor. El control en la fuente requiere de algún mensaje de realimentación (*feedback*) para que la misma conozca la calidad de la sincronización en cada momento y poder actuar en consecuencia. Por tanto, dividiremos las cuatro técnicas anteriores en dos grupos *según su localización* (*en la fuente* o *en el receptor*).

- **Técnicas de Control Básico**

Se trata de técnicas contempladas por prácticamente la totalidad de los algoritmos estudiados.

Control en la Fuente

Las técnicas de control básico que se suelen ejecutar en la fuente suelen basarse en la *introducción de información de sincronización en las propias LDUs enviadas*. En ellas se suele incluir marcas temporales o *timestamps* y/o el *número*

de secuencia o *identificativos*. Por ejemplo, la utilización de *timestamps* no es necesaria cuando, por ejemplo, la generación de las mismas es periódica, caso en el que se utilizarán números de secuencia en las LDUs. Además, en algunos casos, la fuente puede incluir marcas temporales en determinadas LDUs para forzar puntos de sincronización gruesa en el proceso de sincronización entre flujos.

Control en el Receptor

Prácticamente la mayoría de autores proponen la *utilización de buffers* en el receptor. Los *buffers* de recepción se utilizan para mantener en ellos las LDUs hasta que llegue el momento de su reproducción, de acuerdo con cierta información de sincronización, además de para suavizar el efecto de las variaciones del retardo o *jitter* de red.

- **Técnicas de Control Preventivo**

Control en la Fuente

Existen diferentes técnicas para evitar la aparición de asincronía desde la propia fuente transmisora, dependiendo del tipo de contenidos a transmitir.

Cuando se trata de transmisión de flujos de contenidos almacenados, normalmente la fuente podrá *transmitir las LDUs de acuerdo a una cierta información de sincronización* (por ejemplo, *timestamps*). Se utiliza en la mayoría de referencias estudiadas (por ejemplo en el algoritmo VTR, descrito en [ISH95]).

La fuente también puede prevenir una asincronía inicial ya desde el instante inicial de la reproducción de los diferentes flujos, calculando el *Instante Inical de Consumo* de las LDUs y comunicándolo a los receptores antes de enviar dichos flujos, para que todos ellos inicien su reproducción en el mismo instante.

Además, la propia fuente puede *planificar la transmisión según tiempos límite (deadline-based transmisión scheduling)*, técnica válida sólo para transmisión de contenidos almacenados. Si la fuente es capaz de conocer, para cada LDU, su tamaño, *deadline* de transmisión y límites del retardo (o, por lo menos, la función de distribución de probabilidad del retardo) podrá planificar la transmisión de LDUs según dichos requerimientos temporales ([LAM96] y [BAQ96]).

La fuente también puede *utilizar técnicas de entrelazado de LDUs* de diferentes flujos en un único flujo (como, por ejemplo, en [TAS98a] y [TAS98b]). Esta técnica mejora la calidad de la sincronización inter-flujo pero puede degradar la calidad de sincronización intra-flujo, sobre todo en flujos que sean sensibles al *jitter*.

Control en el Receptor

En ciertos casos, el proceso reproductor del receptor podrá *realizar 'Saltos' preventivos (descartes) y/o 'pausas' preventivas (repeticiones o inserciones)* dependiendo de la longitud del *buffer* de reproducción ([LIT92], [KÖH94] y [LAM96]). También es posible insertar LDUs duplicadas o vacías en vez de realizar '*pausas'* en la reproducción. En el caso de utilizar sistemas de codificación multinivel, como, por ejemplo, MPEG, se pueden descartar ciertas LDUs de tramas menos importantes de acuerdo con la ocupación del *buffer* de recepción.

Si es capaz de estimar el valor del retardo de red sufrido por las LDUS, el receptor podrá *cambiar el tiempo de espera de las mismas en buffers* a partir de dicha estimación ([RAM94], [ISH95], [ROT95] y [ROT97]). Existen autores ([RAM94]) que proponen alargar o acortar los períodos de silencio de los flujos de audio de las aplicaciones, técnica que no será válida para aplicaciones con sonidos musicales donde los sonidos son tan importantes como los periodos de silencio.

- **Técnicas de Control Reactivo**

En cuanto a las técnicas necesarias para recuperar la sincronización ante situaciones de asincronía, después de su aparición, también podemos encontrar varias, ejecutadas bien en la fuente o bien en el receptor.

Control en la Fuente

Una de las técnicas consiste en *ajustar el tiempo de transmisión* cambiando el período de transmisión. Si la fuente es capaz de conocer la asincronía existente en la reproducción de los diferentes flujos (por ejemplo, a partir de información de realimentación enviada por los receptores), podrá cambiar la temporización de la transmisión (por ejemplo, cambiando el período de transmisión)

Por otro lado, si la fuente detecta que al receptor le resulta difícil la recuperación de la asincronía detectada puede *decrementar el número de LDUs transmitidas* (esta técnica es utilizada, por ejemplo, en [ISH95]). Podemos citar como ejemplo, el caso de la sincronización de flujos de audio y vídeo, en la que, por ejemplo, si se ha detectado una situación de asincronía que persiste, la fuente podría dejar de transmitir el flujo de vídeo temporalmente. De esta manera, cuando el receptor se recuperase de la situación de asincronía, la fuente podría reanudar la transmisión de dicho flujo.

Control en el Receptor

Desde el punto de vista del receptor, éste también puede realizar determinadas acciones para recuperarse ante situaciones de asincronía detectadas.

La técnica más popular, debido a su facilidad de implementación, ha sido la de *realizar 'Saltos' reactivos (descartes) y/o 'pausas' reactivos (repeticiones).* Esta técnica ha sido utilizada por numerosos autores. Como ejemplo, podemos citar el caso en el que un receptor detecta que el instante de reproducción de la LDU que está procesando ya ha pasado por llegar tarde, y puede optar bien por reproducirla y descartar LDUs consecutivas que ya hubiera recibido ([AND91], [RAV93], [ISH95], [ROT95], [ISH97], [ROT97], [TAS98a], [TAS98b], [XIE99], [ISH00], [TAS00], [ISH01a], [ISH02] e [ITO02]), o bien por descartarla directamente ([RAM94] y [CHE96]). En caso de flujos con sistema de codificación multinivel, como, por ejemplo, MPEG, se intentará descartar sólo LDUs correspondientes a tramas poco prioritarias ([ISH97]). Por otro lado, cuando se produce *underflow* del *buffer* de recepción, el receptor puede optar por reproducir la última LDU de forma repetida hasta que llegue la siguiente.

También pueden optar por realizar '*estiramientos' y 'acortamientos' de la duración de la reproducción*. Para recuperarse gradualmente de una situación de asincronía sin empeorar la calidad de la reproducción (es decir, sin que lo noten los usuarios), el receptor puede acortar o alargar la duración de la reproducción de cada LDU hasta que se recupere la sincronización ([ISH95], [ROT95], [KOU96], [ROT97], [ISH97], [ROT97], [TAS98a], [TAS98b], [ISH00], [TAS00], [ISH01a], [ISH02] e [ITO02]). Acortar la duración implica una reproducción rápida (pero sin '*saltos'* de LDUs) mientras que un alargamiento implica una ralentización de la reproducción. Esta técnica también se utiliza para adaptar la reproducción de un flujo a la reproducción de otro. En el caso de flujo de voz, en determinadas aplicaciones, se podría acortar y alargar sólo las LDUs correspondientes a periodos de silencio para afectar lo mínimo posible a la calidad percibida por los usuarios.

La técnica de alargar la duración de la reproducción es similar a la de realizar '*pausas*' preventivas ya que ésta última puede realizarse mediante alargamiento de la duración de salida. Sin embargo, la primera es reactiva mientras que la última es preventiva.

Diversos autores proponen la *utilización de un tiempo virtual que se puede expandir y contraer para conseguir la sincronización buscada*, de tal manera que las LDUs son reproducidas según un eje de tiempos virtual diferente del tiempo real. Esta técnica ha sido utilizada en numerosas propuestas entre las que destacan [AND91], [ISH95], [LAM96], [ISH97], [TAS98a], [TAS98b], [XIE99], [ISH00], [TAS00], [ISH01a], [ISH02] e [ITO02]. El tiempo virtual se expande o se contrae en función del *jitter* sufrido por las LDUs recibidas.

Normalmente, para obtener una sincronización inter-flujo, se emplea un esquema *maestro/*esclavo, definiendo uno de los flujos como *maestro* y el resto como *esclavos*. Existen algunas propuestas en las que los roles de *maestro* y *esclavos* son intercambiados dinámicamente, permitiendo la *conmutación maestro-*

esclavo ([ROT95], [ROT97] y [XIE99]). De esta forma, cuando la asincronía correspondiente a un flujo *esclavo* determinado aumenta por encima de un valor umbral, el receptor puede conmutar los roles pasando a considerar a dicho flujo como *maestro*, realizando las adaptaciones apropiadas.

- **Técnicas de Control Común**

En la literatura consultada, se han encontrado una serie de técnicas que se pueden emplear tanto para prevenir (preventivas) como para corregir asincronías (reactivas), bien desde la fuente o bien desde el receptor.

Control en la Fuente

En determinados casos ([RAM93], [ISH95] y [BIE96]) se propone que la fuente realice *'saltos' y/o 'pausas' en la transmisión,* de acuerdo con información de realimentación recibida de los receptores (tanto para prevenir las asincronías como para corregirlas). La fuente incluso podría enviar LDUs vacías, en vez de '*saltar',* cuando la tasa de captura es menor que la tasa de transmisión ([ZAR96])

Cuando se trata de contenidos almacenados, las fuentes pueden *adelantar de forma dinámica la temporización de la transmisión según una estimación de retardo de red*, bien realizada por la propia fuente o bien obtenida mediante información de realimentación enviada por los receptores. Por ejemplo, se puede adelantar la reproducción '*saltando*' LDUs en la transmisión.

En [AND91], se propone *ajustar la tasa de captura* por medio de ajustes de la frecuencia de reloj del dispositivo de captura, según la calidad de la sincronización obtenida. El mismo autor también propone *interpolar datos* para ajustar la tasa de salida.

También se puede utilizar la técnica denominada *Media Scaling*. Por ejemplo, *layered multicast* es una técnica de este tipo y se puede transmitir más o menos flujos dependiendo de las condiciones de la red ([CAN96], [TAS98b]). Se puede cambiar, por ejemplo la resolución temporal o espacial del flujo de vídeo dependiendo de la carga de la red.

Control en el Receptor

En cuanto a técnicas de control común en el receptor se puede citar la de *ajustar la tasa de reproducción* mediante la modificación de la frecuencia de reloj del dispositivo de reproducción, de acuerdo con la calidad de sincronización obtenida. En [AND91], [ROT95] y [ROT97], el receptor ajusta la tasa de reproducción dependiendo de la longitud de la cola de espera de reproducción.

En [AND91] también se propone realizar *interpolación de datos* en el receptor para ajustar la tasa efectiva de salida.

4.3.2. Técnicas utilizadas por el Algoritmo Propuesto

En este apartado se indicará qué técnicas de las descritas en el apartado anterior han sido utilizadas en el algoritmo propuesto en la Tesis.

- **Técnicas de Control Básico**

En cuanto a las técnicas que incluyen la mayoría de algoritmos de sincronización estudiados, nuestro algoritmo utiliza tanto técnicas de sincronización en la fuente como en el receptor.

Control en la Fuente

La fuente de un flujo multimedia, introduce *información de sincronización en las LDUs*. El algoritmo propuesto hace uso de los *timestamps* y *números de secuencia* que incluyen los paquetes RTP y RTCP.

Control en el Receptor

Por otro lado, en el receptor, se considera la *utilización de buffers* para almacenar las LDUs conforme se van recibiendo, hasta que llegue su instante de reproducción obtenido tras la aplicación del algoritmo de sincronización. De esta manera, además, se podrá suavizar el efecto del *jitter* de red.

- **Técnicas de Control Preventivo**

Como se ha comentado, el algoritmo propuesto es válido tanto para flujos de contenidos *en directo* como para flujos de contenidos almacenados.

Control en la Fuente

El algoritmo ya prevé que la fuente evite una asincronía inicial ya desde el instante inicial de la reproducción de los diferentes flujos, calculando el *Instante Inicial de Consumo* de las LDUs, del flujo considerado como *maestro*, y enviando dicha información a los receptores mediante un mensaje RTCP TIN, antes de enviar las primeras LDUs de dicho flujo. Éste proceso es válido tanto para contenidos almacenados como *en directo*.

Posteriormente, si se trata de flujos de contenidos almacenados no multiplexados, la fuente podrá utilizar cualquiera de las técnicas explicadas, es decir, *transmitir las LDUs de acuerdo a una cierta información de sincronización*

y, también, podrá *planificar la transmisión según los deadline de las mismas.* En caso de transmitir flujos de contenidos *en directo*, la fuente no podría utilizar ninguna de estas dos técnicas de control preventivo.

Control en el Receptor

El proceso reproductor del receptor, cuando recibe un paquete APP ACT procedente de la fuente sincronizadora, podrá *realizar 'saltos' preventivos (descartes) y/o 'pausas' preventivas (repeticiones)* dependiendo del resultado del algoritmo.

- **Técnicas de Control Reactivo**

Control en la Fuente

Aunque no se ha incluido en la especificación, en el algoritmo propuesto podría incluirse que las fuentes transmisoras, al recibir la información de realimentación en los mensajes RTCP RR, pudieran realizar alguna acción correctora como, por ejemplo, *ajustar del tiempo de transmisión* cambiando el período de transmisión o bien *decrementar el número de LDUs transmitidas.*

Control en el Receptor

En el algoritmo propuesto, el receptor realizará determinadas acciones para recuperarse ante situaciones de asincronía detectadas.

Se utiliza la popular técnica de realizar *'saltos' reactivos (descartes) y/o 'pausas' reactivos (repeticiones)* en el proceso reproductor del receptor al detectar una situación de asincronía (será detectada al analizar los paquetes APP ACT que envíe la fuente). Además, el receptor *descartará* todas las LDUs que lleguen tarde y, por tanto, su instante de reproducción ya haya pasado.

Por otro lado, como consecuencia de los ajustes realizados en el proceso reproductor del receptor (tanto en la sincronización inter-flujo como en la de grupo), aumentando o disminuyendo el valor del *playout delay* de las LDUs, se producirá un *'estiramiento' y 'acortamiento' de la duración de la reproducción.*

Como se ha explicado anteriormente, para obtener la sincronización de grupo, se emplea un esquema *maestro/esclavo*, definiendo uno de los receptores como *maestro* y el resto como *esclavos*. Como se ha visto, el algoritmo permite *cambiar los roles* de *maestro* y *esclavo* entre los receptores en función de la técnica de selección de la referencia escogida. No se tratará de una *conmutación maestro/esclavo* en cuanto a flujos sino en cuanto a receptores para obtener la referencia necesaria para la sincronización de grupo.

Como se comentó en el capítulo 2, en [ISH01b] y [ISH03], aparece una evaluación de 9 combinaciones de cuatro técnicas de sincronización reactivas (‘*saltos’*, descartes, *‘estiramiento’* y *‘acortamiento’* de la duración de la reproducción y contracción y expansión del tiempo virtual) sobre una simulación con flujos de audio y vídeo.

- **Técnicas de Control Común**

Control en la Fuente

Tal como se ha comentado anteriormente, aunque no se ha contemplado en la propuesta, se podría incluir en el algoritmo propuesto que las fuentes transmisoras, al recibir la información de realimentación en los mensajes RTCP RR, pudieran realizar acciones correctoras como, por ejemplo, *realizar ‘saltos’ y/o ‘pausas’* (tanto preventivos como reactivos) o *ajustar la tasa de captura* por medio de ajustes de la frecuencia de reloj del dispositivo de captura.

Control en el Receptor

En este caso, el algoritmo propuesto no contempla la utilización de ninguna técnica de este tipo.

4.3.3. Clasificación de los Algoritmos de Sincronización Multimedia

La tabla 4.4 muestra una posible clasificación de los algoritmos estudiados[(3)], según las técnicas descritas anteriormente y otros factores de interés:

- *Relojes*. Se indicará si la señal de reloj está sincronizada globalmente o está disponibles sólo localmente.

- *Límites del retardo*. Necesidad de conocer a priori los límites del retardo de red o la función de distribución de probabilidad del mismo.

- *Periodicidad de generación de LDUs*. Se indicará si el algoritmo está pensado para trabajar con transmisión de flujos en los que la generación de LDUs es periódica o no.

[3] En el capítulo 2 de Estado del Arte se puede encontrar una descripción breve de cada uno de los trabajos referenciados en la tabla 4.4.

- *Contenidos almacenados o en directo.* Algunos algoritmos están diseñados bien para la transmisión de contenidos almacenados, bien para la transmisión de contenidos *en directo*, o bien para ambos tipos de contenidos.

- *Tipo de sincronización* incluida en el algoritmo (intra-flujo, inter-flujo y de grupo). En este caso se han clasificado aquellos algoritmos que proponen bien sincronización inter-flujo o bien de grupo, no así los que sólo proponen sincronización intra-flujo (en [LAO02] aparece una clasificación de los mismos).

- Relación *maestro*/*esclavo*. Si existe relación *maestro*/*esclavo* entre los flujos de información.

- *Utilización de Feedbacks*. Se indicará si el algoritmo hace uso de mensajes de realimentación o *feedback* enviados por los receptores a las fuentes.

- *Información de Sincronización*. Indica el tipo de información, útil para la obtención de sincronización, que se incluye en las LDUs.

- *Localización* de las técnicas de sincronización. Indica si el control está basado en la fuente, en el receptor o en ambos.

- *Uso de RTP.* Indica si el algoritmo propuesto se implementa haciendo uso de RTP/RTCP, o bien si propone su utilización como posible u opcional.

- *Técnicas de Sincronización.* Se incluyen las técnicas de sincronización, descritas en el apartado anterior, que incluye cada algoritmo.

En la primera columna, denominada *Algoritmo*, se presenta el nombre que los autores le han asignado a su propuesta y las principales referencias bibliográficas del mismo. En caso de no haberle asignado ningún nombre sólo aparece la referencia.

Los huecos en la tabla significan que no se contempla el factor en cuestión o que no se hace referencia al mismo en la o las publicaciones relacionadas.

Puede apreciarse en la tabla que existen una gran variedad de algoritmos, definidos en multitud de condiciones y entornos, por lo que en cada uno se utilizan combinaciones de diversas técnicas distintas de las utilizadas por otros protocolos y, por tanto, tampoco es tarea fácil intentar relacionar y comparar los algoritmos estudiados de forma cualitativa.

Tabla 4.4. Tabla Comparativa (1/3)

Tabla 4.4. Tabla Comparativa (2/3)

Tabla 4.4. Tabla Comparativa (3/3)

4.4. Conclusiones

Tal y como se comentó en el capítulo 3, el *retardo*, el *jitter* y la *desviación en las tasas de consumo* de cada reproductor, son las principales causas que generan asincronía en un sistema multimedia distribuido. Todas ellas, junto con otras adicionales, como puedan ser la carga en la estación o la asignación incorrecta de prioridades entre procesos, pueden tener como efecto la necesidad de solucionar los siguientes requerimientos: sincronización del instante inicial de consumo, sincronización gruesa y fina y sincronización de grupo, minimizando las probabilidades de *underflow* y *overflow* en los *buffers* de los receptores.

Como solución a todo ello, se ha propuesto un *algoritmo de sincronización de grupo de flujos multimedia*, basado en un tiempo global que se utilizará en el proceso de sincronización para aumentar la precisión de la misma y permitir su uso en redes corporativas con límites de retardo extremo a extremo acotados. Dicho algoritmo hace uso del protocolo RTP/RTCP y, para ello, se proponen una serie de modificaciones en el formato de ciertos paquetes RTCP, incluso se propone la creación de paquetes RTCP APP nuevos. La estructura del algoritmo en cuanto a funcionalidad está basada en el protocolo *Feedback* ([RAM92]), pero añadiendo la utilización de un tiempo global proporcionado por el protocolo NTP, tal y como se propone en el protocolo *Feedback Global* ([GUE97]).

En este caso, la especificación de la sincronización se realiza mediante marcas de tiempo en los paquetes RTP en el instante de la captura y el cálculo de la planificación de la reproducción sincronizada se realiza en tiempo de ejecución, aprovechando la referencia de tiempos global proporcionada por NTP.

El algoritmo descrito ha sido diseñado para garantizar, en una aplicación multimedia distribuida, la sincronización del instante de inicio de la reproducción y la reproducción posterior sincronizada de los diferentes flujos de información involucrados, tanto dependientes como independientes del tiempo (sincronización inter-flujo). Además se consigue que todos los receptores reproduzcan al mismo tiempo los diferentes flujos (sincronización de grupo).

Por tanto, el algoritmo consta de tres partes bien diferenciadas dedicadas a sincronización intra-flujo, sincronización de grupo y sincronización inter-flujo, centrándose principalmente en solucionar la sincronización inter-flujo y de grupo. Para ello se basa en la utilización de dos esquemas *maestro/esclavo*.

La sincronización inter-flujo, se basa en la definición de uno de los flujos como *maestro* y el resto serán considerados como flujos *esclavos* y sus procesos de reproducción sincronizarán su estado con el del flujo *maestro*. Para ello ha sido

necesaria la utilización de un mecanismo local de intercambio de mensajes entre procesos, denominado *mbus*.

Por otro lado, el proceso de sincronización de grupo persigue la sincronización de la reproducción de los diferentes flujos en todos los receptores de forma sincronizada, es decir, de forma simultánea. Para ello se elige un estado de reproducción del flujo *maestro* como *referencia* (según varias técnicas descritas) y se fuerza a los diferentes receptores a ajustar el estado de reproducción de dicho flujo al estado de referencia. El estado de referencia es calculado por la fuente del flujo *maestro* (considerada como *fuente sincronizadora*), a partir de información de realimentación de los receptores, y es distribuido a los receptores mediante un nuevo tipo de mensaje RTCP (RTCP ACT), definido para la especificación del algoritmo.

Tanto en un tipo de sincronización como en otro, los receptores realizarán ajustes del estado de la reproducción de sus flujos mediante la modificación del valor del retardo de reproducción o *playout delay* de las LDUs.

Al utilizar mensajes RTCP, se consigue mantener una muy baja carga de información de control y mensajes dedicados a la sincronización, en comparación al número total de LDUs transmitidas.

Podemos concluir que el algoritmo de sincronización propuesto resultará apropiado para sistemas multimedia distribuidos con varias fuentes y varios receptores, donde se realice una transmisión *multicast* (si las red lo permite) de flujos individuales no multiplexados, a través de una red determinista o con una cierta calidad de servicio garantizada, donde los retardos máximos sean limitados y conocidos a priori.

ALGORITMO	Reloj	Límites Retardo	Periodicidad generación LDUs	Contenidos almacenados o en directo	Sincr. Intra, Inter y de grupo	*Maestro/ Esclavo (M/E)*	Utiliz. de *Feedbacks*	Inform. de Sincroniz.	Localiz.	Uso de RTP	Técnicas de sincronización
ALGORITMO PROPUESTO	**Global**	**Conoc.**	**Periódicos y no periódicos**	**Ambos**	**Inter. y de grupo**	***flujos M/E receptores M/E***	**Sí**	**Timestamp Nº secuencia**	**Fuente y receptor**	**Sí**	- **Instante inicial de consumo.** - ***'saltos'* y *'pausas'* reactivos.** - **Expansión virtual del tiempo.** - **Conmutación *maestro/esclavo* en cuanto a receptores (sincr. de grupo).**
ACME(*) [AND91]	Global	-	Periódicos	Ambos	Intra e Inter.	M/E y sin relación	-	Timestamp	Receptor	No	- '*saltos*' y '*pausas*' reactivos. - Expansión virtual del tiempo. - Ajuste de tasas de E/S. - Interpolación de datos.
[LIT92] [LIT93]	Global	-	Periódicos	Almacenados	Inter.	Sin relación.	Sí (pero no los utiliza)	Nº secuencia	Receptor	No	- Decremento del número de flujos. - Inserción y descarte de LDUs. - *'Saltos'* y '*pausas*' reactivos para cambiar la tasa de reproducción.
Feedback Protocol **[RAN92], [RAM92], [RAM93],[RAM93b], [RAM95], [RAN95a], [RAN95b]**	Local	Conoc.	Periódicos	Almacenados	Intra e inter	M/E	Sí	Timesptamp Nº secuencia	Fuente y receptor	No	- *'Saltos'* y *'pausas'* en la fuente.
MCP(*) [YAV92], [YAV94]	Global	-	Periódicos y no periódicos		Inter y de grupo	-	No	Timestamp Nº secuencia Info. de contexto	Receptor	No	- *'Saltos'* y *'pausas'* en el receptor. - *Descarte de LDUs.*
FSP (*) [ESC94]	Global	Descon.	Periódicos y no periódicos	Ambos	Inter	-	No	Timestamp	Receptor	No	- Contracción y expansión de la duración de la reproducción.
VTR(*) [ISH95] , [TAS98a], [TAS98b], [ISH01a]	Global y local	Descon.	Periódicos y no periódicos	Ambos	Intra e inter	M/E	Sí	Timestamp Nº secuencia	Fuente y Receptor	No: [ISH95, TAS98a, ISH01a] Sí: [TAS98b]	- Cambio del tiempo de *buffer* según estimación del retardo. - Decremento del número de flujos. - *'Pausas'* preventivas. - *'Saltos'* y *'pausas'* reactivos. - *'Saltos'* en la fuente. - Acortar y extender duración de reproducción. - Expansión y contracción del tiempo virtual. - Media Scaling.
ASP (*) [ROT95], [ROT97]	Global	-	-	Sin distinción	Intra e inter	M/E	-	Timestamp	Receptor	No	- Instante inicial de transmisión y de consumo. - *'Saltos'* y *'pausas'* (duplicados) reactivos. - Cambio del tiempo de *buffer* según estimación del retardo. - Acortar y extender duración de reproducción. - Conmutación *maestro-esclavo*. - Ajuste de la tasa de reproducción.

ALGORITMO	Reloj	Límites Retardo	Periodicidad generación LDUs	Contenidos almacenados o en directo	Sincr. Intra, Inter y de grupo	*Maestro/ Esclavo (M/E)*	Utiliz. de *Feedbacks*	Inform. de Sincroniz.	Localiz.	Uso de RTP	Técnicas de sincronización
***Concord Algorithm* [SHI95]**	Global y local	Conoc.	Periódicos	-	Intra e inter	M/E y sin relación	-	Timestamp en 1er paquete o paquetes especiales	-	No	- Descarte de LDUs. - 'Pausas' en el flujo *esclavo.*
[AKY96]	Global	Conoc.	-	-	Intra, inter y de grupo	flujos M/E receptores M/E (chairman)	-	Timestamp en 1er paquete		No	- Instante inicial de consumo. - Ajustes en tasa de reproducción (del reloj del receptor). - Conmutación *maestro/esclavo.* (*chairman*) en cuanto a receptores.
[BAQ96]	-	Conoc.	-	Almacenados	-	Sin relación	No	*Deadline* de reproducción	Fuente	No	- Planificación de la transmisión basada en *deadlines.*
[BIE96]	Local	Descon.	-	Almacenados	Intra e inter	Sin relación	Sí	Timestamp Nº secuencia	Fuente y receptor	Opc.	- Instante inicial de transmisión y consumo. - *'Saltos'* y *'pausas'* preventivos en la fuente.
***MultiSynch* [CHE96]**	-	-	Periódicos	-	Intra e inter	Sin relación	-	Timestamp	Receptor	No	- *'saltos'* (descartes) y *'pausas'* reactivos. - Asignación de prioridades.
[KOU96]	Global	Descon.	Periódicos	Directo	Intra e inter (*lip-synch*)	Sin relación	-	Timestamp	Receptor	Sí	- Expansión y contracción de la duración de la reproducción.
SSP$^{(*)}$ [LAM96]	Local	Conoc.	Ambos	Almacenados	-	-	Sí en fase inicial de conexión	Timestamp	Fuente y receptor	No	- Instante inicial de consumo. - Planificación de la transmisión basada en *deadlines.* - Asignación de prioridades. - *'Saltos'* preventivos (descarte selectivo) y *'pausas'* (duplicados) o retardos 'intencionados'. - Expansión de tiempo virtual.
[LIU96]	Local	Descon.	Ambos	Sin distinción	Intra e inter	Sin relación	-	Timestamp	Receptor	No	- *'Saltos'* (descartes) y *'pausas'* reactivos.
[RAN96]	Global	Conoc.	Periódicos	Almacenados	Intra e inter	M/E y sin relación	-	Timestamp (en algunas LDUs)	Fuente y receptor	No	- *'Saltos'* y *'pausas'* reactivos en el receptor o en la fuente.
[SON96]	Local	Descon.	-	Ambos	Inter	-	Sí	Timestamp	Fuente (Servidor de sincr.)	No	- *'Saltos'* y *'pausas'* en la fuente. - Esquema que puede variar la precisión de detección de asincronía.
MSTP$^{(*)}$ [YAN96]	-	Descon.	Ambos	Ambos	Inter	M/E	-	Nº sec. LDU flujo *maestro a sincronizar*	Receptor	No	- Descarte de LDUs.

ALGORITMO	Reloj	Límites Retardo	Periodicidad generación LDUs	Contenidos almacenados o en directo	Sincr. Intra, Inter y de grupo	*Maestro/ Esclavo (M/E)*	Utiliz. de *Feedbacks*	Inform. de Sincroniz.	Localiz.	Uso de RTP	Técnicas de sincronización
[ZAR96]	Local	Descon. (*Jitter* máximo)	Periódicos	-	Inter	Sin relación.	Sí	Timestamp Nº secuencia	Fuente	No	- *'Saltos'* (o inserción de LDUs de 'ruido') y *'pausas'* en la fuente.
***Feedback Global Protocol* [GUE97]**	Global	Conoc.	Periódicos	Ambos	Intra e Inter.	M/E	Sí	Timestamp Nº secuencia	Receptor	No	- *'Saltos'* y *'pausas'* reactivos en el receptor
***Synchronization maestro scheme (VTR modificado)* [ISH97], [ISH99], [ISH01c], [ISH02]**	Global	Conoc.	Ambos.	Almacenados (en [ISH02] ambos a la vez)	De grupo (VTR) Intra e inter.	flujos M/E receptores M/E	Sí	Timestamp Nº secuencia	Receptor	No	- Mismos que VTR ([ISH95]).
[QIA97]	Local	-	Periódicos	Almacenados	Inter (*lip-synch*)	M/E	Sí	Timestamp	Fuente y receptor	No	- Ajuste inicial en tasa de transmisión. - Ajustes en tasa de reproducción. - *'Saltos'* y *'pausas'*. - Descarte de LDUs.
Algoritmo *bucket* [DIO99]	Global	Conoc.	Ambos	En directo	Inter y de grupo	Sin relación	No	Timestamp Nº secuencia	Receptor	Sí	- '*Saltos'* (descartes) y '*pausas'* (duplicados) de LDUs en el receptor.
[XIE99]	Local	Descon.	Ambos	En directo	Intra e inter	M/E	No	Timestamp	Receptor	No	- *'Saltos'* y '*pausas'* reactivos. - Expansión y contracción del tiempo virtual. - Conmutación de *Maestro-Esclavo*.
[GON00]	Global o local	Conoc.	Ambos	directo	Intra e inter	-	Solo en fase inicial	-	Receptor	No	- Descarte de paquetes. - Reducción períodos de silencio. - Contracción y expansión del tiempo de reproducción. - Inserción de *'gaps'*.
RVTR(*) [TAS00]	Global	Descon.	Ambos	Ambos	Intra e inter	M/E	Sí	Timestamp Nº secuencia	Fuente (retrans.) y receptor	Sí	- Mismos que VTR ([ISH95]). - Contracción y expansión del tiempo de reproducción. - Retransmisiones con reconocimientos negativos.
[KUO01]	Global	Descon.	Periódicos	En directo	Intra e inter	-	Sí	Timestamp Nº secuencia	Receptor	Sí	- Ajustes en tasa de reproducción.

(*)

ACME	*Abstractions for Continuous Media*	***MSTP***	*Multimedia Synchronization Transport Protocol*
ASP	*Application Synchronization Protocol*	***RVTR***	*Retransmission with VTR*
FSP	*Flow Synchronization Protocol*	***SSP***	*Stream Synchronization Protocol*
MCP	*Multi-Flow Conversation Protocol*	***VTR***	*Virtual Time Rendering Algorithm*

Capítulo 5

Transmisión de Datos utilizando RTP/RTCP

5.1. Introducción

El protocolo RTP está siendo ampliamente utilizado para la transmisión de audio y vídeo sobre redes IP, mientras que los datos no críticos (texto, datos de usuario, ficheros, etc.) están siendo enviados mediante el uso de TCP o UDP sobre IP. Ya que ni TCP ni UDP proporcionan información temporal de los datos que transportan, no se puede sincronizar su presentación con la de otros flujos de datos, como son el audio y el vídeo.

Entre los ejemplos más comunes de transmisión de datos en sistemas multimedia distribuidos podemos citar los siguientes:

- Transmisión de datos de texto en presentaciones multimedia. Por ejemplo, se podría representar el texto correspondiente a una exposición oral para

personas con problemas auditivos y que las palabras aparecieran al mismo tiempo que se fueran pronunciando. Lógicamente, en este caso las relaciones temporales se crearían a posteriori para la presentación de forma '*sintética*'.
- Exposición de transparencias o diapositivas con un comentario de audio, donde el cambio de transparencia o diapositiva se deba realizar de forma sincronizada con el comentario de audio.
- Inclusión de información del inicio, del contenido y del final de la emisión de un vídeo por Internet que facilitara a los videograbadores de un receptor iniciar y parar la grabación del vídeo.
- Transmisión en una emisión de radio, junto a la señal de audio, información de tráfico, de la estación y de las canciones, tal como se realiza en el sistema *Radio Data System* o RDS.
- Información sobre el puntero del ratón en una teleconferencia, donde la posición del ratón se maneje a distancia.
- Datos provenientes de sensores distribuidos en una red que se representen en una aplicación que integre diferentes tipos de datos. Un ejemplo podría ser el de control visual de una cadena de fabricación donde existan cámaras de video filmando la cadena de montaje y, al mismo tiempo, se reflejen en las mismas u otras pantallas datos resultado del proceso (por ejemplo, gráficas en tiempo real, unidades montadas por segundo, etc.). Otro ejemplo sería el de un sistema de monitorización del tráfico de una ciudad en tiempo real, con múltiples monitores reflejando imágenes de diferentes cámaras distribuidas por toda la ciudad, incluyendo mensajes de texto (como subtítulos) indicando el estado de la circulación en cada momento, datos recibidos de sensores colocados en diferentes puntos de la ciudad.

Actualmente no existe ninguna RFC que indique cómo transportar datos genéricos mediante RTP, pero sí existen varias RFCs que indican cómo transportar algunos tipos de datos. Entre estas últimas, podemos citar la RFC 2793 ([HEL00]), que define cómo transportar datos de conversaciones de texto; la RFC 2862 ([CIV00]) que define cómo transportar las coordenadas de punteros dinámicos; y la RFC 2833 ([SCH00]) para el transporte de dígitos multifrecuencia (DTMF), tonos y señales telefónicas. Además, como se ha comentado en el capítulo 2 dedicado al Estado del Arte, en [BRA02] se presenta una solución basada en RTP para la difusión de flujos multimedia para redes de distribución de contenidos o CDN (*Content Distribution Networks*), basada en un mecanismo denominado *cueing protocol* para proporcionar información temporal, estructura e información de identidad en los flujos multimedia (por ejemplo, para conseguir enviar la información propia del sistema RDS). De los cuatro documentos anteriores, los que más interesan, desde el punto de vista de la presente Tesis, son las dos primeras RFCs que serán tratadas a lo largo de este capítulo.

Al utilizar RTP/RTCP para transportar datos de cualquier tipo, se puede aprovechar el uso de las marcas temporales que estos protocolos proporcionan para

poder sincronizar la presentación de los datos con la de otros flujos dependientes del tiempo, como puedan ser los flujos de audio y vídeo.

5.2. Transmisión de Texto utilizando RTP/RTCP

Según el tipo de las aplicaciones donde se emplee esta técnica, el texto a enviar puede ser introducido por los usuarios desde el teclado, mediante reconocimiento de voz u otros métodos de entrada.

Como ya se ha comentado, la transmisión de texto utilizando RTP se define en la RFC 2793, *RTP Payload for Text Conversation*, desarrollada por Hellstrom ([HEL00]), que describe cómo transportar los contenidos de una sesión de conversación de texto en paquetes RTP. Dichos contenidos están especificados en la recomendación T.140 de la ITU-T.

El formato de la carga útil de los paquetes RTP propuesto en dicha RFC contiene también la posibilidad opcional de incluir redundancia con datos de texto de paquetes ya transmitido para, así, reducir el riesgo de pérdidas de texto debido a pérdidas de paquetes. La codificación de la redundancia sigue la RFC 2198 (*RTP Payload for Redundant Audio Data*, [PER97]).

5.2.1. Estructura del Texto y Codificación UTF-8

5.2.1.1. Estructura del Texto

Basándonos en la norma ITU-T T.140, los caracteres de texto enviados deberán ser transmitidos usando la codificación ISO 10646-1 con transformación a UTF-8, que se explica a continuación.

UTF-8 es un tipo de codificación de caracteres que tiene su origen en un proyecto del grupo de investigación internacional *X/Open Joint Internationalization Group* (*XOJIG*), que pretendía crear un sistema de transformación de caracteres para un sistema de ficheros seguro, que fuera compatible con los sistemas UNIX, soportando texto multilingüe en una codificación simple. Para llevar a cabo la tarea de unificar los sistemas de escritura mundiales en un solo tipo de datos, la norma ISO 10646-1 define un carácter multiocteto llamado UCS (*Universal Character Set*).

La norma UCS define dos codificaciones multiocteto: Codificación de dos octetos por carácter (UCS-2) y la Codificación de cuatro octetos por carácter (UCS-4), capaces de direccionar los primeros 64K caracteres del UCS (el *plano*

básico multilingüe o BMP). La diferencia entre esta última y el estándar *Unicode* (que también cubre el mismo plano) es que, este último define ciertas propiedades adicionales para los caracteres pero no incluye la codificación UCS-4.

En ocasiones, UCS no es compatible con ciertas aplicaciones y protocolos existentes en el ámbito comercial, por lo que surge la necesidad de crear un mecanismo que sea capaz de adaptar todos estos formatos. Esto constituye lo que se denomina *Formatos de Transformación UCS* (UTF o *UCS Transformation Formats*), cada uno con sus propias características. UTF-1 sólo tiene carácter histórico y fue eliminada de la norma 10646 hace bastantes años. UTF-7 posee la suficiente calidad como para codificar el repertorio multilingüe BMP, utilizando sólo los 7 bits más altos de cada octeto para codificar los caracteres US-ASCII. UTF-8 utiliza 8 bits y será explicado más adelante. UTF-16 se usa para transformar un subconjunto del repertorio UCS-4 en parejas de valores UCS-2.

Las codificaciones UCS-2 y UCS-4 tienen el problema de que la implementación se complica en ciertos programas y protocolos, que asumen caracteres de 7 u 8 bits, como es el caso de numerosas aplicaciones que utilizan el sistema US-ASCII de 8 bits. Esto ya no depende sólo de la forma de implementar la aplicación, sino del propio sistema operativo o plataforma sobre la que se esté trabajando. Incluso las plataformas más modernas que soportan caracteres de 16 bits, son incapaces de procesar UCS-4.

UTF-8 es capaz de codificar tanto UCS-2 como UCS-4, variando el número de octetos. La cantidad de los octetos que se necesiten, así como el valor de cada uno de ellos, depende del valor entero asignado al carácter en ISO 10646.

La norma T.140 requiere que el medio de transporte tenga un cierto control de los paquetes de texto, con el fin de evitar posibles fallos de la red, tales como pérdidas, paquetes duplicados o desórdenes de paquetes. Para que estos mecanismos puedan ser implementados en el receptor es necesario hacer uso de los datos contenidos en las cabeceras de los paquetes. A pesar de ello, no todos los problemas se pueden solucionar con las cabeceras, como es el caso de las pérdidas que necesitan del uso de redundancia para ser compensadas.

Según se define en la norma ITU-T T.140, el contenido de los bloques de texto T.140 de los paquetes RTP debe encontrarse en formato UTF-8, tanto los bloques primarios como los redundantes.

Si las aplicaciones enviaran la información textual en formato ASCII, en la zona y ámbito de uso en la que se ha desarrollado la presente Tesis no habría problema, ya que se utiliza un sistema de escritura basado en caracteres latinos. Todo esto es extensible para los países europeos, América, parte de Oceanía y parte de África. En cambio, si se estuviera en países como Turquía, Japón, China, etc.,

los símbolos utilizados para la escritura en estos países no coinciden con la simbología del alfabeto latino, por tanto no se encuentran dentro del sistema ASCII. Este problema limitaría el uso de las aplicaciones en los lugares anteriormente citados. La solución viene de mano del sistema de codificación UTF-8, que pretende extender el conjunto de símbolos utilizables a más allá de los 256 caracteres. De esta forma se pueden incluir todas las letras pertenecientes a los sistemas de escritura arábigo, chino, japonés, etc.

Después de lo comentado, se puede apreciar lo importante que supone tener un sistema de estas características, además de la posibilidad de implantación de las aplicaciones basadas en el mismo en cualquier país del mundo.

5.2.1.2. Algoritmo UTF-8

Cabe destacar una característica muy importante de la codificación UTF-8 antes de pasar a analizarla más detenidamente. Cuando se introducen valores de código comprendidos entre 00h y 7Fh (0 a 255 en el código de caracteres ASCII), el algoritmo no realiza ninguna codificación, transmitiendo la secuencia directa como texto llano. Si se transmitiese el texto 'Hola', simplemente se transmitirían los códigos binarios correspondientes a cada carácter ASCII, del siguiente modo:

Carácter	'H'	'o'	'l'	'a'
ASCII/UCS(*)	072h	111h	108h	097h
Secuencia	1001000	1101111	1101100	1100001

(*) los valores UCS se encuentran en hexadecimal.

Tabla 5.1. Secuencia UTF-8 de la cadena 'Hola'

En UTF-8, los caracteres se codifican utilizando secuencias de uno a seis octetos. En el caso de que una secuencia esté formada por un solo octeto, éste tendrá el bit más significativo a '0'. Los siete bits restantes se usan para codificar el valor del carácter. El octeto quedará de esta forma:

Rango UCS-4	Secuencia UTF-8
00h – 7Fh	0xxxx

Tabla 5.2. Codificación UTF-8 de secuencias de un octeto

Cuando la secuencia a transmitir está formada por más de un octeto, es decir, por n octetos, con n>1 (valores de código comprendidos entre 80h y 7FFFFFFF), se debe seguir un proceso con el que se consiga separarla en los octetos que sean necesarios, siempre que no excedan de 6.

Rango UCS-4	Secuencia UTF-8					
80h - 7FFh	110xxxxx	10xxxxxx				
800h - FFFFh	1110xxxx	10xxxxxx	10xxxxxx			
1 0000-001F FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx		
20 0000-03FF FFFF	111110xx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	
400 0000-7FFF FFF	1111110x	10xxxxxx		…		10xxxxxx

Tabla 5.3. Codificación UTF-8 de secuencias de varios octetos

El bit más significativo del primer octeto se pondrá a ‘1’ indicando que la secuencia es multiocteto. Los bits siguientes indican el número de octetos que siguen al octeto primario, de forma que por cada octeto añadido se inserta un ‘1’ después del primer bit. A continuación se coloca un bit a ‘0’. Una vez completada esta secuencia de bits, se empieza a insertar los bits de información (en los bits x que aparecen en la tabla). En cada octeto que siga al primario, el valor de los dos primeros bits será ‘10’, para indicar que es un octeto secundario. En cada octeto secundario se tendrán, por tanto, 6 bits para insertar información útil.

De forma esquemática, los pasos a seguir para codificar los datos de UCS-4 a UTF-8 es la siguiente:

- Determinar el número de octetos requeridos, teniendo en cuenta el valor del carácter y la primera columna de la tabla anterior.

- Preparar los bits de mayor orden de cada octeto tal y como muestra la segunda columna de la tabla.

- Rellenar los bits marcados como *x* con los valores del carácter a codificar, empezando por los bits de menor orden del valor del carácter y colocándolos primero en el último octeto de la secuencia y así sucesivamente. Para un solo octeto no se haría nada, pero en el caso de tener dos octetos, por ejemplo, para el primer octeto, se deberá desplazar el valor del carácter 6 bits hacia la derecha (por ejemplo, el valor 328, que en binario es ‘101001000’, al desplazarlo quedaría 000000101) y, a continuación, se realiza una operación AND con el valor 1Fh (0001 1111). A continuación, se realizará una OR del valor obtenido con el valor hexadecimal C0h (1100 0000). Con esto, en el ejemplo propuesto, queda el primer octeto de la siguiente forma: ‘110001011. Para el segundo octeto, se realiza una AND del valor del carácter con el valor 3Fh (0011 1111) y, a continuación, una OR de dicho valor con 80h (1000 0000), quedando en el ejemplo de la siguiente manera ‘1000 1000’. Finalmente quedará la secuencia codificada como se desea: ‘11000101 10001000’.

En recepción, será necesario un decodificador que haga el proceso exactamente inverso al que se acaba de describir, con tal de poder recuperar la información del mensaje original.

5.2.2. Estructura del paquete RTP

Como cualquier paquete RTP, un paquete RTP de texto se compone de cabecera y cuerpo (carga útil). La estructura de la cabecera será la misma que se especifica en la RFC 1889, sin embargo, la carga útil del paquete está compuesta por una estructura de bloque T.140, tal y como define la RFC 2793.

Un bloque T.140, puede contener uno o más caracteres, dependiendo de si el sistema de transmisión utilizado es de caracteres simples o de líneas de texto. El bloque T.140 no siempre tiene la misma forma, dependiendo su estructura de si se trabaja con redundancia o sin ella. En los paquetes que contengan redundancia, los datos correspondientes a la parte no redundante, se llamarán *texto primario*.

En la figura 5.1 se presenta el formato de un paquete simple (cabecera más carga útil con datos T.140), es decir, sin redundancia.

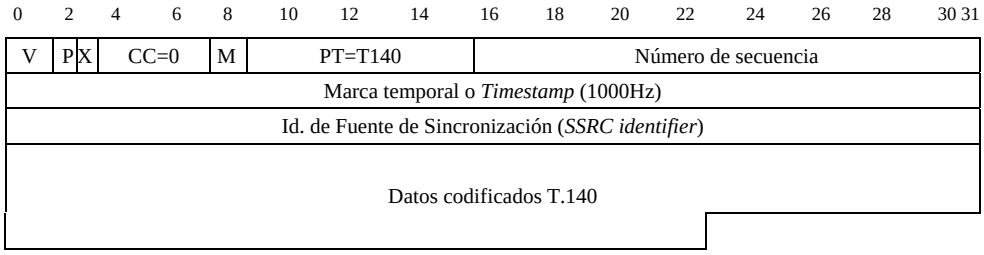


Figura 5.1. Estructura de un paquete RTP de texto sin redundancia

En este caso, el campo tipo (*Payload Type* o PT) tomará el valor 'T140'. En los paquetes que contengan redundancia tomará el valor 'RED'. En el campo *Timestamp* se colocará una marca de tiempo correspondiente al momento en el que se generó la entrada de texto primario del paquete según un reloj con una frecuencia de 1000 Hz.

Para este tipo de paquetes de datos no se especifican cabeceras adicionales a la de RTP. Cuando existe redundancia, la citada cabecera va seguida de una o más *cabeceras de bloques de redundancia*, una para cada bloque de datos redundantes incluidos en el paquete.

Por otro lado, como las comunicaciones no son totalmente perfectas, puede suceder que se pierda un paquete por cualquier causa, como, por ejemplo, una desconexión temporal espontánea debida a que un router de la red haya quedado fuera de servicio. Hasta que la red vuelva a ser estable puede pasar un cierto tiempo durante el cual se va a perder información que, si es de texto, va a tener diferentes grados de prioridad. Por ejemplo, un paquete de control SR RTCP, no es de vital importancia con respecto a la información de texto que se está enviando. La pérdida de un paquete SR implica que durante un tiempo no se va a obtener información del sistema remoto, pero en el momento en que se reestablezca la comunicación, se va a poder recuperar esa información. En cambio, un paquete RTP cuyo contenido sea información textual T.140 no se debe perder, porque si esto sucediese se produciría un 'vacío de información' notable y el usuario final lo percibiría.

Para evitar esta pérdida de paquetes, la RFC 2793 señala la posibilidad de incluir información de redundancia, a partir de la cual se puede recuperar información perdida. El método para agregar redundancia al paquete de datos, como ya se ha comentado, se describe en la RFC 2198 ([PER97]). Si se incluye redundancia, cuando se envía un paquete, los datos que se generan en el momento del envío, formarán la parte denominada *datos primarios*. Estos datos se insertan al final del paquete, concretamente después de los datos redundantes que son los datos no primarios y están compuestos por el conjunto de los 'n' paquetes T.140 enviados con anterioridad. El número de paquetes anteriores que serán colocados como datos redundantes dependerá del tipo de aplicación y de las condiciones bajo las que se ejecute. Tal como define la RFC 2198, el número mínimo de paquetes redundantes es uno, y el número máximo es elegido por el programador o por la propia aplicación, que deberá tener en cuenta parámetros del programa y de la red, tales como la memoria disponible, ancho de banda, tamaño del *buffer* de transmisión y recepción, etc.

Si no se conocen las condiciones de la red a través de la que se van a realizar las transmisiones, se recomienda la utilización de un único bloque de redundancia. El formato del paquete con un bloque de redundancia se muestra en la figura 5.2. Tal y como se ha mencionado anteriormente, el campo PT de la cabecera contiene el identificador 'RED', indicando que el paquete contiene redundancia. En este caso, al añadir el bloque de redundancia también aparece una especie de extensión de cabecera formada por las denominadas *cabeceras de bloque de redundancia*, cuya función es la de identificar los datos contenidos en cada bloque redundante T.140. Cada uno de los bloques adicionales que se inserten en el paquete llevará esta cabecera, además del cuerpo de datos T.140 correspondiente.

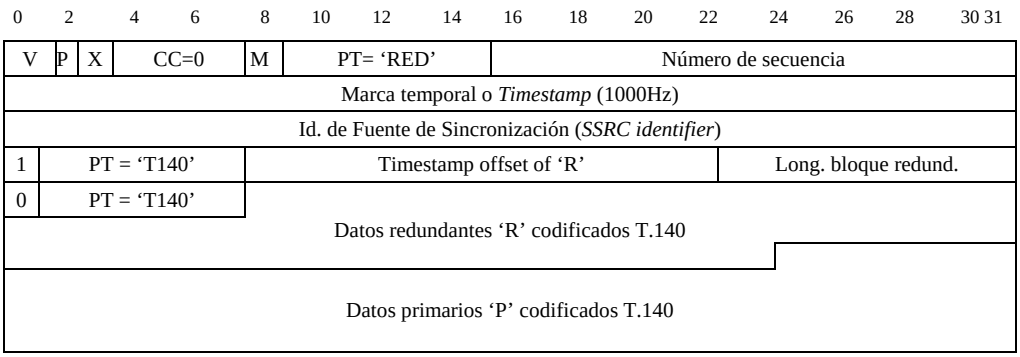


Figura 5.2. Estructura de un paquete RTP de texto con redundancia

- El primer bit de esta cabecera se denomina *more* y sirve para indicar si hay más bloques de datos a continuación.
- El campo *PT* (*Payload Type*) indica el contenido del paquete, que, en este caso, al ser texto, siempre contendrá 'T140'.
- El campo *Timestamp offset of 'R'*, en este caso, contiene el valor correspondiente a la diferencia entre el tiempo en que se generó el paquete al que pertenecen dichos datos redundantes y el tiempo indicado por el *timestamp* de los datos primarios.
- El campo *longituddel bloque redundante* contiene la longitud del bloque de datos redundantes T140. De esta forma se puede saber la longitud de este bloque para poder extraerlo correctamente en recepción.

El bloque de datos primario se coloca al final del mismo, no necesita una cabecera de bloque completa pudiéndose abreviar ya que la información relativa al mismo se puede extraer de la propia cabecera del paquete RTP y de la longitud total del paquete, y sólo será necesario un pequeño bloque formado por el bit *more*, que contiene el valor '0' (indicando que ya no quedan más bloques) y el PT que, en este caso, también será 'T.140'.

La utilización de redundancia introduce una mejor protección contra pérdidas de texto, pero estas pérdidas no vienen indicadas directamente en la información que contiene el paquete, por lo que se precisará de un mecanismo adicional para cuantificarlas. Para ello, se deberá realizar un análisis completo de los diferentes campos de la cabecera del paquete.

Dado que los paquetes de texto no se transmiten a intervalos regulares de tiempo, la marca de tiempo o *timestamp* contenida en la cabecera no es suficiente para identificar la pérdida de un paquete, a menos que se introduzca información extra en la cabecera. Los números de secuencia de la información redundante no se proporcionan en la cabecera de bloques de redundancia correspondiente, por tanto se deben proporcionar mecanismos adicionales para situar correctamente los datos

redundantes dentro de la secuencia de paquetes T.140 de recepción. Se siguen las siguientes pautas:

- Cada bloque de datos redundantes debe contener la misma información que el bloque T140 transmitido previamente en otro paquete como datos primarios y, además, el offset del *timestamp* debe coincidir con la diferencia de tiempos entre el tiempo actual y el *timestamp* del paquete original.
- Los datos redundantes deben ser posicionados por orden de antigüedad, situando el bloque redundante más reciente, al final del área de redundancia.
- Si se ha elegido una generación de redundancia de 'n' paquetes, todos los 'n' paquetes deben ser colocados en orden en el área de redundancia.

Si se siguen todos los puntos anteriores, es bastante sencillo conocer el número de secuencia de cada paquete siguiendo el siguiente método:

- Extraer el paquete actual y guardar el número de secuencia.
- Cada paquete que se va extrayendo se restará una unidad al número de secuencia de los datos primarios y el valor obtenido será el número de paquete que se estará analizando.

A continuación se muestra, en pseudocódigo, el procedimiento general:

```
Variable auxiliar: numseq_paq_actual        //almacena nº secuencia del paquete
Flag  fin de análisis: flagfin = 'no terminado'
Extrae_cabecera RTP();
Extrae_paquete_de_datos_primario(); //calcula tamaño  datos redundantes
Numseq =  número de secuencia del paquete de datos primario
...
WHILE  (flagfin = 'no terminado')   DO
{
        Numseq--          // disminuye en una unidad el número de secuencia

        Extrae_cabecera redundante();

        IF primer_bit_cabecera = 0          //si no hay más bloques de redund.
              flagfin = 'fin del análisis'
        ENDIF

        Copia_bloque_T140 (tamaño length )    //length es la longitud en bytes

        numseq_paq_actual = Numseq;
        ...
} // end while
```

Las aplicaciones deberán disponer de un *buffer* de salida que almacene los 'n' últimos paquetes enviados, que se incluirán como bloques de redundancia al paquete que se envíe en cada instante. Siguiendo este sistema, en el momento en que se encuentren huecos en los números de secuencia de los paquetes recibidos, se extraerán los bloques de redundancia correspondientes a dichos huecos. Además se realizarán las comprobaciones necesarias para determinar si se corresponden con el contenido de los paquetes que se han perdido. En caso de no utilizar redundancia, se podría, incluso, implementar algún tipo de técnica de retransmisión para recuperar paquetes perdidos.

5.3. Transmisión de Coordenadas de Punteros utilizando RTP/RTCP

En la mayoría de presentaciones o conferencias se presenta algún tipo de información, apoyándose normalmente en gráficos, imágenes y punteros. En aplicaciones de conferencias remotas es de vital importancia que los datos referentes a la posición del puntero se envíen de forma fiable y sincronizada con los flujos de audio y vídeo para no tener problemas de comprensión de la exposición.

Para este propósito se podría enviar toda la información visual del monitor del transmisor pero esto resultaría problemático ya que mientras que los gráficos necesitan ser enviados con una gran resolución espacial, los movimientos del puntero solo necesitan ser muestreados y transmitidos a una gran resolución temporal para que las acciones del transmisor con el puntero puedan ser visualizadas de forma sincronizada con los flujos de audio y vídeo de la presentación. En muchos casos esta sincronización es de gran importancia para el entendimiento de la presentación como, por ejemplo, en el caso de un ponente que esté presentando varias alternativas señalando un gráfico donde aparezcan datos de cada alternativa y las esté comparando constantemente con el puntero.

Mediante RTP/RTCP se pueden enviar las coordenadas del puntero dinámico tal y como define la RFC 2862, que propone el formato de la carga útil de los paquetes RTP para transportar dichas coordenadas. El formato aparece en la figura 5.3.

La asignación de un tipo de paquete RTP (campo *PT*) no se define en la RFC 2862 pero en ella se indica que deberá ser definido en los perfiles particulares para cada clase de aplicaciones.

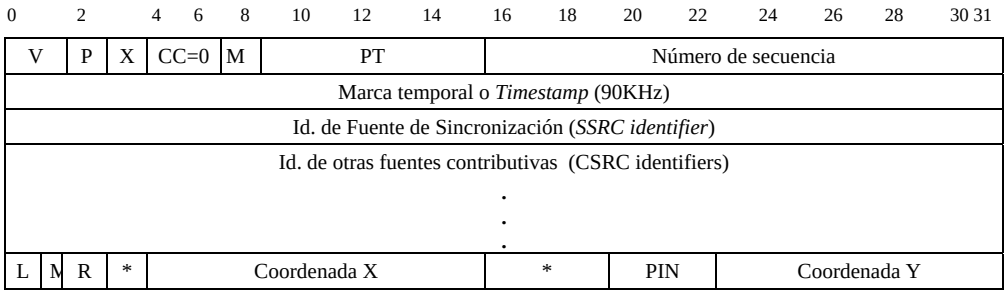


Figura 5.3. Estructura de un paquete RTP con coordenadas X e Y de un puntero en tiempo real

En este caso, según la RFC, en el campo *Timestamp* se colocará una marca de tiempo correspondiente al momento en el que se generó la posición del puntero codificada en el paquete según un reloj con una frecuencia de 90 kHz.

Se codifican las coordenadas X e Y con respecto al extremo superior izquierdo de la ventana asociada al gráfico visualizado. Son representadas como una fracción de la longitud del correspondiente borde (horizontal o vertical) de la ventana utilizando 12 bits sin signo.

Los bits L (*Left*), R (*Right*) y M (*Middle*) son bits de efectos especiales (*pointer special effects flags*) y el uso de los mismos se deja abierto y dependerá de la aplicación, siendo definidos fuera de banda (*out-of-band*). Las aplicaciones pueden ignorar estos bits.

El campo *PIN* (*Pointer Icon Number*), de 3 bits sirve para seleccionar un número o tipo de icono para el puntero. La asociación entre los número de *PIN* y los iconos de puntero deben ser establecidos fuera de banda. Un valor de 0 indica el icono por defecto.

Además, tal como se puede apreciar en la figura anterior, existen dos bits de comprobación que deberán contener el valor '0' (MZB o *Must Be Zero*).

5.4. Posibilidad de Cifrado

Para la transmisión segura de datos utilizando RTP a través de una red de comunicaciones se puede implementar algún tipo de protección de los datos cifrándolos antes de ser introducidos en los paquetes RTP, mediante algoritmos de encriptación como, por ejemplo, MD5 (*Message Digest 5*), definido en la RFC 1321 ([RIV92]), donde se puede encontrar ejemplos de código de implementación

del mismo. MD5 es un algoritmo de encriptación de clave privada que toma como entrada un mensaje de cualquier longitud y lo cifra a partir de una clave, de tal manera que sólo podrá ser descifrado por todos aquellos que conozcan dicha clave.

Se debe remarcar que la encriptación puede ralentizar el proceso de transmisión y recepción de la información, aumentando el retardo extremo a extremo de los datos (debido al cifrado y descifrado de la información) y afectar a la interactividad y al proceso de sincronización.

5.5. Aplicación de transmisión de Texto utilizando RTP/RTCP

Como ya se ha comentado, en el marco de la presente tesis se ha desarrollado una aplicación de transmisión de texto mediante RTP/RTCP que forma parte del trabajo realizado sobre sincronización multimedia durante el desarrollo de la misma. La aplicación ha sido desarrollada en la Escuela Politécnica Superior de Gandia, y en colaboración con D. Gunnar Hellstrom, autor de la RFC 2793 (*RTP Payload for Text Conversation*, [HEL00]), de la empresa sueca *OMNITOR AB* y miembro del IETF, que ha facilitado ejemplos de código relacionado con su RFC, escritos en java. En la figura 5.4. aparece la ventana principal de la aplicación desarrollada.

Esta aplicación constituye un software experimental, que valiéndose de un protocolo para transmisión en tiempo real (RTP), trata de conseguir una comunicación lo más rápida y eficaz posible entre diversos usuarios conectados a una o varias redes, mediante una conversación de carácter textual. El interfaz gráfico ha sido desarrollado mediante TCL/TK y el resto de la aplicación con MS Visual C++ 6.0.

Se ha integrado junto con las dos aplicaciones de MBone de audio y vídeo (*rat* y *vic*) modificadas para el desarrollo de la Tesis, formando una plataforma real donde se ha podido probar el funcionamiento del protocolo propuesto y donde se podrán probar en el futuro nuevas ideas y modificaciones de los protocolos de sincronización multimedia futuros.

En esta aplicación, se utiliza RTP/RTCP para transportar datos de texto introducido por el usuario desde el teclado (aunque se podría tomar de otras fuentes, por ejemplo, reconocimiento de voz, datos de sensores, etc.). Haciendo uso de las marcas temporales que proporcionan dichos protocolos, se podrá sincronizar la presentación del texto (datos) con la de los flujos de audio y vídeo de las otras aplicaciones.

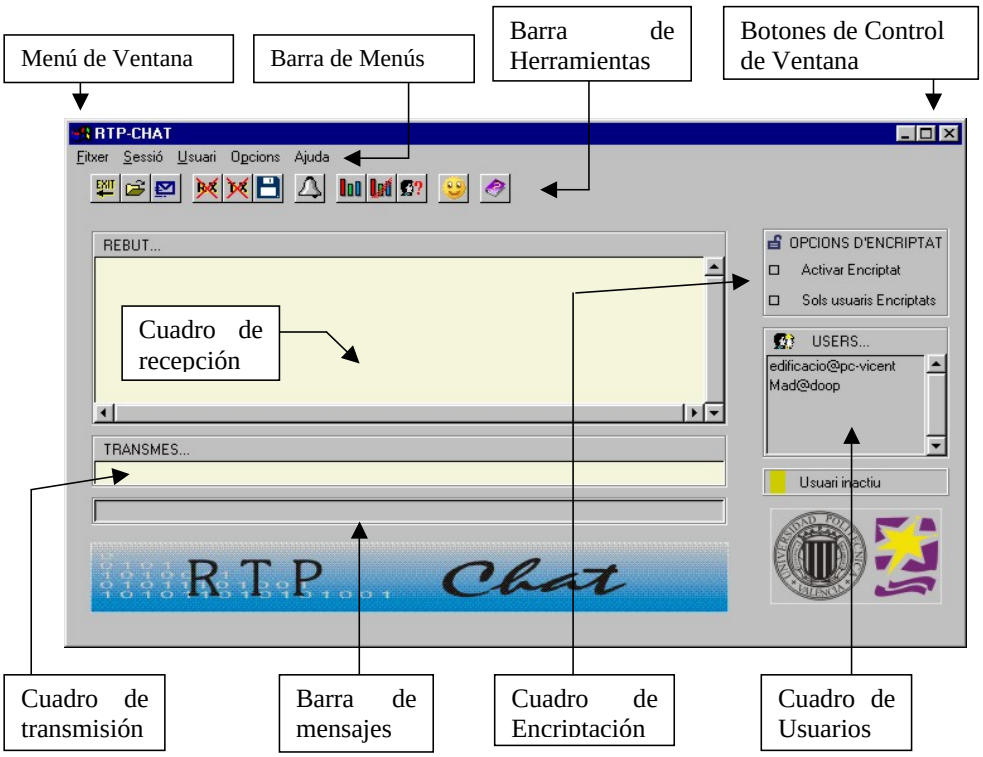


Figura 5.4. Ventana principal de la aplicación

En el desarrollo de la aplicación se ha seguido la RFC 2793, y la parte de codificación del texto a transmitir sigue la norma ITU-T.140. El texto transmitido sufre una conversión al formato UTF-8, basado en caracteres de 8 bits, que permite la adaptación de esta aplicación con las tipografías de casi todos los lenguajes del mundo.

La aplicación ofrece la posibilidad de establecer una conexión tanto *unicast* *multicast*. Dado que el programa puede ser utilizado por varios usuarios simultáneamente, en una misma conexión, se lleva una gestión de los mismos, mediante bases de datos donde se almacena información de cada uno de ellos.

Además, también se considera la posibilidad de coexistencia de varios usuarios transmisores y receptores simultáneamente, siendo necesaria la implementación de un sistema de colas para evitar la pérdida de información cuando el número de usuarios crece de forma significativa.

Tal y como se ha comentado anteriormente, la aplicación permite conversaciones privadas mediante el uso de encriptación. Para ello se ha

implementado el sistema de encriptación basado en el algoritmo de autenticación MD5.

5.5.1. Paquetes RTP implementados

En la aplicación se han utilizado diferentes paquetes de datos distinguiéndose entre paquetes de datos de texto y paquetes de datos con avisos o información de la sesión. Entre los primeros están los paquetes *LINEMSG* y *LINEMSGRED* y entre los segundos están los paquetes *BELL* y *SESSIONMSG*, que se explican a continuación.

- Paquete *LINEMSG*: Se trata del tipo de paquete de datos básico, sin redundancia (ver figura 5.1). La información básica que lleva es una línea de texto.
- Paquete *LINEMSGRED*: Paquete de datos que contiene datos redundantes del paquete anterior (ver figura 5.2).
- Paquete *BELL*. Es el utilizado para dar una alerta (visual, sonora, o ambas) al usuario seleccionado desde la interfaz gráfica.
- Paquete *SESSIONMSG*. Contiene información detallada relativa a parámetros de sesión del usuario local. Se utiliza en el momento del establecimiento de la conexión y para modificar los parámetros de la sesión.

Como es lógico, en la aplicación también se han utilizado los paquetes típicos del protocolo RTCP, como son los informes del emisor y del receptor (RR, SR), de descripción de la fuente (SDES) y el paquete de adiós (BYE), con los formatos especificados en la RFC 1889.

5.5.2. Funciones del Codificador y Decodificador UTF-8

En la aplicación se ha implementado un codificador/decodificador UTF-8, que actúa de forma transparente al sistema. Mediante el lenguaje de programación ‘C’, se han creado dos funciones que, implementando el algoritmo descrito anteriormente, realizan las tareas de transformar de UCS-4 a UTF-8, y viceversa.

La cabecera de la *función de codificación* tiene el siguiente aspecto:

*void writeUTF(int *cadenaEntrada, int longcadena, __int8 *textosalida, int *lc_sal)*

, cuyos parámetros son los siguientes:

- *cadenaEntrada*: es un puntero a la cadena de caracteres que contiene el mensaje a transformar a UTF-8. En la aplicación presentada será la cadena que se desea transmitir por el canal de comunicaciones. Se debe notar que dicha cadena estará compuesta por un conjunto de caracteres de 16 bits, por lo cual, si se tiene un conjunto de caracteres ASCII (8 bits), antes de pasar dicho conjunto a la función se deberá convertir a cadena de caracteres de 16 bits.
- *longcadena*: longitud de la cadena de entrada (*CadenaEntrada*).
- *textosalida*: puntero a la cadena de salida. Mediante este puntero, la función retorna la cadena de entrada ya convertida a una secuencia de octetos (8 bits), tal como se ha descrito en el algoritmo de transformación.
- *lc_sal*: devuelve el valor de la longitud de la nueva cadena de salida.

La cabecera de la *función de decodificación* tiene un aspecto muy parecido al anterior

*void readUTF(__int8 *buffer, int *cadenaSalida, int longitud, int *lc_sal2)*

, y los parámetros que se le pasan son los siguientes:

- *buffer*: puntero a la cadena de caracteres (de 8 bits) que se desea decodificar. En la aplicación presentada será la cadena que se recibirá por el canal de comunicaciones.
- *cadenaSalida*: cadena de caracteres de 16 bits de salida. Mediante este puntero, la función devuelve la cadena decodificada. Si lo que se recibe es texto ASCII, se deberá convertir esta cadena de salida a una cadena de caracteres de 8 bits.
- *longitud*: longitud de la cadena de entrada (*buffer*).
- *lc_sal2*: mediante este puntero, la función devuelve el tamaño de la cadena decodificada (*cadenaSalida*).

5.6. Conclusiones

La mayoría de aplicaciones de transmisión de flujos multimedia que utilizan RTP/RTCP han sido desarrolladas para transmitir flujos continuos, como los de audio y vídeo. Se ha detectado una falta de herramientas de transmisión de datos no continuos, como por ejemplo, de texto, de ficheros, de coordenadas de punteros, etc., cuyas restricciones temporales no son tan severas como las de los flujos continuos. En este capítulo, y *como contribución adicional de la Tesis*, tras un estudio preliminar de las diferentes RFCs que proponen transmisión de datos no continuos sobre RTP/RTCP, se propone utilizar el formato de carga útil definido en la RFC 2793 ([HEL00]), con un solo bloque de redundancia, para transmitir

cualquier tipo de datos de información que se puedan codificar como una cadena de caracteres. Dicha RFC define cómo transportar texto sobre paquetes RTP, de acuerdo con lo definido en la recomendación UIT-T.140 y codificando los caracteres mediante codificación UTF-8. La estructura de la secuencia de caracteres dependerá de la aplicación.

Por ejemplo, se podría aplicar a aplicaciones de televigilancia o telecontrol donde existen dispositivos que disponen de salidas RS-232 por la que transmiten datos bien binarios o bien en formato ASCII siguiendo una determinada estructura (por ejemplo, la sentencia NMEA de un receptor GPS, la salida en formato ASCII de una estación meteorológica, etc.). Estos datos pueden ser convertidos a caracteres (o grupos de n bits) y, posteriormente ser convertidos a formato UTF-8 para poder ser incluidos en paquetes RTP, tal y como se ha explicado en el capítulo.

Para demostrar la propuesta realizada, se ha implementado una aplicación de transmisión de datos de texto basada en la RFC anterior. Ya que se trata de una transmisión de texto en tiempo real y las aplicaciones más extendidas de este tipo sobre redes IP son las de *Chat*, se la ha dotado de una interfaz y de unas utilidades semejantes a la mayoría de dichas aplicaciones. Sin embargo, la finalidad última de esta aplicación es demostrar la viabilidad de utilización de RTP/RTCP para transmisión de datos no continuos.

Como contribuciones adicionales de la Tesis, en este capítulo, además de la aplicación anterior, también se han desarrollado en ‘C’ los algoritmos de codificación y decodificación UTF-8, así como la inclusión y tratamiento de redundancia (de texto) en los paquetes RTP y el cifrado MD-5 del texto en la aplicación.

Capítulo 6

Evaluación Objetiva y Resultados

6.1. Introducción

Como ya se ha comentado en capítulos anteriores, el concepto de sincronización en las aplicaciones multimedia abarca, por un lado la definición de relaciones temporales entre varios flujos de información y, por otro lado, el cumplimiento de dichas relaciones en el instante de la reproducción. Los protocolos y algoritmos de sincronización son necesarios para asegurar dicha sincronización durante toda la sesión multimedia con unos niveles mínimos de calidad.

Respecto a la calidad de una presentación multimedia se puede distinguir entre *Calidad Objetiva* y *Calidad Subjetiva*. La *Calidad Objetiva*, en general, se refiere a las diferencias entre la presentación original y la reproducida, en términos de resolución, colores, tasa de reproducción, nivel de sincronización, etc. La *Calidad Subjetiva* se refiere a la percepción de la calidad de la presentación (en cuanto a los mismos parámetros) por parte de un observador humano.

Por tanto, la evaluación del funcionamiento de los protocolos de sincronización multimedia se puede evaluar de dos formas: objetiva y subjetivamente. En este capítulo nos vamos a ocupar de la *evaluación objetiva* del comportamiento de nuestro algoritmo, analizando ciertos parámetros característicos, tales como la asincronía (medida en milisegundos o en número de LDU's) entre diferentes flujos, el número de veces que se supera un umbral establecido, la cantidad de mensajes de control utilizados para mantener la sincronización dentro de un margen especificado, etc. También se deben considerar efectos producidos por los retardos y las variaciones de los mismos o *jitter* (tanto los introducidos por la red como los debidos a los dispositivos reproductores), la desviación de los relojes de los distintos equipos involucrados en un sistema multimedia, así como el efecto que introducen las diferentes acciones correctoras del algoritmo de sincronización. La *evaluación subjetiva* se presenta en el próximo capítulo.

R. Steinmetz, en [STE96] investigó el efecto que producían sobre la percepción humana las diferencias de tiempos de reproducción de diferentes flujos en una presentación multimedia. Parte de las conclusiones de dicho trabajo aparecen en la tabla 3.1 del capítulo 3. Como conclusión principal, señala que una diferencia de ±80 milisegundos, entre los flujos de audio y vídeo en una transmisión multimedia, estaría dentro de una sincronización considerada de alta calidad, pero que una diferencia que excediera los ±160 milisegundos se consideraría asincronía y no sería tolerada por los usuarios. Como para la sincronización de grupo, no está clara cuál es la diferencia máxima permitida entre los procesos reproductores del receptor *maestro* y los receptores *esclavos*, será necesario hacer una evaluación subjetiva del comportamiento de nuestro algoritmo. No obstante, como se ha tomado el flujo de audio como flujo *maestro* y su reproducción conjunta en todos los receptores será la que indique el grado de sincronización de grupo, se ha tomado como valor de asincronía máxima permitida que garantice una correcta sincronización entre los procesos reproductores de dicho flujo en todos los receptores, el valor correspondiente a *flujos de audio poco acoplados en modo diálogo con varios participantes* que, según Steinmetz, es de ±120 milisegundos. Según dicho autor, en la reproducción de un diálogo en el que los datos de audio de los participantes provienen de diferentes fuentes se pueden tolerar asincronías inferiores a dicho valor ([STE96]).

6.2. Descripción de los Escenarios de Prueba

Para la evaluación del algoritmo propuesto, se han adaptado y combinado las aplicaciones MBone modificadas en la Tesis (*vic* y *rat*) para conseguir dos tipos diferentes de aplicaciones multimedia de prueba:

- Aplicación de Aprendizaje a Distancia.
- Aplicación de Televigilancia.

Ambas se han evaluado en un entorno LAN (red de laboratorio) y en un entorno de CAMPUS (red de interconexión de los Campus de Valencia y Gandia de la Universidad Politécnica de Valencia).

Para la implementación del algoritmo propuesto, en ambos escenarios se ha tomado como *flujo maestro* el de audio que es más restrictivo y se ha seleccionado, de forma manual, uno de los receptores como *receptor maestro* mediante la opción correspondiente de la *interfaz gráfica de usuario* de la aplicación *rat* (desactivándose, por tanto, los algoritmos de selección de receptor *maestro* en la fuente sincronizadora). Esto último se ha realizado para simplificar la ejecución del algoritmo.

En ambas aplicaciones, los procesos de reproducción de los flujos de vídeo (*esclavos*) se sincronizan al proceso de reproducción del flujo de audio (*maestro*), de acuerdo con el diagrama de la figura 6.1, explicado en el capítulo 4.

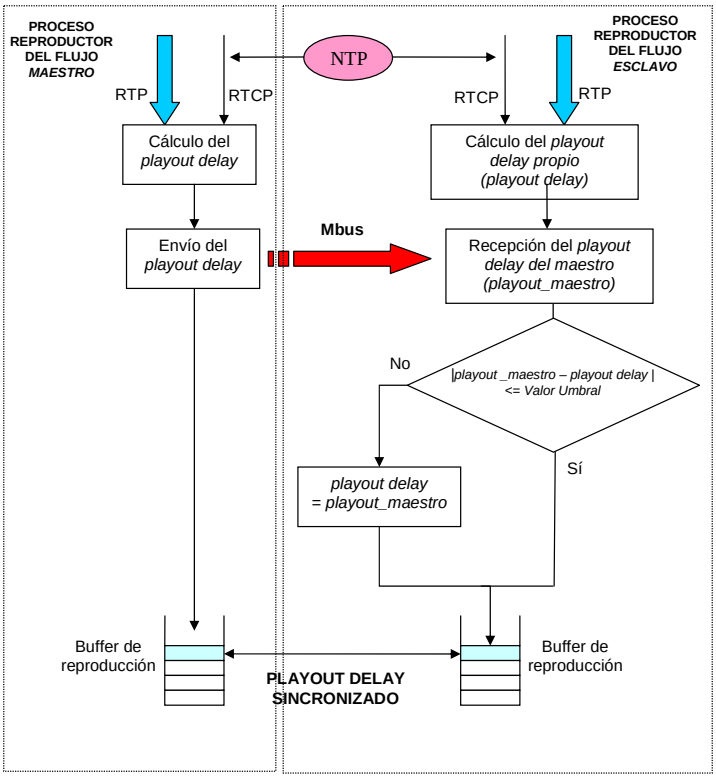


Figura 6.1. Diagrama para el cálculo de reproducción sincronizada

Se puede apreciar en el mismo que el proceso reproductor del flujo *maestro* calcula el valor de su *playout delay* y lo envía al proceso reproductor del flujo *esclavo*, que lo comparará con su propio *playout delay* y que, en caso de detectar una desviación por encima de un valor umbral, desechará el valor propio y tomará el valor recibido. Se fija un valor umbral para evitar continuos ajustes que afectarían negativamente a la calidad de la reproducción del flujo *esclavo*.

6.2.1. Aplicación de Aprendizaje a Distancia

El siguiente escenario de prueba nos ha servido para verificar que el algoritmo propuesto se puede utilizar para conseguir la sincronización de grupo necesaria, por ejemplo, en *Aplicaciones de Aprendizaje a Distancia*, en las que el profesor o tutor envíe un vídeo de una película (flujo de contenido almacenado) y, al mismo tiempo, en determinados instantes, pueda hacer comentarios (flujo de contenido en directo) sobre las imágenes que se están viendo de la película y los estudiantes pueden discutir dichos comentarios.

El objetivo que se persigue es conseguir una reproducción simultánea de los flujos, tanto de contenidos almacenados como en directo, para todos los estudiantes. Incluso aunque sólo se enviara la película, es decir, el flujo de contenidos almacenados, cada LDU (por ejemplo, una trama de vídeo) debería ser reproducida simultáneamente en los diferentes destinos (estudiantes).

Como se ha comentado, se ha evaluado el comportamiento del algoritmo implementado en esta aplicación en dos entornos: LAN y CAMPUS, que se describen a continuación.

6.2.1.1. Entorno LAN

Para verificar el comportamiento anterior, el primer entorno donde se ha evaluado el algoritmo, mostrado en la figura 6.2, está constituido por un servidor de flujos multimedia (transmitiendo audio y vídeo) y varios receptores reproductores, interconectados mediante una red de área local (LAN).

La prueba se ha realizado con 11 equipos, un transmisor multimedia y 10 receptores, en uno de los laboratorios del Campus de Gandia, cuyas características técnicas aparecen en la tabla 6.1.

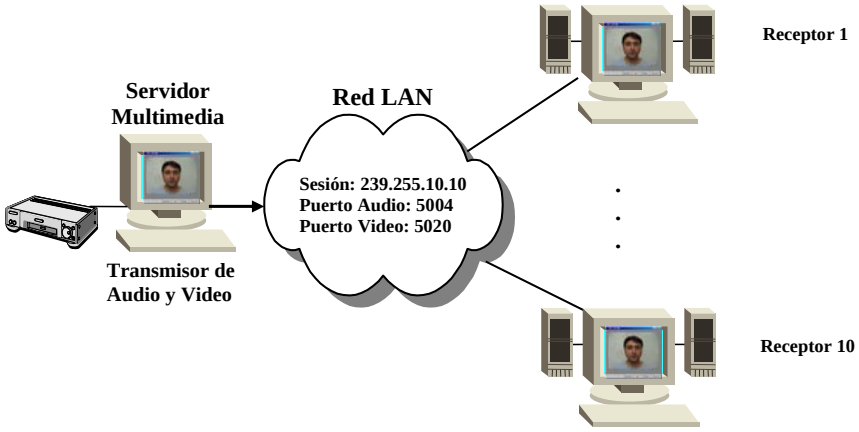


Figura 6.2. Entorno LAN de prueba de la *Aplicación de Aprendizaje a Distancia*

Equipo	Procesador	RAM (MB)	S.O.	Tarjeta Audio	Tarjeta Vídeo
Servidor Multimedia	PIV a 900MHz	512	W98	Sound Blaster AudioPCI	AverMedia EzCapture
PC1 a PC10	ATHLON 1600+ XP a 1,4 GHz	128	W98	Creative Sound Blaster PCI 128	-

Tabla 6.1. Características técnicas de los equipos empleados en transmisión y recepción (aprendizaje a distancia)

Todos los equipos estaban conectados a un dispositivo concentrador o hub de 16 puertos a 10Mbps, marca ACCTON, modelo ETHERHUB 16S.

6.2.1.2. Entorno CAMPUS

Para verificar el comportamiento en un entorno más amplio, con mayor jitter y retardo de red, el segundo entorno donde se ha evaluado el algoritmo, mostrado en la figura 6.3, está constituido por un servidor de flujos multimedia (transmitiendo audio y vídeo) ubicado en el Campus de Vera de la Universidad Politécnica de Valencia (en adelante, UPV), en la ciudad de Valencia, y los diez receptores ubicados en el Campus de la Escuela Politécnica Superior de Gandia, en la ciudad de Gandia, a unos 70 kilómetros de distancia, haciendo uso de las conexiones de datos existentes entre los dos Campus.

El entorno se muestra en la figura 6.3 y las características técnicas de los 11 equipos involucrados son las mismas que para la prueba en red local (entorno

LAN). Por otro lado, los dispositivos de interconexión utilizados en este caso son los siguientes[1]

- Routers CISCO 7200 VXR
- HUB de16 puertos a 10Mbps, marca ACCTON, modelo ETHERHUB 16S.

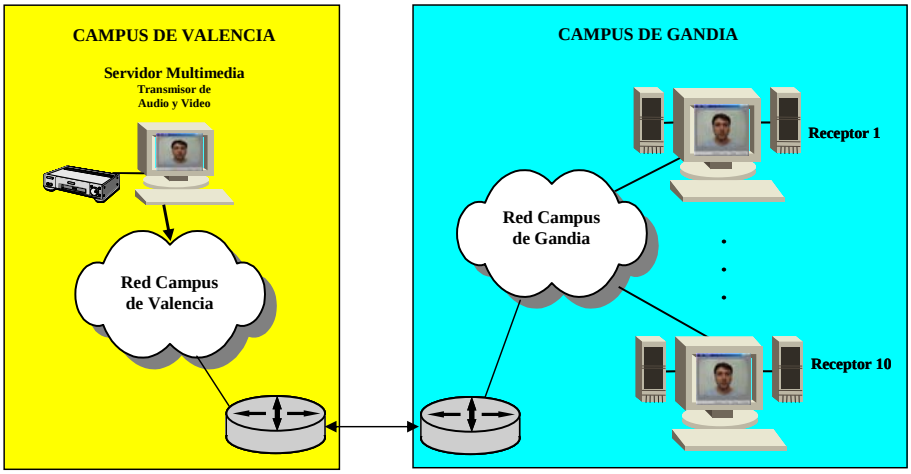


Figura 6.3. Entorno de CAMPUS, de prueba de la *Aplicación de Aprendizaje a Distancia*

6.2.2. Aplicación de Televigilancia

También se ha evaluado la posibilidad de utilizar el algoritmo de sincronización propuesto en una *Aplicación de Televigilancia*. En una sala o puesto de control con uno o varios dispositivos receptores, se sincronizará la información multimedia que se reciba desde varios dispositivos o fuentes remotas, que estén transmitiendo flujos de información multimedia (p.ej. audio y vídeo), todos ellos con relojes sincronizados globalmente con NTP.

La diferencia, en este caso, con respecto a la aplicación anterior radica en el número de fuentes independientes de flujos multimedia. En el escenario de prueba diseñado para esta aplicación, existen tres fuentes independientes: dos de vídeo y una de audio. Las fuentes de vídeo transmiten vídeo mediante la herramienta *vic* y la fuente de audio transmite audio mediante la herramienta *rat*.

[1] Datos facilitados por el Centro de Procesado de Datos de la UPV

En cuanto a los receptores, en la sala de control, existirán tres receptores, también independientes, uno para cada flujo, que, por tanto, ejecutarán herramientas independientes.

Del mismo modo que en la aplicación anterior, en esta aplicación también se ha evaluado el comportamiento del algoritmo propuesto en los entornos LAN y de CAMPUS, descritos a continuación.

6.2.2.1. Entorno LAN

En este caso, el escenario para evaluar el algoritmo propuesto será el mostrado en la figura 6.4. La prueba de evaluación se ha realizado con 6 equipos, con tres fuentes (una de audio y dos de vídeo) y tres receptores (también uno de audio y dos de vídeo), cuyas características técnicas aparecen en la tabla 6.2.

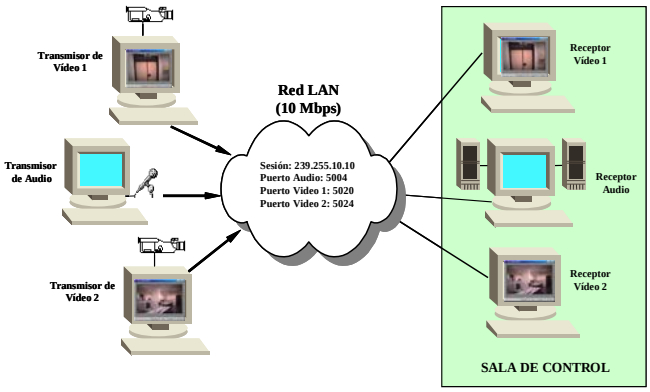


Figura 6.4. Entorno LAN de prueba de la *Aplicación de Televigilancia*

Equipo	Procesador	RAM (MB)	S.O.	Tarjeta Audio	Tarjeta/Cámara Vídeo
Transmisor Vídeo 1	PIV 900MHz	512	W98	-	AverMedia EzCapture/Cámara SONY CCX-Z11E
Transmisor Vídeo 2	PIII 933 MMX	128	W98	-	Cámara USB Creative Webcam Plus
Transmisor Audio	ATHLON 1600+ XP a 1,4 GHz	128	W98	Creative Sound Blaster PCI 128	-
Receptores	ATHLON 1600+ XP a 1,4 GHz	128	W98	Creative Sound Blaster PCI 128	-

Tabla 6.2. Características técnicas de los equipos empleados en transmisión y recepción (televigilancia)

En este entorno, todos los equipos están conectados a un HUB de 16 puertos a 10Mbps, marca ACCTON, modelo ETHERHUB 16S.

6.2.2.2. Entorno CAMPUS

En este caso, del mismo modo que para la aplicación anterior, los tres transmisores están conectados a diferentes segmentos de red en el Campus de Valencia, y los tres receptores están conectados al segmento de red de uno de los laboratorios del Campus de Gandia, tal como se aprecia en la siguiente figura.

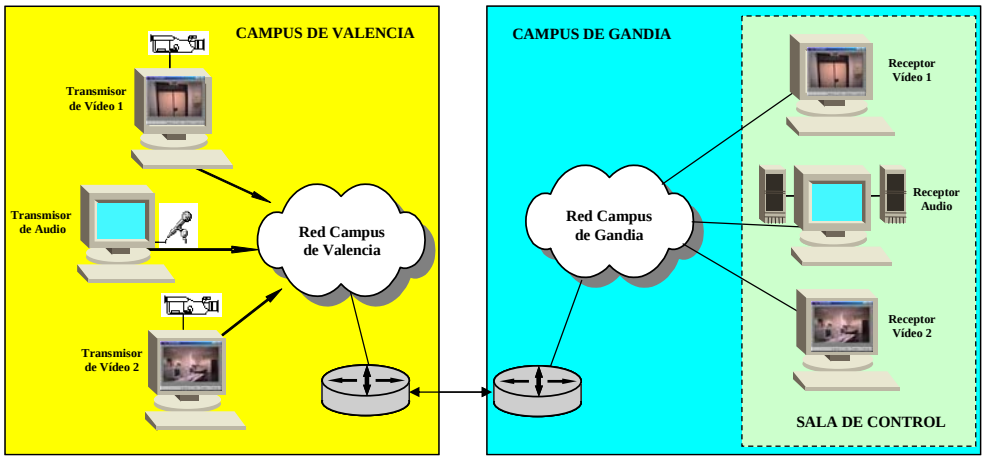


Figura 6.5. Entorno CAMPUS de prueba de la *Aplicación de Televigilancia*

Los 6 equipos utilizados en este entorno son los mismos que para el entorno LAN. Los equipos de interconexión y la configuración es la misma que para las pruebas en entorno CAMPUS para la Aplicación de Aprendizaje a Distancia.

6.3. Evaluación Objetiva

Como se ha comentado anteriormente, en este apartado se va a presentar la evaluación objetiva del algoritmo propuesto en ambas aplicaciones y en los dos entornos descritos en el apartado anterior.

Para poder hacer una evaluación objetiva del *Algoritmo de Sincronización de Grupo basado en RTP/RTCP y NTP* desarrollado e implementado sobre las aplicaciones MBone de audio y vídeo seleccionadas (*rat* y *vic*), se han tomado una serie de medidas de ciertos parámetros significativos en tiempo real, para lo cual se

ha tenido que añadir, al código fuente de las propias aplicaciones, unas funciones de monitorización y captura de dichos parámetros.

Los parámetros más importantes y significativos elegidos para realizar la evaluación objetiva de las prestaciones del algoritmo propuesto han sido los siguientes:

- El *retardo de reproducción (playout delay) de los flujos recibidos.* Será el retardo, en milisegundos, que sufren las LDUs desde el instante de su envío hasta el instante de su reproducción de los flujos de audio y de vídeo.
- *Número de secuencia (LDU) reproducida* en los receptores de los distintos flujos.
- Los *ajustes*, en milisegundos, realizados en los procesos reproductores de los receptores del flujo *maestro* como consecuencia de las acciones de resincronización, al recibirse un paquete APP TIN o APP ACT, para conseguir la sincronización de grupo (distribuida) esperada.
- Los *ajustes*, en milisegundos, entre los procesos reproductores de los flujos *esclavos* para conseguir la sincronización inter-flujo (local) esperada.
- *Valor cuadrático medio de las diferencias* en el estado de reproducción entre receptores (sincronización de grupo distribuida) y localmente (sincronización inter-flujo). Si se obtienen los valores medios de las diferencias medidas, es posible que valores negativos compensen a valores positivos de las mismas a la hora de realizar el cálculo, enmascarando el comportamiento real. Es por ello que el valor cuadrático ofrece una mayor información.
- *Número total de paquetes RTP enviados* del flujo *maestro*.
- *Número y tipo de mensajes* de control enviados por los equipos involucrados en la sesión.

Por último, destacar que los valores de estos parámetros, debido a que se han tomado de forma local e independiente en cada uno de los receptores, se almacenan junto con el instante de tiempo NTP en que son capturados, y, así, poder interpretarlos posteriormente y poder comparar, temporalmente (sobre un eje de tiempos NTP común), los datos obtenidos de todos los receptores.

Como se ha comentado anteriormente, en ambos escenarios, todos los equipos se sincronizan mediante un tiempo global proporcionado por el protocolo NTP. La herramienta utilizada para dicha sincronización es el cliente NTP denominado *NTPTime*, y se ha configurado para que obtenga la referencia de tiempos de un servidor NTP. Para la evaluación en entorno LAN, este servidor será local, es decir, estará conectado a la misma red, con sistema operativo *Windows 2000 Server* y el *Servicio de Tiempos NTP* activado. Para la evaluación en entorno de CAMPUS se ha escogido un servidor de tiempos primario de la Red Iris, *hora.rediris.*es, de *stratum* 1. En ambas sesiones de evaluación, en todos los

equipos, los clientes NTP se configuran para que sincronicen sus relojes, automáticamente cada 30 segundos, con el tiempo proporcionado por el servicio NTP de dichos servidores, tal como se muestra en la figura 6.6.

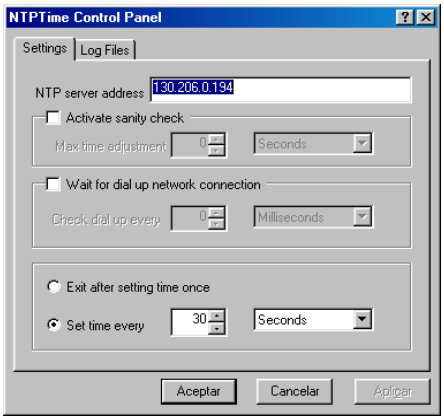


Figura 6.6. Configuración del cliente NTP

Para la transmisión de audio mediante la aplicación *rat*, se ha utilizado una codificación GSM con una duración de los paquetes de 20 ó 40 milisegundos, un muestreo de 8KHz (mono). Por otra parte, el flujo de vídeo transmitido mediante la aplicación *vic* se ha configurado para enviarse haciendo uso de una codificación H.261 de calidad normal (según menú de configuración de la herramienta *vic*), 25 fps (tramas por segundo). Estos datos están reflejados en la tabla 6.3.

Parámetro	Audio	Vídeo
Esquema de codificación	GSM	H.261
Tasa de LDUs (LDUs/s)	8000 muestras/s	25 tramas/s
Tiempo inter-LDU (ms)	20-40	40

Tabla 6.3. Especificaciones de los flujos de audio y vídeo

Para la evaluación del algoritmo en las dos aplicaciones, primero se ha evaluado el comportamiento sin la utilización del algoritmo propuesto para, posteriormente, poder comparar y observar las mejoras introducidas por éste. Por tanto, para ambos entornos se han realizado las siguientes pruebas:

A) Evaluación sin implementar el algoritmo propuesto.
B) Evaluación implementando el algoritmo propuesto.

6.3.1. Evaluación y Resultados del Algoritmo de Sincronización obtenidos en la Aplicación de Aprendizaje a Distancia.

Para esta aplicación, en todos los equipos se ejecutará una herramienta, desarrollada en la presente Tesis, que engloba en una misma interfaz gráfica las aplicaciones de audio, vídeo y texto (*rat*, *vic* y la herramienta de transmisión de texto sobre RTP implementada). Aunque se haya incluido la herramienta de transmisión de texto sobre RTP, no se ha evaluado durante las pruebas, evaluándose solamente la sincronización entre los flujos de audio y vídeo que es más restrictiva. En la figura 6.7 se muestra la ventana de inicio de dicha aplicación, donde se aprecian los argumentos utilizados por cada una de las herramientas.

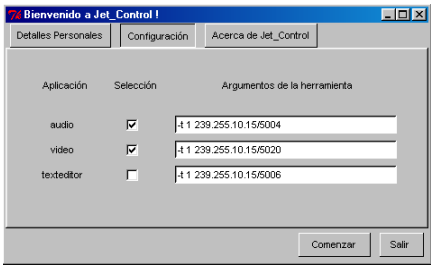


Figura 6.7. Herramienta de control de la interfaz gráfica

En la figura 6.8 se muestra la ventana principal de la aplicación vista desde el Servidor Multimedia durante las pruebas realizadas.

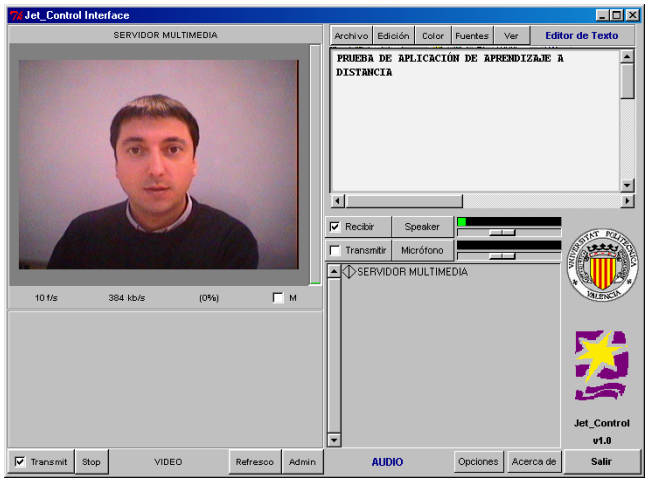


Figura 6.8. Herramienta de transmisión de audio, vídeo y texto

Puede observarse que se divide en tres partes, correspondientes a las tres herramientas seleccionadas. La parte de la izquierda corresponde a la herramienta de vídeo, la parte superior derecha a la herramienta de texto y la parte inferior derecha a la herramienta de audio.

A continuación se presenta la evaluación de esta aplicación en los dos entornos comentados.

6.3.1.1. Entorno LAN

La prueba consistió en enviar los flujos de audio y vídeo de una grabación, de 10 minutos de duración aproximada, de un noticiario de la televisión local de Gandia (TVG).

A) SIN ALGORITMO

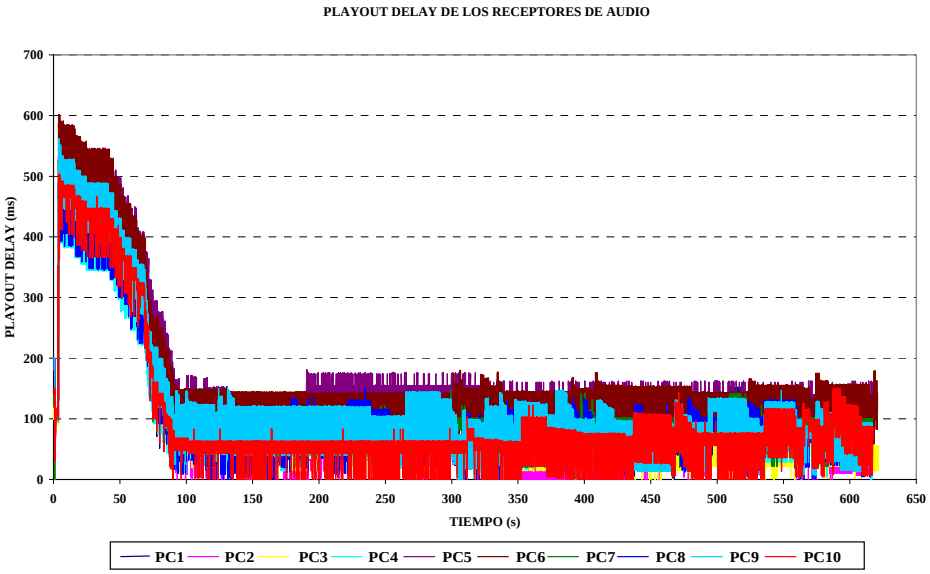
Se han transmitido los flujos de audio y vídeo desde el Servidor de Flujos Multimedia hacia 10 receptores (PC1 a PC10) situados en la misma red local. En este caso, aunque no se ejecute el algoritmo propuesto, se han tenido que sincronizar todos los relojes de los equipos involucrados, mediante NTP, para poder disponer de una referencia de tiempos común y poder comparar los parámetros obtenidos durante la sesión. Además, todos tienen, en la herramienta *rat*, desactivada la opción de sincronización mediante el algoritmo propuesto.

- ***Retardos de reproducción (playout delay) de los procesos reproductores del flujo maestro de todos los receptores***

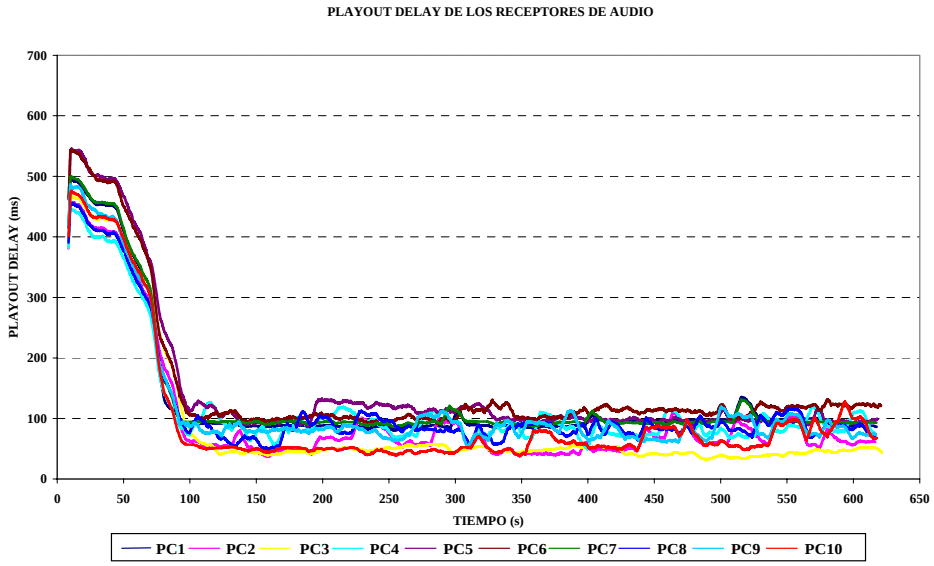
Como se ha explicado anteriormente, el valor de retardo de reproducción o *playout delay* es el tiempo que transcurre desde que la fuente envía una LDU hasta el momento de su reproducción en el receptor(2). A continuación, se muestra la figura 6.9, que representa el retardo de reproducción o *playout delay* del flujo de audio experimentado por los diferentes receptores. La gráfica representa el intervalo de tiempo correspondiente a toda la sesión. Si los procesos reproductores de los receptores estuvieran sincronizados, estos valores deberían ser similares en cada instante de tiempo en todos ellos.

[2] El parámetro *playout delay* no contempla el retardo desde que se capturan los datos y se generan las LDUs hasta que realmente son transmitidas. En este caso, este retardo es el mismo para todos los receptores puesto que hay un único transmisor del flujo *maestro*.

Para suavizar las fluctuaciones de la gráfica y para mostrar con más claridad la trama de valores, en la figura 6.9b se ha representado la *línea de tendencia de media móvil* (en adelante, media móvil) de cada una de las series de datos representados en la figura 6.9a.



a) Valores reales



b) Media móvil (100 muestras)

Figura 6.9. *Playout delay* del flujo de audio de los receptores (aprendizaje a distancia - LAN - sin algoritmo)

La media móvil representada utiliza conjuntos de 100 valores de los datos de las series, realiza un promediado de los mismos y utiliza el valor obtenido como punto de la línea representada. En ambas figuras, se observa cómo hay un salto en el retardo de reproducción de los receptores al comienzo de la sesión. Esto es debido a que la fuente también está ejecutando la herramienta *vic*, de transmisión de vídeo, que afecta al consumo de recursos de la máquina, lo que se traduce en un retardo extra de procesamiento que, como es lógico, incrementará al retardo de reproducción de los receptores al no enviar inmediatamente los paquetes RTP.

En la figura 6.9b se puede apreciar que las diferencias entre los valores del *playout delay* de los receptores están por debajo de los 100 milisegundos, lo cual nos podría inducir a pensar que los procesos reproductores estarían sincronizados, por estar dentro del límite de asincronía máxima fijado, de ±120 milisegundos. Como se verá más adelante esto no es así, ya que no se cumple la sincronización entre las LDUs que está reproduciendo cada uno en cada instante.

- ***Número de Secuencia de las LDUs del flujo maestro reproducidas en cada receptor***

La mejor forma de verificar el grado de asincronía entre los procesos reproductores del flujo *maestro* de todos los receptores será comprobar si las mismas LDUs se están reproduciendo, en cada receptor, en los mismos instantes de tiempo o no.

En primer lugar, se va a comprobar si el instante de inicio de la reproducción coincide en todos los receptores.

Para ello, en la figura 6.10 se ha representado la primera LDU que reproduce cada receptor (el número de la primera LDU transmitida por la fuente se toma de forma aleatoria, [SCH96]).

Se puede observar que no todos los receptores inician la reproducción en el mismo instante, tal como era de esperar. Existe una diferencia máxima de 220 milisegundos entre el inicio de la reproducción de los receptor PC3 y PC7 (amarillo y verde, respectivamente) y el inicio de la reproducción del receptor PC5 (morado).

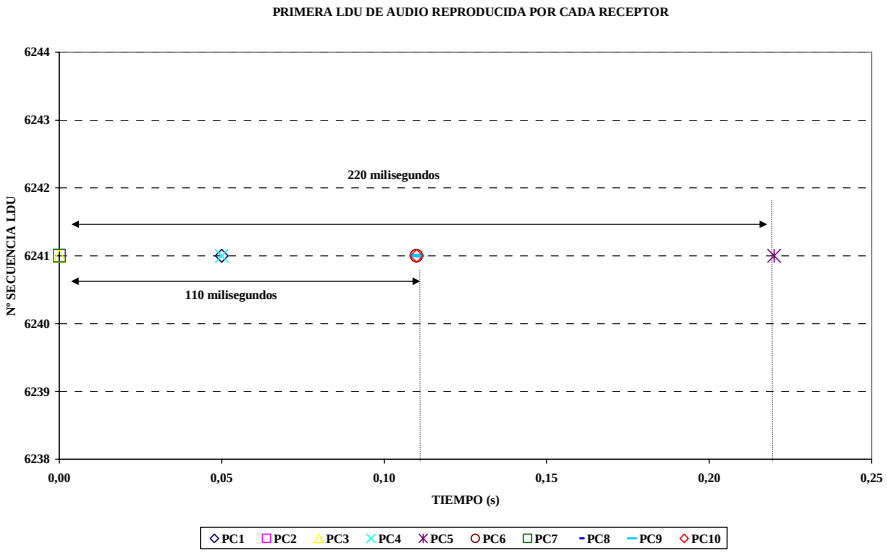


Figura 6. 10. Primera LDU del flujo *maestro* reproducida (aprendizaje a distancia - LAN - sin algoritmo)

A continuación, en la figura 6.11, se muestra una gráfica donde se representa el número de secuencia de las LDUs reproducidas por cada receptor en función del tiempo. Sólo se ha presentado el fragmento correspondiente a los últimos 2 segundos de la sesión de medida, para que se pueda apreciar mejor la información presentada en la gráfica.

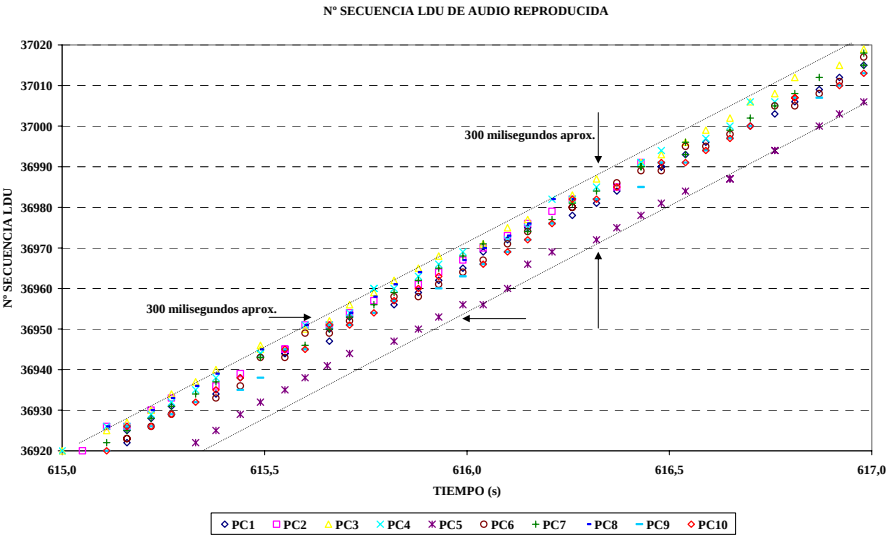


Figura 6.11. Número de secuencia de las LDUs de audio reproducidas (aprendizaje a distancia - LAN - sin algoritmo)

Lógicamente, para que existiera una total sincronización entre los procesos de reproducción del flujo de audio (*maestro*) de todos los receptores, sería necesario que todos reprodujeran la misma LDU (es decir, el mismo número de secuencia de LDU) en el mismo instante de tiempo, con lo que en la gráfica deberían coincidir todos los puntos y aparecer una sola línea. En esta gráfica se aprecia que esto no es así, ya que muestra el desfase en la reproducción del flujo de audio de los diferentes receptores en un mismo instante de tiempo. Teniendo en cuenta que cada LDU de audio contiene muestras correspondientes a 20 milisegundos, en la gráfica la separación entre líneas horizontales equivale a 200 milisegundos. Si nos fijamos en un instante de tiempo y trazamos una línea vertical, podemos observar que el desfase entre los procesos de reproducción de los receptores que aparecen en los extremos (receptor PC3, en color amarillo, y receptor PC5, en color morado) es de 15 LDUs, aproximadamente, que se corresponden con unos 300 milisegundos de asincronía entre ambos receptores. Por otro lado, si nos fijamos, por ejemplo, en la LDU número 36.950, vemos que el receptor PC5 la reproduce con un retraso de unos 300 milisegundos, aproximadamente, con respecto al resto.

De esta manera, por tanto, se comprueba que, en ausencia de algoritmo de sincronización, existe una asincronía en los procesos de reproducción que sobrepasa el límite de ±120 milisegundos.

B) CON EL ALGORITMO DE SINCRONIZACIÓN DE GRUPO

A continuación, se muestra la evaluación del algoritmo para una transmisión de audio y vídeo desde el servidor de flujos multimedia hacia 10 receptores, con una duración de 10 minutos, aproximadamente. Como en el caso anterior, y ya que la utilización de un tiempo global común en todos los equipos es un requisito del algoritmo propuesto, los relojes de los mismos están sincronizados globalmente mediante NTP. Además de contribuir al funcionamiento del algoritmo, este aspecto nos permitirá tener una referencia de tiempos común a la hora de tomar medidas y poder comparar los valores de los parámetros obtenidos durante la sesión.

A diferencia de la configuración anteriormente evaluada, en este caso todos tienen la opción de sincronización mediante el algoritmo propuesto, de la aplicación *rat*, activada (figura 6.12) y, desde la *interfaz gráfica de usuario* de la misma aplicación, en el servidor de flujos multimedia, se han configurado los parámetros necesarios para la correcta ejecución del algoritmo (figura 6.13), con la finalidad de conseguir mantener los valores de asincronía dentro de los límites que aparecen en la tabla 6.4.

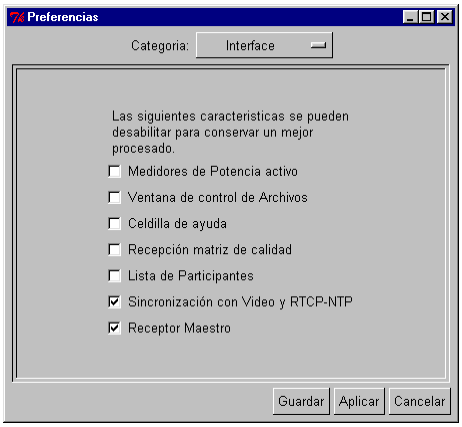


Figura 6.12. Configuración del receptor *maestro*

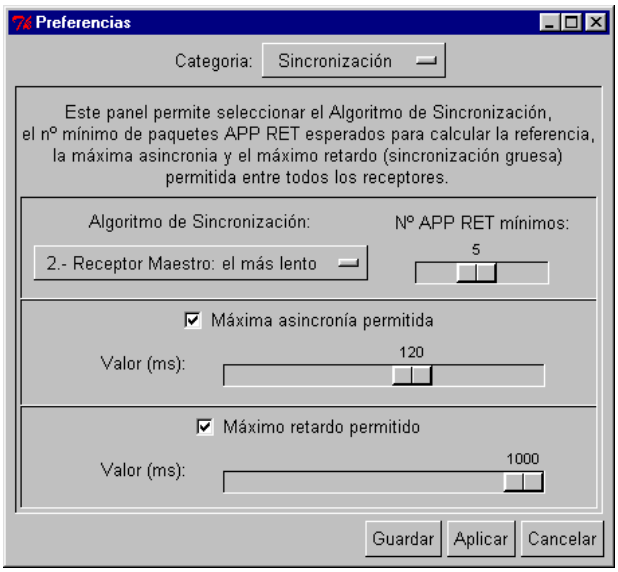


Figura 6.13. Configuración de la fuente de audio

Parámetro	**Valor**
Asincronía Máxima permitida entre los reproductores del flujo de audio. (*Sincronización de Grupo, distribuida*)	±120 ms
Máximo retardo de reproducción (*playout delay*) permitido en los reproductores de audio	1 s
Asincronía Máxima permitida entre los procesos reproductores de los flujos de audio y vídeo. (*Sincronización inter-flujo, local*)	±80 ms

Tabla 6.4. Valores límite a cumplir
(aprendizaje a distancia - LAN - sin algoritmo)

Además, se ha seleccionado uno de los receptores como receptor *maestro*, el cual será el tomado como referencia para la sincronización de grupo entre receptores. Concretamente, se ha elegido al PC10 como receptor *maestro*, apareciendo en las gráficas referenciado como *PC_ MAESTRO*. Será el único que tenga la opción de *Receptor Maestro* activada en la aplicación *rat* (figura 6.12).

A continuación, se presentan una serie de figuras conteniendo gráficas obtenidas durante la sesión que demuestran el buen comportamiento del algoritmo de sincronización de grupo propuesto, durante la misma.

En primer lugar, se van a presentar gráficas de la sincronización de grupo distribuida y la sincronización del instante inicial de consumo, con respecto al flujo *maestro*. En segundo lugar se mostrarán las gráficas con los resultados de la evaluación de la sincronización inter-flujo entre los flujos *maestro* y *esclavo*, localmente en cada receptor.

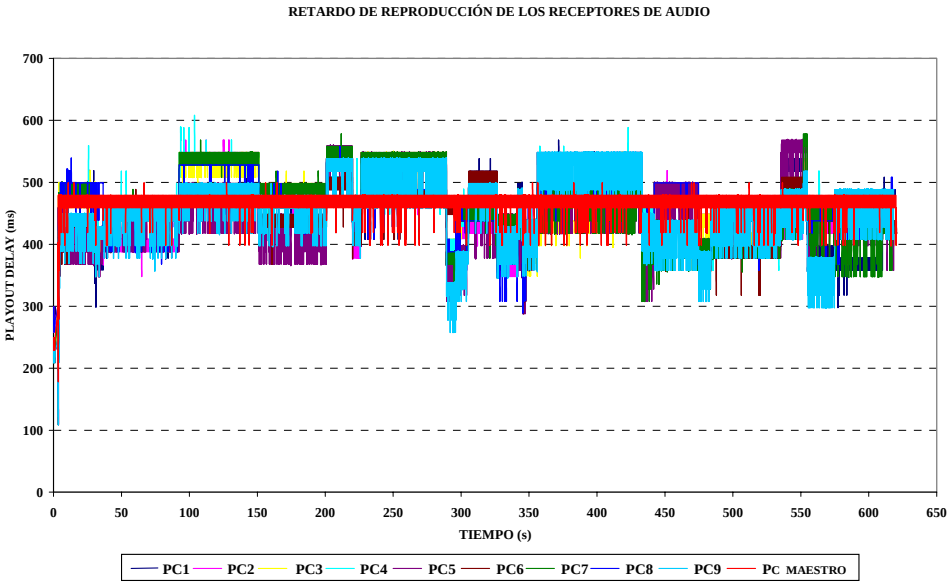
B1. Sincronización de Grupo (distribuida)

- ***Retardos de reproducción (playout delay) de los procesos reproductores del flujo maestro de todos los receptores***

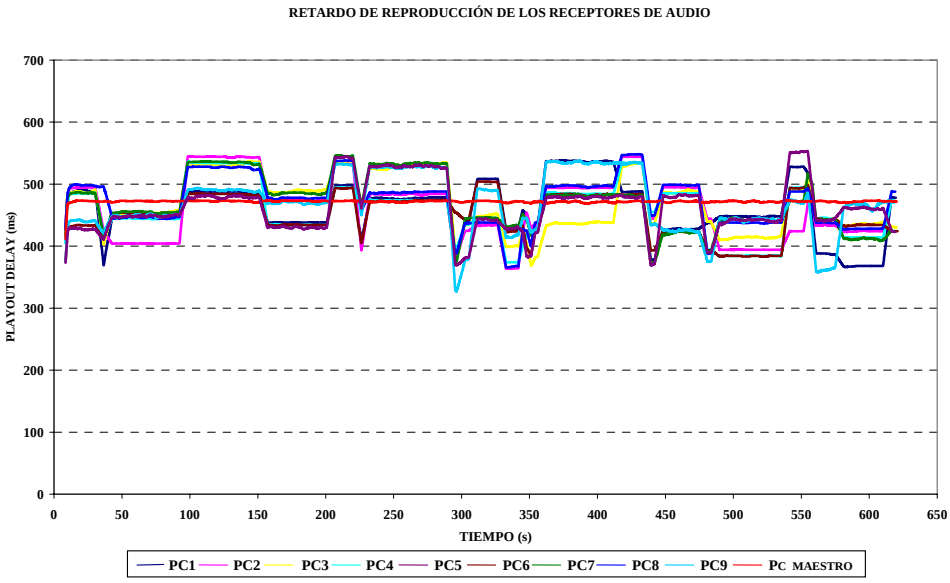
En la figura 6.14 se representa el retardo de reproducción (*playout delay*) del flujo de audio experimentado por los 10 receptores.

Como en el caso anterior, para suavizar las fluctuaciones de la gráfica y mostrar con más claridad la trama de valores, en la figura 6.14b se ha representado la *media móvil* de la serie de datos representados en la figura 6.14a, a partir de conjuntos de 100 muestras. En ambas figuras se representa el tiempo completo de la sesión (10 minutos, aproximadamente).

En la figura 6.14a, se puede observar el salto en el retardo de reproducción de los receptores al comienzo de la sesión debido al consumo inicial de recursos de la fuente debido al inicio de las aplicaciones. Posteriormente, los retados de reproducción (*playout delay*) de los flujos de audio de los receptores *esclavos* se van adaptando en determinados puntos al retardo de reproducción del receptor *maestro* (*PC_MAESTRO*, en rojo), tomado como referencia, cuyos valores de *playout delay* están en torno a 480 milisegundos.



a) Valores reales



b) Media móvil (conjuntos de 100 muestras)

Figura 6.14. *Playout delay* del flujo de audio de los receptores (aprendizaje a distancia - LAN - con algoritmo)

Para ver esto con mayor claridad, se muestra la figura 6.15, donde se ha representado la misma gráfica pero sólo con un receptor *esclavo* (PC1) y, además, se han incluido los ajustes que ha sufrido el proceso reproductor de dicho receptor, debido a las operaciones de '*saltos*' y '*pausas*' efectuadas por las acciones de resincronización del algoritmo. Un ajuste positivo implica un aumento del retardo de reproducción ('*pausa*'), mientras que un ajuste negativo implica una disminución del retardo de reproducción ('*salto*'), tal y como se puede apreciar en la figura.

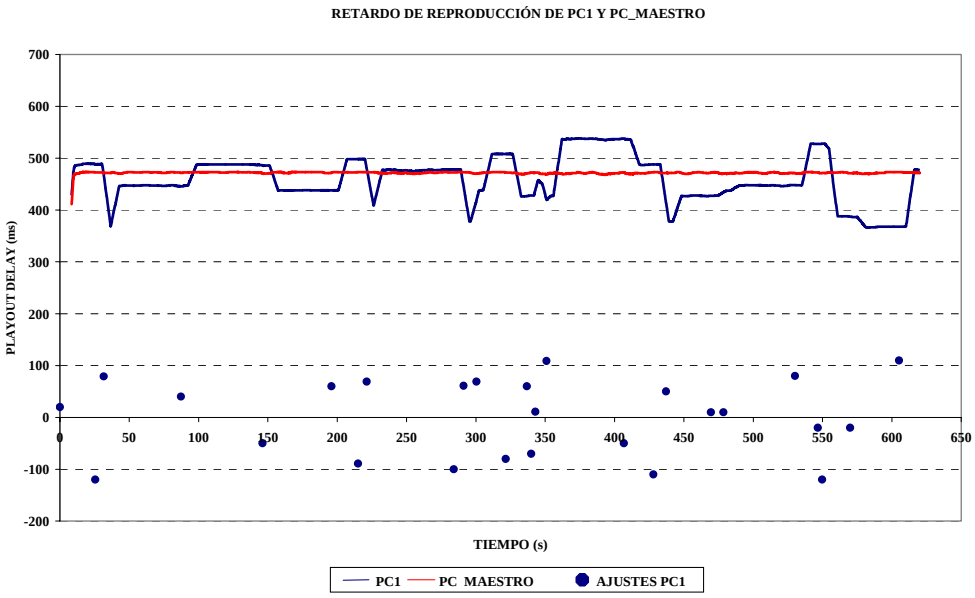


Figura 6.15. *Playout delay* del flujo *maestro* del receptor *maestro* y de un único receptor *esclavo*. Ajustes en el receptor *esclavo* (aprendizaje a distancia - LAN - con algoritmo)

El proceso reproductor del flujo *maestro* (audio) en el receptor *maestro* (PC_MAESTRO) no sufre ajustes, ni debidos a la sincronización de grupo ni a la sincronización inter-flujo, por lo que su estado de reproducción variará menos, tal como se comprueba en ambas figuras.

A continuación, se muestran los valores de los parámetros más representativos obtenidos de todos los receptores durante la sesión, que son los siguientes:

- *Máxima asincronía positiva y negativa detectadas*: valores máximos, tanto negativos como positivos, de la diferencia detectada entre el retardo de reproducción (*playout delay*) del flujo de audio (*maestro*) del receptor tomado como *maestro* y el del receptor *esclavo*. Un valor de asincronía

positivo indica que el proceso reproductor del receptor *esclavo* se encuentra adelantado con respecto al proceso del receptor *maestro*. Por el contrario, un valor de asincronía negativo indica que el proceso reproductor del receptor *esclavo* se encuentra retrasado con respecto al proceso del receptor *maestro*.

- *Minima asincronía detectada*: valor absoluto mínimo de la diferencia detectada entre el retardo de reproducción del flujo de audio (*maestro*) del receptor tomado como *maestro* y el del receptor *esclavo*.
- *Valor medio y desviación estándar de la asincronía detectada*. Valor medio y desviación estándar de la asincronía detectada entre los reproductores del flujo de audio (*maestro*). El valor medio será un valor orientativo ya que existen valores positivos y negativos que, al calcular el valor medio, se anulan unos a otros.
- *Valor cuadrático medio de la asincronía detectada*. Representa la media del cuadrado de los valores de asincronía detectados entre los reproductores del flujo de audio (*maestro*). Se trata de un parámetro mucho más útil que los anteriores ya que tiene en cuenta los valores absolutos de las diferencias.
- *Número de mensajes enviados por el servidor*. Representa el número de mensajes de 'acción' (APP ACT) enviados por el servidor.
- *Número de mensajes de sincronización 'gruesa' adicionales enviados por el servidor*. Es el número de paquetes APP TIN que el servidor envía para garantizar que no se llegue a situaciones de *overflow* de los *buffers* de recepción, incluyendo el paquete APP TIN inicial, cuyo objetivo es el de garantizar que todos los receptores inicien la reproducción en el mismo instante de tiempo.
- *Número de mensajes recibidos del servidor*. Número de mensajes de 'acción'(APP ACT) recibidos del servidor. Debido a las pérdidas es posible que no coincida con el valor de mensajes del mismo tipo enviados por el servidor.
- *Número de mensajes Feedback enviados por el receptor*. Representa el número de paquetes con información de realimentación (*feedback*) que el receptor ha enviado al servidor (mensajes RR Extendidos).
- *Asincronía máxima permitida audio-audio con respecto al receptor maestro*. Valor configurado desde la *interfaz gráfica de usuario* del servidor que indica el valor máximo de asincronía permitida entre los procesos reproductores del flujo *maestro* de los diferentes receptores. Si el servidor detecta que un proceso reproductor de algún receptor *esclavo* se adelanta o se retrasa con respecto al del receptor *maestro*, superando dicho valor, enviará un paquete de '*acción*' (APP ACT) para que los procesos de los receptores corrijan su estado de reproducción.
- *Número asincronías audio-audio detectadas*: Número de veces que se ha superado el valor anterior al calcular la asincronía existente entre los

procesos de reproducción del flujo *maestro* (de audio) en cada receptor con respecto al proceso de reproducción de dicho flujo en el receptor *maestro*.

- *'Saltos' realizados por el flujo de audio (maestro)*: Número total de LDUs del flujo *maestro* que no se han consumido en los receptores, debido al proceso de sincronización de grupo.
- *'Pausas' realizadas por el flujo de audio (maestro)*: Número total de LDUs del flujo *maestro* cuya reproducción en cada receptor es una repetición de una LDU ya consumida, debido al proceso de sincronización de grupo.
- *Número de paquetes RTP enviados.* Número de paquetes RTP conteniendo muestras del flujo *maestro* enviados por el servidor durante la sesión.
- *Duración muestra (milisegundos)*. Intervalo de tiempo cuyas muestras se incluyen en cada paquete RTP. En nuestro caso en cada paquete RTP se codifica el flujo de audio correspondiente a 20 milisegundos.

Los resultados obtenidos en la sesión evaluada se muestran en la tabla 6.5.

En primer lugar se presentan los valores máximos, tanto positivos como negativos de las asincronías detectadas por cada receptor. Puede observarse que en los receptores PC2, PC3, PC5, PC7 y PC9 se ha superado el valor máximo de 120 milisegundos por lo que se habrán producido asincronías que deberán haber sido corregidas por nuestro algoritmo.

En la tabla también puede apreciarse la poca cantidad de paquetes de control que envía el servidor durante la sesión. El servidor envía 1 sólo paquete APP TIN para indicar el instante de inicio de la reproducción (*Instante Inicial de Consumo*) y, posteriormente, durante la sesión, envía 26 paquetes APP ACT de acción indicando acciones de resincronización en los receptores. En este caso, no ha sido necesario que el servidor enviara más paquetes de sincronización gruesa (APP TIN) para evitar situaciones de *overflow* en los *buffers* de los receptores. Se recuerda que, durante la sesión, se enviaría un paquete de este tipo cada vez que el servidor detectara que el retardo de reproducción (*playout delay*) del receptor *maestro* está llegando a un umbral (de 1 segundo, que es la cantidad configurada desde la *interfaz gráfica de usuario* del servidor en la prueba realizada). Puede comprobarse que esto no sucede durante la prueba en la figura 6.14 anterior, ya que los retardos de reproducción de los receptores no llegan a sobrepasar 1 segundo.

En esta prueba, el servidor ha detectado en 26 ocasiones que, como mínimo, un proceso reproductor de algún receptor estaba retrasado o adelantado más de 120 milisegundos con respecto al proceso reproductor del receptor *maestro*. Es por ello que, el servidor, durante la sesión, envía 26 paquetes de '*acción*' APP ACT, frente a los 31170 paquetes RTP de datos del flujo de audio (que contienen muestras de audio codificadas correspondientes a un fragmento de audio de 20 milisegundos de duración).

ESTADÍSTICAS RECEPTOR	**PC1**	**PC2**	**PC3**	**PC4**	**PC5**	**PC6**	**PC7**	**PC8**	**PC9**	**PC10** (MAESTRO)
Máx. asincronía positiva detectada (ms)	110	140	90	110	111	110	90	120	110	-
Máx. asincronía negativa detectada (ms)	-120	-140	-140	-100	-160	-90	-160	-119	-200	-
Mín. asincronía detectada (ms)	10	9	0	9	0	0	0	0	0	-
Valor medio asincronía detectada (ms)	0,35	1,12	-5,42	2	0	0,35	-1,58	1,12	3,04	-
Desviación estándar de asincronías detectadas (ms)	74,24	74,82	64,47	60	73	54,98	67,06	68,65	77,58	-
Valor cuadrático medio de la asincronía detectada (ms^2)	5300	5383	4026	3512	5149	2906	4327	4533	5797	-
Nº mensajes de '*acción*' (APP TIN) enviados por servidor	1									
Nº mensajes de '*acción*' (APP ACT) enviados por servidor	26									
Nº mensajes recibidos del servidor (APP ACT)	26	26	26	26	26	26	26	26	26	26
Nº de mensajes *Feedback* enviados por el receptor (RR Ext.)	227	221	233	224	225	225	224	220	230	226
Asincronía máx. permitida audio-audio con respecto al receptor *maestro*	±120 ms									
Nº asincronías audio-audio	0	2	1	0	1	0	2	1	2	-
'*Saltos*' realizados por el flujo de audio (*maestro*) debido a la sincr. de grupo	41	42	38	33	38	25	40	37	38	-
'*Pausas*' realizadas por el flujo de audio (*maestro*) debido a la sincr. de grupo	40	40	31	34	40	30	39	33	43	-
Nº de paquetes RTP enviados	31170									
Duración muestra flujo de audio (ms)	20									

Tabla 6.5. Datos obtenidos de todos los receptores (aprendizaje a distancia - LAN - con algoritmo)

Los mensajes de control enviados por el transmisor (1 APP TIN y 26 APP ACT, es decir, 27 mensajes de control RTCP) apenas suponen un 0,087 % del total de los paquetes transmitidos (de datos y de control). Por otro lado, los mensajes de control enviados por los receptores (en total 2255 mensajes RTCP RR extendidos) apenas suponen un 6,74 % del total de los paquetes transmitidos (de datos y de control) por las aplicaciones de audio durante la sesión y un 7,23 % con respecto al número total de paquetes de datos enviados (31170 paquetes RTP). En la tabla 6.5 se puede apreciar que los receptores que han enviado un mayor y un menor número de mensajes RR extendidos han sido PC3 (con 233 mensajes) y PC8 (con 220 mensajes), respectivamente.

Hay que tener en cuenta que nuestro algoritmo no ha supuesto ningún aumento del número de paquetes de control RTCP ya que el número de estos paquetes que envía cada receptor es el mismo que enviaría aunque no se ejecutara el algoritmo. Esto es debido a que para enviar la información de realimentación, necesaria para el correcto funcionamiento del mismo, se utilizan los paquetes RR que envía RTCP, pero añadiéndoles una extensión con datos útiles para nuestro algoritmo (número de secuencia de LDU y tiempo NTP de su reproducción) aumentando muy poco la carga introducida por RTCP en la red, que, por otro lado, es mínima.

Tomando como asincronía máxima, entre flujos de audio, ±120 milisegundos, se puede comprobar cómo los receptores, en general, sufren de 0 a 2 situaciones de asincronía durante los 10 minutos que dura la sesión (situaciones en las que se detecta que se ha superado dicha asincronía máxima), que son cantidades mínimas. En concreto, de los nueve receptores (se excluye el receptor *maestro*), 3 receptores no han detectado ninguna asincronía, 3 receptores han detectado una única situación de asincronía y 3 receptores han detectado dos situaciones de asincronía. Gracias a las acciones correctoras del algoritmo se ha recuperado la sincronización rápidamente y se ha mantenido, a los procesos reproductores, dentro de los límites de asincronía configurados (±120 milisegundos).

La suma del número total de asincronías detectadas por los receptores es 9, valor distinto a las 26 ocasiones en que el servidor detecta asincronía en algún receptor. Esto es debido a que la contabilización de asincronías en cada receptor se hace en base a los ajustes realizados en cada uno de los receptores en el momento de recibir el mensaje APP ACT enviado por el servidor, ya que es posible que en ese instante el ajuste no supere el valor máximo de ±120 milisegundos y, por tanto, no se contabilizará como asincronía detectada por el receptor.

Un valor significativo para la evaluación de la sincronización de grupo obtenida es el del parámetro *Valor cuadrático medio de la asincronía detectada*, que, como se puede observar, no supera el valor 14400 milisegundos^2 (que corresponde al cuadrado del valor máximo de asincronía permitido, que es de ±120

milisegundos). Como puede apreciarse en la tabla, en ningún receptor dicho valor medio supera el valor de 14400 milisegundos[2] comentado anteriormente, consiguiéndose valores adecuados de sincronización de grupo (distribuida) para flujos de audio poco acoplados en modo diálogo, tal como establecen los valores de referencia utilizados.

Para conseguir mantener el grado de sincronización de grupo dentro de dicho límite, puede observarse en la tabla que, en los receptores, el algoritmo de sincronización ha provocado acciones de resincronización que se traducen en *'saltos'* y *'pausas'* en el proceso de reproducción del flujo *maestro*. También aparecen en la tabla la cantidad de *'saltos'* y *'pausas'* que sufren los procesos reproductores del flujo *maestro*. El receptor que más *'saltos'* ha experimentado en el proceso reproductor del flujo de audio ha sido el receptor PC2, que habrá dejado de consumir 42 LDUs, mientras que el receptor cuyo proceso reproductor ha sufrido menos *'saltos'* ha sido PC6, que habrá dejado de reproducir 25 LDUs. En cuanto a las *'pausas'*, el receptor que más *'pausas'* ha experimentado en el proceso reproductor del flujo de audio ha sido el receptor PC9, repitiendo la reproducción de una LDU anterior en 43 ocasiones, mientras que el receptor cuyo proceso reproductor ha sufrido menos *'pausas'* ha sido PC6, que habrá reproducido 30 LDUs duplicadas. Se puede deducir de estos datos que el proceso reproductor del receptor PC6 era el que más se aproximaba al punto de reproducción del receptor *maestro* durante la sesión, mientras que los procesos reproductores que más se alejaban eran los de PC1 y PC2.

En la figura 6.16 aparece la distribución de los ajustes sufridos por los procesos reproductores del flujo *maestro* de los receptores, al recibir un paquete de '*acción*' (APP ACT) enviado por la fuente al detectar asincronía en alguno de los receptores. Estos valores reflejan la diferencia, en tiempo, del estado de reproducción de los receptores con respecto al proceso reproductor del receptor *maestro*. Un ajuste positivo indica que el proceso reproductor del receptor se encuentra adelantado con respecto al proceso del receptor *maestro* y debe realizar una pausa igual, en tiempo, a la cantidad de ajuste.

Por el contrario, un ajuste negativo indica que el proceso reproductor del receptor se encuentra retrasado con respecto al proceso del receptor *maestro* y debe realizar un salto en su reproducción igual, en tiempo, a la cantidad de ajuste. Por tanto, los ajustes reflejan los errores de sincronización entre los procesos reproductores de los receptores y el proceso reproductor del receptor *maestro*. Dicho error de sincronización debería mantenerse por debajo del valor comentado anteriormente de ±120 milisegundos (marcado en rojo). Se observa en la figura que este valor es superado en muy pocas ocasiones durante la sesión.

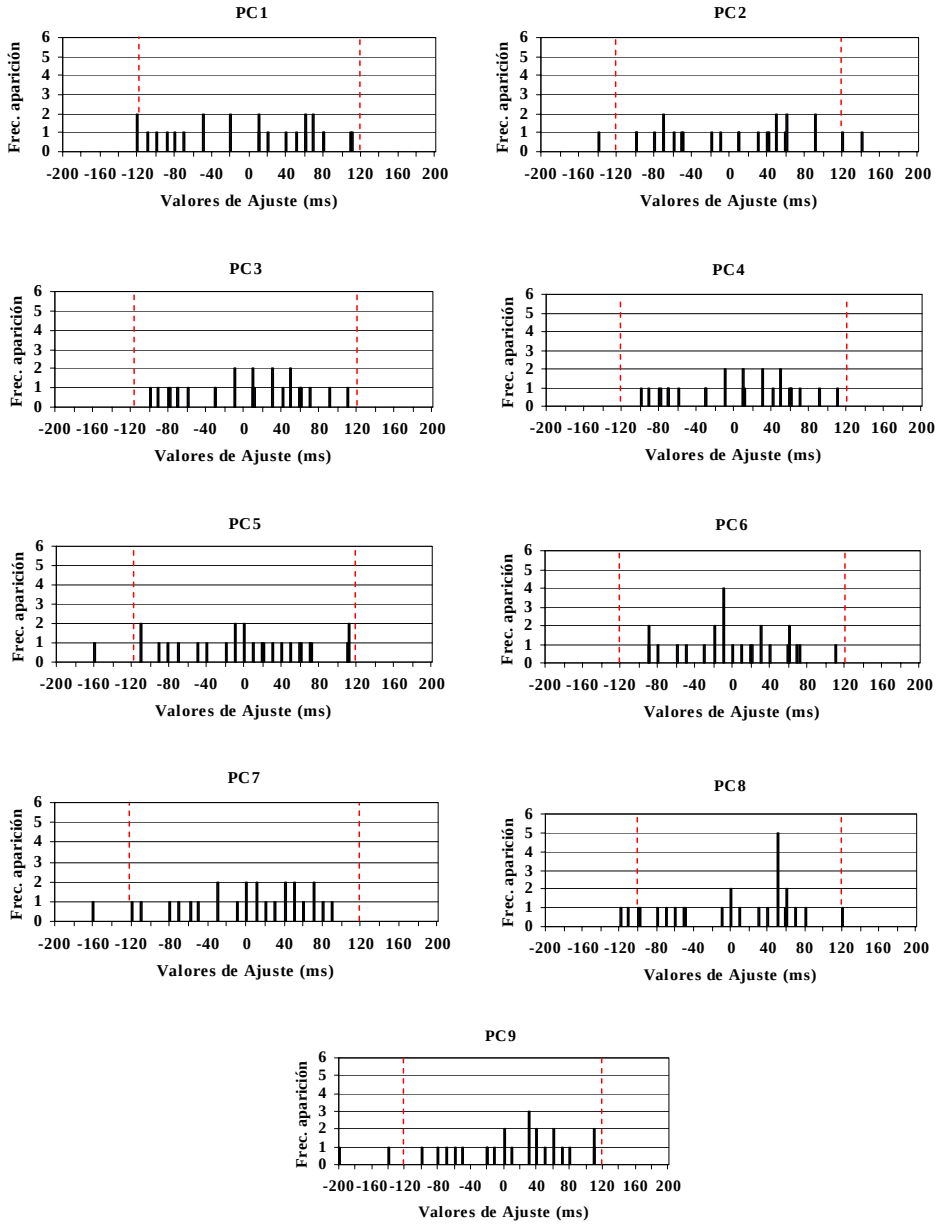


Figura 6.16. Distribución de los valores de ajuste del proceso de reproducción del flujo *maestro* de cada receptor (aprendizaje a distancia - LAN - con algoritmo)

En la figura 6.17 se muestran las gráficas de cada receptor con el valor cuadrático de dichos ajustes, a la largo de los 10 minutos de la sesión. Se ha resaltado, mediante una línea roja, el valor de 14400 milisegundos^2, máximo permitido para una sincronización de calidad entre flujos de audio poco acoplados.

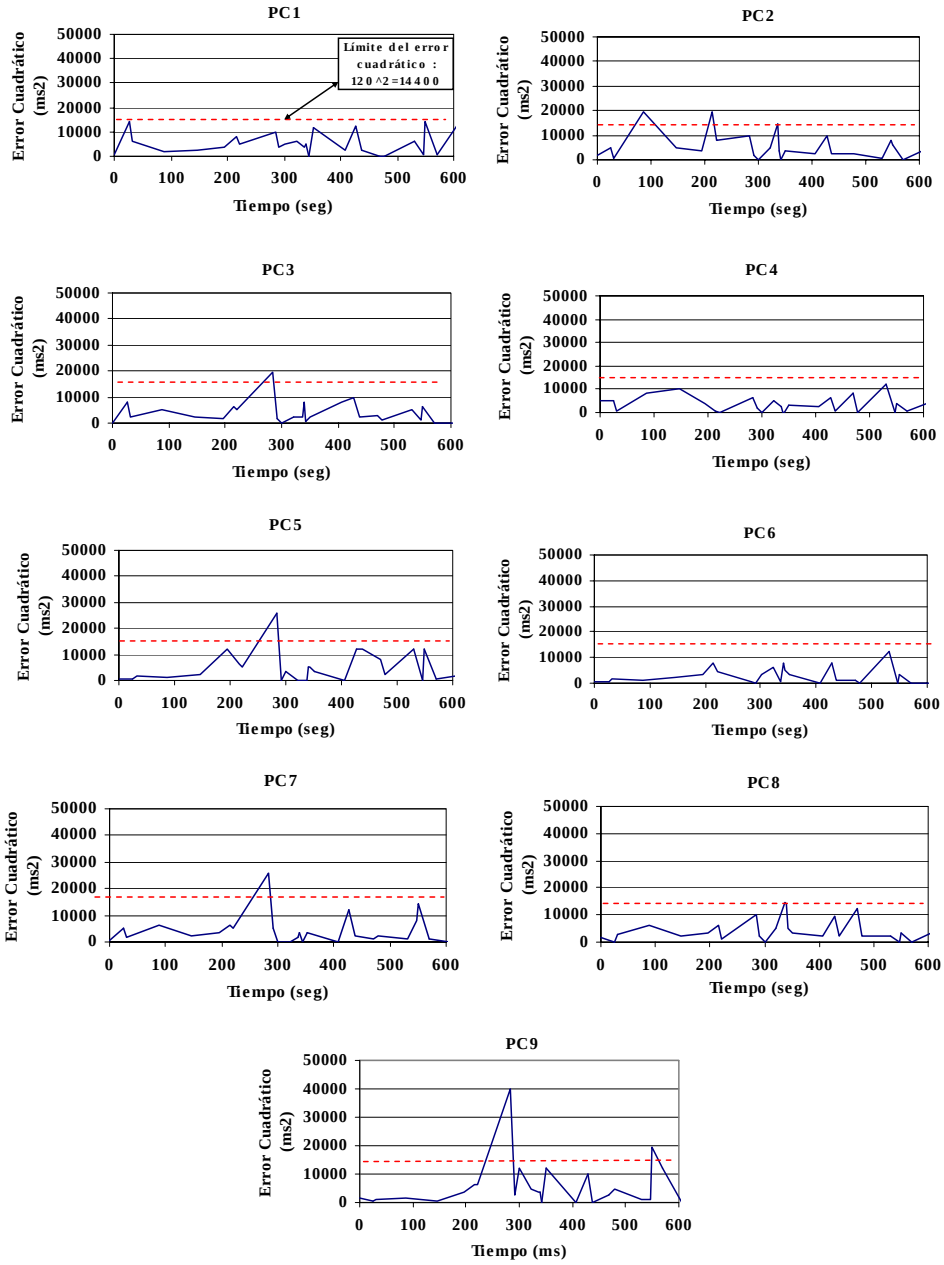


Figura 6.17. Valor Cuadrático del Error de Sincronización (aprendizaje a distancia - LAN - con algoritmo)

Se puede apreciar que en algunos receptores este valor es sobrepasado en ocasiones muy concretas y que rápidamente la situación de asincronía es detectada y corregida por el algoritmo. Sin embargo, la mayor parte del tiempo de la sesión los valores están por debajo de dicho valor. Cabe destacar que sólo se representan los errores de cada receptor, almacenados en los instantes en que éstos reciben un mensaje de '*acción*' que el servidor envía cuando detecta asincronías por encima de 120 milisegundos. Si no envía mensajes APP ACT es porque no se supera dicho valor y, por consiguiente, se supone que el error cuadrático de todos los receptores está por debajo de 14400 milisegundos^2.

En la tabla 6.6 aparecen los valores más representativos comentados, sin tener en cuenta los datos correspondientes al receptor *maestro*.

ESTADÍSTICAS RECEPTOR	**VALORES**
Máxima asincronía positiva detectada (ms)	140 (PC2)
Máxima asincronía negativa detectada (ms)	-200 (PC9)
Mínima asincronía detectada (ms)	0
Máximo valor cuadrático medio de la Asincronía detectada (ms^2)	5797 (PC9)
Media del valor cuadrático medio de todos los receptores	4548,11
Nº mensajes de '*acción*' (APP TIN) enviados por servidor	1
Nº mensajes de '*acción*' (APP ACT) enviados por servidor	26
Nº máximo de mensajes recibidos del servidor (APP ACT)	26
Nº máximo/mínimo mensajes *Feedback* enviados por un receptor (RR Ext.)	233 (PC3)/220(PC8)
Asincronía máxima permitida audio-audio con respecto al receptor *maestro* (± ms)	±120
Nº máximo/mínimo de asincronías audio-audio en un receptor	2 (PC2, PC7, PC9) 0 (PC1, PC4 , PC6)
Nº máximo/mínimo de '*saltos*' realizados por el flujo de audio (*maestro*) de un receptor debido a la sincronización de grupo	42 (PC2) 25 (PC6)
Nº máximo/mínimo de '*pausas*' realizadas por el flujo de audio (*maestro*) de un receptor debido a la sincronización de grupo	43 (PC9) 30 (PC6)
Nº de paquetes RTP enviados	31170
Duración muestra flujo de audio (ms)	20

Tabla 6.6. Valores más representativos
(aprendizaje a distancia - LAN - con algoritmo)

Del análisis de todos estos datos se puede concluir que, en entorno LAN, *el mecanismo de sincronización de grupo de flujos multimedia ha funcionado correctamente a lo largo de la sesión evaluada.*

- ***Número de Secuencia de las LDUs del flujo maestro reproducidas en cada receptor***

Como se ha indicado anteriormente, la mejor forma de corroborar el resultado del algoritmo propuesto será la de verificar el número de secuencia de la LDU del flujo *maestro* que reproduce cada receptor en cada instante.

En primer lugar se ha comprobado que el instante de inicio de la reproducción coincide en todos los receptores ya que todos los receptores comenzaron la reproducción de la primera LDU transmitida (cuyo número escogido aleatoriamente por la fuente, en este caso, fue el 41448) en el mismo instante de tiempo, tal y como se muestra en la figura 6.18.

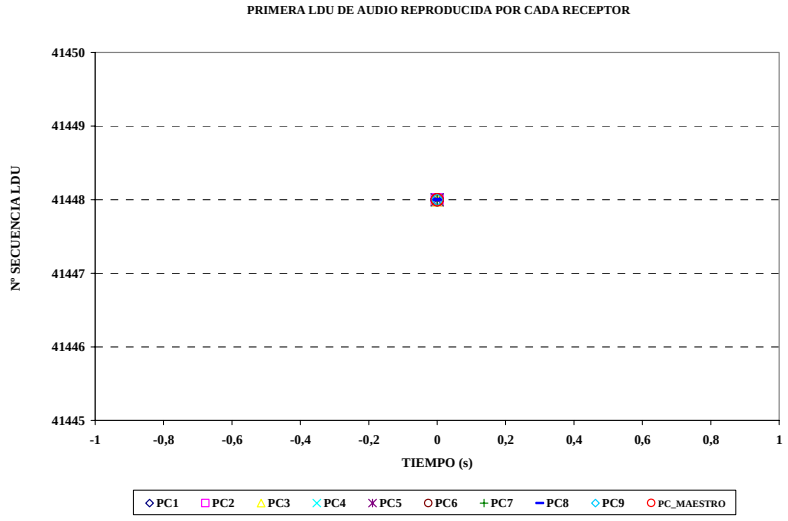


Figura 6.18. Primera LDU del flujo *maestro* reproducida (aprendizaje a distancia - LAN - con algoritmo)

Podemos concluir, pues, que *el mecanismo de sincronización del instante inicial de consumo ha funcionado correctamente*.

A continuación, en la figura 6.19 se representan todos los datos obtenidos durante la sesión de evaluación. Como se ha comentado anteriormente, sólo se han obtenido datos estadísticos en los receptores al recibir paquetes de acción provenientes del servidor. Es por ello que aparecen discontinuidades en la gráfica.

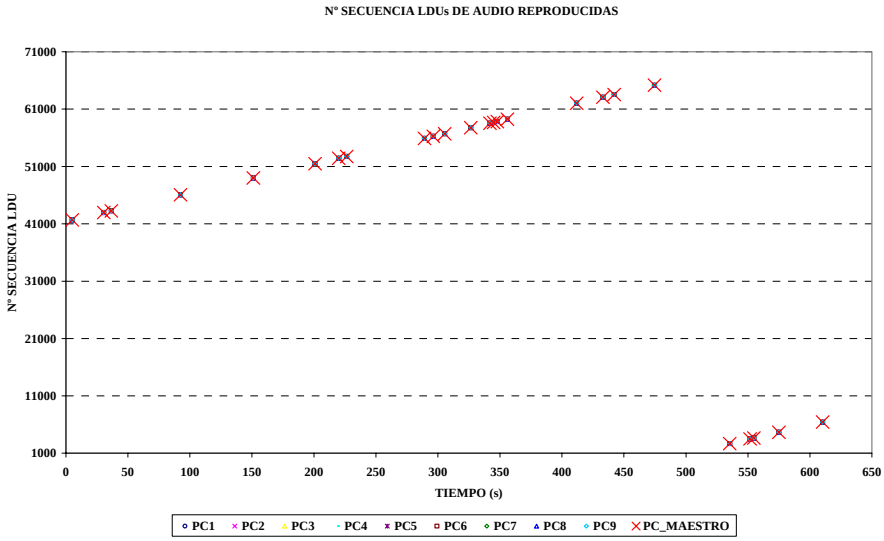


Figura 6.19. Número de secuencia de las LDUs de audio reproducidas (aprendizaje a distancia - LAN - con algoritmo)

Como en la figura anterior no se aprecia bien el grado de sincronización de grupo obtenida, en la figura 6.20 se muestra un intervalo de tiempo reducido (1,5 segundos), donde se representa el número de secuencia de una de las LDUs que cada receptor está reproduciendo en dicho intervalo, y así poder analizar el grado de sincronización existente durante el mismo.

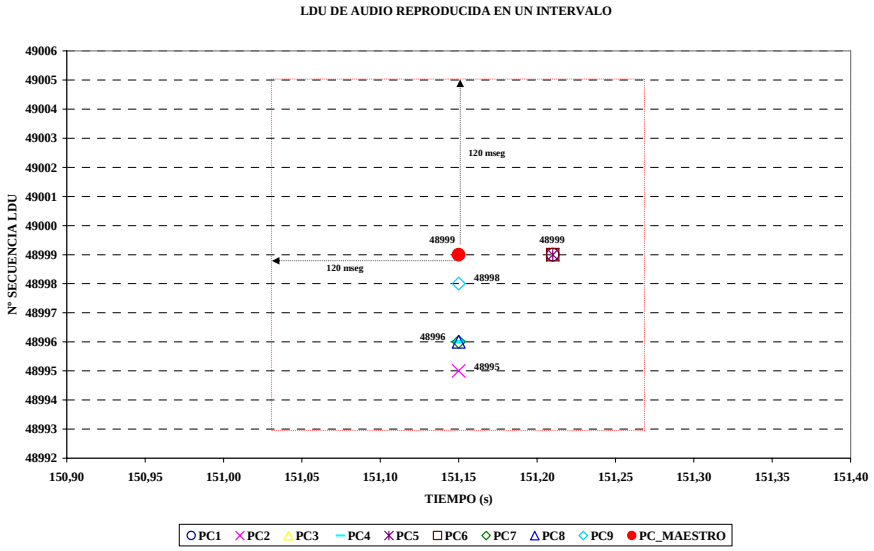


Figura 6.20. Núm. secuencia LDU reproducida en intervalo reducido (aprendizaje a distancia - LAN - con algoritmo)

En la figura 6.20 se puede observar cómo cuando el receptor *maestro* (punto grueso de color rojo) está reproduciendo la LDU número 48999, el receptor PC9 (en color azul claro) está reproduciendo la LDU 48998 (por tanto, está retrasado 20 milisegundos con respecto al receptor *maestro*); los receptores PC3, PC4, PC7 y PC8 están reproduciendo la LDU 48996 (por tanto, están retrasados 60 milisegundos con respecto al receptor *maestro*), el receptor PC2 (en rosa) está reproduciendo la LDU 48995 (por tanto, está retrasado 80 milisegundos con respecto al receptor *maestro*). Por otro lado, también se puede apreciar que los receptores PC1, PC5 y PC6 reproducirán la LDU 48999, pero 60 milisegundos después (por tanto, están atrasados 60 milisegundos con respecto al receptor *maestro*). Como se puede observar, en este caso, las diferencias en el estado de reproducción de los receptores *esclavos*, con respecto al proceso del receptor *maestro*, están dentro de los límites de asincronía máxima permitidos (±120 milisegundos, representados por un cuadrado de color rojo, de 120 milisegundos de lado, centrado en la muestra reproducida por el receptor *maestro*) para flujos de audio poco acoplados en modo diálogo ([STE96]).

En la figura 6.21 se han representado datos de 26 muestras obtenidas durante la sesión, superpuestas, tomando diferencias relativas, de tiempos y de número de LDU reproducida en cada instante (eje *x* y eje *y*, respectivamente). En el eje *y* se muestra la diferencia en el número de LDU reproducida con respecto a la LDU reproducida por el PC_MAESTRO. De esta forma, en el instante 0, se representa la LDU que el PC_MAESTRO está reproduciendo (punto rojo central). El resto de receptores puede que estén reproduciendo la misma LDU (diferencia 0 con respecto al PC_MAESTRO) o pueden estar adelantados (diferencias positivas) o retrasados (diferencias negativas). Se estará sincronizado si los puntos representados están dentro del cuadro rojo que representa una asincronía de ±120 milisegundos.

En la figura hay representados 235 puntos correspondientes a la muestra reproducida por el receptor *maestro* (en rojo, centrada en el punto (0,0)) y cada una de las muestras reproducidas por cada receptor en los 26 instantes muestreados. Se puede observar en la misma que en la reproducción de estas muestras apenas se han producido asincronías, ya que, prácticamente la totalidad de los puntos están dentro del cuadro, y que los puntos están repartidos por dentro del mismo, indicando que los puntos de reproducción de los receptores oscilan alrededor del punto de reproducción del receptor *maestro* en cada instante. Aunque no se aprecia el número de puntos que coinciden (debido a la superposición), en la figura, sólo se han quedado fuera del cuadro rojo 4 puntos, es decir un 1,7% del total de las muestras representadas (que son 26 muestras, frente a las 31170 enviadas en la sesión).

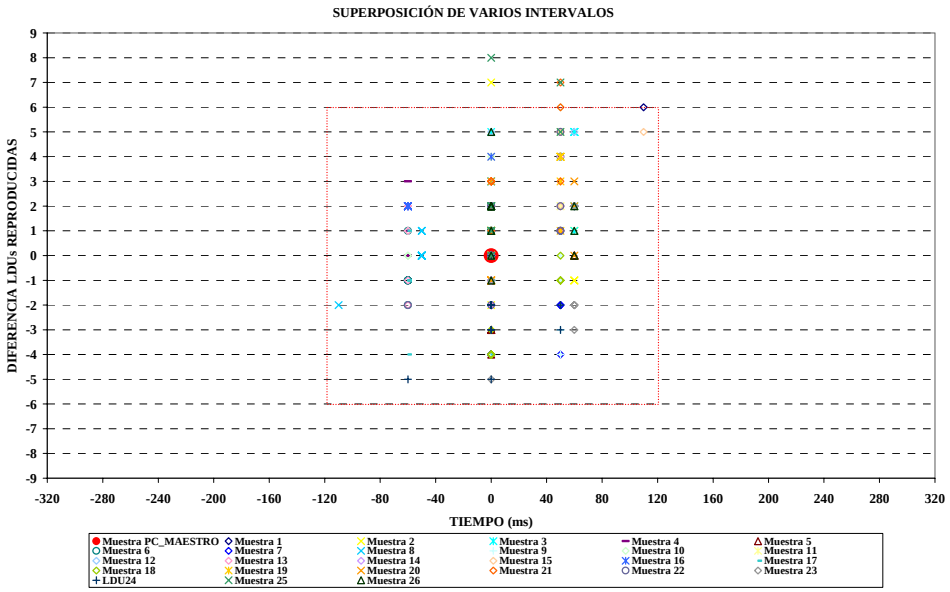


Figura 6.21. Superposición de 26 muestras
(aprendizaje a distancia - LAN - con algoritmo)

B2. Sincronización inter-flujo (local) en cada receptor

Veamos ahora el comportamiento de la sincronización inter-flujo, localmente en cada receptor, es decir, de la asincronía existente entre los procesos de reproducción de los flujos *maestro* (de audio) y *esclavo* (de vídeo) en un mismo receptor.

La asincronía máxima permitida entre los flujos de audio y vídeo que se ha fijado como objetivo respetar es de ±80 milisegundos para una sincronización inter-flujo, dato obtenido de [STE96], donde, según Steinmetz, una sincronización de calidad se consigue si se mantiene el error de sincronización dentro de los límites de ±80 milisegundos entre los flujos de audio y vídeo. Por otro lado, también se indica en el mismo trabajo que los errores de sincronización que sobrepasen los ±160 milisegundos son detectables e inaceptables para los usuarios, mientras que los errores comprendidos en las regiones intermedias (comprendidas entre -160 y -80 y entre +80 y +160 milisegundos) son detectados pero pueden ser aceptados por los usuarios. Por tanto, *nuestro algoritmo funcionará correctamente, en este entorno, si mantiene los valores de asincronía entre los procesos reproductores por debajo de los límites de ±80 milisegundos, corrigiendo las posibles asincronías detectadas que superen dichos límites y, en todo caso, cuando*

se superare en situaciones puntuales, se deberá mantener por debajo del límite superior de ±160 milisegundos.

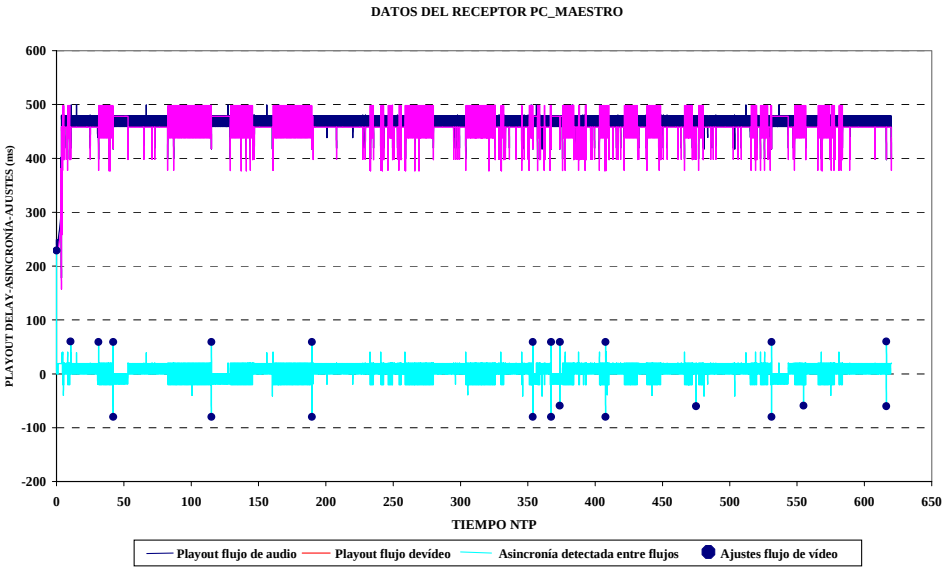
- ***Retardo de reproducción (playout delay) de los procesos reproductores de los flujos maestro y esclavo***

Una manera de ver si los dos flujos se reproducen simultáneamente, será analizando los retardos de reproducción (*playout delay*) de ambos en cada instante. Si existiera una total sincronización, deberían ser prácticamente iguales, ya que muestras de distintos flujos que se generan en el mismo instante deben ser reproducidas en el mismo instante. En la práctica no serían exactamente iguales ya que se debe tener en cuenta que el parámetro *playout delay* no contempla el retardo desde que se capturan los datos y se generan las LDUs hasta que realmente son transmitidas. Dicho retardo será distinto para cada flujo debido a los procesos de captura y codificación. A pesar de ello, las diferencias en cuanto a dicho retardo serán mínimas de un flujo a otro, con respecto al valor real del *playout delay*, y, por tanto podemos utilizar el parámetro *playout delay* para observar si están sincronizados o no los procesos reproductores de los flujos evaluados en cada receptor.

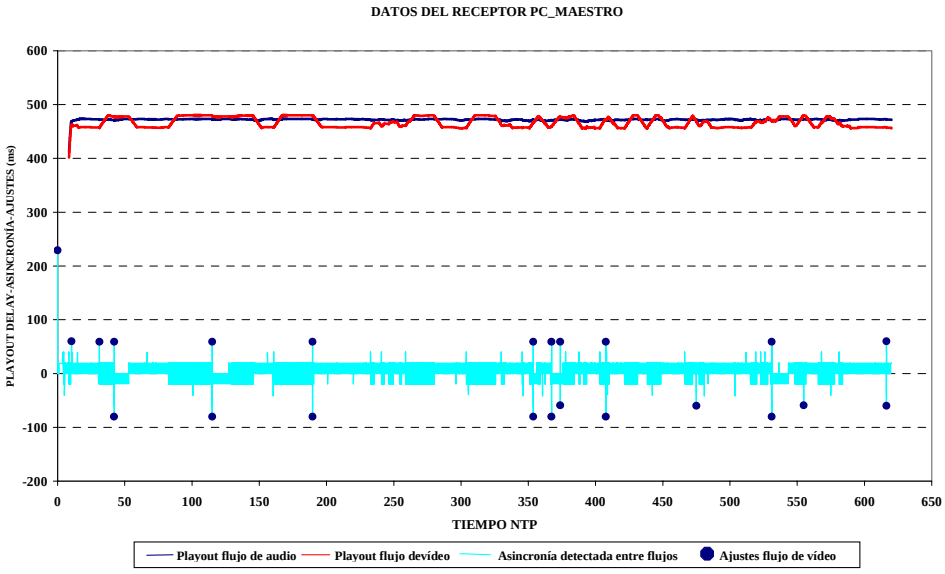
Sin pérdida de generalidad, dado que los comportamientos son muy similares, no se considera necesario mostrar el comportamiento de cada uno de los receptores *esclavos*, por lo que sólo se incluirá el comportamiento de uno de ellos y el del propio receptor *maestro*. En las figuras 6.22 y 6.23 (las gráficas 6.22a y 6.23a representan los datos reales, mientras que las gráficas 6.22b y 6.23b reflejan el resultado de suavizar los datos reales con la media móvil de los datos tomando conjuntos de 100 muestras), se muestran los retardos de reproducción de los flujos *maestro* (audio) y *esclavo* (vídeo) correspondientes a los receptores PC_MAESTRO y PC1, respectivamente. Para poder entender mejor el significado de las gráficas, también se ha incluido en las mismas la asincronía detectada entre los procesos reproductores de ambos flujos y los ajustes que ha sufrido el proceso reproductor del flujo *esclavo* como consecuencia de la aplicación del algoritmo propuesto. En ambas se puede apreciar que en los receptores, al iniciar las aplicaciones, el proceso de reproducción del flujo *esclavo* (vídeo) sufre un ajuste muy elevado, debido, como ya se ha comentado, al aumento del retardo de reproducción del flujo *maestro* debido al consumo de recursos producido por el arranque de los procesos de codificación de las herramientas utilizadas en la fuente.

Durante la sesión los procesos de reproducción del flujo *esclavo* en los receptores PC_MAESTRO y PC1 sufren 23 y 31 ajustes, respectivamente, que, como se puede comprobar, superan todos los ±40 milisegundos. Esto es lógico, ya que en las aplicaciones se ha configurado dicho valor de 40 milisegundos como

umbral de asincronía máxima permitida por el proceso reproductor del flujo de audio antes de enviar un mensaje de sincronización a través de *mbus*.

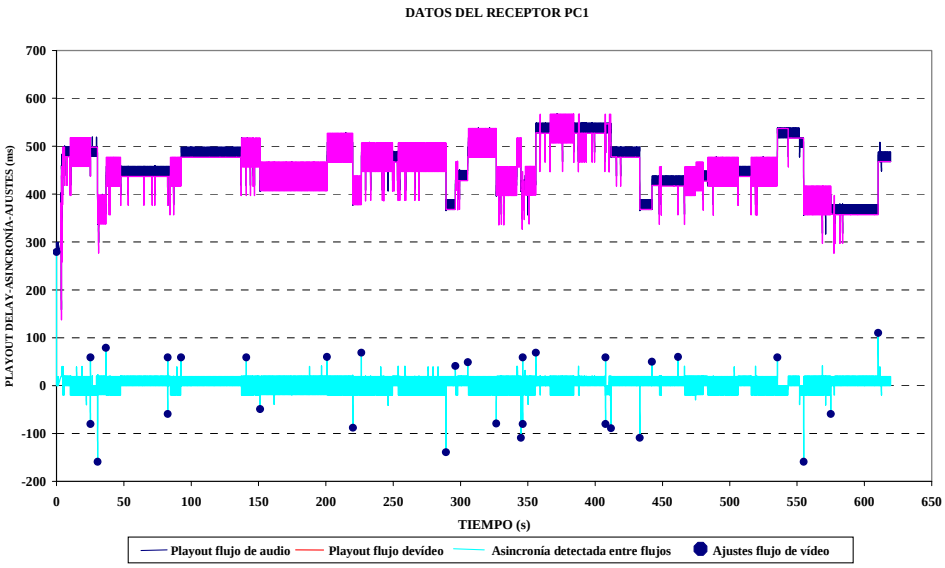


a) Datos reales

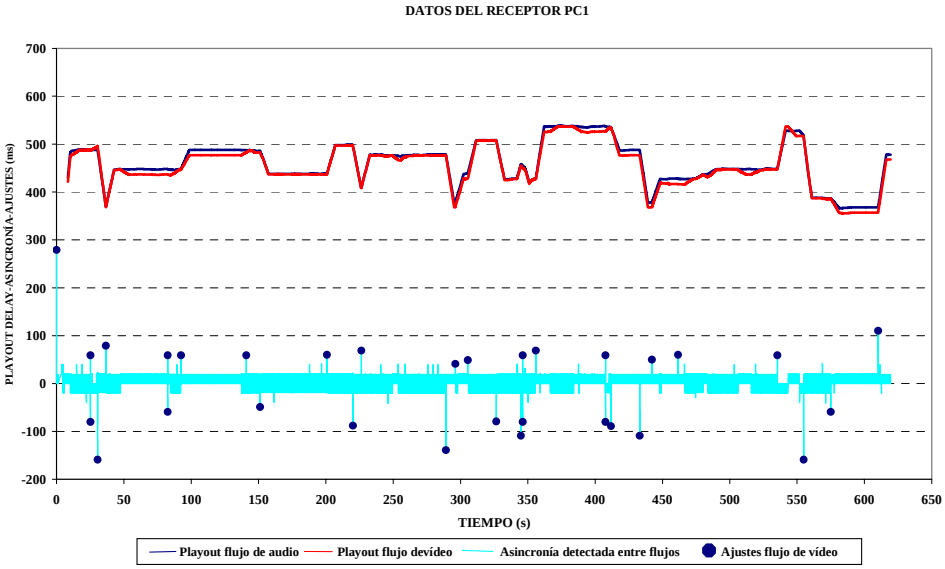


b) Media móvil (conjuntos de 100 muestras)

Figura 6.22. *Playout delay* de los dos flujos reproducidos por PC_MAESTRO, asincronía detectada y ajustes del reproductor del flujo *esclavo* (aprendizaje a distancia - LAN - con algoritmo)



a) Datos reales



b) Media móvil (conjuntos de 100 muestras)

Figura 6.23. *Playout delay* de los dos flujos reproducidos por PC1, asincronía detectada y ajustes del proceso reproductor del flujo *esclavo* (aprendizaje a distancia - LAN - con algoritmo)

Tal y como se puede apreciar, las asincronías detectadas durante la sesión están, durante la mayor parte del tiempo, dentro de los límites de ±80 milisegundos esperados. De ahí que los valores cuadráticos medios obtenidos de la misma sean menores que 6400 milisegundos^2. Como era de esperar y como ya ha sido justificado anteriormente, los ajustes sufridos por el proceso reproductor del receptor *maestro* son inferiores a los del resto de receptores (aunque sólo se haya mostrado el comportamiento del receptor PC1). Como ya se ha comentado, el proceso reproductor del flujo tomado como *maestro* (de audio) en el receptor *maestro* no sufre ajustes ni debidos a la sincronización de grupo ni a la sincronización inter-flujo, por lo que el estado de reproducción de los flujos de audio y vídeo sufrirán menos desviaciones.

A continuación, en la tabla 6.7, se muestran los datos más representativos en cuanto a la sincronización inter-flujo, obtenidos de todos los receptores, que son los siguientes:

- *Máxima asincronía positiva y negativa detectadas*: Máximo valor positivo y negativo de la diferencia detectada entre el retardo de reproducción (*playout delay*) del flujo *maestro* (de audio) y el retardo de reproducción del flujo *esclavo* (de vídeo). Un valor de asincronía positivo indica que el proceso reproductor del flujo esclavo se encuentra adelantado con respecto al proceso reproductor del flujo *maestro*. Por el contrario, un valor de asincronía negativo indica que el proceso reproductor del flujo *esclavo* se encuentra retrasado con respecto al proceso reproductor del flujo *maestro*.
- *Mínima asincronía detectada*: Mínimo valor absoluto de la diferencia detectada entre el retardo de reproducción del flujo *maestro* (audio) y el retardo de reproducción del flujo *esclavo* (vídeo).
- *Valor medio y desviación estándar de la asincronía detectada*. Valor medio y desviación estándar de la asincronía detectada (valores orientativos) entre ambos procesos reproductores.
- *Valor cuadrático medio de la asincronía detectada*. Valor medio del cuadrado de los valores de asincronía detectados entre ambos procesos reproductores.
- *Nº mensajes mbus enviados*. Número total de mensajes *mbus* enviados por el proceso reproductor del flujo *maestro* al proceso reproductor del flujo *esclavo*.
- *Asincronía máxima permitida audio-vídeo por la aplicación*. Valor configurado desde la *interfaz gráfica de usuario* de las aplicaciones que indica el valor máximo de asincronía entre los procesos reproductores de los flujos de audio y vídeo permitido por las mismas en el receptor. Si se supera este valor, el algoritmo pondrá en marcha las acciones oportunas para recuperar un estado de sincronización con unos valores de asincronía menores a dicho valor configurado.

- *Número de asincronías audio-vídeo detectadas*: Número de veces que se ha superado el valor de asincronía máxima fijado en ±80 milisegundos, al calcular la asincronía existente entre los procesos de reproducción de los flujos *maestro* y *esclavo* en cada receptor.
- *'Saltos' realizados por el flujo de vídeo (esclavo)*: Número total de LDUs del flujo *esclavo* que no se han consumido, debido al proceso de sincronización inter-flujo.
- *'Pausas' realizadas por el flujo de vídeo (esclavo)*: Número total de LDUs del flujo *esclavo* cuya reproducción es una repetición de una LDU ya consumida, debido al proceso de sincronización inter-flujo.
- *Duración muestra del flujo esclavo*. Intervalo de tiempo cuyas muestras se incluyen en cada paquete RTP. En nuestro caso se envían 25 tramas por segundo con lo que cada trama tendrá una duración de 40 milisegundos.

Si nos fijamos en los valores máximos de la asincronía (positivos y negativos) detectada en los receptores, vemos que, a excepción del receptor *maestro*, todos han sufrido alguna situación de asincronía ya que se supera el límite de ±80 milisegundos, pero sólo en el caso de PC9 se supera el límite de -160 milisegundos en el valor de máximo valor negativo de la asincronía detectada. Esto último sucedió en dos momentos puntuales de la sesión, tal y como se comprobará más adelante.

También se puede apreciar que el receptor *maestro* es el que menores valores de asincronías detecta. Esto es lógico pues el proceso reproductor del flujo *maestro* no sufre ajustes debidos al algoritmo de sincronización de grupo (distribuida) ya que su estado de reproducción es tomado como referencia por el mismo y no es modificado.

Cuando se detecta que la asincronía entre flujos supera un determinado umbral (en las aplicaciones se ha fijado un umbral de ±40 milisegundos), el proceso reproductor del flujo *maestro* envía un mensaje *mbus* al proceso reproductor del flujo *esclavo* para que corrija o ajuste su estado de reproducción y se recupere la sincronización.

En la tabla aparece el número de mensajes *mbus* enviados. El receptor que más mensajes *mbus* ha enviado fue PC3, con 74 mensajes, mientras que el que menos mensajes envió fue el receptor PC8, con 25 (exceptuando a PC_MAESTRO, con 23).

ESTADÍSTICAS RECEPTOR	**PC1**	**PC2**	**PC3**	**PC4**	**PC5**	**PC6**	**PC7**	**PC8**	**PC9**	**PC10 (MAESTRO)**
Máx. asincronía positiva detectada (ms)	110	120	90	70	111	110	90	120	110	60
Máx. asincronía negativa detectada (ms)	-159	-139	-159	-139	-159	-109	-159	-150	-199	-80
Mín. asincronía detectada (ms)	0	0	0	0	0	0	0	0	0	0
Valor medio de la Asincronía detectada (ms)	4	1	4	6	7	7	5	16	9	4
Desviación estándar de las asincronías detectadas (ms)	18	19	15	16	18	17	18	12	17	17
Valor cuadrático medio de la asincronía detectada (ms^2) [3]	321,4	368,43	244,18	294,73	370,52	346,68	369,69	403,31	388,60	310,08
Nº mensajes *mbus* enviados	31	29	74	26	70	27	52	25	58	23
Asincronía audio-vídeo permitida	Umbral configurada: ±40ms [4]					Máxima permitida: ±80ms				
NºAsincronías Audio-Vídeo	9	9	8	7	9	7	13	8	7	1
'*Saltos*' realizados por el flujo de vídeo (*esclavo*) debido a la sincr. inter-flujo	32	32	24	24	50	26	51	23	44	20
'*Pausas*' realizadas por el flujo de vídeo (*esclavo*) debido a la sincr. inter-flujo	23	18	82	17	51	20	34	19	45	13
Duración Muestra flujo de vídeo (ms)	40									

Tabla 6.7. Datos obtenidos de todos los receptores
(aprendizaje a distancia - LAN - con algoritmo)

[3] Para el cálculo de este valor no se ha tenido en cuenta el ajuste inicial debido al inicio de las aplicaciones en cada receptor.

[4] Se ha seleccionado una asincronía máxima entre los procesos reproductores de audio y de vídeo, permitida desde las aplicaciones, de ±40 milisegundos, para que se inicien las acciones correctoras antes de que se supere el límite máximo permitido (que será de ±80 milisegundos), lo cual se contabilizaría como una situación de asincronía.

En la figura 6.24 se muestra la distribución de los ajustes que ha sufrido el proceso reproductor del flujo *esclavo* (de vídeo) de cada receptor para adaptarse al estado del proceso reproductor del flujo *maestro* (de audio). Puede observarse que los ajustes que superan los límites establecidos de ±80 milisegundos son mínimos en todos los receptores, así como que no existen ajustes por debajo de 40 milisegundos (por el umbral fijado comentado anteriormente). La mayoría de ajustes están en las franjas [-80, -40] y [40,80] milisegundos, como era de esperar.

La superación del límite de ±80 milisegundos en cada receptor ha sido contabilizada como una situación de asincronía. En la tabla anterior, también aparece el número de asincronías detectadas en cada receptor. El receptor en el que más asincronías se detectaron fue PC7 con 13, mientras que, por otra parte, los receptores que menos situaciones de asincronías detectaron fueron PC4, PC6 y PC9, con 7, a excepción del receptor *maestro* (PC_MAESTRO), con 1.

Para corregir dichas situaciones de asincronías, los procesos reproductores del flujo *esclavo* de cada receptor han tenido que modificar su estado mediante acciones de *'saltos'* y *'pausas'*. El proceso reproductor del receptor que más *'saltos'* realizó durante la sesión fue el de PC7 con 51 *'saltos'*, mientras que el del receptor PC8 fue el que menos *'saltos'* sufrió, con 23 (el receptor *maestro* realizó 20). Ello significa que los procesos reproductores de PC7 y de PC8 han dejado de reproducir 51 y 23 tramas, respectivamente. Por otro lado, el proceso reproductor del receptor que más *'pausas'* ha sufrido durante la sesión fue el de PC3 con 82 *'pausas'* (es decir, repitiendo en 82 ocasiones la reproducción de una LDU anterior), mientras que los de los receptores PC7 y PC_MAESTRO fueron los que menos *'pausas'* sufrieron, con 17 y 13, respectivamente (habrán reproducido 13 y 17 tramas duplicadas, respectivamente).

Al igual que para la sincronización de grupo, para el caso de la sincronización inter-flujo el parámetro más representativo será el valor cuadrático del error de sincronización. Si nos centramos en el valor cuadrático medio de las diferencias entre los estados de los procesos reproductores de ambos flujos, el algoritmo funcionará correctamente si es capaz de mantener dicho valor, es decir, el valor cuadrático medio de las diferencias detectadas entre los procesos reproductores de ambos flujos, por debajo del valor 6400 milisegundos^2 (correspondiente a una asincronía de ±80 milisegundos) o, en el peor de los casos, por debajo del valor 25600 milisegundos^2 (correspondiente a una asincronía de ±160 milisegundos). En la tabla anterior también puede apreciarse cómo el mínimo valor medio del error cuadrático de la asincronía le corresponde al receptor PC4, con 294,73 milisegundos^2, mientras que el receptor con el valor medio más alto fue el receptor PC8, con 403,31 milisegundos^2.

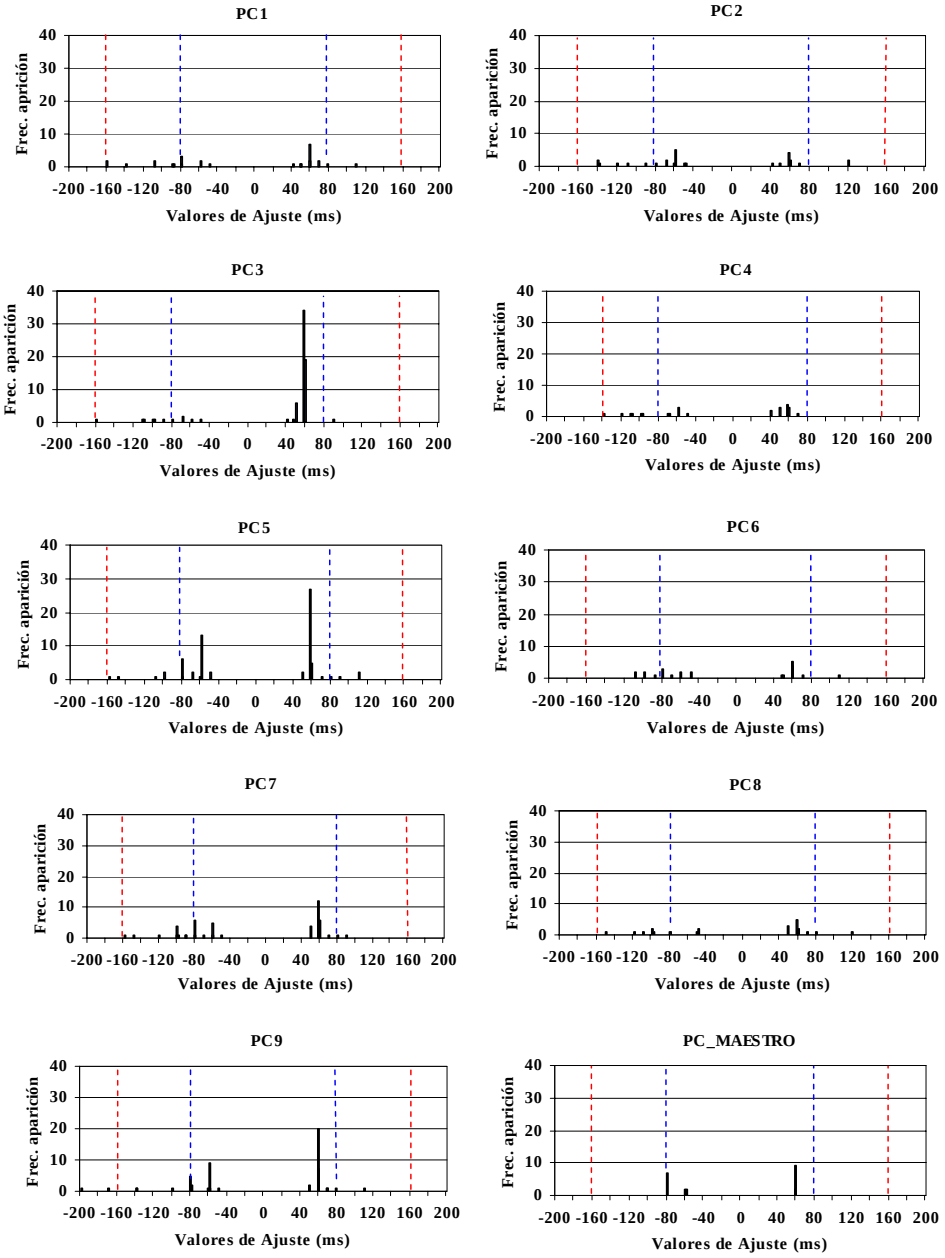


Figura 6.24. Número de veces que aparece un determinado ajuste en el proceso reproductor del flujo *esclavo* (vídeo)
(aprendizaje a distancia - LAN - con algoritmo)

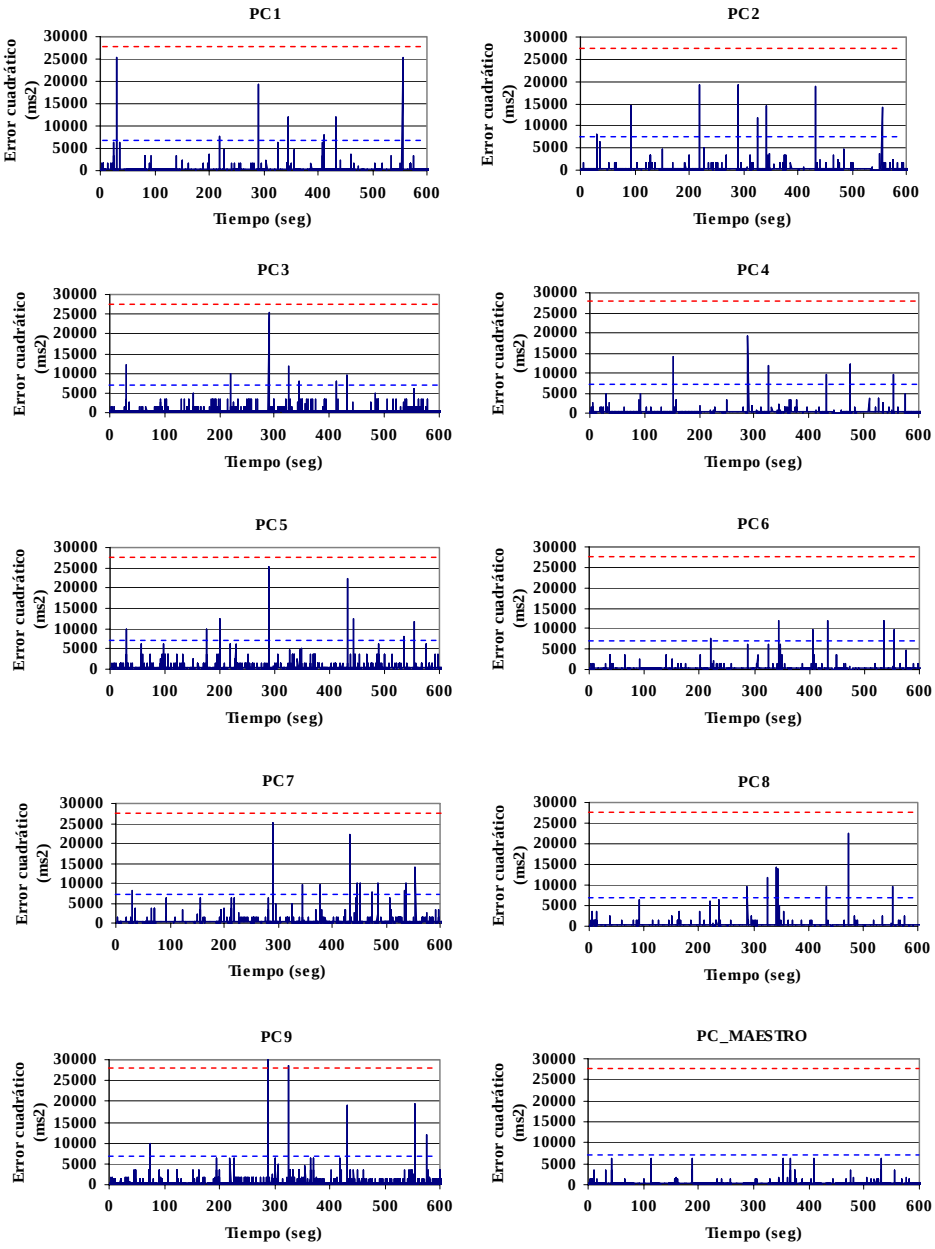


Figura 6.25. Valor cuadrático del error de sincronización inter-flujo (aprendizaje a distancia - LAN - con algoritmo)

En la tabla 6.8 se presentan los valores más representativos de los parámetros analizados.

ESTADÍSTICAS RECEPTOR	**VALORES**
Máxima Asincronía positiva detectada (ms)	120 (PC2 y PC8)
Máxima Asincronía negativa detectada (ms)	-199 (PC9)
Mínima asincronía detectada (ms)	0
Valor Cuadrático medio máximo de la Asincronía detectada en un receptor (ms^2)	403,31 (PC8)
Media de los valores cuadráticos medios	310,75
Nº máximo/mínimo de mensajes *mbus* enviados por un receptor	74 (PC3) 25 (PC8) 23 (PC_MAESTRO)
Asincronía Máxima permitida audio-vídeo (± ms)	±80
Asincronía Máxima configurada audio-vídeo (± ms)	±40
Nº máximo/mínimo de Asincronías Audio-Vídeo detectadas en un receptor	13 (PC7) 7 (PC4, PC6, PC9) 1 (PC_MAESTRO)
Nº máximo/mínimo de *'saltos'* realizados por el flujo de vídeo (*esclavo*) de un receptor debido a la sincronización inter-flujo	51(PC7) 23 (PC8) 20 (PC_MAESTRO)
Nº máximo/mínimo de *'pausas'* realizadas por el flujo de vídeo (*esclavo*) de un receptor debido a la sincronización inter-flujo	82 (PC3) 17 (PC4) 13 (PC_MAESTRO)
Duración Muestra flujo de vídeo (ms)	40

Tabla 6.8. Valores más representativos
(aprendizaje a distancia - LAN - con algoritmo)

6.3.1.2. Entorno CAMPUS

En este entorno, se ha evaluado la transmisión de los mismos flujos de audio y vídeo utilizados en la evaluación del entorno anterior, pero, en este caso, el Servidor de Flujos Multimedia que realiza la transmisión está situado en el Campus de Vera en Valencia, mientras que los 10 receptores están situados en uno de los laboratorios del Campus de Gandia.

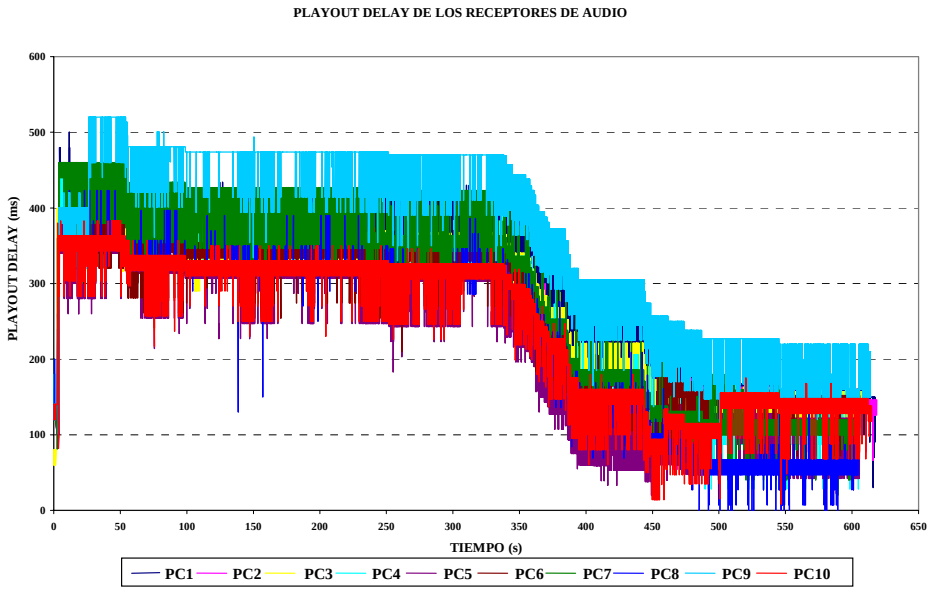
A) SIN ALGORITMO

Al igual que con la evaluación en entorno LAN, en este caso, primero también se ha evaluado el comportamiento sin la utilización del algoritmo propuesto para, posteriormente, poder comparar y observar las mejoras introducidas por éste.

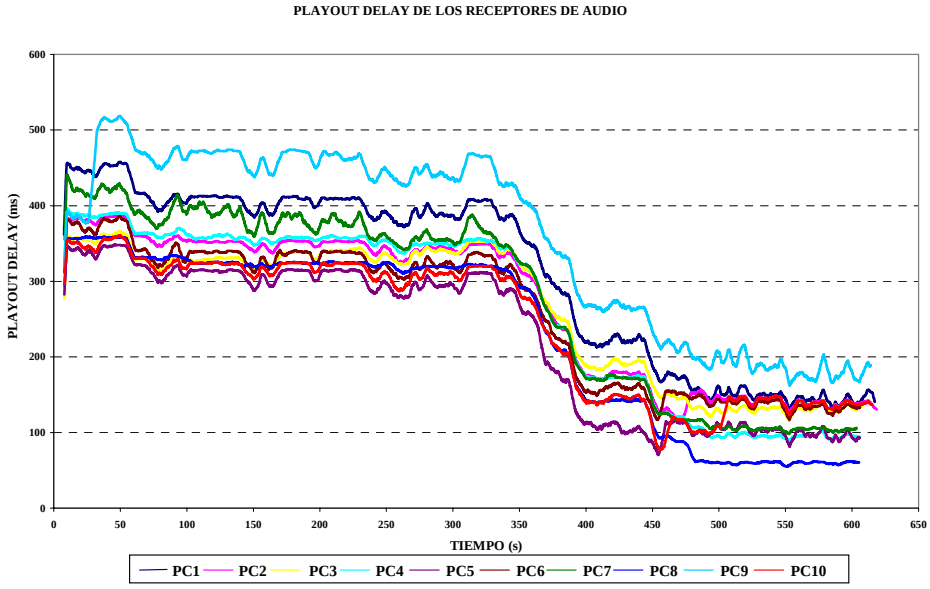
- ***Retardos de reproducción (playout delay) de los procesos reproductores del flujo maestro de todos los receptores***

A continuación, se muestra la figura 6.26 que representa el *playout delay* del flujo de audio experimentado por los diferentes receptores durante toda la sesión evaluada. Para suavizar las fluctuaciones de la gráfica con los valores reales obtenidos (figura 6.26a) y mostrar con más claridad la trama de valores, en la figura 6.26b, se ha representado la *media móvil* de la serie de valores reales, a partir de conjuntos de 100 muestras.

En la figura 6.26b se puede apreciar que entre el estado de reproducción con menor *playout delay*, correspondiente al receptor PC5 (representado en color morado), y el estado de reproducción con mayor retardo, correspondiente al receptor PC9 (representado en azul claro), se excede, en media, el límite de asincronía de ±120 milisegundos, fijado para flujos de audio poco acoplados en modo diálogo.



a) Valores reales



b) Media móvil

Figura 6.26. *Playout delay* del flujo de audio de los receptores (aprendizaje a distancia - CAMPUS - sin algoritmo)

- ***Número de Secuencia de las LDUs del flujo maestro reproducidas en cada receptor***

En primer lugar, se ha comprobado que, en ausencia de algoritmo de sincronización, el instante de inicio de la reproducción no coincide en todos los receptores. En la figura 6.27 se representa la primera LDU de la sesión reproducida por cada receptor. En ella se puede apreciar que los reproductores PC2 y PC3 inician la reproducción de la LDU número 25368 con un desfase de 110 milisegundos entre sus procesos reproductores.

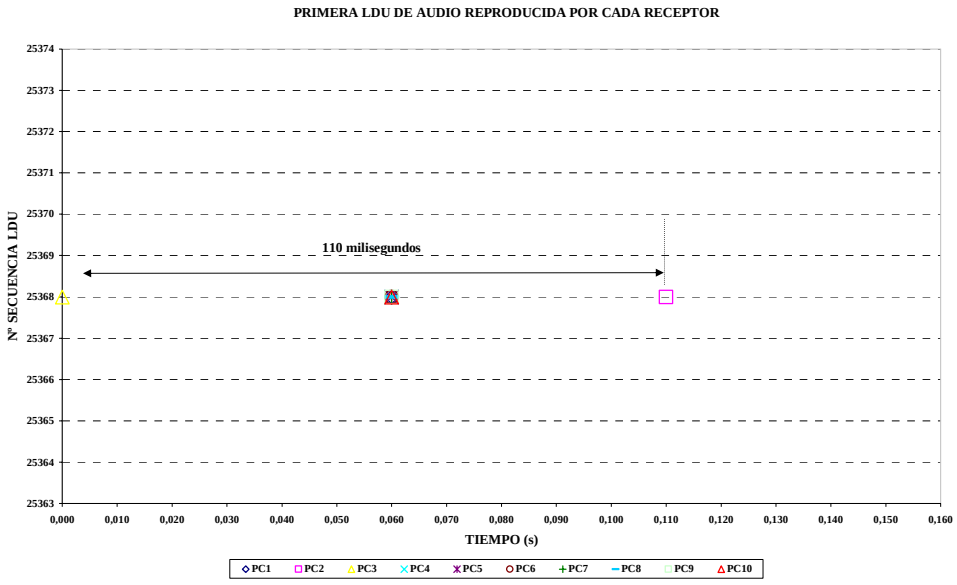


Figura 6.27. Primera LDU del flujo de audio reproducida (aprendizaje a distancia - CAMPUS - sin algoritmo)

En la siguiente figura, se muestran los números de secuencia de las LDUs reproducidas por cada receptor en función del tiempo, durante un fragmento de 3 segundos, para poder apreciar mejor la información representada. Se muestra el desfase existente entre los procesos reproductores del flujo de audio en los diferentes receptores en un mismo instante de tiempo. Si nos fijamos en un instante de tiempo y trazamos una línea vertical, podemos observar que el desfase entre los procesos de reproducción de los receptores que aparecen en los extremos (receptor PC10, en color rojo, y receptor PC9, en color azul claro) es, aproximadamente, de unas 100 LDUs, que se corresponden con unos 2000 milisegundos de asincronía entre ambos receptores. En la figura se ha marcado un punto donde esta diferencia es de 108 LDUs, correspondiendo a un adelanto del proceso reproductor del receptor PC10 equivalente a 2160 milisegundos.

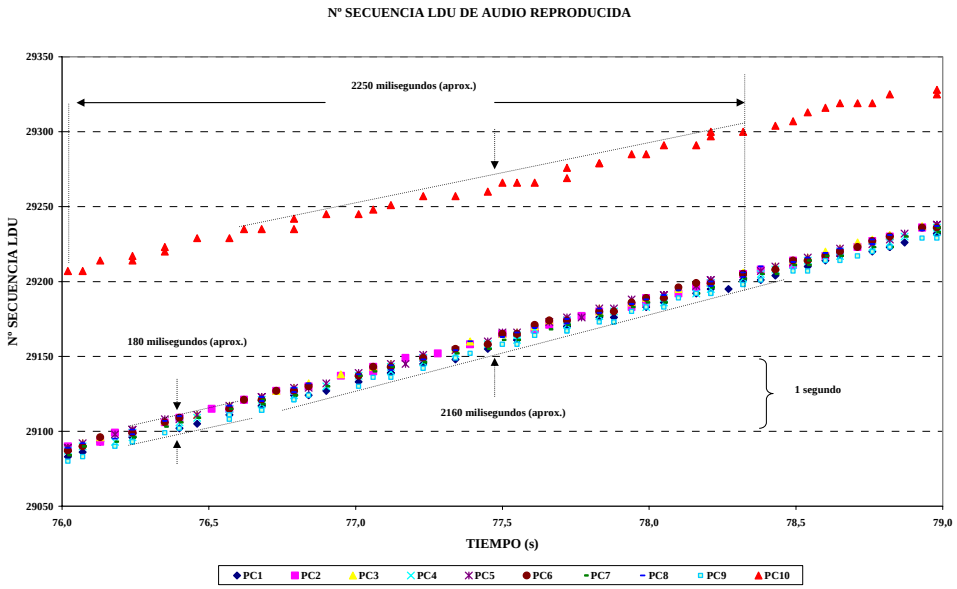


Figura 6.28. Número de secuencia de las LDUs de audio reproducidas (aprendizaje a distancia - CAMPUS - sin algoritmo)

Si nos fijamos en la LDU número 29.207 vemos que el PC10 la reproduce unos 2250 milisegundos antes que el resto. En la figura también se puede apreciar que, con la excepción de PC10, las asincronías entre el resto de receptores están dentro de un margen de 180 milisegundos.

Se comprueba, por tanto, que, en ausencia de algoritmo de sincronización, existe una asincronía en los procesos de reproducción que sobrepasa el límite fijado, de ±120 milisegundos.

B) CON EL ALGORITMO DE SINCRONIZACIÓN DE GRUPO

A diferencia de la configuración anteriormente evaluada, en este caso, por un lado, todos los receptores habrán activado, desde la *interfaz gráfica de usuario* de las aplicaciones, la opción de sincronización mediante el algoritmo propuesto y, por otro lado, al transmisor o Servidor de Flujos Multimedia, desde su *interfaz gráfica de usuario*, se le han configurado los parámetros necesarios para la correcta ejecución del algoritmo (ver figuras 6.12 y 6.13), con la finalidad de obtener valores de asincronía dentro de los límites que aparecen en la siguiente tabla:

En este caso también se ha seleccionado el receptor PC10 como el receptor *maestro*, en adelante PC_MAESTRO, que será el tomado como referencia para la sincronización de grupo entre receptores.

Parámetro	**Valor**
Asincronía Máxima permitida entre los reproductores del flujo de audio (*Sincronización de Grupo distribuida*).	±120 ms
Máximo retardo de reproducción permitido en los reproductores de audio.	1 s
Asincronía Máxima permitida entre los procesos reproductores de los flujos de audio y vídeo (*Sincronización inter-flujo local*).	±80 ms

Tabla 6.9. Valores límite a cumplir
(aprendizaje a distancia - CAMPUS - con algoritmo)

A continuación, se presenta un conjunto de figuras con datos obtenidos durante la sesión que demuestran el buen comportamiento del algoritmo de sincronización de grupo propuesto durante la misma, para el entorno CAMPUS evaluado.

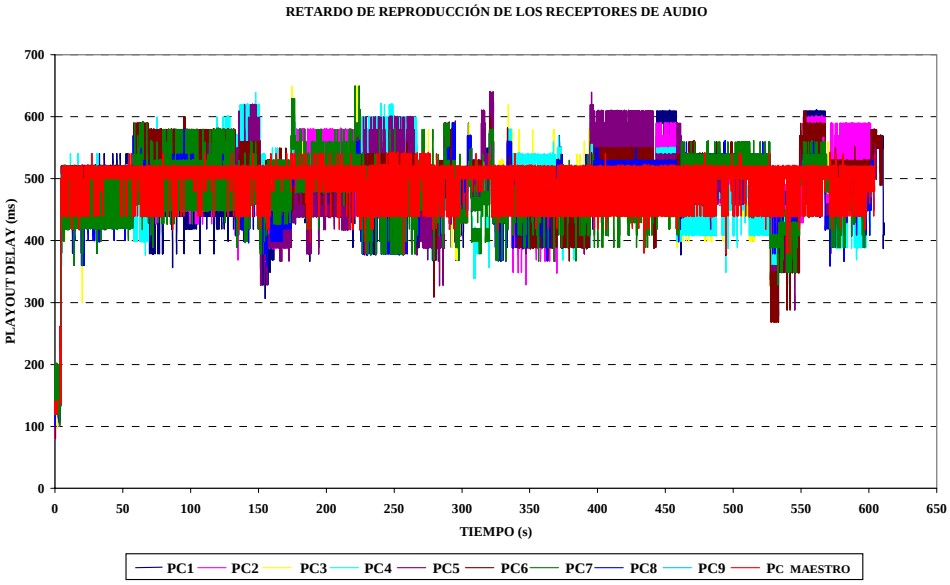
En primer lugar, se presentan una serie de gráficas que confirman el buen funcionamiento de la sincronización de grupo distribuida y de la sincronización del instante inicial de consumo, con respecto al flujo *maestro*. En segundo lugar se evaluará la sincronización inter-flujo entre los flujos *maestro* y *esclavo*, localmente en cada receptor.

B1. Sincronización de Grupo (distribuida)

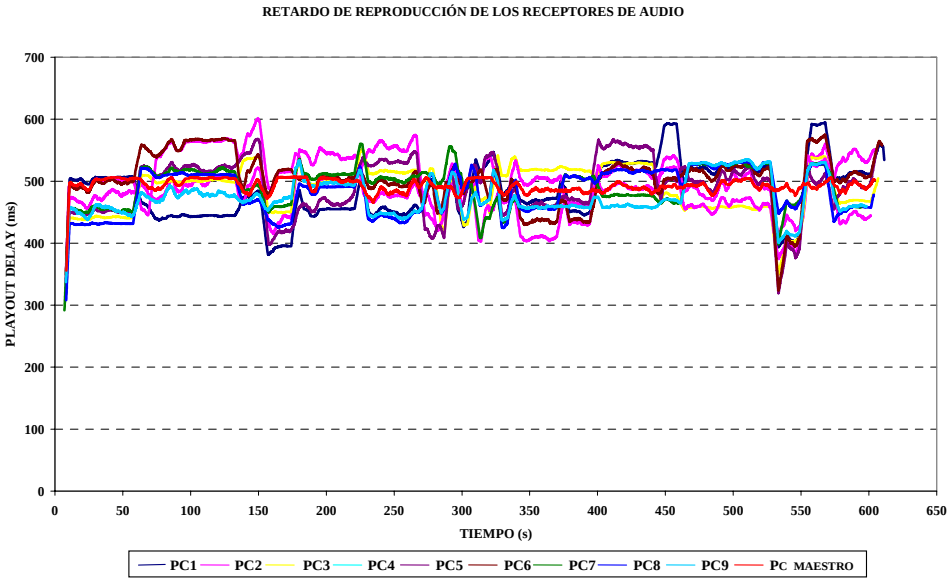
- ***Retardos de reproducción (playout delay) de los procesos reproductores del flujo maestro de todos los receptores***

A continuación se muestra la figura 6.29 que representa el retardo de reproducción (*playout delay*)de los procesos reproductores del flujo de audio, experimentado por los 10 receptores.

Como en el caso anterior, para suavizar las fluctuaciones de la gráfica y mostrar con más claridad la trama de valores, en la figura 6.29b se ha representado la *media móvil* de la serie de datos representados en la figura 6.29a, a partir de conjuntos de 100 muestras. En ambas se representa el tiempo completo de la sesión.



a) Valores Reales



b) Media móvil (conjuntos de 100 muestras)

Figura 6.29. *Playout delay* del flujo de audio de los receptores (aprendizaje a distancia - CAMPUS - con algoritmo)

Se puede observar el salto inicial en el retardo de reproducción de los receptores debido al arranque de las aplicaciones y al consumo de recursos inicial.

Se comprueba que los retados de reproducción de los procesos reproductores del flujo de audio de los receptores *esclavos* se van adaptando en determinados puntos al retardo de reproducción del proceso del receptor *maestro* (PC_MAESTRO, en rojo), tomado como referencia (cuyos valores están entorno a los 500 milisegundos durante la sesión).

Para apreciar mejor este efecto, se muestra la figura 6.30, donde se ha representado la misma gráfica pero sólo con un receptor *esclavo* (PC1) y, además, se han incluido los ajustes que ha sufrido el proceso reproductor de dicho receptor, debido a las operaciones de '*saltos*' y '*pausas*' efectuadas por las acciones de resincronización del algoritmo.

El proceso reproductor del flujo *maestro* en el receptor *maestro* no sufre ajustes, ni debidos a la sincronización de grupo ni a la sincronización inter-flujo, por lo que su estado de reproducción fluctuará menos, tal como se comprueba en las dos figuras anteriores.

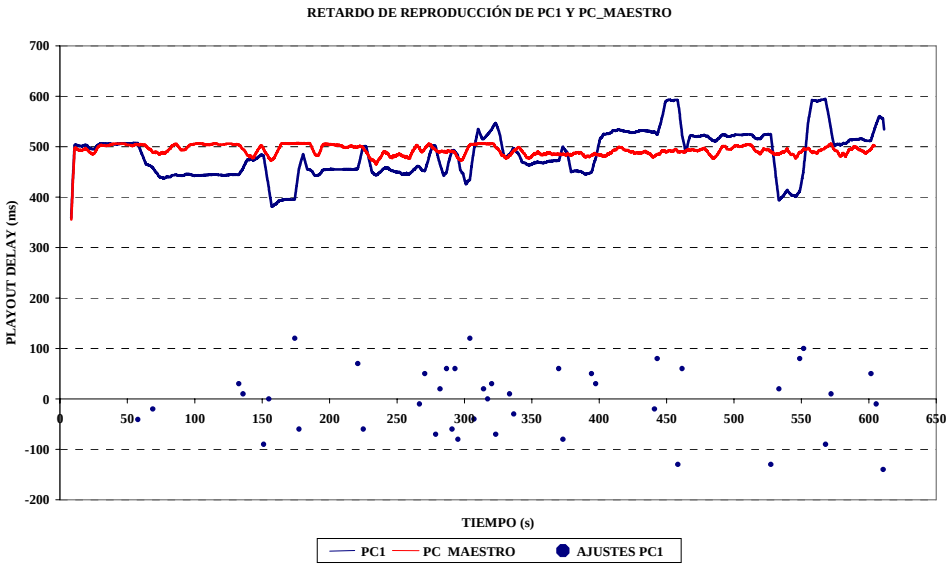


Figura 6.30. *Playout delay* del flujo *maestro* del receptor *maestro* y de un único receptor *esclavo*. Ajustes en el receptor *esclavo*
(aprendizaje a distancia - CAMPUS - con algoritmo)

En la tabla 6.10 se muestran, para este entorno, los mismos parámetros representativos evaluados en entorno LAN.

ESTADÍSTICAS RECEPTOR	**PC1**	**PC2**	**PC3**	**PC4**	**PC5**	**PC6**	**PC7**	**PC8**	**PC9**	**PC10** (MAESTRO)
Máx. asincronía positiva (ms)	120	120	160	140	100	140	110	110	100	-
Máx. asincronía negativa (ms)	-140	-90	-110	-130	-120	-190	-130	-130	-130	-
Mín. asincronía (ms)	0	0	±10	0	0	0	0	0	0	-
Valor medio de la Asincronía (ms)	-2,12	1,44	1,88	-1,28	2,17	1,17	0,23	2,17	0,73	-
Desv. estándar de las asincronías detectadas (ms)	67,36	54,47	65,76	62,82	56,44	63,88	56,88	60,23	53,8	-
Valor cuadrático medio asincronía (ms^2)	4437	2897	4225	3850	3113	3985	3154	3544	2821	-
Nº mensajes de '*acción*' (APP TIN) enviados	1									
Nº mensajes de '*acción*' (APP ACT) enviados	42									
Nº mensajes recibidos (APP ACT)	42	42	42	42	42	42	42	42	42	42
Nº mensajes *Feedback* enviados (RR Ext.)	225	222	222	225	219	230	213	227	220	230
Asincronía máx. permitida audio-audio	±120 ms									
Nº asincronías audio-audio	3	0	1	2	0	2	2	1	2	-
'*Saltos*' realizados por el flujo de audio (*maestro*) debido a la sincr. de grupo	60	42	57	51	44	50	40	43	39	-
'*Pausas*' realizadas por el flujo de audio (*maestro*) debido a la sincr. de grupo	56	46	58	50	44	48	39	53	38	-
Nº de paquetes RTP enviados	30200									
Duración muestra flujo de audio (ms)	20									

Tabla 6.10. Datos obtenidos de todos los receptores (aprendizaje a distancia - CAMPUS - con algoritmo)

En primer lugar se presentan los valores máximos, tanto positivos como negativos de las asincronías detectadas por cada receptor. Puede observarse que en todos los receptores, excepto en PC2 y PC5, se ha superado el valor máximo de 120 milisegundos por lo que se habrán producido asincronías que deberán haber sido corregidas pro nuestro algoritmo.

En ella también puede apreciarse la poca cantidad de paquetes de control que envía el servidor durante la sesión. El servidor envía 1 sólo paquete APP TIN para indicar el instante de inicio de la reproducción (*Instante Inicial de Consumo*) y, posteriormente, durante la sesión envía 42 paquetes APP ACT de acción, indicando acciones de resincronización en los receptores.

En este caso, tampoco ha sido necesario que el servidor enviara más paquetes de sincronización gruesa (APP TIN) para evitar situaciones de *overflow* en los *buffers* de los receptores. Puede comprobarse en la figura 6.29, que el *playout delay* no supera en ninguno de los receptores el tiempo de un segundo configurado, desde la *interfaz gráfica de usuario* de la aplicación *rat* del servidor, como el límite máximo que si era sobrepasado forzaba el envío de un paquete APP TIN durante la sesión.

Si se compara con la evaluación en entorno LAN, en este caso el servidor ha enviado 42 paquetes APP ACT, frente a los 26 enviados en aquella. Como era de esperar, en este entorno el servidor ha detectado en más ocasiones (42, más de un 50% más) que, como mínimo, uno de los procesos reproductores de algún receptor estaba retrasado o adelantado más de 120 milisegundos con respecto al proceso reproductor del receptor *maestro*.

El servidor, durante la sesión evaluada, ha enviado 30200 paquetes RTP con datos de audio (cada uno con muestras de audio correspondientes a 20 milisegundos de duración). Por otro lado, los mensajes de control enviados por el mismo (1 APP TIN y 42 APP ACT, es decir, 43 mensajes de control RTCP) apenas suponen un 0,14 % del total de los paquetes transmitidos (de datos y de control). Por otro lado, los mensajes de control enviados por los receptores (en total 2233 mensajes RTCP RR extendidos, 22 menos que en la evaluación en entorno LAN, donde se enviaron 2255) apenas suponen un 6,88 % del total de los paquetes transmitidos (de datos y de control) por las aplicaciones de audio durante la sesión y un 7,39 % con respecto al número total de paquetes de datos enviados (30200 paquetes RTP). Se puede comprobar que estos valores son muy parecidos a los obtenidos para el entorno LAN. En la tabla anterior se puede apreciar que los receptores que han enviado un mayor y un menor número de mensajes RR extendidos han sido PC6 y PC_MAESTRO (con 230 mensajes) y PC7 (con 213 mensajes), respectivamente, valores similares a los obtenidos en entorno LAN.

Tomando como asincronía máxima, entre flujos de audio, ±120 milisegundos, se puede comprobar cómo los receptores, en general, sufren de 0 a 3 situaciones de asincronía durante la sesión (situaciones en las que se ha superado dicho valor), que son cantidades mínimas. En concreto, de los nueve receptores, 2 receptores no han detectado ninguna asincronía, 2 receptores han detectado una sola situación de asincronía, 4 receptores han detectado dos situaciones de asincronía y 1 receptor (PC1) ha detectado tres situaciones de asincronía. En este caso, también, se ha recuperado la sincronización rápidamente y se ha mantenido dentro de los límites de asincronía configurados (±120 milisegundos).

Si analizamos el valor del parámetro *Valor cuadrático medio de la asincronía detectada*, se puede observar que en ningún receptor supera el valor 14400 milisegundos^2 (cuadrado del valor máximo de asincronía permitido de ±120 milisegundos), consiguiéndose valores adecuados de sincronización de grupo (distribuida) para flujos de audio poco acoplados en modo diálogo.

En la tabla también aparecen la cantidad de '*saltos*' y '*pausas*' que sufren los procesos reproductores del flujo de audio (*maestro*) para ajustarse al punto de reproducción instantáneo del proceso del receptor *maestro*. El receptor que más '*saltos*' ha experimentado en el proceso reproductor del flujo de audio ha sido el receptor PC1 con 60 '*saltos*' (dejando de reproducir 60 LDUs), mientras que el receptor cuyo proceso reproductor ha sufrido menos '*saltos*' ha sido PC9, que dejó de reproducir 39 LDUs. En cuanto a las '*pausas*', el receptor que más '*pausas*' ha experimentado en su proceso reproductor de audio ha sido el receptor PC3, con 58 '*pausas*' (es decir, repitió en 58 ocasiones la reproducción de una LDU anterior), mientras que el receptor cuyo proceso reproductor ha sufrido menos '*pausas*' ha sido PC7, con 39 repeticiones de LDUs. A la vista de los datos de la tabla, parece que el proceso reproductor del receptor PC9 era el que más se aproximaba al punto de reproducción del receptor *maestro* durante la sesión, con 39 '*saltos*' y 38 '*pausas*', mientras que el que más se alejaba fue PC1, con 60 '*saltos*' y 56 '*pausas*'. En este entorno, a pesar de realizarse más operaciones de '*acción*', es decir, más ajustes en los procesos reproductores de los receptores (debidos a la recepción de un mensaje de acción del servidor, APP ACT), los valores máximo y mínimo de dichos ajustes son similares a los obtenidos durante la evaluación en un entorno LAN.

En la figura 6.31 se muestra, para cada receptor, la distribución de los ajustes que ha sufrido su proceso reproductor del flujo *maestro*, como consecuencia de la recepción de un paquete de '*acción*' (APP ACT) enviado por el servidor al detectar alguna situación de asincronía en alguno de los receptores. Tal como ya se ha comentado anteriormente, los ajustes reflejan los errores de sincronización entre los procesos reproductores de los receptores y el proceso reproductor del receptor *maestro* y, para una sincronización de calidad entre flujos de audio poco acoplados, deberían mantenerse por debajo del valor comentado

anteriormente de ±120 milisegundos. En dichas gráficas puede apreciarse, también, que los procesos reproductores de los receptores sufren más ajustes que en un entorno LAN.

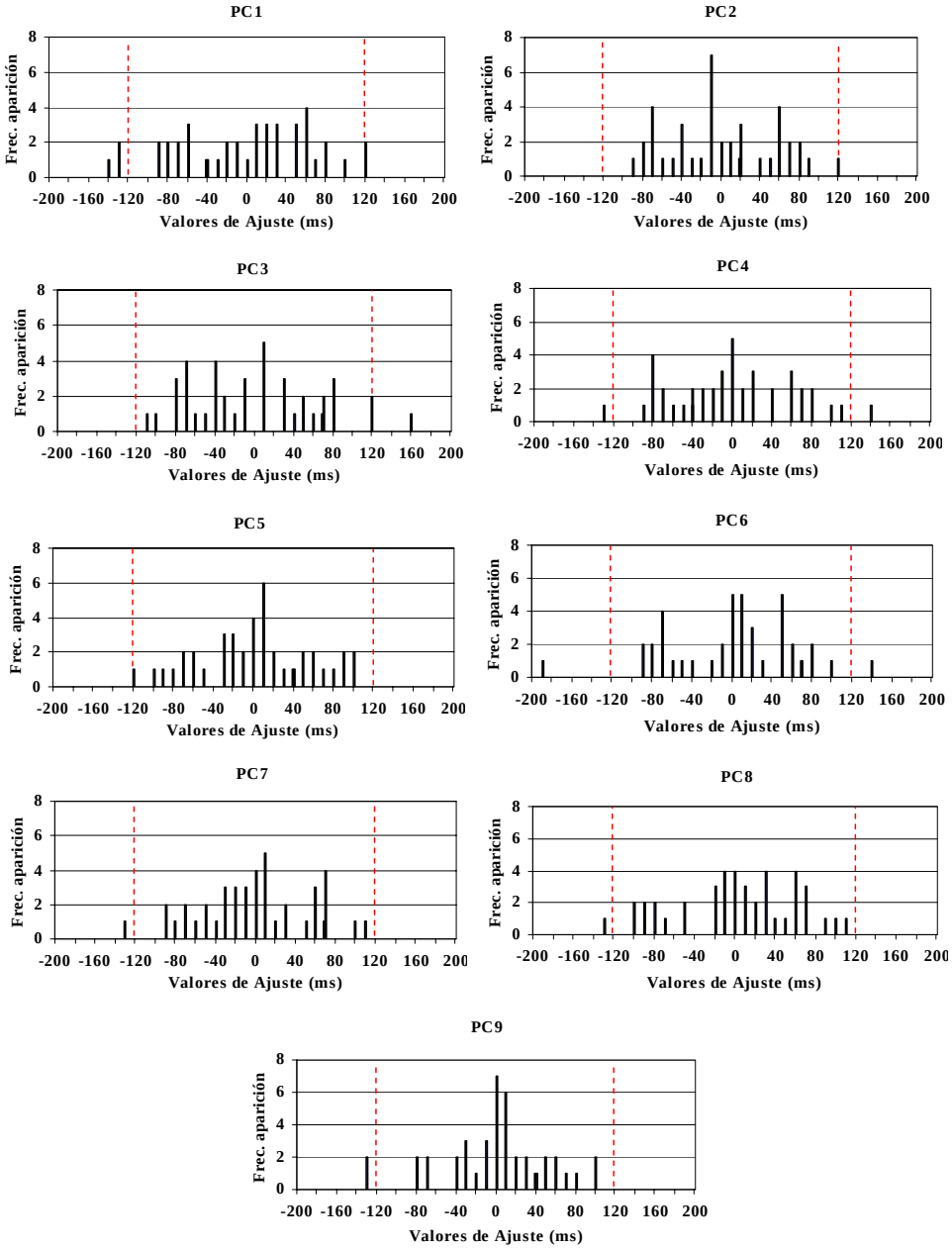


Figura 6.31. Distribución de los valores de ajuste del proceso de reproducción del flujo *maestro* de cada receptor (aprendizaje a distancia - CAMPUS - con algoritmo)

En la figura 6.32 se muestra para cada receptor el valor cuadrático de dichos ajustes a la largo de la sesión. Se ha resaltado, mediante una línea roja, el valor de 14400 milisegundos^2, que es el valor máximo permitido.

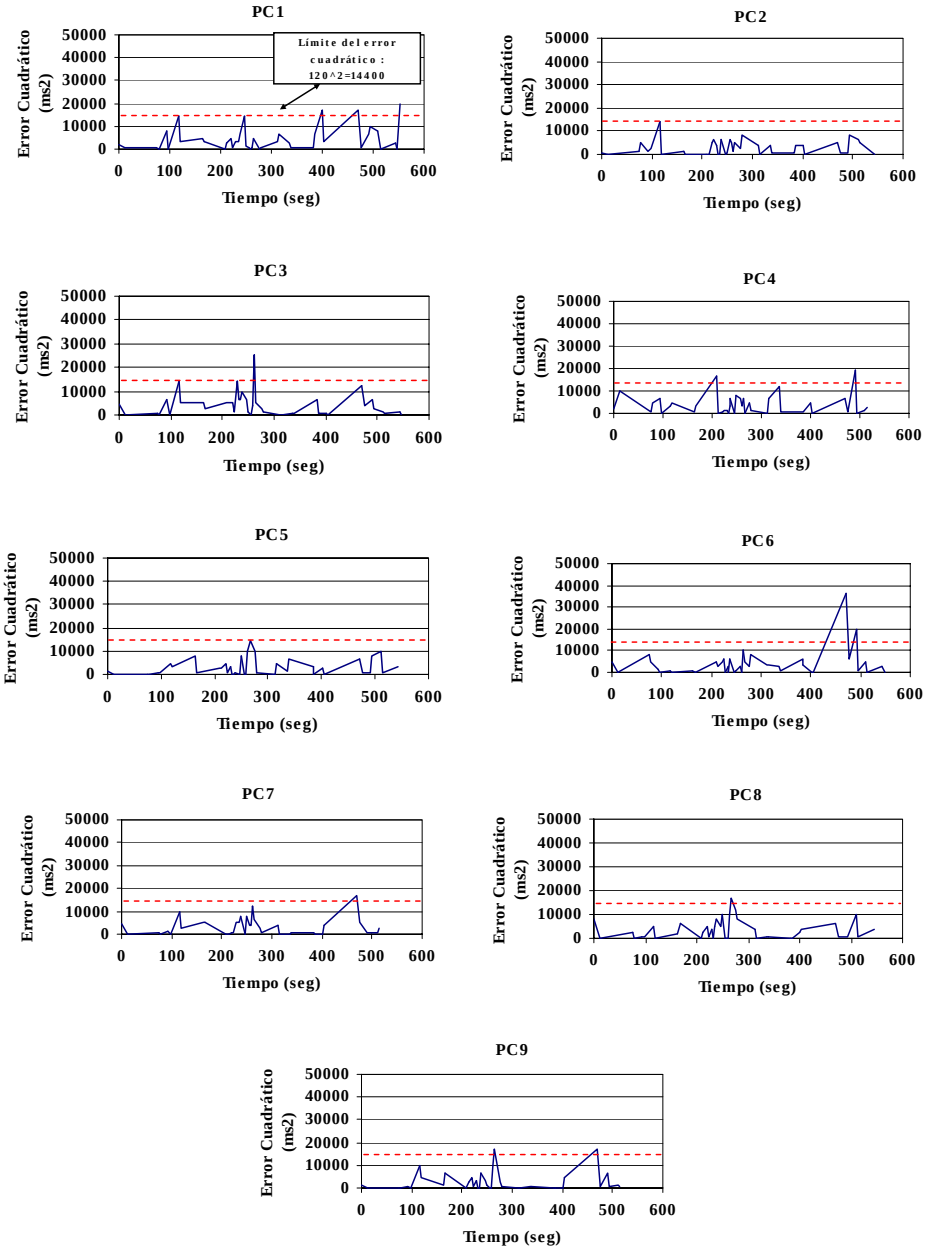


Figura 6.32. Valor Cuadrático del Error de Sincronización (aprendizaje a distancia - CAMPUS - con algoritmo)

En este entorno algunos receptores sobrepasan este valor en ocasiones muy concretas (sin embargo, el valor medio del error cuadrático de la sesión no lo supera), pero rápidamente se detecta la situación de asincronía y es corregida por el algoritmo. La mayor parte del tiempo de la sesión los valores están por debajo de dicho valor. Sólo se han representado errores de cada receptor, almacenados en los instantes en que éstos reciben un mensaje de '*acción*' que el servidor envía cuando detecta asincronías por encima de 120 milisegundos. Si no envía mensajes es porque no detecta que se supere dicho valor y, por consiguiente, se supone que el error cuadrático de todos los receptores esta por debajo de 14.400 milisegundos^2.

En la tabla 6.11 aparecen los valores más representativos comentados (excluyendo los del receptor *maestro*).

ESTADÍSTICAS RECEPTOR	**VALORES**
Máxima asincronía positiva detectada (ms)	160 (PC3)
Máxima asincronía negativa detectada (ms)	-190 (PC6)
Valor mínimo de asincronía detectado	0
Máximo valor cuadrático medio de la asincronía detectada (ms^2)	4437
Media del valor cuadrático medio de todos los receptores	3558,44
Nº mensajes de '*acción*' (APP TIN) enviados por servidor	1
Nº mensajes de '*acción*' (APP ACT) enviados por servidor	42
Nº máximo de de mensajes recibidos del servidor (APP ACT)	42
Nº máximo de mensajes *Feedback* enviados por receptor (RR Ext.)	230 (PC6 y PC_MAESTRO)
Nº mínimo de mensajes *Feedback* enviados por receptor (RR Ext.)	213 (PC7)
Asincr. Máx. permitida audio-audio respecto al receptor *maestro*	±120 ms
Nº máximo/mínimo de asincronías audio-audio en un receptor	3 (PC1) 0 (PC2, y PC5)
Nº máximo/mínimo de *'saltos'* realizados por el flujo de audio (*maestro*) de un receptor debido a la sincronización de grupo	60 (PC1)/ 39 (PC9)
Nº máximo/mínimo de *'pausas'* realizadas por el flujo de audio (*maestro*) de un receptor debido a la sincronización de grupo	58 (PC3) 39 (PC7)
Número de paquetes RTP enviados	30200
Duración muestra flujo de audio (ms)	20

Tabla 6.11. Valores más representativos
(aprendizaje a distancia - CAMPUS - con algoritmo)

Si comparamos la media de los valores medios de cada uno de los receptores, que en este caso es de 3558,44 milisegundos^2, con respecto al mismo valor obtenido en la evaluación en entorno LAN, de 4548,11 milisegundos^2 (tabla 6.11), podemos observar que en este caso es inferior. Esto puede ser debido a que, en este caso, al realizarse más ajustes que en entorno LAN, se ha mantenido el error cuadrático en unos niveles inferiores a los obtenidos en dicho entorno.

Del análisis de los datos anteriores se puede concluir que, en entorno CAMPUS, *el mecanismo de sincronización de grupo de flujos multimedia ha funcionado correctamente a lo largo de la sesión evaluada.*

- ***Número de Secuencia de las LDUs del flujo maestro reproducidas en cada receptor***

Veamos si el número de secuencia de la LDU que reproduce cada receptor en cada instante coincide para corroborar el funcionamiento del algoritmo.

En primer lugar se ha comprobado que el instante de inicio de la reproducción coincide en todos los receptores ya que todos los receptores comenzaron la reproducción de la primera LDU transmitida (cuyo número escogido aleatoriamente por la fuente, en este caso, fue el 7895) en el mismo instante de tiempo, tal como se puede apreciar en la siguiente figura.

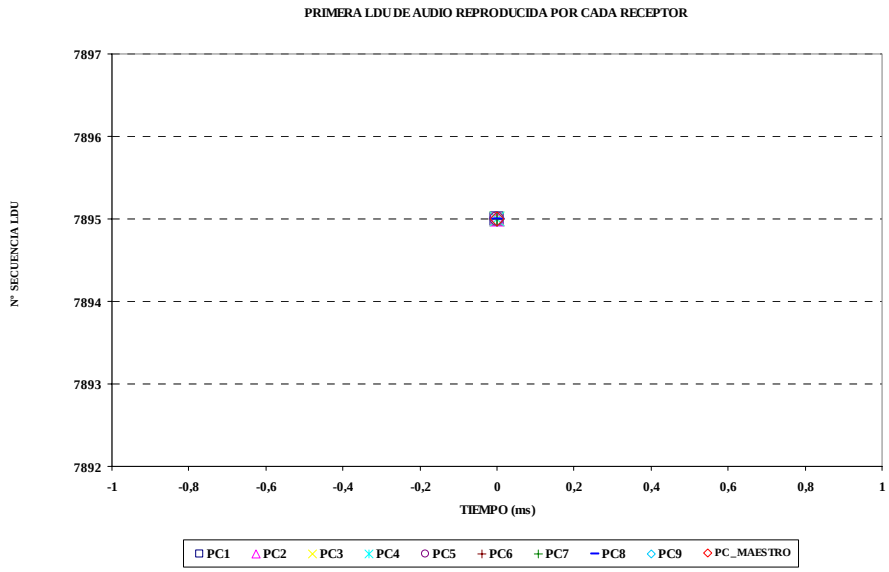


Figura 6.33. Primera LDU del flujo *maestro* reproducida (aprendizaje a distancia - CAMPUS - con algoritmo)

Podemos concluir, pues, que *el mecanismo de sincronización del instante inicial de consumo ha funcionado correctamente* también en este entorno.

A continuación, en la figura 6.34 se representan un conjunto de LDUs cuyos datos han sido obtenidos durante la sesión de evaluación.

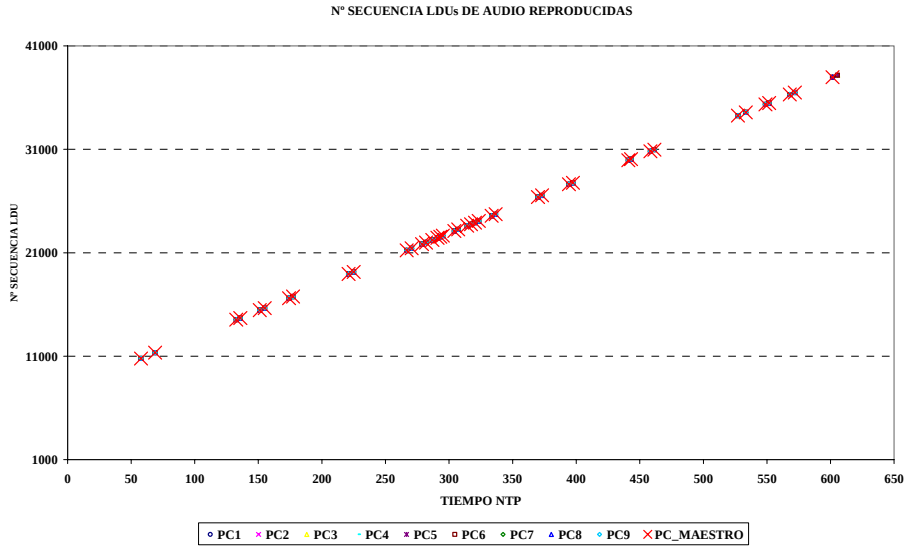


Figura 6.34. Número de secuencia de las LDUs de audio reproducidas

La figura 6.35 muestra un intervalo reducido de tiempo donde se representa el número de secuencia de una de las LDUs que cada receptor está reproduciendo en dicho intervalo para poder, así, analizar mejor el grado de sincronización existente durante el mismo.

En dicha figura se puede observar cómo cuando el receptor *maestro* (en rojo) está reproduciendo la LDU número 23252, los receptores PC3, PC4, PC7 y PC9 están reproduciendo la misma (por tanto, están sincronizados con el *maestro*), el receptor PC8 (en azul) está reproduciendo la LDU 23249 (por tanto, está retrasado 60 milisegundos con respecto al receptor *maestro*); los receptores PC2 y PC6 están reproduciendo la LDU 23255 (por tanto, están adelantados 60 milisegundos con respecto al receptor *maestro*). Por otro lado, el receptor PC5 (en morado) habrá reproduciendo la LDU 23252, 50 milisegundos antes (por tanto, estará adelantado 50 milisegundos con respecto al receptor *maestro*), mientras que el receptor PC1 habrá reproducido la LDU 23249, también 50 milisegundos antes (con lo que, si mantiene la tasa de reproducción de una LDU cada 20 milisegundos, reproduciría la LDU 23252 unos 10 milisegundos después que el maestro y, por tanto, irá retrasado ese tiempo con respecto a aquél).

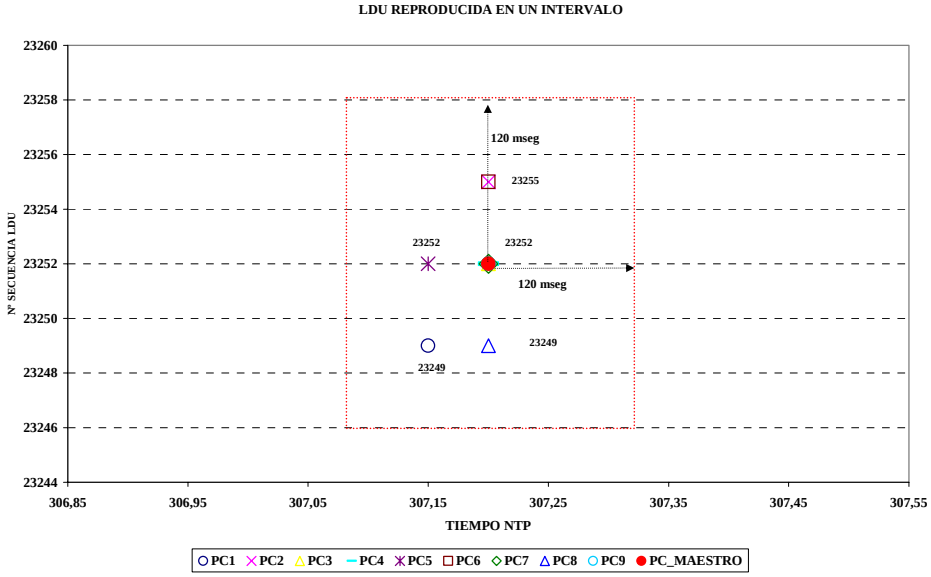


Figura 6.35. Número de secuencia de LDU reproducida en un intervalo reducido (aprendizaje a distancia - CAMPUS - con algoritmo)

Como se puede observar, en el intervalo representado, las diferencias en el estado de reproducción de los receptores *esclavos*, con respecto al proceso del receptor *maestro*, están dentro de los límites de asincronía máxima permitidos (±120 milisegundos, representados por un cuadrado de color rojo, centrado en la muestra reproducida por el receptor *maestro*) para flujos de audio poco acoplados en modo diálogo.

En la figura 6.36 se han representado 40 muestras obtenidas durante la sesión, superpuestas, tomando diferencias relativas, de tiempos (eje *x*) y de número de LDU reproducida en cada instante (eje *x* y eje *y*, respectivamente). Se estará sincronizado si los puntos representados están dentro del cuadro rojo que representa una asincronía de ±120 milisegundos.

En la figura hay representados 370 puntos correspondientes a la muestra reproducida por el receptor *maestro* (en rojo, centrada en el punto (0,0)) y cada una de las muestras reproducidas por cada receptor en 40 instantes diferentes. Se puede observar en la misma que en la reproducción de estas muestras apenas se han producido asincronías, ya que, prácticamente la totalidad de los puntos están dentro del cuadro, y que los puntos están repartidos por dentro del mismo, indicando que los puntos de reproducción de los receptores oscilan alrededor del punto de reproducción del receptor *maestro* en cada instante.

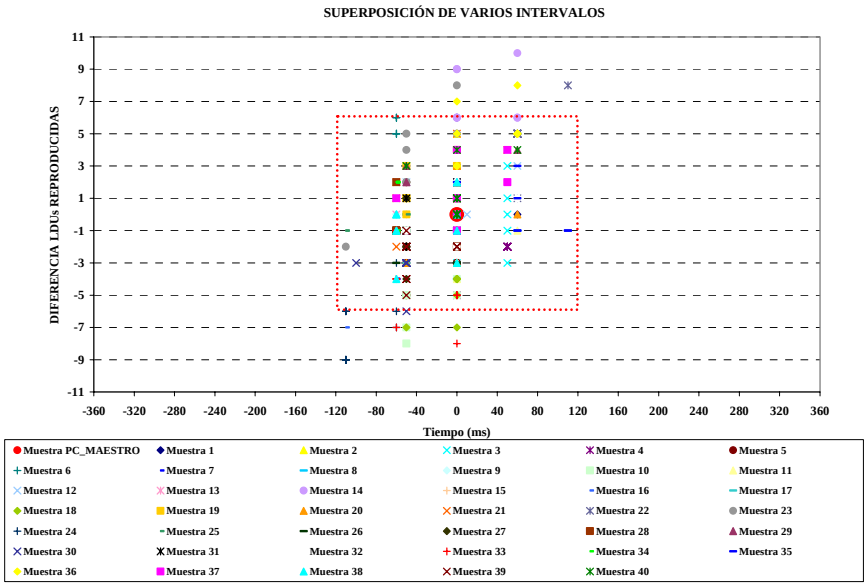


Figura 6.36. Superposición de 40 muestras
(aprendizaje a distancia - CAMPUS - con algoritmo)

Aunque no se aprecia el número de puntos que coinciden (debido a la superposición), en la figura sólo se han quedado fuera del cuadro rojo 34 puntos, es decir un 9,1 % del total de las muestras representadas (que son 40 muestras, frente a las 30200 enviadas en la sesión).

Si se compara con el resultado obtenido en la evaluación LAN, en este caso se producen más situaciones de asincronía, tal y como ya habíamos detectado anteriormente con los parámetros evaluados anteriormente. No obstante, el número de puntos que caen fuera del cuadro sigue siendo muy pequeño (menos del 10 %). Además, se ha analizado el orden de los puntos que caen fuera del cuadro, y se ha detectado que no es consecutivo por lo que el algoritmo corrige estas situaciones de asincronía ya que en el conjunto de muestras posteriores a un conjunto donde había asincronía se ha comprobado minuciosamente que caen todas dentro del cuadro.

B2. Sincronización inter-flujo (local) en cada receptor

Veamos ahora el comportamiento de la sincronización inter-flujo, localmente en cada receptor, entre los procesos de reproducción de los flujos *maestro* y *esclavo*. Como en el caso anterior, la asincronía máxima permitida entre

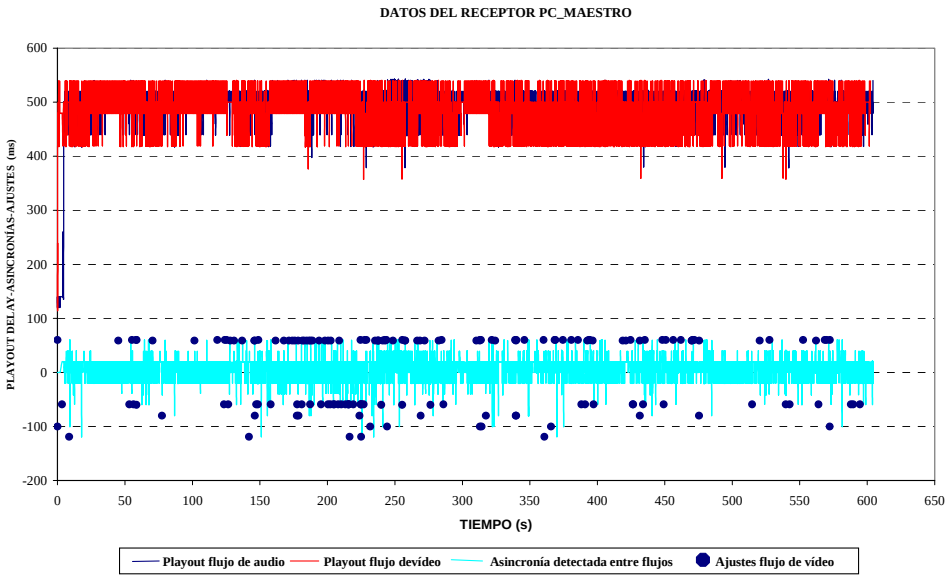
los flujos de audio y vídeo fijada como objetivo a respetar es de ±80 milisegundos para una sincronización inter-flujo de buena calidad. *El algoritmo propuesto funcionará correctamente, en este entorno, si mantiene los valores de asincronía entre los procesos reproductores por debajo de los límites de ±80 milisegundos, corrigiendo las posibles asincronías detectadas que superen dichos límites. En caso de superarse en situaciones puntuales, se deberá mantener siempre por debajo del límite superior de ±160 milisegundos.*

- ***Retardo de reproducción (playout delay) de los procesos reproductores de los flujos maestro y esclavo***

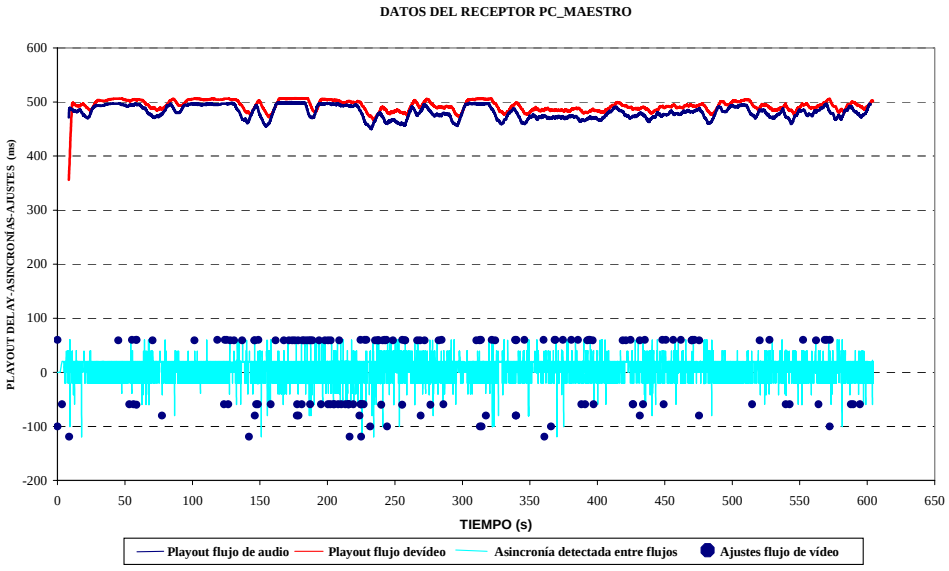
A continuación, se va a comprobar si ambos flujos se reproducen simultáneamente, mediante el análisis de su *playout delay* en cada instante.

En este caso, tampoco se considera necesario mostrar el comportamiento de cada uno de los receptores *esclavos*, por lo que sólo se incluirá el comportamiento de uno de ellos y el del propio receptor *maestro*. En las figuras 6.37 y 6.38 (las gráficas 6.37*a* y 6.38*a* representan los datos reales, mientras que las figuras 6.37*b* y 6.38*b* reflejan el resultado de suavizar los datos reales con la media móvil de los datos tomando conjuntos de 100 muestras), se muestran los retardos de reproducción de los flujos *maestro* (audio) y *esclavo* (vídeo) correspondientes a los receptores PC_MAESTRO y PC1, respectivamente. Para poder entender mejor las gráficas, también se ha incluido la asincronía detectada entre los procesos reproductores de ambos flujos y los ajustes que ha sufrido el proceso reproductor del flujo *esclavo*, como consecuencia de la aplicación del algoritmo propuesto. También se puede apreciar en ambas, el elevado ajuste inicial que sufre el proceso de reproducción del flujo *esclavo* (vídeo) debido a los motivos ya explicados.

Durante la sesión, los procesos de reproducción del flujo *esclavo* en los receptores PC_MAESTRO y PC1 sufren 163 y 76 ajustes, respectivamente, que, como se puede comprobar, son todos de más de 40 milisegundos (valor configurado en las aplicaciones como umbral de asincronía máxima permitida por las mismas antes de enviar un mensaje de sincronización a través de *mbus*). Tal y como se puede apreciar, las asincronías detectadas durante la sesión están, durante la mayor parte del tiempo, dentro de los límites de ±80 milisegundos esperados. De ahí que los valores cuadráticos medios de la misma sean menores que 6400 milisegundos^2. Además, como era de esperar, y como ya ha sido justificado anteriormente, los ajustes sufridos por el proceso reproductor del receptor *maestro* son inferiores a los del resto de receptores (aunque sólo se haya mostrado el comportamiento de PC1). En la tabla 6.12 se muestran los datos más representativos en cuanto a la sincronización inter-flujo.

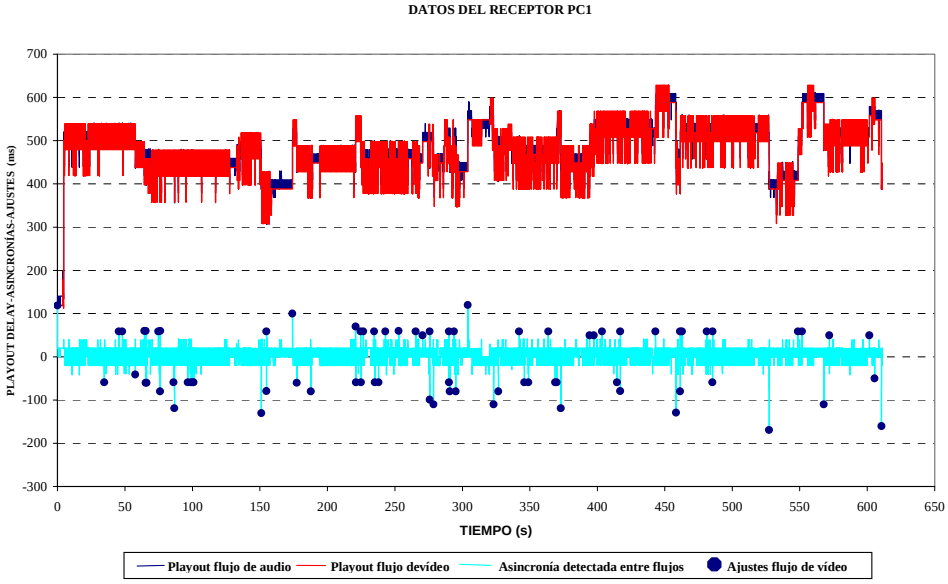


a) Datos reales

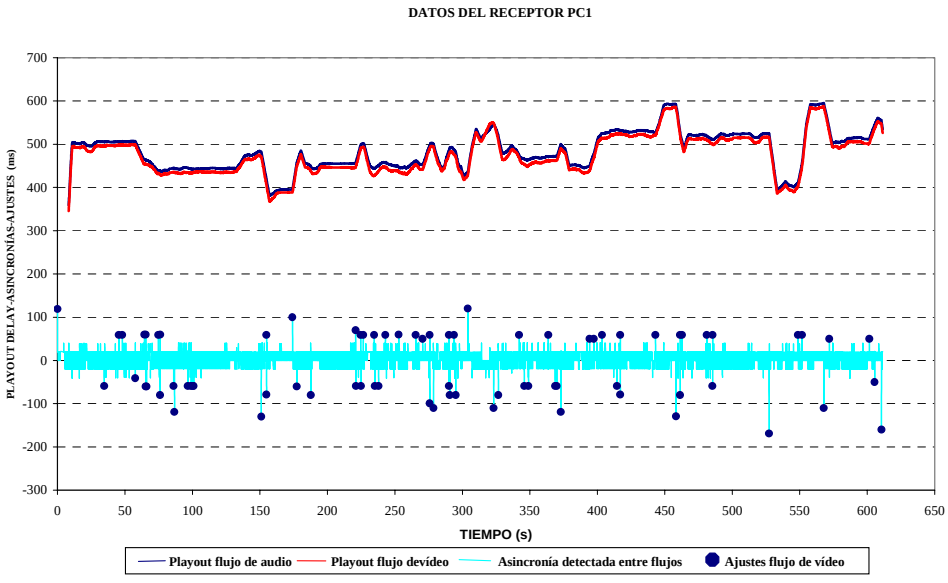


b) Media móvil (conjuntos de 100 muestras)

Figura 6.37. *Playout delay* de los dos flujos reproducidos por PC_MAESTRO, asincronía detectada y ajustes del reproductor del flujo *esclavo* (aprendizaje a distancia - CAMPUS - con algoritmo)



a) Datos reales



b) Media móvil (conjuntos de 100 muestras)

Figura 6.38. *Playout delay* de los dos flujos reproducidos por PC1, asincronía detectada y ajustes del reproductor del flujo *esclavo* (aprendizaje a distancia - CAMPUS - con algoritmo)

ESTADÍSTICAS RECEPTOR	**PC1**	**PC2**	**PC3**	**PC4**	**PC5**	**PC6**	**PC7**	**PC8**	**PC9**	**PC10** (MAESTRO)
Máx. asincronía positiva (ms)	120	110	160	129	100	119	110	110	80	60
Máx. asincronía negativa (ms)	-169	-170	-169	-199	-139	-229	-169	-139	-169	-119
Mínimo valor de la asincronía detectada.	0	0	0	0	0	0	0	0	0	0
Valor medio de la asincronía (ms)	8	11	13	11	5	9	5	4	5	11
Desviación estándar de las asincronías detectadas (ms)	18	21	17	19	18	19	19	19	20	19
Valor cuadrático medio de la asincronía (ms^2)[5]	387,37	542,05	441	483,25	361,31	451,97	379,18	385,59	430,29	459,12
Nº mensajes mbus enviados	76	367	84	177	78	180	134	126	159	163
Asincronía máxima audio-vídeo (±ms) configurada	Umbral Configurada: ±40[6]					Máxima permitida: ±80ms				
Nº asincronías audio-vídeo	13	11	14	23	12	15	9	10	13	12
'*Saltos*' realizados por el flujo de vídeo (*esclavo*) debido a la sincr. inter-flujo	72	186	73	120	69	109	101	104	136	101
'Pausas' realizadas por el flujo de vídeo (*esclavo*) debido a la sincr. inter-flujo [2]	46	282	56	131	49	137	93	79	85	136
Duración muestra flujo de vídeo (ms)	40									

Tabla 6.12. Datos obtenidos de todos los receptores
(aprendizaje a distancia - CAMPUS - con algoritmo)

[5] Para el cálculo de estos valores no se ha tenido en cuenta los ajustes debidos al inicio de las aplicaciones.

[6] Se ha seleccionado una asincronía máxima entre los procesos reproductores de audio y de vídeo, permitida desde las aplicaciones, de ±40 milisegundos, para que se inicien las acciones correctoras antes de que se supere el límite máximo permitido que será de ±80 milisegundos, lo cual se contabilizaría como una situación de asincronía.

Si nos fijamos en los valores (positivos y negativos) de máxima asincronía detectada en los receptores, vemos que todos han sufrido alguna situación de asincronía ya que se supera el límite de ±80 milisegundos. Además se puede apreciar que en algún momento los receptores PC1, PC2, PC3, PC6, PC7 y PC9 han superado el límite superior de ±160 milisegundos en el valor de asincronía. Esto sucede en momentos puntuales y la situación es rápidamente corregida por el algoritmo, tal y como se comprobará posteriormente. También se puede apreciar que el receptor *maestro* es el que menores valores máximos de asincronía detecta.

Cuando se detecta que la asincronía entre flujos supera un determinado umbral (como en el entorno LAN, en las aplicaciones se ha fijado un umbral de ±40 milisegundos), el proceso reproductor del flujo *maestro* envía un mensaje *mbus* al proceso reproductor del flujo *esclavo* para que corrija o ajuste su estado de reproducción y se recupere la sincronización. En la tabla anterior aparecen el número de mensajes *mbus* enviados. El receptor que más mensajes *mbus* ha enviado fue PC2, con 367 mensajes, mientras que el que menos mensajes envió fue el receptor PC1, con 76.

En la figura 6.39 se muestra la distribución de los ajustes que ha sufrido el proceso reproductor del flujo *esclavo* (de vídeo) de cada receptor para adaptarse al estado del proceso reproductor del flujo *maestro* (de audio). Puede observarse que los ajustes que superan los límites establecidos de ±80 milisegundos son mínimos en todos los receptores, así como que no existen ajustes por debajo de 40 milisegundos. La mayoría de los ajustes están en las franjas [-80, -40] y [40, 80] milisegundos, como era de esperar.

La superación del límite de ±80 milisegundos en cada receptor ha sido contabilizada como una situación de asincronía. En la tabla anterior aparece el número de asincronías detectadas en cada receptor. El receptor que en el que más asincronías se detectaron fue PC4 con 23, mientras que, por otra parte, el receptor que menos situaciones de asincronías detectó fue el receptor PC7, con 9 asincronías detectadas (menos, incluso, que las detectadas por el receptor maestro). Como ya se ha comentado, para corregir dichas situaciones de asincronías, los procesos reproductores del flujo *esclavo* de cada receptor han tenido que corregir su estado mediante acciones de '*saltos*' y '*pausas*'. El proceso reproductor del receptor que más '*saltos*' realizó durante la sesión fue el de PC2, dejando de reproducir 186 LDUs', mientras que el del receptor PC5 fue el que menos '*saltos*' sufrió, con 69 LDUs no reproducidas. Por otro lado, el proceso reproductor del receptor que más '*pausas*' ha sufrido durante la sesión fue el de PC2 con 282 repeticiones de una LDU anterior, mientras que el del receptor PC1, con 46, fue el que menos '*pausas*' sufrió. Cabe destacar que, en este caso, el receptor *maestro* no fue el que menos ajustes realiza en la reproducción del flujo *esclavo*, como se podría esperar.

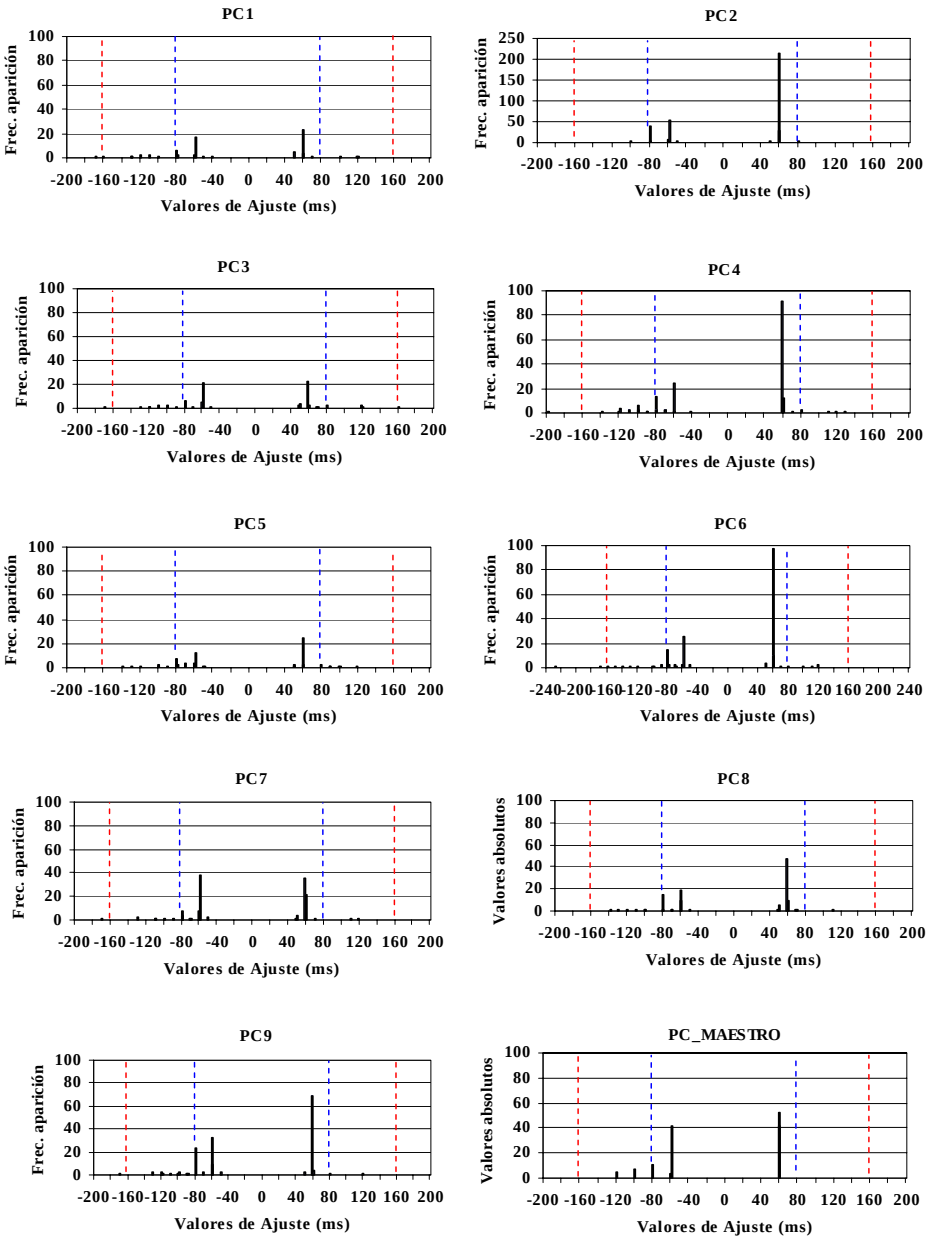


Figura 6.39. Número de veces que aparece un determinado ajuste en el proceso reproductor del flujo *esclavo*
(aprendizaje a distancia - CAMPUS - con algoritmo)

Al igual que para la sincronización de grupo, para evaluar la sincronización inter-flujo, un parámetro significativo es el valor cuadrático del error de sincronización. Si nos centramos en el valor cuadrático medio de las diferencias entre los estados de los procesos reproductores de ambos flujos, el algoritmo funcionará correctamente si es capaz de mantener dicho valor, es decir, el valor cuadrático medio de las diferencias detectadas entre los procesos reproductores de ambos flujos, por debajo del valor 6400 milisegundos^2 (correspondiente a una asincronía de ±80 milisegundos) o, en el peor de los casos, por debajo del valor 25600 milisegundos^2 (correspondiente a una asincronía de ±160 milisegundos). En la tabla 6.12 puede apreciarse que el mínimo valor medio del error cuadrático de la asincronía le corresponde al receptor PC5, con 361,31 milisegundos^2. Por otro lado el receptor con el valor medio más alto es el receptor PC2, con 542,05 milisegundos^2. Estos valores son ligeramente más altos que los obtenidos en el entorno LAN. Si comparamos la media de dichos valores, que es de 432,11 milisegundos^{2,} con el mismo valor obtenido en la evaluación en entorno LAN, que era de 310,75 milisegundos^2 (tal como aparece en la tabla 6.8), vemos que, en este caso, es 100 unidades superior, tal como era de esperar.

En la figura 6.40 se muestran una serie de gráficas con la distribución de los valores cuadráticos de los errores de sincronización detectados en las aplicaciones durante la sesión. Puede observarse que durante prácticamente la totalidad del tiempo de la sesión los valores permanecen por debajo de los 6400 milisegundos^2, superándose esta cantidad en muy pocas ocasiones. Aún así, durante la sesión evaluada, la cantidad de 25600 milisegundos^2 no es superada por 3 de los receptores (PC5, PC8 y PC_MAESTRO), es superada sólo en una ocasión por 5 receptores (PC1, PC3, PC4, PC7 y PC9) y en dos ocasiones por 2 receptores (PC2 y PC6).

A pesar de los instantes puntuales donde se supera la asincronía máxima aceptada por los usuarios, la *evaluación subjetiva* del algoritmo propuesto, ha sido favorable, calificando los ‘efectos extraños’ detectados como imperceptibles o perceptibles pero poco molestos, tal como se verá posteriormente, en el capítulo correspondiente a dicha evaluación.

En la tabla 6.13 se presentan los valores más representativos comentados.

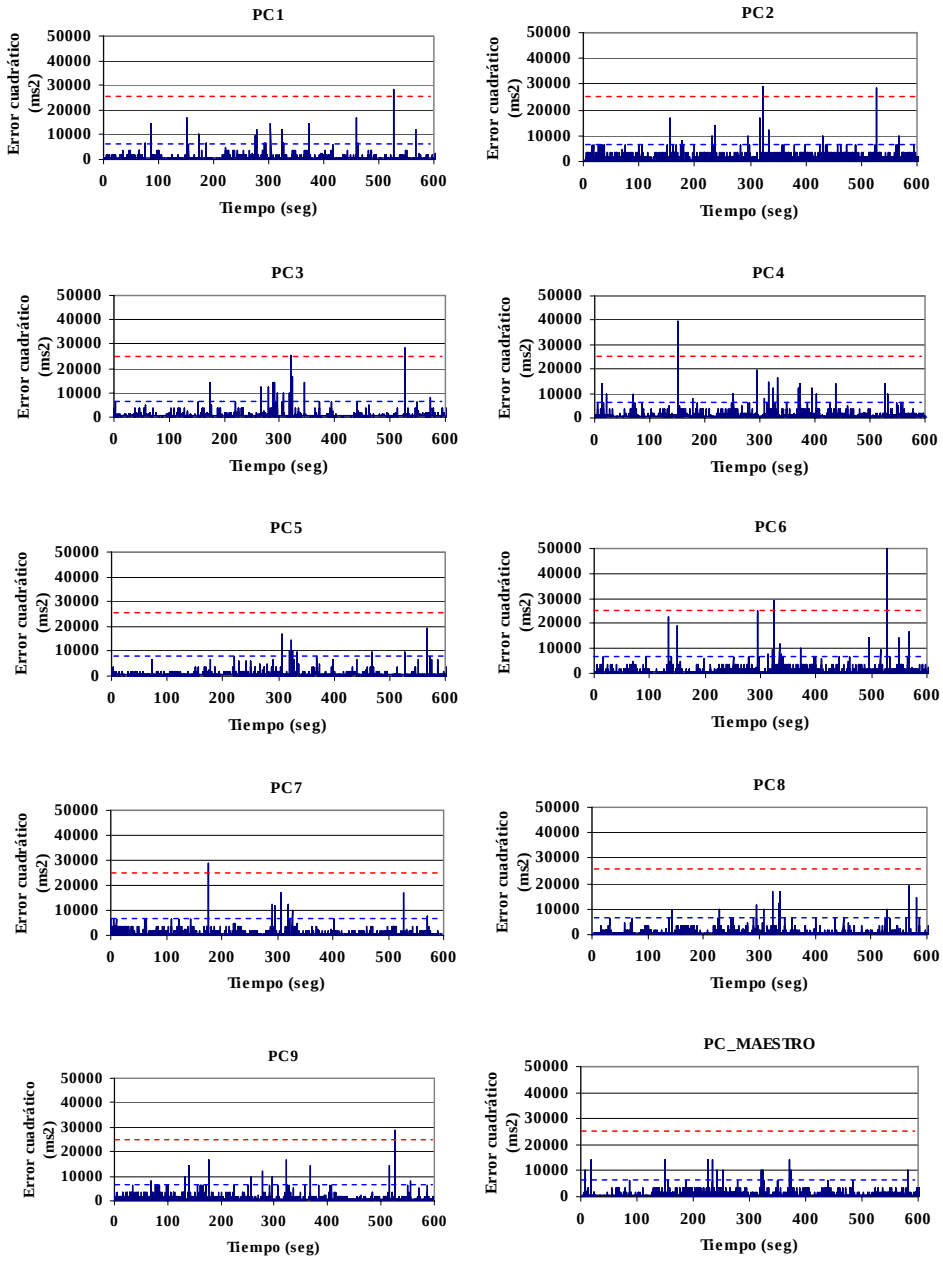


Figura 6.40. Valor cuadrático del error de sincronización inter-flujo (aprendizaje a distancia - CAMPUS - con algoritmo)

ESTADÍSTICAS RECEPTOR	VALORES MEDIOS
Máx. asincronía positiva detectada (ms)	160 (PC3)
Máx. asincronía negativa detectada (ms)	-229 (PC6)
Mínima asincronía detectada (ms)	0
Valor cuadrático medio máximo de la asincronía detectada en un receptor (ms^2)	542,05
Media de los valores cuadráticos medios	432,11
Nº máximo/mínimo de mensajes mbus enviados por un receptor	367 (PC2) 76 (PC1)
Asincronía máxima permitida audio-vídeo (± ms)	±80
Asincronía máxima configurada audio-vídeo (± ms)	±40
Número máximo/mínimo de asincronías audio-vídeo detectadas en un receptor	23 (PC4) 9 (PC7)
Nº máximo/mínimo de '*saltos*' realizados por el flujo de vídeo (*esclavo*) de un receptor debido a la sincr. inter-flujo	186(PC2) 69 (PC5)
Nº máximo de '*pausas*' realizadas por el flujo de vídeo (*esclavo*) de un receptor debido a la sincr. inter-flujo	282 (PC2) 46 (PC1)
Duración muestra flujo de vídeo (ms)	40

Tabla 6.13. Valores más representativos
(aprendizaje a distancia - CAMPUS - con algoritmo)

6.3.2. Evaluación y Resultados del Algoritmo de Sincronización obtenidos en la Aplicación de Televigilancia.

En este apartado se muestra la evaluación objetiva de la aplicación de televigilancia en los entornos LAN y CAMPUS, de dos formas: por un lado, sin ejecutar el algoritmo de sincronización propuesto y, por otro lado, ejecutando el algoritmo, para, así, poder comparar los resultados obtenidos.

Ya que en la aplicación de televigilancia se transmiten flujos independientes desde fuentes independientes, tal y como se explicó en el apartado 6.2.2, para poder obtener la sincronización de la reproducción de los flujos entre

los receptores (sincronización de grupo) en dicha aplicación será necesario simular la reproducción de uno de los flujos (elegido de entre los involucrados en la propia aplicación), de forma ficticia, en aquellos receptores en los que no corresponde su reproducción. En nuestro caso se ha escogido el flujo de audio, por ser más restrictivo, y será el considerado como flujo *maestro* o de referencia, tanto para la sincronización de grupo (distribuida), como, también, para la sincronización inter-flujo (local) en cada receptor.

Por tanto, para poder incluir el mecanismo de sincronización distribuido propuesto, en todos los receptores se tendrá que ejecutar algún tipo de herramienta que se encargue de garantizar dicha sincronización. Aprovechando las modificaciones realizadas en las herramientas utilizadas en la aplicación anterior, se han realizado una serie de modificaciones en la herramienta de audio (*rat*) para que sea ésta la encargada de mantener la sincronización. El escenario implementado se muestra en la siguiente figura:

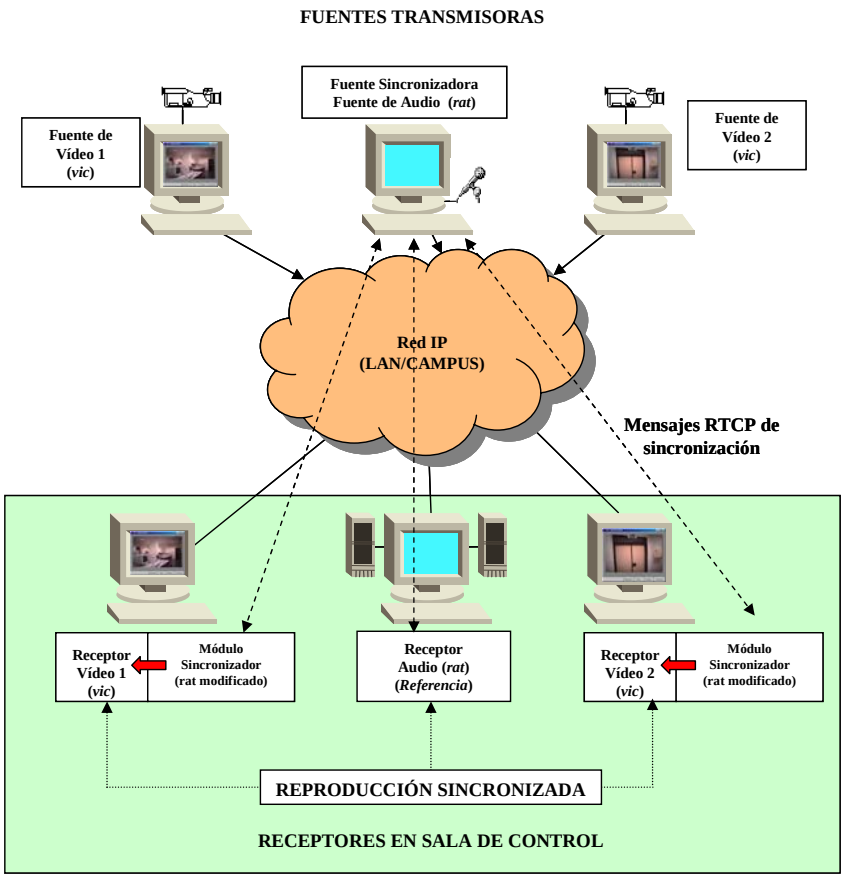


Figura 6.41. Escenario de la aplicación de Televigilancia

En este escenario tampoco se implementa ningún algoritmo de selección de un receptor como *maestro* sino que se fija como *receptor maestro* a *aquel que esté reproduciendo el flujo maestro*. Por tanto, el receptor *maestro* será el receptor del flujo de audio, que estará ejecutando la herramienta *rat*, y que será el que esté emitiendo el sonido por sus altavoces. Por otro lado, en los otros dos receptores se ejecutará la herramienta correspondiente al flujo *esclavo* (*vic*) y, además, como herramienta de sincronización de grupo, la *versión modificada* de la herramienta *rat* que servirá para la sincronización de grupo entre receptores. En adelante, a esta versión modificada de la herramienta *rat*, se la llamará '*rat modificado*'. La modificación realizada es muy sencilla y consiste, simplemente, en la eliminación de todo el procesamiento de los paquetes RTP de audio de la aplicación, dejando únicamente la ejecución del mecanismo de sincronización de grupo propuesto (y por lo tanto, no emitirá audio por los altavoces). A todos los efectos, se trata de un *módulo sincronizador de grupo* que se ejecuta en los receptores *esclavos*, cuya función será la de mantener sincronizada la reproducción de los flujos de vídeo con la del flujo de audio en el receptor *maestro*. De esta manera se aprovecha el trabajo realizado modificando el código fuente de la aplicación *rat* y no se ha tenido que desarrollar una aplicación específica para la sincronización de grupo.

En cada una de las fuentes se ha ejecutado la herramienta correspondiente, es decir, *rat* en la fuente transmisora del flujo de audio (que será la *fuente sincronizadora*) y *vic* en cada una de las fuentes transmisoras de flujo de vídeo. Por otro lado, en el receptor del flujo de audio (receptor *maestro* o de referencia para la sincronización de grupo) se ha ejecutado la herramienta *rat*, mientras que en los dos receptores de los flujos de vídeo 1 y 2, se ha ejecutado la herramienta *vic* junto con la herramienta sincronizadora ('*rat modificado*').

En este caso, mediante el algoritmo de sincronización de grupo entre receptores se conseguirá que todos ellos reproduzcan el flujo de audio de forma sincronizada (de forma real en uno de ellos y de forma ficticia en los demás). Por otro lado, mediante el algoritmo de sincronización inter-flujo (que sólo se ejecutará en los receptores de flujos de vídeo ya que en el receptor de audio sólo se reproduce un flujo), el proceso reproductor ficticio forzará la sincronización de la reproducción del flujo de vídeo, consiguiendo indirectamente la sincronización del mismo con el flujo de audio que está reproduciéndose realmente en el receptor de referencia.

Por tanto, como realmente es como si se reprodujera el flujo de audio en todos los receptores, se va a analizar objetivamente el comportamiento de los procesos reproductores de dicho flujo para evaluar la sincronización de grupo entre los receptores. Posteriormente, se evaluará la sincronización inter-flujo realizada entre los flujos de audio y vídeo en los dos receptores de vídeo, comprobando los errores de sincronización detectados. Por último se comprobará mediante el

análisis del *playout delay* de los 3 flujos de la aplicación el grado de sincronización conseguido entre sus procesos reproductores.

6.3.2.1. Entorno LAN

A continuación se muestra la evaluación realizada para la transmisión de los tres flujos multimedia (uno de audio y dos de vídeo) desde tres fuentes independientes, hacia tres receptores independientes situados en la misma red local (red de laboratorios del Campus de Gandia). Se transmitieron los flujos de audio y vídeo capturados en directo desde el laboratorio por dos cámaras de vídeo y un micrófono, cada uno conectado a un transmisor independiente (ordenador personal equipado con los dispositivos de captura adecuados).

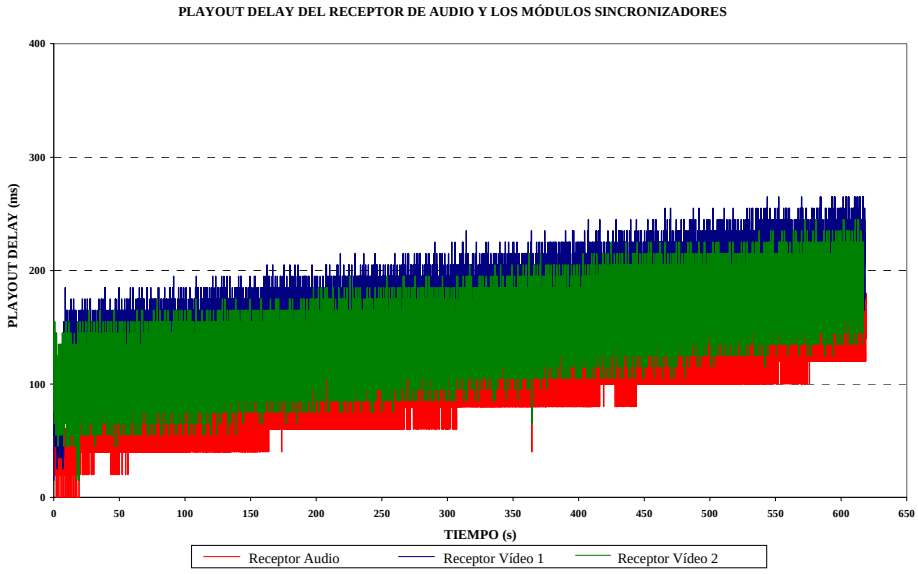
A) SIN ALGORITMO

En este caso, también se ha sincronizado los relojes de todos los equipos involucrados mediante NTP para poder disponer de la referencia de tiempos común con el fin de comparar los parámetros obtenidos durante la sesión, pero no se ejecuta ningún algoritmo de sincronización.

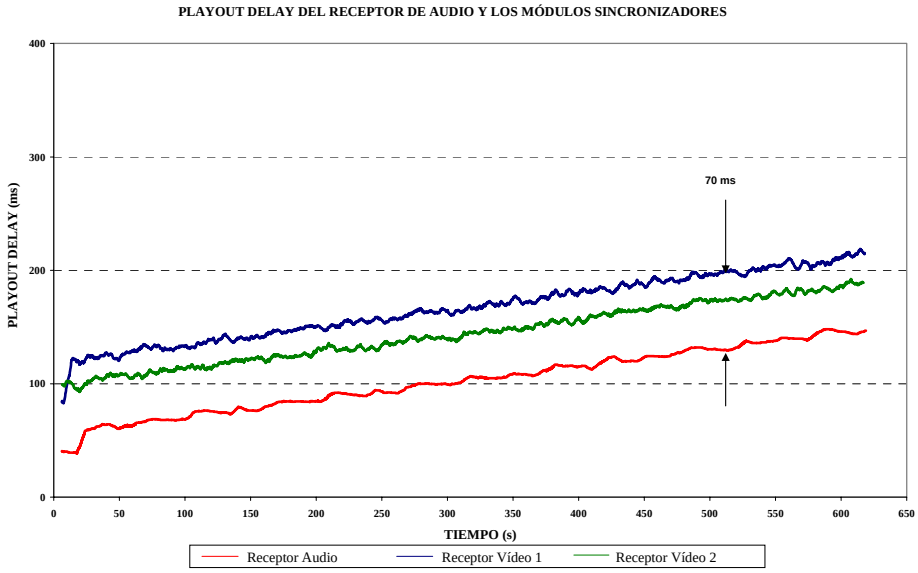
- ***Retardos de reproducción (playout delay) de los procesos reproductores del flujo de referencia (audio) en todos los receptores***

A continuación, en la siguiente figura se muestra el *playout delay* calculado por los procesos reproductores del flujo de audio y flujos de referencia. Con independencia de la versión de la herramienta ejecutada (*rat* completo o *'rat modificado'*), en todas ellas se calculará el *playout delay* que proporcionará una idea del grado de sincronización de grupo existente entre los receptores.

En ambas gráficas puede apreciarse que, durante la sesión, la diferencia entre los valores calculados por el receptor de audio y los calculados por los módulos sincronizadores de los receptores de vídeo, se mantiene entre 50 y 80 milisegundos, aumentando ligeramente a lo largo de la sesión. Con el tiempo se habrían superado valores de 120 milisegundos, tomados como valor máximo permitida en la aplicación anterior.



a) Valores reales



b) Media móvil (conjuntos de 100 muestras)

Figura 6.42. *Playout delay* calculado por el receptor de audio y por los módulos de sincronización
(televigilancia - LAN - sin algoritmo)

- ***Números de secuencia de las LDUs del flujo de audio(maestro) reproducidas (real o ficticiamente) en cada receptor***

Si nos fijamos en la primera LDU reproducida real o ficticiamente por los tres receptores, representada en la figura 6.43, vemos que existe ya una asincronía en el instante inicial de reproducción. Por un lado el receptor de audio y los receptores de vídeo reproducen la primera LDU (cuyo valor escogido por el transmisor para la misma es 55.755), con un desfase de 170 milisegundos.

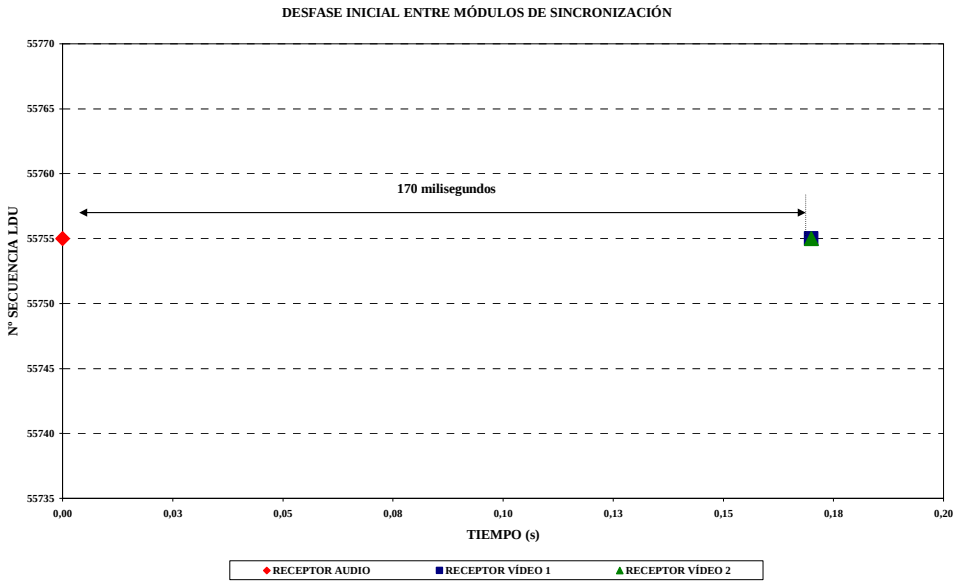


Figura 6.43. Primera LDU del flujo de audio reproducida
(real y ficticiamente)
(televigilancia - LAN - sin algoritmo)

Si observamos, ahora, los números de secuencia de las LDUs del flujo de audio reproducidas de forma real (receptor de audio) o de forma ficticia (receptores de vídeo), representados en la figura 6.44, vemos que en el intervalo de 1 segundo representado existe una asincronía entre el receptor de audio y los receptores de vídeo que está en torno a los 150 milisegundos (en este caso, se ha utilizado la opción por defecto de la aplicación *rat*, en la que las LDUs del flujo de audio contienen muestras correspondientes a una duración de 40 milisegundos). Se ha elegido un segundo del final de la sesión donde la asincronía se ve más claramente ya que, como se ha indicado anteriormente, ésta aumentaba a medida que avanzaba la sesión.

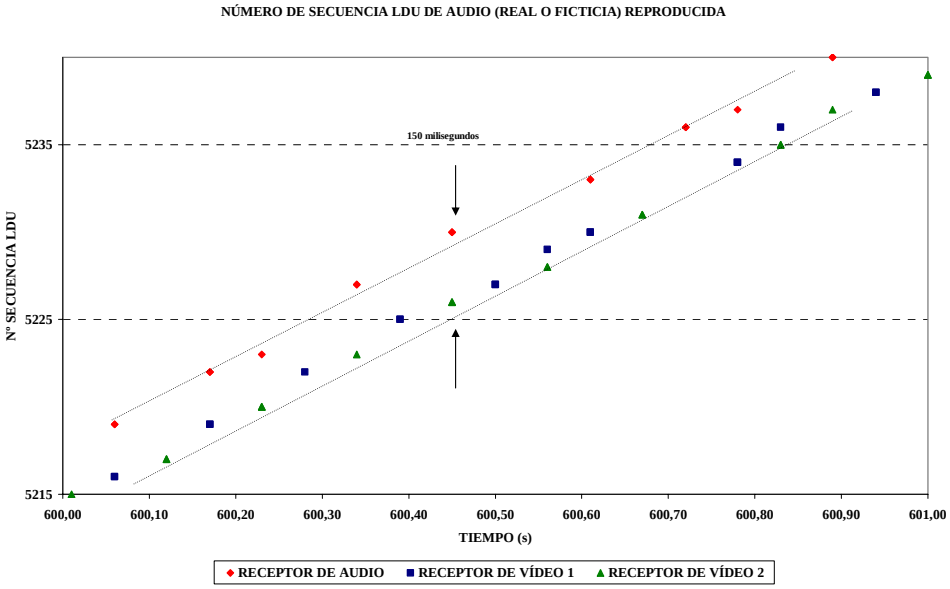


Figura 6.44. Número de secuencia de las LDUs de audio reproducidas (real y ficticiamente)
(televigilancia - LAN - sin algoritmo)

- ***Instante de inicio de la reproducción de los diferentes receptores***

También se ha comprobado que los diferentes receptores inician la reproducción de los tres flujos 'reales' (en cada receptor un flujo diferente) en diferentes instantes de tiempo.

En la siguiente figura se muestra el instante en que cada receptor reproduce el contenido del primer paquete RTP del flujo correspondiente. Se aprecia claramente que existe un desfase en el inicio de la reproducción que llega casi a los 10 segundos entre el inicio de la reproducción del flujo de audio en el receptor de audio y el inicio del flujo de vídeo 1, y un desfase de unos 6,1 segundos entre el inicio de la reproducción del flujo de audio y el inicio de la reproducción del flujo de vídeo 2.

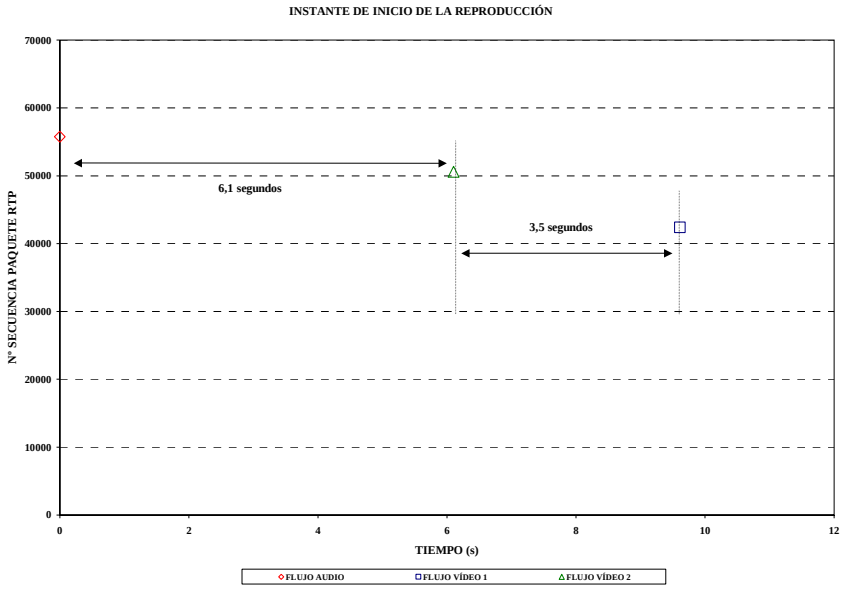


Figura 6.45. Instante de inicio de la reproducción de los tres flujos (televigilancia - LAN - sin algoritmo)

Con los resultados obtenidos podemos afirmar, por tanto, que, en ausencia de algoritmo de sincronización, existe una asincronía en los procesos de reproducción de los tres flujos involucrados en la aplicación de televigilancia implementada.

B) CON EL ALGORITMO DE SINCRONIZACIÓN DE GRUPO

En este caso se ha repetido la prueba anterior pero esta vez activando las opciones de sincronización en las herramientas utilizadas. Para esta aplicación no se ha configurado ningún límite de asincronía máxima permitida entre flujos de audio ya que, realmente, sólo existe un receptor que va a reproducir audio (emitiendo sonido). Además, en las aplicaciones se ha configurado el umbral superior de ±80 milisegundos como máxima asincronía permitida entre los procesos ficticios de reproducción de audio de los módulos sincronizadores (‘*rat modificado*’) y los procesos reproductores de vídeo (*vic*), para evitar que se estén constantemente enviando mensajes *mbus* internamente en los dos receptores de flujo de vídeo.

Como se ha comentado, se ha tomado como *receptor maestro* de referencia al receptor del flujo de audio para la sincronización de grupo (distribuida) y al flujo

de audio ficticio como *flujo maestro* para la sincronización inter-flujo en los receptores de vídeo. Los procesos reproductores del flujo de vídeo en los receptores *esclavos* recibirán mensajes *mbus* procedentes del módulo de sincronización (‘*rat modificado*’) y se ajustarán al estado que dichos módulos les indiquen. Los módulos de sincronización recibirán mensajes RTCP de ‘acción’ (APP ACT) del servidor que traducirán en mensajes *mbus* hacia los procesos reproductores del flujo de vídeo del receptor.

Por tanto, en esta aplicación existirán los dos procesos de sincronización que incluye el algoritmo propuesto: por un lado, el de sincronización de grupo (integrado en la herramienta *rat* en el receptor de audio y los módulos de sincronización en los receptores de vídeo) y, por otro lado, el de sincronización inter-flujo (integrado en los módulos de sincronización y la herramienta *vic*).

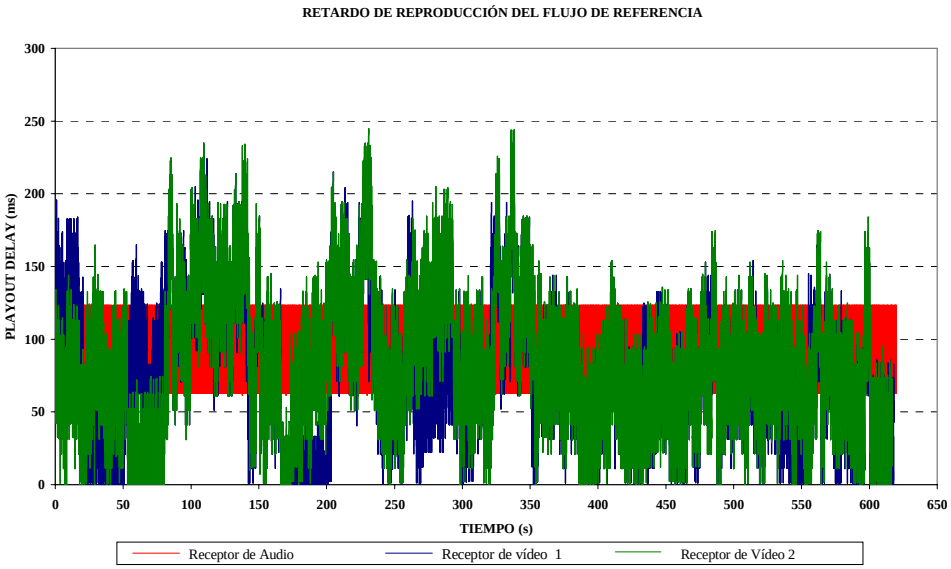
En primer lugar, se van a presentar gráficas del funcionamiento de la sincronización de grupo distribuida y la sincronización del instante inicial de consumo, con respecto al flujo de referencia. En segundo lugar se evaluará la sincronización inter-flujo entre el flujo ficticio de referencia y el flujo de vídeo, localmente, en cada uno de los receptores de flujo de vídeo. En tercer lugar, se evaluará el grado de sincronización en la reproducción de los tres flujos reales, cada uno reproducido en un receptor independiente.

B1. Sincronización de Grupo (distribuida)

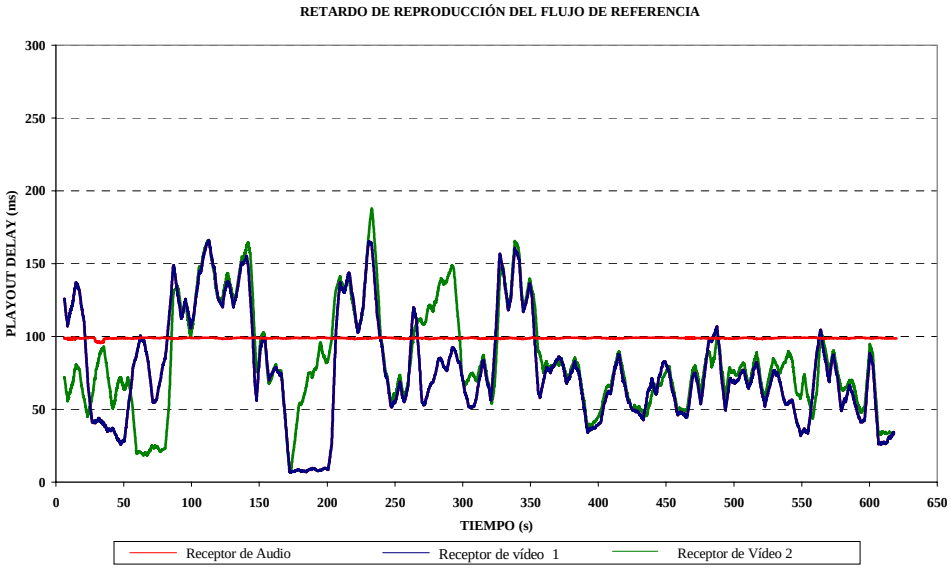
- ***Retardos de reproducción (playout delay) de los procesos reproductores del flujo de referencia de todos los receptores***

A continuación se muestra la figura 6.46, que representa el retardo de reproducción del flujo de audio experimentado por los 3 receptores. En la figura 6.46b representa la *media móvil* de la serie de datos representada en la figura 6.46a, a partir de conjuntos de 100 muestras. En ambas figuras se representa el tiempo completo de la sesión evaluada, de unos 10 minutos, aproximadamente.

Puede observarse que los retados de reproducción de los flujos ficticios de audio en los receptores de vídeo se van adaptando en determinados puntos al retardo de reproducción del receptor de audio tomado como referencia o receptor *maestro* (en rojo). En la figura 6.47, se desglosa la gráfica anterior para los dos receptores de vídeo incluyendo los ajustes que sufre el proceso reproductor ficticio de dicho receptor, debido a las operaciones de ‘*saltos*’ y ‘*pausas*’ efectuadas por las acciones de resincronización del algoritmo propuesto.

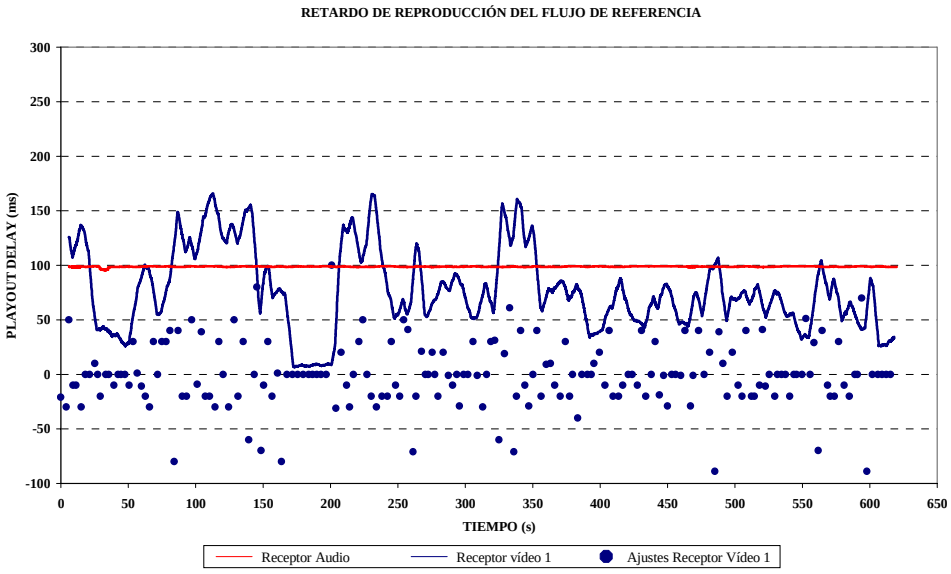


a) Valores reales

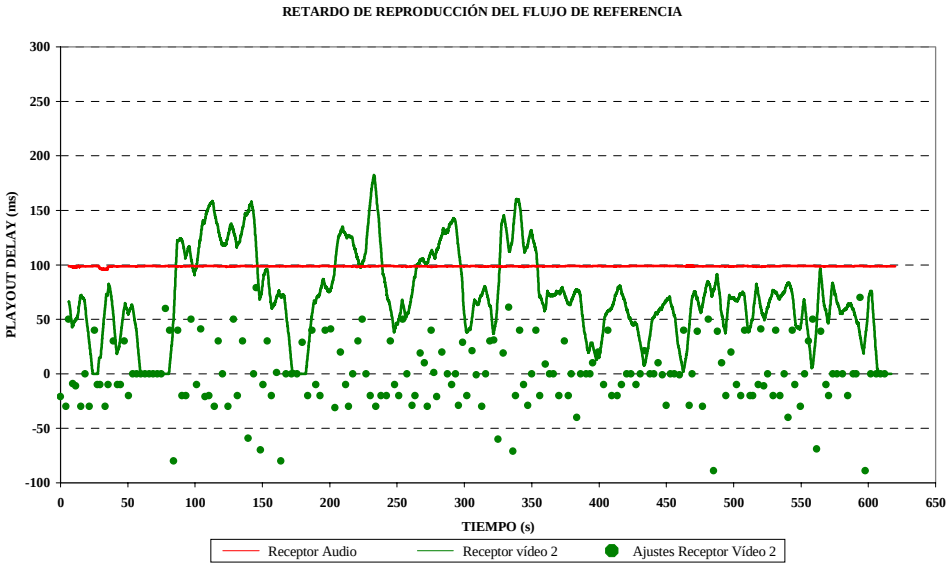


b) Media móvil (conjuntos de 100 muestras)

Figura 6.46. *Playout delay* del flujo de referencia de los receptores (televigilancia - LAN - con algoritmo)



a) Receptor de vídeo 1



b) Receptor de vídeo 2

Figura 6.47. *Playout delay* del flujo *maestro de* referencia en cada receptor de vídeo. Ajustes en el receptor de vídeo
(televigilancia - LAN - con algoritmo)

El proceso reproductor del flujo *maestro* (audio) en el receptor de referencia (el receptor del flujo de audio) no sufre ajustes, ni debidos a la sincronización de grupo ni a la sincronización inter-flujo, por lo que su estado de reproducción variará menos, tal como se comprueba en las figuras anteriores.

A continuación, se muestran los valores de los parámetros evaluados que son prácticamente los mismos que en la aplicación de aprendizaje a distancia, con algunas matizaciones:

- *Máxima asincronía positiva y negativa detectada*: valores máximos positivo y negativo de la diferencia detectada entre el retardo de reproducción del flujo de audio (*maestro*) del receptor de audio (referencia) y los calculados por los módulos sincronizadores en los dos receptores de vídeo.
- *Valor mínimo de la asincronía detectada.* Valor absoluto mínimo de todos los valores de asincronía detectados por los procesos reproductores de audio (reales o ficticios).
- *Valor medio y desviación estándar de la asincronía detectada*. Valor medio y desviación estándar de la asincronía detectada entre los procesos reproductores del flujo de audio ficticio (módulos de sincronización) y el proceso del receptor de audio real. El valor medio será un valor orientativo ya que existen valores positivos y negativos que, al calcular el valor medio, se anulan unos a otros.
- *Valor cuadrático medio de la asincronía detectada*. Representa la media del cuadrado de los valores de asincronía detectados entre los procesos reproductores de audio ficticio (módulos de sincronización) y el proceso del receptor de audio.
- *Número de mensajes enviados por la fuente sincronizadora*. Número de mensajes de 'acción' (APP ACT) enviados por la fuente sincronizadora.
- *Número de mensajes de sincronización 'gruesa' adicionales enviados por el servidor*. Es el número de paquetes APP TIN que la fuente sincronizadora envía para garantizar que no se llegue a situaciones de *overflow* de los *buffers* de recepción, incluyendo el paquete APP TIN inicial, cuyo objetivo es el de garantizar que todos los receptores inicien la reproducción en el mismo instante de tiempo.
- *Número de mensajes recibidos del servidor*. Número de mensajes de 'acción'(APP ACT) recibidos de la fuente sincronizadora. Debido a las pérdidas es posible que no coincida con el valor de mensajes del mismo tipo enviados por el servidor.
- *Número de mensajes Feedback enviados por el receptor*. Representa el número de paquetes con información de realimentación (*feedback*) que el receptor ha enviado al servidor (número de mensajes RR Extendidos enviados por el receptor).

- *Asincronía máxima permitida audio ficticio-audio real del receptor de audio (receptor maestro)*. Valor configurado desde la *interfaz gráfica de usuario* de la fuente sincronizadora que indica el valor máximo de asincronía, en milisegundos, entre los procesos reproductores del flujo *maestro* (en este caso, de audio) de los diferentes receptores.
- *'Saltos' realizados por el flujo de audio ficticio*: Número total de LDUs del flujo de audio ficticio que no se habrían consumido en los receptores, debido al proceso de sincronización de grupo.
- *'Pausas' realizadas por el flujo de audio (maestro)*: Número total de LDUs del flujo de audio ficticio, que en caso de reproducirse en los receptores de vídeo serían una repetición de una LDU ya consumida, debido al proceso de sincronización de grupo.
- *Número de paquetes RTP enviados.* Número de paquetes RTP conteniendo muestras del flujo de audio enviados por la fuente de audio (fuente sincronizadora) durante la sesión evaluada.
- *Duración muestra*. Intervalo de tiempo cuyas muestras se incluyen en cada paquete RTP de audio. En este caso, en cada paquete RTP se ha codificado el flujo audio correspondiente a 40 milisegundos.

Los resultados obtenidos en la sesión evaluada se muestran en la tabla 6.14.

En este caso, el servidor envía 1 sólo paquete APP TIN al comienzo de la sesión, para indicar el instante de inicio de la reproducción (*Instante Inicial de Consumo*) y, posteriormente, a lo largo de la misma, envía 191 paquetes de 'acción' APP ACT, indicando acciones de resincronización en los receptores, frente a los 15500 paquetes RTP de datos del flujo de audio (que contienen muestras de audio codificadas correspondientes a un fragmento de audio de 40 milisegundos de duración). Este número de mensajes de control es mayor que el de mensajes enviados en el tipo de aplicación anterior. Ello es debido a que, para esta aplicación, no se ha fijado un umbral de asincronía máxima tolerada entre procesos reproductores de audio en cada receptor antes de enviar mensajes de acción por parte de la fuente sincronizadora.

Los mensajes de control enviados por la fuente sincronizadota (1 APP TIN y 191 APP ACT, es decir, 192 mensajes de control RTCP) suponen un 1,2 % del total de los paquetes transmitidos (de datos y de control) considerando únicamente el flujo de audio.

Por otro lado, los mensajes de control enviados por los receptores (en total 614 mensajes RTCP RR extendidos) apenas suponen un 3,7 % del total de los paquetes transmitidos (de datos y de control) por las aplicaciones de audio durante la sesión y un poco menos del 4 % con respecto al número total de paquetes de datos enviados (15500 paquetes RTP de audio).

ESTADÍSTICAS RECEPTOR	**RECEPTOR VÍDEO 1**	**RECEPTOR VÍDEO 2**	**RECEPTOR AUDIO (MAESTRO)**
Máxima asincronía positiva detectada (ms)	79	100	-
Mínima asincronía negativa detectada (ms)	-89	-89	-
Mínima asincronía detectada (ms)			
Valor medio de la asincronía detectada (ms)	-0,79	-0,79	-
Desviación estándar de las asincronías detectadas (ms)	29,63	29,40	-
Valor cuadrático medio de la asincronía detectada (ms^2)	874	860	-
Nº mensajes de '*acción*' (APP TIN) enviados por la fuente sincronizadora	1		
Nº mensajes de '*acción*' (APP ACT) enviados por la fuente sincronizadora	191		
Nº mensajes recibidos del servidor (APP ACT)	191	191	191
Número de mensajes *Feedback* enviados por el receptor (RR Extendidos)	205	208	201
Asincronía máxima audio-audio *permitida y configurada,* con respecto al receptor *maestro*	0 ms		
'*Saltos*' realizados por el flujo de audio (*maestro*) debido a la sincr. de grupo	40	35	-
'*Pausas*' realizadas por el flujo de audio (*maestro*) debido a la sincr. de grupo	49	45	-
Nº de paquetes RTP enviados	15500		
Duración muestra flujo de audio (ms)	40		

Tabla 6.14. Datos obtenidos de los tres receptores (televigilancia - LAN - con algoritmo)

El valor del parámetro *Valor cuadrático medio de la asincronía detectada*, proporciona una idea del grado de sincronización de grupo obtenida. Se puede observar que los valores de ambos receptores son similares, de 874 y 860 milisegundos [2], por debajo de 14400 milisegundos^2 (correspondiente a valores de asincronía de ±120 milisegundos).

Para mantener la sincronización de grupo (distribuida), puede observarse en la tabla que, en los receptores de vídeo, el algoritmo propuesto ha provocado acciones de resincronización que se traducen en *'saltos'* y *'pausas'* en el proceso ficticio de reproducción del flujo de audio. El receptor que más discontinuidades en su proceso de reproducción ficticia ha experimentado ha sido el receptor de vídeo 1 con 40 *'saltos'* y 49 *'pausas'* (frente a los 35 'saltos y 45 '*pausas*' que experimentó el proceso del receptor de vídeo 2.

A continuación, en la figura 6.48 aparece la distribución de los ajustes que han sufrido los procesos reproductores del flujo *ficticio de referencia* de los receptores de vídeo, como consecuencia de la recepción de paquetes de '*acción*' (APP ACT) enviados por la fuente sincronizadora. Como se ha comentado, los ajustes reflejan los errores de sincronización entre los procesos reproductores ficticios (módulos sincronizadores) y el proceso reproductor del receptor de audio. Dicho error influirá posteriormente en la sincronización entre los tres flujos reproducidos realmente en la aplicación (uno de audio y dos de vídeo).

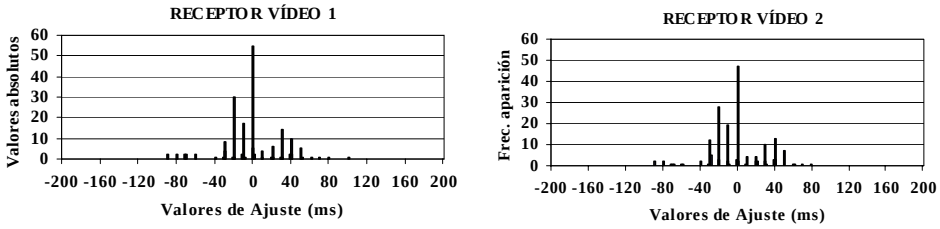


Figura 6.48. Distribución de los valores de ajuste del proceso de reproducción ficticio de cada receptor (televigilancia - LAN - con algoritmo)

En la figura 6.49 se muestran las gráficas de cada receptor con el valor cuadrático de dichos ajustes a la largo de la sesión. Se puede observar que se trata de unos valores bastante bajos (por debajo de 14400 milisegundos^2).

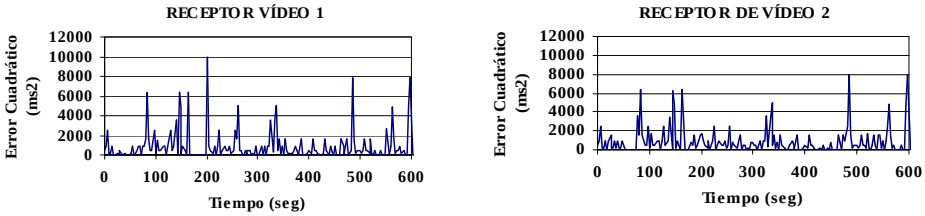


Figura 6.49. Valor Cuadrático del Error de Sincronización de Grupo (televigilancia - LAN - con algoritmo)

Del análisis de estos datos se puede concluir que, en entorno LAN, *el mecanismo de sincronización de grupo de flujos multimedia ha funcionado correctamente en la aplicación de televigilancia, a lo largo de la sesión evaluada.*

- ***Número de Secuencia de las LDUs del flujo de referencia reproducidas en cada receptor (real o ficticiamente)***

Como ya se ha indicado anteriormente, la mejor forma de comprobar el buen funcionamiento del algoritmo propuesto es la de verificar el número de secuencia de la LDU del flujo *maestro* que reproduce cada proceso reproductor de dicho flujo, en cada uno de los tres receptores, en cada instante de tiempo.

En primer lugar, se ha comprobado que el instante de inicio de la reproducción del flujo de referencia coincide en todos los receptores ya que todos los receptores comenzaron la reproducción de la primera LDU transmitida (cuyo número de secuencia elegido aleatoriamente por la fuente fue 16584) en el mismo instante de tiempo, tal y como se muestra en la figura 6.50.

Podemos concluir, pues, que, también en este caso, *el mecanismo de sincronización del instante inicial de consumo ha funcionado correctamente.*

En la figura 6.51, se han representado datos de 40 muestras obtenidas durante la sesión, superpuestas, tomando diferencias relativas, de tiempos (eje *x*) y de número de LDU reproducida en cada instante (eje *x* y eje *y*, respectivamente), tal y como se hizo anteriormente. Como se han obtenido 191 muestras durante la sesión de evaluación, para que se pueda visualizar mejor la información se ha representado una muestra de cada 5 obtenidas.

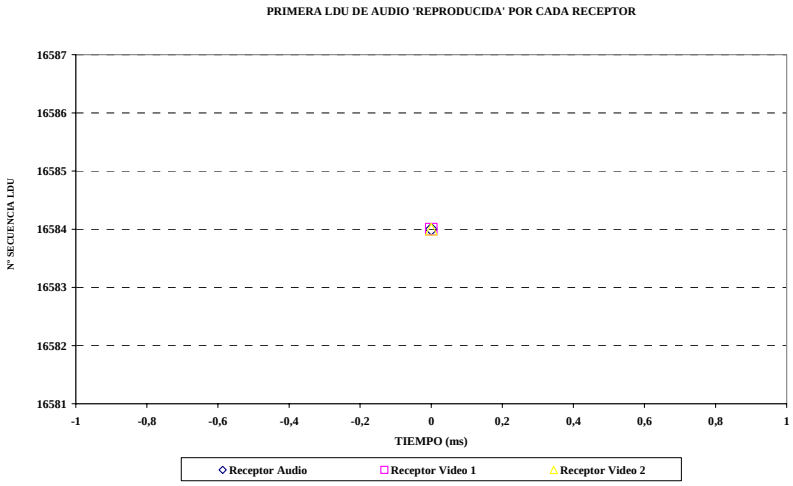


Figura 6.50. Primera LDU del flujo *maestro* de referencia reproducida (real o ficticiamente)
(televigilancia - LAN - con algoritmo)

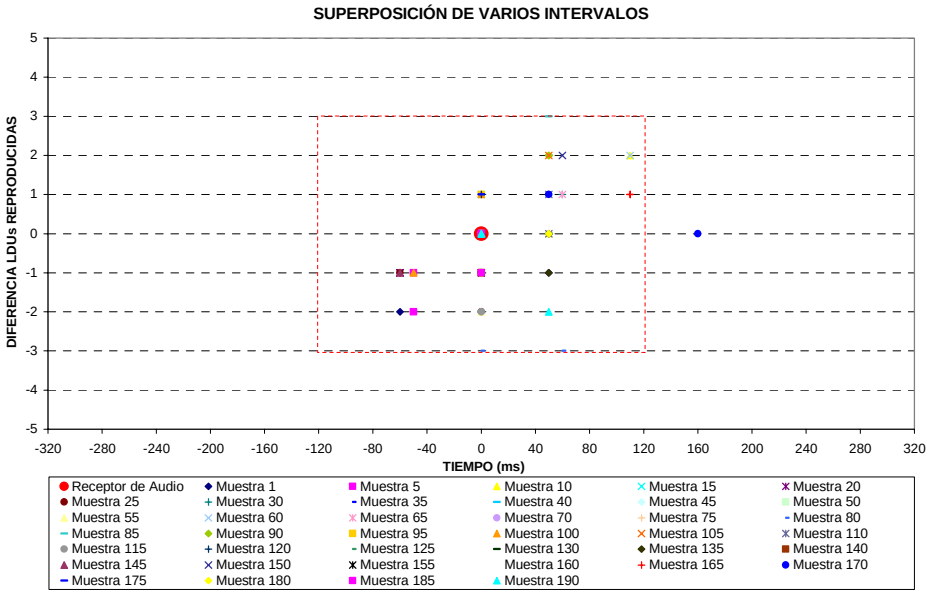


Figura 6.51. Superposición de 40 muestras
(televigilancia - LAN - con algoritmo)

En la total, en la figura hay representados 121 puntos correspondientes a la muestra reproducida por el receptor de audio (en rojo, centrada en el punto (0,0)) y las muestras reproducidas, de forma ficticia, por los receptores de vídeo en los 40 instantes muestreados. Se puede observar en la misma que, en las muestras representadas, a excepción de un punto, el resto están dentro del cuadro rojo (que representa una desviación de ±120 milisegundos), y que los puntos están repartidos por dentro del mismo, indicando que los puntos de reproducción ficticios de los receptores de vídeo oscilan alrededor del punto de reproducción del receptor de audio en cada instante.

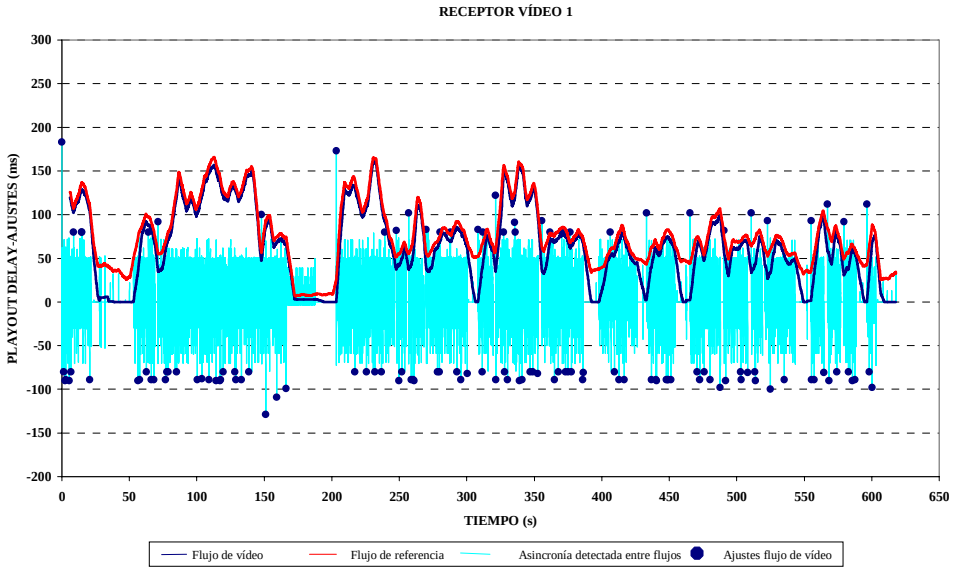
B2. Sincronización inter-flujo (local) en cada receptor de vídeo

Como se ha comentado, la sincronización inter-flujo sólo se realizará localmente en los receptores de los flujos de vídeo que es donde se reproducen dos flujos, el de vídeo y el de referencia (aunque la reproducción de este último sólo se realice de forma ficticia)

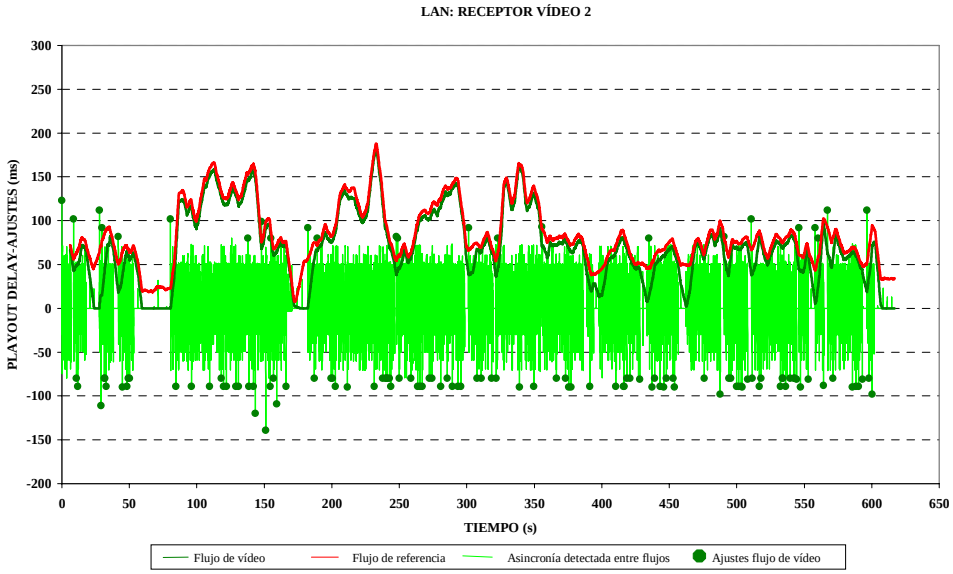
- ***Retardo de reproducción (playout delay) de los procesos reproductores de los flujos maestro (de referencia) y esclavo***

Una manera de ver si los dos flujos (el de referencia y el de vídeo) se reproducen simultáneamente, será analizando los retardos de reproducción (*playout delay*) de ambos en cada instante. A pesar de los aspectos comentados anteriormente, podemos utilizar el parámetro *playout delay* para observar si están sincronizados o no los procesos reproductores de los flujos evaluados en cada receptor.

A continuación se muestra el comportamiento de cada uno de los receptores de vídeo. En la figura 6.52, se muestra la media móvil del *playout delay* de ambos flujos, correspondientes a los dos receptores de vídeo, tomada a partir de conjuntos de 100 muestras de los datos almacenados. También se muestra la asincronía detectada entre el flujo principal (de vídeo) y el flujo de referencia ficticio (de audio) y los ajustes que ha sufrido el proceso reproductor del flujo de vídeo, como consecuencia de las acciones de resincronización provocadas por el algoritmo propuesto.



a) Receptor de vídeo 1



b) Receptor de vídeo 2

Figura 6.52. Media móvil (conjuntos de 100 muestras) del *Playout delay* de los dos flujos reproducidos por los receptores de vídeo, asincronía detectada y ajustes del proceso reproductor del flujo *esclavo*
(televigilancia - LAN - con algoritmo)

Durante la sesión los procesos de reproducción del flujo de vídeo en ambos receptores sufren 128 y 126 ajustes (en los receptores de vídeo 1 y vídeo 2, respectivamente), todos ellos de más de 80 milisegundos (pero la mayoría en torno a este valor) ya que, para esta aplicación, éste es el valor configurado en las herramientas como asincronía máxima permitida entre flujo *maestro* y flujo *esclavo*, antes de enviar un mensaje de sincronización a través de *mbus*.

A continuación, se van a mostrar los valores de los parámetros más representativos, relacionados con el proceso de sincronización inter-flujo, obtenidos de los dos receptores de vídeo, que son los siguientes:

- *Máxima asincronía positiva y negativa detectada*: Máximo valor positivo y negativo de la diferencia detectada entre el retardo de reproducción (*playout delay*) del flujo *maestro* (de audio) y el retardo de reproducción del flujo *esclavo* (de vídeo).
- *Mínima asincronía detectada*: Mínimo valor, en valor absoluto, de la diferencia detectada entre el retardo de reproducción del flujo *maestro* (de audio) y el retardo de reproducción del flujo *esclavo* (de vídeo).
- *Valor medio y desviación estándar de la asincronía detectada*. Valor medio y desviación estándar de la asincronía detectada entre ambos procesos reproductores.
- *Valor cuadrático medio de la asincronía detectada*. Representa la media del cuadrado de los valores de asincronía detectados entre ambos procesos reproductores.
- *Número de mensajes mbus enviados*. Número total de mensajes *mbus* enviados por el proceso reproductor del flujo *maestro* al proceso reproductor del flujo *esclavo*.
- *Asincronía máxima permitida audio-vídeo por la aplicación*. Valor configurado desde la *interfaz gráfica de usuario* de las aplicaciones que indica el valor máximo de asincronía entre los flujos de audio y vídeo permitidos por las mismas en el receptor.
- *'Saltos' realizados por el flujo de vídeo (esclavo)*: Número total de LDUs del flujo *esclavo* que no se han consumido, debido al proceso de sincronización inter-flujo.
- *'Pausas' realizadas por el flujo de vídeo (esclavo)*: Número total de LDUs del flujo *esclavo* cuya reproducción es una repetición de una LDU ya consumida, debido al proceso de sincronización inter-flujo.
- *Duración muestra del flujo esclavo*. Intervalo de tiempo cuyas muestras se incluyen en cada paquete RTP. En nuestro caso se envían 25 tramas por segundo con lo que cada trama tendrá una duración de 40 milisegundos.

Los resultados se muestran en la tabla 6.15.

ESTADÍSTICAS RECEPTOR	**RECEPTOR VÍDEO 1**	**RECEPTOR VÍDEO 2**
Máxima asincronía detectada (milisegundos)	183	123
Mínima asincronía detectada (milisegundos)	-129	-139
Valor medio de la Asincronía detectada (milisegundos)	11,73	10,78
Desviación estándar de las asincronías detectadas (milisegundos)	58,09	57,54
Valor cuadrático medio de la asincronía detectada (miliseg^2) **(7)**	874,14	940,1
Nº mensajes mbus enviados	128	126
Asincronía Máxima audio-vídeo (±milisegundos) *permitida y configurada* en las aplicaciones	±80	
Número Asincronías Audio-Vídeo	128	126
'*Saltos*' realizados por el flujo de vídeo (*esclavo*) debido a la sincronización inter-flujo	199	225
'*Pausas*' realizadas por el flujo de vídeo (*esclavo*) debido a la sincronización inter-flujo	76	53
Duración Muestra flujo de vídeo (milisegundos)	40	

Tabla 6.15. Datos obtenidos de ambos receptores
(televigilancia - LAN - con algoritmo)

En la tabla aparecen el número de mensajes *mbus* intercambiados entre el proceso reproductor ficticio (módulo sincronizador) y el proceso reproductor de vídeo, en ambos receptores. En el receptor de vídeo 1 se han enviado 128 mensajes *mbus*, mientras que en el receptor de vídeo 2 se han enviado 126, coincidiendo con el número de situaciones de asincronía detectadas.

[7] Para el cálculo de este valor no se ha tenido en cuenta el ajuste inicial debido al inicio de las aplicaciones en cada receptor.

En la figura 6.53 se muestra la distribución de los ajustes que ha sufrido el proceso reproductor del flujo de vídeo de cada receptor para adaptarse al estado del proceso reproductor del flujo de audio ficticio que es el tomado como *maestro* por el algoritmo. Puede observarse que los ajustes que superan los límites establecidos de ±80 milisegundos son mínimos en todos los receptores, así como que no existen ajustes por debajo de 80 milisegundos (por el umbral fijado comentado anteriormente). La mayoría de ajustes realizados son de valores cercanos a ±80 milisegundos.

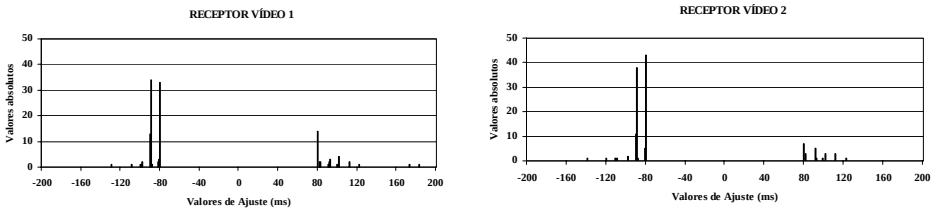


Figura 6.53. Distribución de los ajustes en los procesos reproductores de vídeo (televigilancia - LAN - con algoritmo)

Cabe destacar que existen más ajustes negativos 8por debajo de -80 milisegundos), indicando 'saltos' de esa cantidad de tiempo en el proceso reproductor de vídeo que se encontraría retrasado con respecto al de audio ficticio.

La superación del límite de ±80 milisegundos en cada receptor ha sido contabilizada como una situación de asincronía. En la tabla anterior aparece el número de asincronías detectadas en cada uno de ellos. En ambos se produce un número similar de asincronías: el receptor de vídeo 1 sufre 128 situaciones de asincronía, mientras que el receptor de vídeo 2 sufre 126.

Para corregir dichas situaciones de asincronías, los procesos reproductores del flujo *esclavo* de cada receptor han tenido que modificar su estado de reproducción mediante acciones de *'saltos'* y *'pausas'*. El proceso reproductor de vídeo del receptor de vídeo 1 realizó 199 *'saltos'*, dejando de reproducir dicho número de LDUs, y 76 *'pausas'*, reproduciendo en otras tantas ocasiones una LDU anterior, mientras que el proceso reproductor de vídeo del receptor de vídeo 2 realizó 225 *'saltos'* y 53 *'pausas'*. La mayoría de estos '*saltos*' y '*pausas*' fueron de, aproximadamente, dos LDUs (ya que la mayoría de ajustes están cercanos a ±80 milisegundos).

Al igual que para la sincronización de grupo, para el caso de la sincronización inter-flujo el parámetro más representativo será el valor cuadrático del error de sincronización. Si nos centramos en el valor cuadrático medio de las diferencias entre los estados de los procesos reproductores de ambos flujos, puede apreciarse que el valor medio del error cuadrático de la asincronía en los receptores

de vídeo 1 y 2 es de 874,14 y 940,1 milisegundos^2, respectivamente, muy por debajo de 6400 milisegundos^2 (valor correspondiente a errores de ±80 milisegundos).

En la siguiente figura se muestran las dos gráficas con la distribución de los valores cuadráticos de los errores de sincronización detectados en los dos receptores de vídeo durante la sesión. Prácticamente la totalidad del tiempo los valores permanecen por debajo de los 25600 milisegundos^2 (correspondientes a errores de ±160 milisegundos), superándose esta cantidad en una ocasión por el receptor de vídeo 1. En este caso se supera más veces el umbral inferior de 6400 milisegundos^2. No obstante, como se ha observado anteriormente, el valor medio de este parámetro durante la sesión se mantiene muy por debajo de dicho valor.

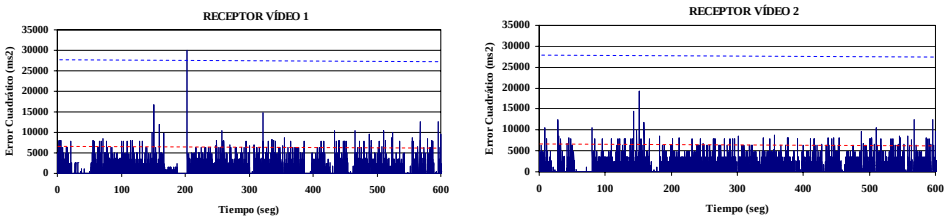


Figura 6.54. Valor cuadrático del error de sincronización inter-flujo (televigilancia - LAN - con algoritmo)

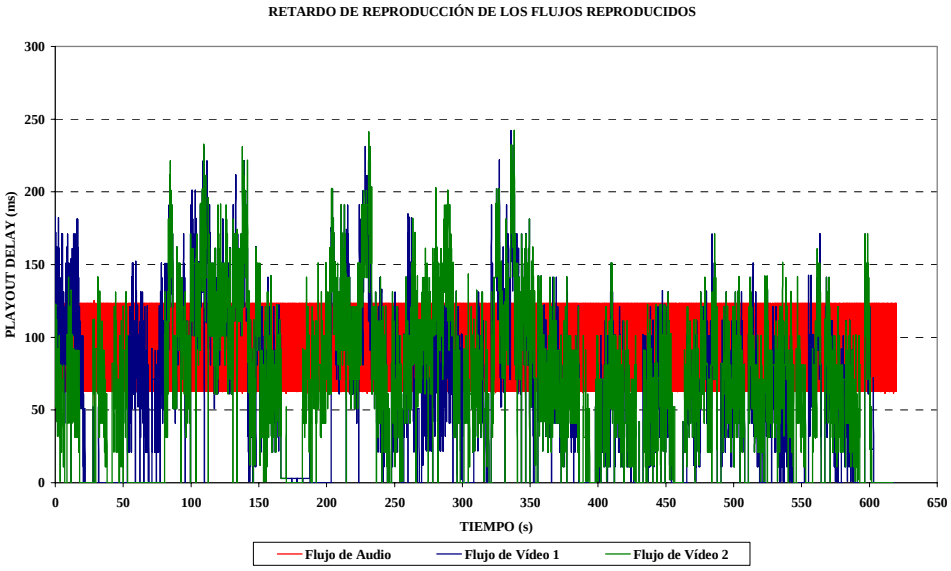
B3. Sincronización entre flujos reproducidos por diferentes receptores

El único parámetro que nos puede servir para saber si la reproducción de los tres flujos (el de audio y los dos de vídeo) están siendo reproducidos, realmente, en la aplicación, de forma sincronizada es el valor del *playout delay* de cada uno de ellos en cada instante. Si están sincronizados, dicho valor debería ser similar en todo momento.

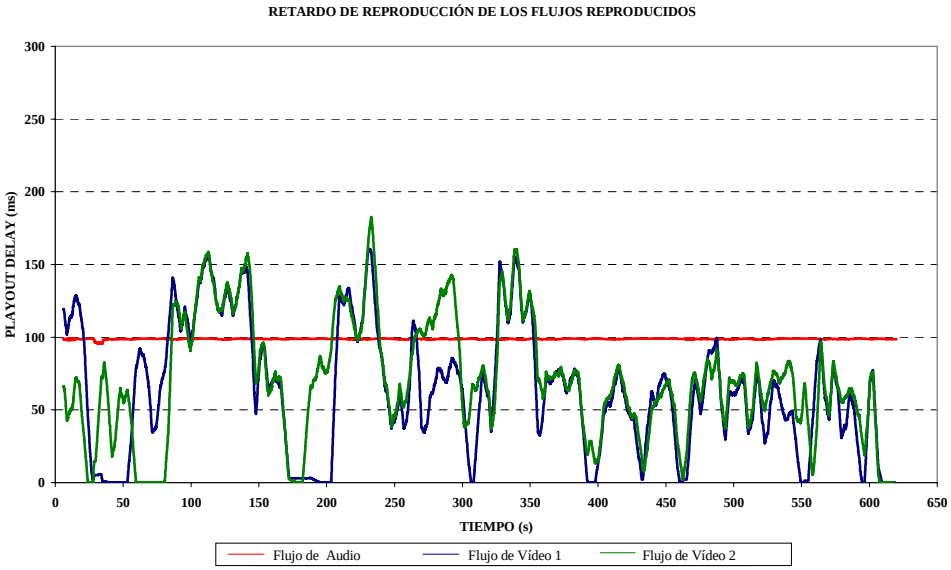
- ***Retardo de reproducción (playout delay) de los procesos reproductores de los tres flujos: el de audio (maestro) y los dos de vídeo (esclavos).***

A continuación se muestra la figura 6.55 que representa el valor del *playout delay* de los tres flujos, calculado por cada uno de los procesos reproductores del flujo principal de cada receptor independiente. En la misma se ha representado un fragmento de unos diez minutos de transmisión, aproximadamente. Puede observarse que los retados de reproducción de los flujos de vídeo de los receptores *esclavos* se van adaptando en determinados puntos al

retardo de reproducción del flujo de audio del receptor *maestro* (en rojo), tomado como referencia para la sincronización de grupo propuesta. Esto ha sido posible gracias a los dos procesos evaluados anteriormente: la sincronización de grupo (distribuida) y la sincronización inter-flujo (local en cada receptor).



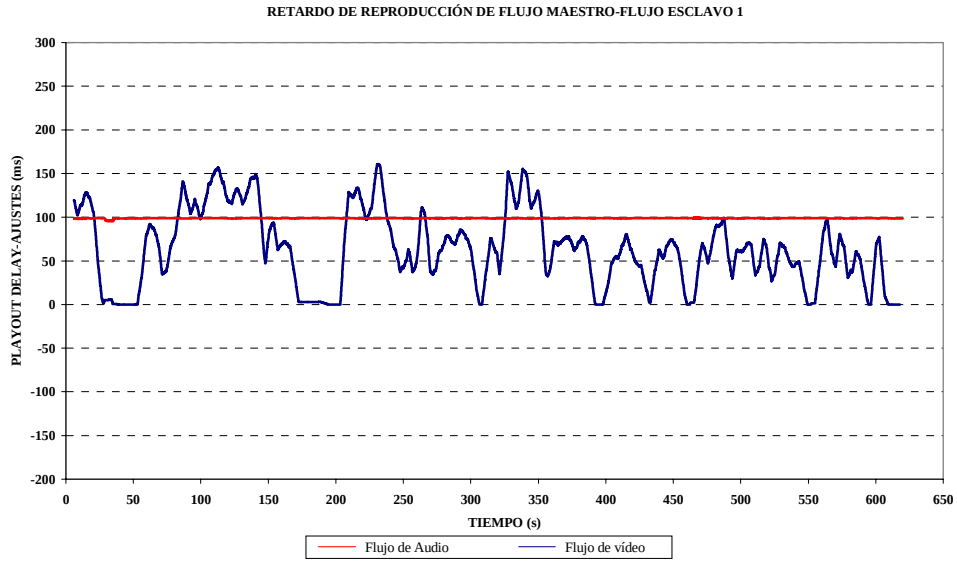
a) Valores reales



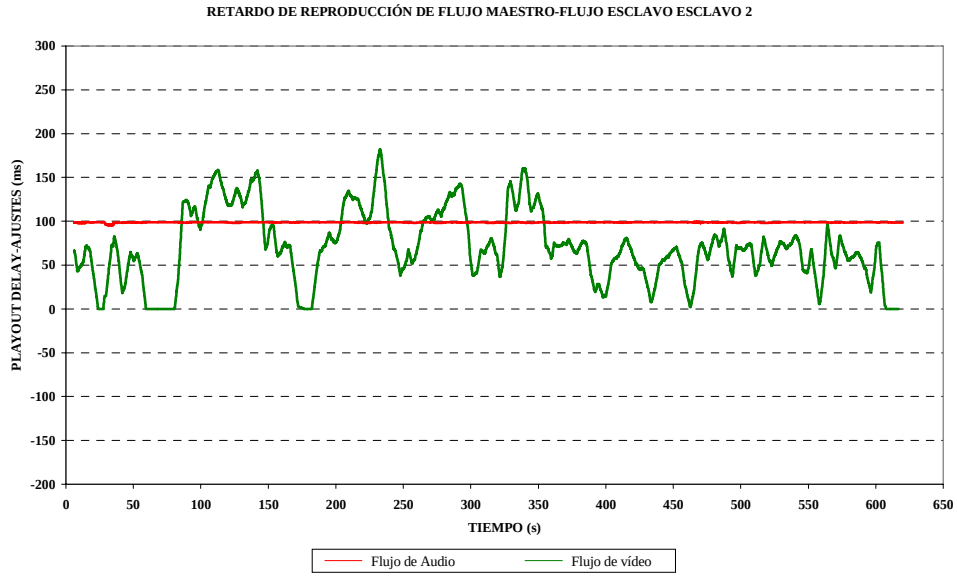
b) Media móvil (conjuntos de 100 muestras)

Figura 6.55. *Playout delay* de los tres flujos involucrados en la sesión (televigilancia - LAN - con algoritmo)

En la figura 6.56 se ha representado la gráfica anterior pero separando el *playout delay* del flujo de vídeo de cada receptor *esclavo*.



a) Flujo esclavo vídeo 1



b) Flujo esclavo vídeo 2

Figura 6.56. *Playout delay* de los flujos *maestro* y *esclavo* en los receptores de vídeo (televigilancia - LAN - con algoritmo)

6.3.2.2. Entorno CAMPUS

A continuación se muestra la evaluación realizada para la transmisión de los mismos tres flujos anteriores desde tres fuentes independientes, en este caso, situadas en el Campus de Vera en Valencia, hacia tres receptores independientes situados en el Campus de Gandia. Los flujos de audio y vídeo fueron capturados, en directo, desde el Centro de Proceso de Datos del Campus de Valencia por dos cámaras de vídeo y un micrófono, conectados cada uno a un transmisor independiente (ordenador personal equipado con los dispositivos de captura adecuados).

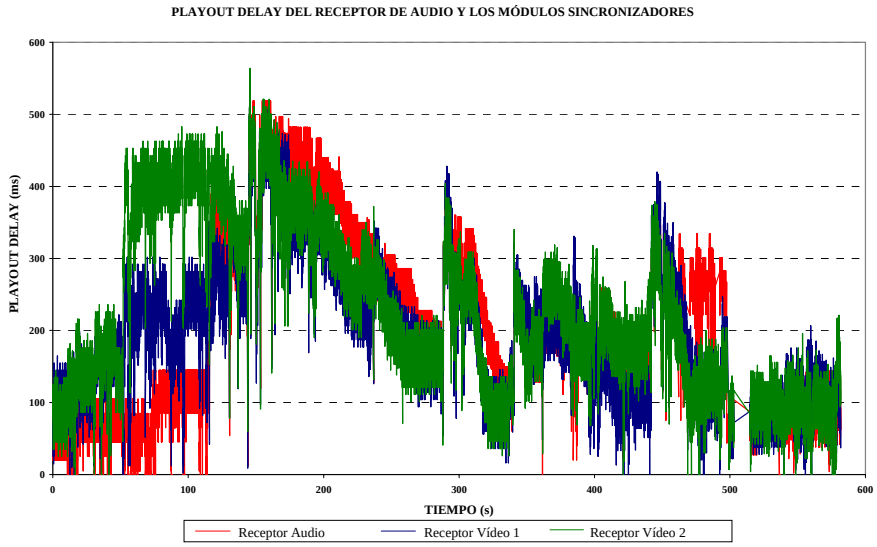
A) SIN ALGORITMO

En este caso, también se ha sincronizado los relojes de todos los equipos involucrados mediante NTP para poder disponer de la referencia de tiempos común con el fin de comparar los parámetros obtenidos durante la sesión, y no se ha ejecutado algoritmo de sincronización alguno.

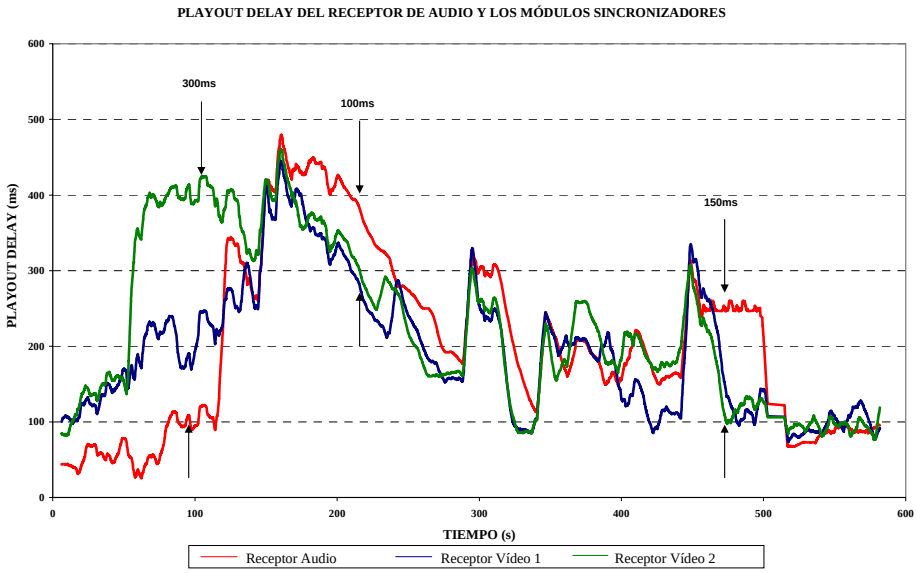
- ***Retardos de reproducción (playout delay) de los procesos reproductores del flujo de referencia (audio) en todos los receptores***

A continuación, en la figura 6.57, se muestra el valor del *playout delay* calculado por los procesos reproductores del flujo de audio (reales y ficticios) en los tres receptores.

En la figura puede apreciarse que, durante la sesión, en determinados momentos, han existido grandes diferencias entre los valores calculados por el receptor de audio y los calculados por los módulos sincronizadores de los receptores de vídeo, llegando a los 300 milisegundos.



a) Valores reales



b) Media móvil (conjuntos de 100 muestras)

Figura 6.57. *Playout delay* calculado por el receptor de audio y por los módulos de sincronización
(televigilancia - CAMPUS - sin algoritmo)

- ***Números de secuencia de las LDUs del flujo de audio (maestro) reproducidas (real o ficticiamente) en cada receptor***

Si nos fijamos en la primera LDU reproducida realmente o ficticiamente por los tres receptores, representada en la figura 6.58, vemos que al calcular el instante de reproducción de dicha LDU (en este caso, se empezó por la número 26.250), se aprecia que existe una asincronía de 50 milisegundos entre los procesos del receptor de audio y los procesos sincronizadores de los receptores de vídeo ya en el momento de iniciarse la reproducción.

Si, por otro lado, nos fijamos, ahora, en los números de secuencia de las LDUs del flujo de audio reproducidas de forma real (receptor de audio) o de forma ficticia (receptores de vídeo), representados en la figura 6.59, vemos que en determinados intervalos existen diferencias notables. En la figura se ha representado un intervalo correspondiente a 1 segundo y en ella se puede apreciar que, en dicho intervalo, existe una asincronía de, aproximadamente, unos 360 milisegundos[8] entre el proceso reproductor de audio del receptor de audio y el proceso ficticio del receptor de vídeo 2.

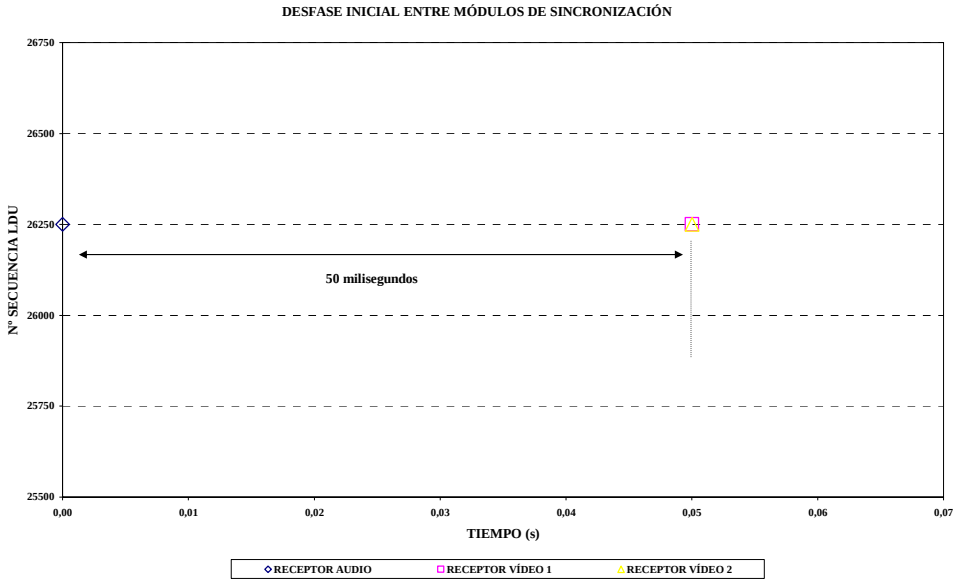


Figura 6.58. Primera LDU del flujo de audio reproducida (real y ficticiamente) (televigilancia - CAMPUS - sin algoritmo)

[8] En esta aplicación se realizó la transmisión del flujo de audio conteniendo los paquetes RTP muestras correspondientes a 40 milisegundos

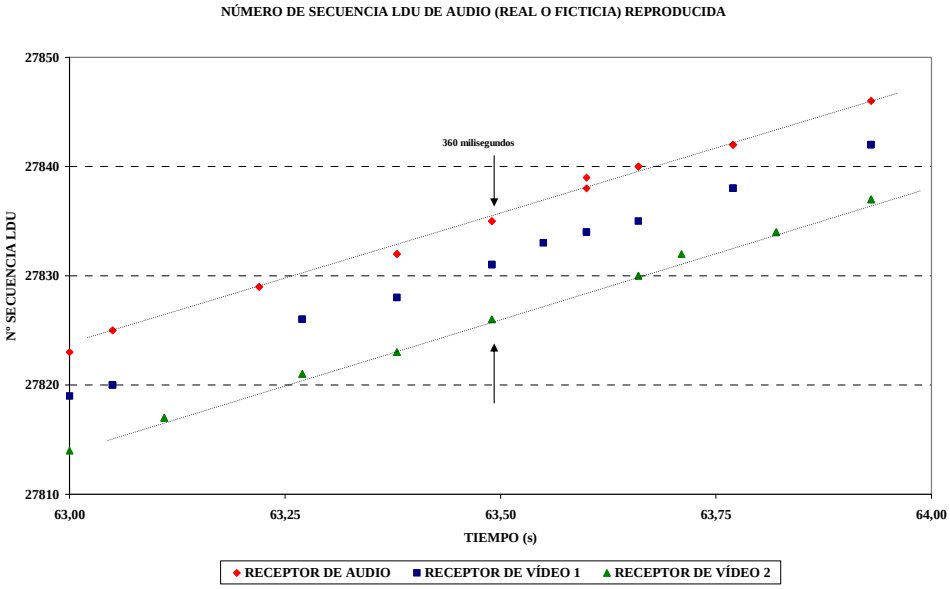


Figura 6.59. Número de secuencia de las LDUs de audio reproducidas (real y ficticiamente)
(televigilancia - CAMPUS - sin algoritmo)

- ***Instante de inicio de la reproducción de los diferentes receptores***

También se ha comprobado que los diferentes receptores inician la reproducción de los tres flujos 'reales' (en cada receptor un flujo diferente) en diferentes instantes de tiempo.

En la figura 6.60 se muestra el instante en que cada receptor reproduce el contenido del primer paquete RTP del flujo correspondiente.

Se aprecia claramente que existe un desfase en el inicio de la reproducción que llega casi a los 10 segundos entre el receptor de audio y el receptor del vídeo 1, y un desfase de unos 5,4 segundos entre el receptor de audio y el receptor del vídeo 2. Con estos resultados obtenidos podemos afirmar, por tanto, que, en ausencia de algoritmo de sincronización, existe una asincronía en los procesos de reproducción de los tres flujos involucrados en la aplicación de televigilancia implementada.

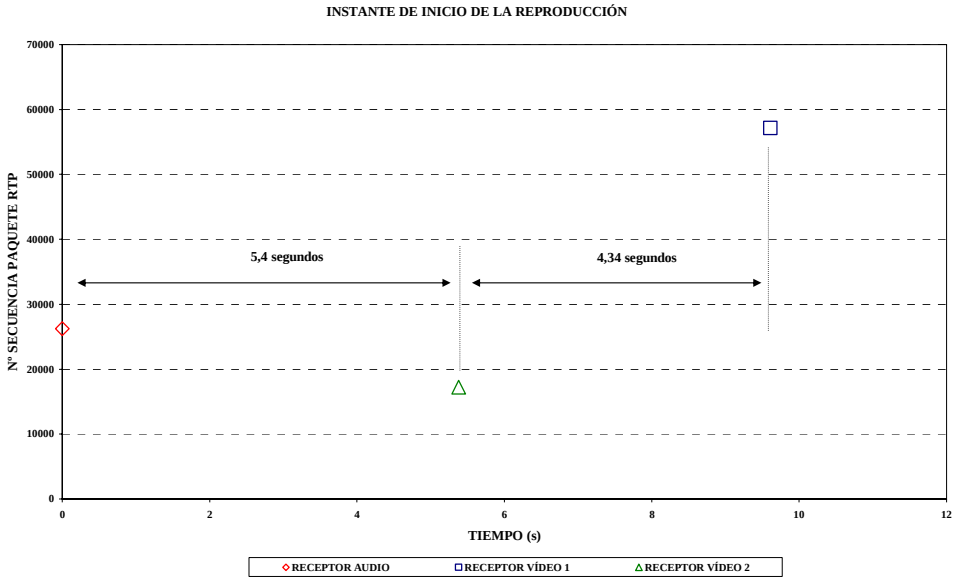


Figura 6.60. Instante de inicio de la reproducción de los tres flujos (televigilancia - CAMPUS - sin algoritmo)

B) CON EL ALGORITMO DE SINCRONIZACIÓN DE GRUPO

A continuación se presentan los resultados de la aplicación activando la sincronización de grupo, tal y como se hizo para el entorno LAN.

Como en dicho entorno, en este caso, en primer lugar, se van a presentar gráficas del funcionamiento de la sincronización de grupo distribuida y la sincronización del instante inicial de consumo, con respecto al flujo de referencia. En segundo lugar se evaluará la sincronización inter-flujo entre el flujo ficticio de referencia y el flujo de vídeo, localmente en cada uno de los receptores de flujo de vídeo. En tercer lugar, se evaluará el grado de sincronización en la reproducción de los tres flujos reales, cada uno reproducido en un receptor independiente.

B1. Sincronización de Grupo (distribuida)

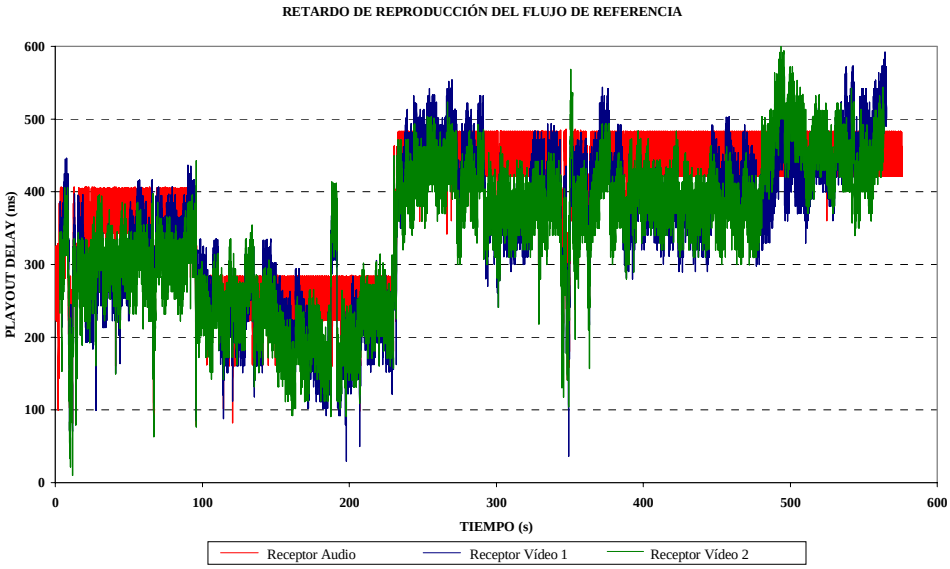
- ***Retardos de reproducción (playout delay) de los procesos reproductores del flujo de referencia de todos los receptores***

La figura 6.61 representa el retardo de reproducción del flujo de audio experimentado por los 3 receptores, tanto los valores reales como la representación de la media móvil de los mismos, a partir de grupos de 100 muestras. En ella puede observarse que los retados de reproducción de los flujos ficticios de audio en los receptores de vídeo se van adaptando en determinados puntos al retardo de reproducción del receptor de audio tomado como referencia o receptor *maestro* (en rojo).

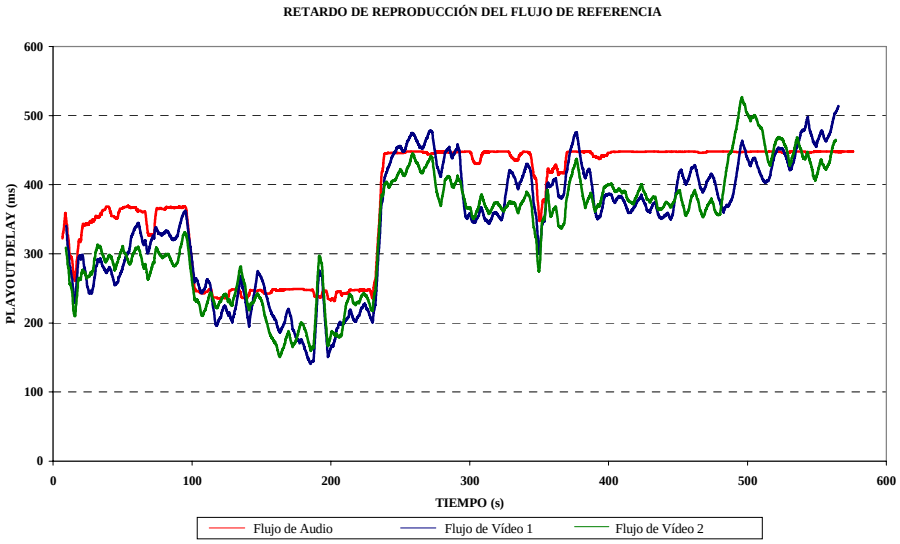
En la figura 6.62, se desglosa la gráfica anterior para los dos receptores de vídeo, mostrando los ajustes sufridos por el proceso reproductor ficticio de dicho receptor, debido a las operaciones de '*saltos*' y '*pausas*' efectuadas por las acciones de resincronización del algoritmo propuesto.

En la tabla 6.16 se muestran los valores de los mismos parámetros evaluados en el entorno anterior. En este caso el servidor envía 1 sólo paquete APP TIN al comienzo de la misma, para indicar el instante de inicio de la reproducción (*Instante Inicial de Consumo*) y, posteriormente, a lo largo de la misma, envía 173 paquetes APP ACT de 'acción', indicando acciones de resincronización en los receptores, frente a los 15300 paquetes RTP de datos del flujo de audio (que contienen muestras de audio codificadas correspondientes a un fragmento de audio de 40 milisegundos de duración). Comparado con los valores obtenidos para la aplicación anterior, se trata de un mayor número de mensajes de control debido a que, para esta aplicación, no se ha fijado un umbral de asincronía máxima tolerada antes de enviar mensajes de acción por parte de la fuente sincronizadora.

Los mensajes de control enviados por la fuente sincronizadota (1 APP TIN y 173 APP ACT, es decir, 174 mensajes de control RTCP) suponen un 1,1 % del total de los paquetes transmitidos (de datos y de control) considerando únicamente el flujo de audio. Por otro lado, los mensajes de control enviados por los receptores (en total 621 mensajes RTCP RR extendidos) apenas suponen un 3,9 % del total de los paquetes transmitidos (de datos y de control) por las aplicaciones de audio durante la sesión y un 4 % con respecto al número total de paquetes de datos enviados (15300 paquetes RTP).

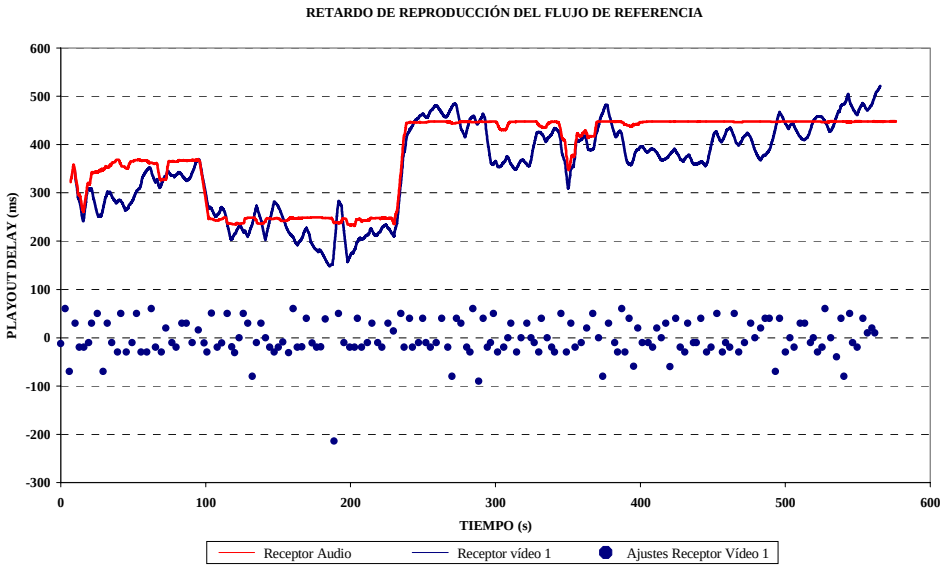


a) Valores reales

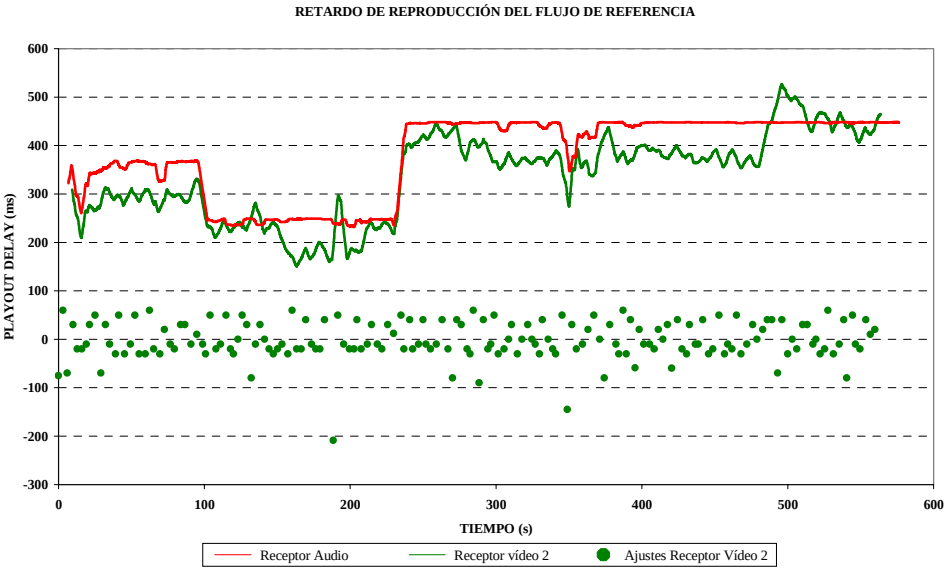


b) Media móvil (conjuntos de 100 muestras)

Figura 6.61. *Playout delay* del flujo de referencia de los receptores (televigilancia - CAMPUS - con algoritmo)



a) Receptor de vídeo 1



b) Receptor de vídeo 2

Figura 6.62. *Playout delay* del flujo *maestro de* referencia en cada receptor de vídeo. Ajustes en el receptor de vídeo
(televigilancia - CAMPUS - con algoritmo)

ESTADÍSTICAS RECEPTOR	**RECEPTOR VÍDEO 1**	**RECEPTOR VÍDEO 2**	**RECEPTOR AUDIO (MAESTRO)**
Máxima asincronía detectada (ms)	60	60	-
Mínima asincronía detectada (ms)	-214	-209	-
Valor medio de la asincronía detectada (ms)	-0,78	-1,88	-
Desviación estándar de las asincronías detectadas (ms)	37,50	39,27	-
Valor cuadrático medio de la asincronía detectada (ms^2)	1399	1537	-
Nº mensajes de '*acción*' (APP TIN) enviados por servidor	1		
Nº mensajes de '*acción*' (APP ACT) enviados por servidor	173		
Nº mensajes recibidos del servidor (APP ACT)	173	173	173
Número de mensajes *Feedback* enviados por el receptor (RR Extendidos)	210	208	203
Asincronía máxima permitida audio-audio con respecto al receptor *maestro* (± milisegundos)	0		
***'Saltos'* realizados por el flujo de audio (*maestro*) debido a la sincronización de grupo**	48	53	-
***'Pausas'* realizadas por el flujo de audio (*maestro*) debido a la sincronización de grupo**	62	62	-
Nº de paquetes RTP enviados	15300		
Duración muestra flujo de audio (ms)	40		

Tabla 6.16. Datos obtenidos de los tres receptores (televigilancia - CAMPUS - con algoritmo)

El valor del parámetro *Valor cuadrático medio de la asincronía detectada*, proporciona una para idea del grado de sincronización de grupo obtenida. Se puede observar que el valor máximo le corresponde al receptor de vídeo 2 con un valor de 1537 milisegundos^2, correspondiente a errores de unos 40 milisegundos, muy por debajo de 14400 milisegundos^2.

Para mantener la sincronización de grupo (distribuida), puede observarse en la tabla que, en los receptores de vídeo, el algoritmo propuesto ha provocado acciones de resincronización que se traducen en *'saltos'* y *'pausas'* en el proceso ficticio de reproducción del flujo de audio. El receptor que más *'saltos'* hubiera experimentado en el proceso ficticio reproductor del flujo de audio ha sido el receptor de vídeo 2 con 53 *'saltos'* (frente a los 48 del receptor de vídeo 1), mientras que los dos receptores han sufrido el mismo número de *'pausas'* (62). Prácticamente el comportamiento de ambos procesos ficticios ha sido el mismo.

A continuación, en la figura 6.63 aparece la distribución de los ajustes que han sufrido los procesos reproductores del flujo *maestro* y de referencia de cada uno de los receptores, como consecuencia de la recepción de paquetes de '*acción*' (APP ACT) enviados por la fuente sincronizadora. Como se ha comentado, los ajustes reflejan los errores de sincronización entre los procesos reproductores ficticios (módulos sincronizadores) y el proceso reproductor del receptor de audio. Dicho error influirá posteriormente en la sincronización entre los tres flujos 'reales', uno de audio y dos vídeo.

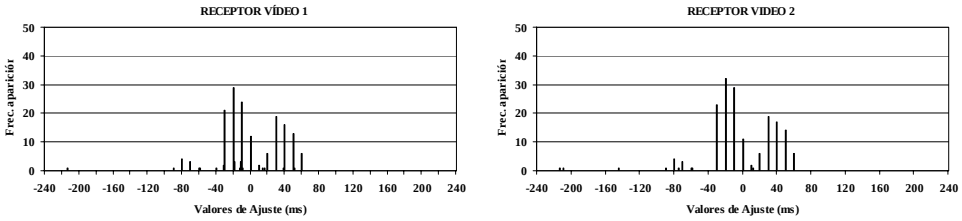


Figura 6.63. Distribución de los valores de ajuste del proceso de reproducción ficticio de cada receptor (televigilancia - CAMPUS - con algoritmo)

En la figura 6.64 se muestran las gráficas de cada receptor con el valor cuadrático de dichos ajustes a la largo de los 10 minutos de la sesión. Se puede observar que se trata de unos valores bastante bajos (por debajo de 14400 milisegundos^2, salvo en una ocasión en el receptor de vídeo 1 y en dos ocasiones en el receptor de vídeo 2).

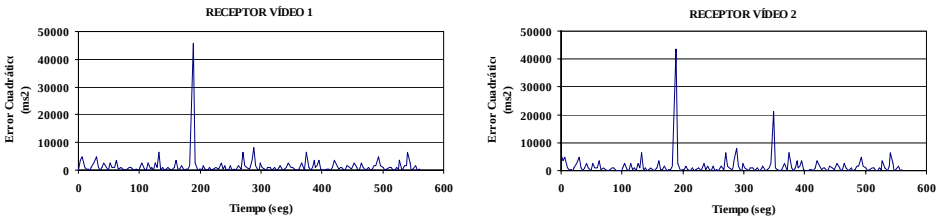


Figura 6.64. Valor Cuadrático del Error de Sincronización de Grupo
(televigilancia - CAMPUS - con algoritmo)

- ***Número de Secuencia de las LDUs del flujo de referencia reproducidas en cada receptor (real o ficticiamente)***

Se va a proceder a analizar el número de secuencia de la LDU del flujo *maestro* que reproduce cada proceso reproductor de dicho flujo en cada instante en cada uno de los tres receptores.

Primero, se ha comprobado que el instante de inicio de la reproducción del flujo de referencia coincide en todos los receptores ya que todos los receptores comenzaron la reproducción de la primera LDU transmitida (en este caso, la número 59951) en el mismo instante de tiempo, tal y como se muestra en la figura 6.65.

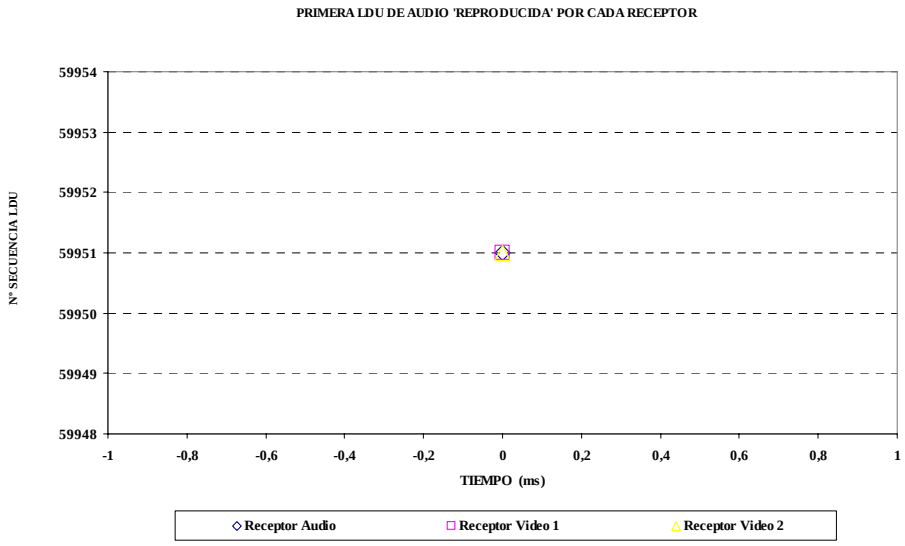


Figura 6.65. Primera LDU del flujo de referencia reproducida
(real o ficticiamente)
(televigilancia - CAMPUS - con algoritmo)

Podemos concluir, pues, que *el mecanismo de sincronización del instante inicial de consumo ha funcionado correctamente*.

En la siguiente figura se han representado datos de 35 muestras obtenidas durante la sesión, superpuestas, tomando diferencias relativas, de tiempos (eje *x*) y de número de LDU reproducida en cada instante (eje *x* y eje *y*, respectivamente), tal y como se hizo anteriormente. Como se han obtenido 173 muestras durante la sesión de evaluación, para que se pueda visualizar mejor la información se ha representado una muestra de cada 5 obtenidas.

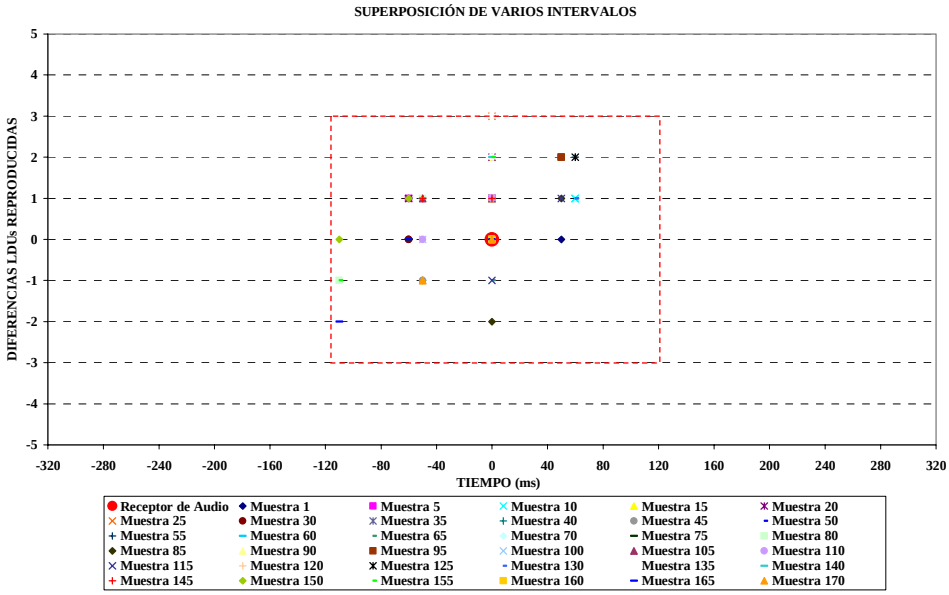


Figura 6.66. Superposición de 36 muestras (televigilancia - CAMPUS - con algoritmo)

En la total, en la figura hay representados 109 puntos correspondientes a la muestra reproducida por el receptor de audio (en rojo, centrada en el punto (0,0)) y las muestras reproducidas, de forma ficticia, por los receptores de vídeo en los 36 instantes muestreados. Se puede observar en la misma que, en las muestras representadas, la totalidad de los puntos están dentro del cuadro rojo (que representa una desviación de ±120 milisegundos), y que los puntos están repartidos por dentro del mismo, indicando que los puntos de reproducción ficticios de los receptores de vídeo oscilan alrededor del punto de reproducción del receptor de audio en cada instante.

B2. Sincronización inter-flujo (local) en cada receptor de vídeo

La sincronización inter-flujo, en esta aplicación, sólo se produce localmente en los receptores de los flujos de vídeo que es donde se reproduce más de un flujo, el de vídeo y el de referencia (aunque la reproducción de este último sólo sea de forma ficticia).

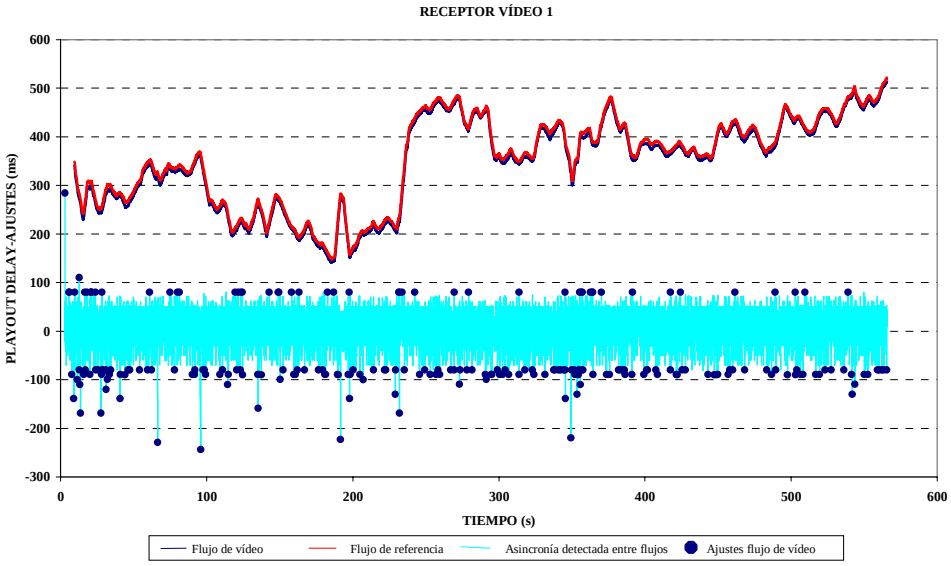
- ***Retardo de reproducción (playout delay) de los procesos reproductores de los flujos maestro (de referencia) y esclavo***

A continuación se analiza los retardos de reproducción (*playout delay*) de ambos flujos. En la figura 6.67, se muestra la media móvil del *playout delay* de los flujos *maestro* (de referencia) y *esclavo* (vídeo) correspondientes a los dos receptores de vídeo, tomada a partir de conjuntos de 100 muestras.

También se muestran la asincronía detectada y los ajustes que ha sufrido el proceso reproductor del flujo de vídeo como consecuencia de la aplicación del algoritmo propuesto.

Durante la sesión los procesos de reproducción del flujo de vídeo en ambos receptores sufren 218 y 221 ajustes (receptores de vídeo 1 y 2, respectivamente), la mayoría de los cuales está en torno a 80 milisegundos ya que éste es el valor configurado en las aplicaciones como asincronía máxima permitida entre flujo *maestro* y flujo *esclavo*, antes de enviar un mensaje de sincronización a través de mbus.

En la tabla 6.17 se muestran los resultados obtenidos en este entorno. En ella aparece el número de mensajes *mbus* intercambiados entre el proceso reproductor ficticio (módulo sincronizador) y el proceso reproductor de vídeo, en ambos receptores, que, como es lógico, coincide con el número de situaciones de asincronía detectadas.



a) Receptor de vídeo 1

a) Receptor de vídeo 2

Figura 6.67. Media móvil (conjuntos de 100 muestras) del *Playout delay* de los dos flujos reproducidos por los receptores de vídeo, asincronía detectada y ajustes del proceso reproductor del flujo *esclavo*
(televigilancia - CAMPUS - con algoritmo)

ESTADÍSTICAS RECEPTOR	**RECEPTOR VÍDEO 1**	**RECEPTOR VÍDEO 2**
Máxima asincronía positiva detectada (ms)	110	313
Mínima asincronía negativa detectada (ms)	-244	-278
Valor medio de la Asincronía detectada (ms)	1,7	1,53
Desviación estándar de las asincronías detectadas (ms)	28,38	28,28
Valor cuadrático medio de la asincronía detectada (ms^2) [9]	1283,66	1284,35
Nº mensajes mbus enviados	218	221
Asincronía Máxima audio-vídeo *permitida y configurada* en las aplicaciones (±milisegundos)	±80	
Número Asincronías Audio-Vídeo	218	221
'*Saltos*' realizados por el flujo de vídeo (*esclavo*) debido a la sincronización inter-flujo	288	286
'*Pausas*' realizadas por el flujo de vídeo (*esclavo*) debido a la sincronización inter-flujo	108	112
Duración Muestra flujo de vídeo (ms)	40	

Tabla 6.17. Datos obtenidos de ambos receptores
(televigilancia - CAMPUS - con algoritmo)

En la figura 6.68 se muestra la distribución de los ajustes que ha sufrido el proceso reproductor del flujo de vídeo de cada receptor para ajustarse al estado del proceso reproductor del flujo ficticio de referencia que es el tomado como *maestro* por el algoritmo. Puede observarse que también en este caso, los ajustes que superan los límites establecidos de ±80 milisegundos son mínimos en todos los receptores, así como que no existen ajustes por debajo de 80 milisegundos (por el

[9] Para el cálculo de este valor no se ha tenido en cuenta el ajuste inicial debido al inicio de las aplicaciones en cada receptor.

umbral fijado comentado anteriormente). La mayoría de ajustes son muy cercanos a ±80 milisegundos.

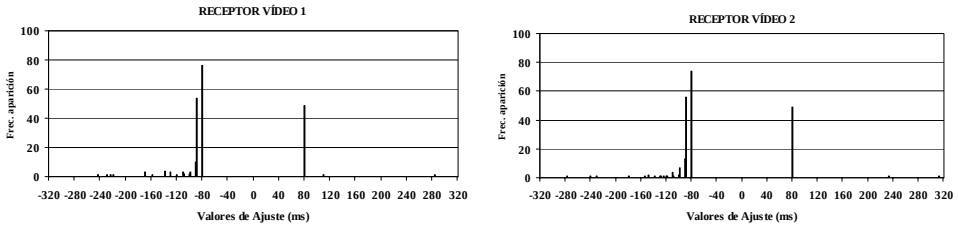


Figura 6.68. Distribución de los ajustes en los procesos reproductores de vídeo (televigilancia - CAMPUS - con algoritmo)

Cabe destacar que existe un mayor número de ajustes negativos , la mayoría cercanos a -80 milisegundos, indicando '*saltos*' de esa cantidad de tiempo al proceso reproductor del flujo de vídeo que estaba retrasado. Este efecto es el mismo encontrado en la aplicación en entorno LAN.

En la tabla anterior aparece el número de asincronías detectadas en cada receptor (superación del límite de ±80 milisegundos). En ambos receptores se produce un número similar de situaciones de asincronía, que, como ya se ha comentado, es de 218 en el receptor de vídeo 1 y de 221 en el receptor de vídeo 2.

Para corregir dichas situaciones de asincronías, los procesos reproductores del flujo *esclavo* de cada receptor han tenido que corregir su estado mediante acciones de *'saltos'* y *'pausas'*. El proceso reproductor de vídeo del receptor de vídeo 1 realizó 288 *'saltos'* y 108 *'pausas'*, mientras que el proceso reproductor de vídeo del receptor de vídeo 2 realizó 286 *'saltos'* y 112 *'pausas'*. La mayoría de estos '*saltos*' y '*pausas*' fueron de aproximadamente dos LDUs (ya que la mayoría de ajustes son cercanos a los ±80 milisegundos.

Si nos centramos en el valor cuadrático medio de las diferencias entre los estados de los procesos reproductores de ambos flujos, puede apreciarse que el valor medio del error cuadrático de la asincronía en los receptores de vídeo 1 y 2 es de 1283,66 y 1912 milisegundos^2, respectivamente, muy por debajo del valor 6400 milisegundos^2 correespondiente a valores de error de ±80 milisegundos.

En la siguiente figura se muestran las dos gráficas con la distribución de los valores cuadráticos de los errores de sincronización detectados en los dos receptores de vídeo durante la sesión. Puede observarse que durante la sesión, los valores permanecen por debajo del límite máximo de 25600 milisegundos^2 (correspondientes a errores de ±160 milisegundos), superándose esta cantidad en ocasiones aisladas.

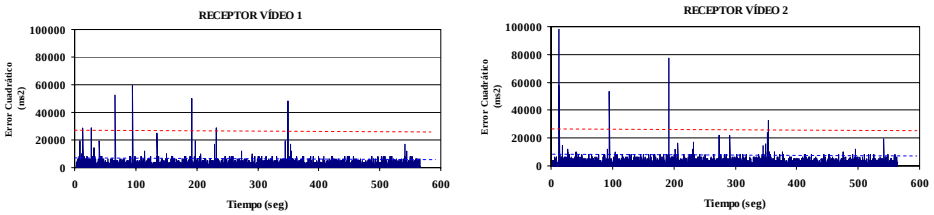


Figura 6.69. Valor cuadrático del error de sincronización inter-flujo (televigilancia - CAMPUS - con algoritmo)

B3. Sincronización entre flujos reproducidos por diferentes receptores

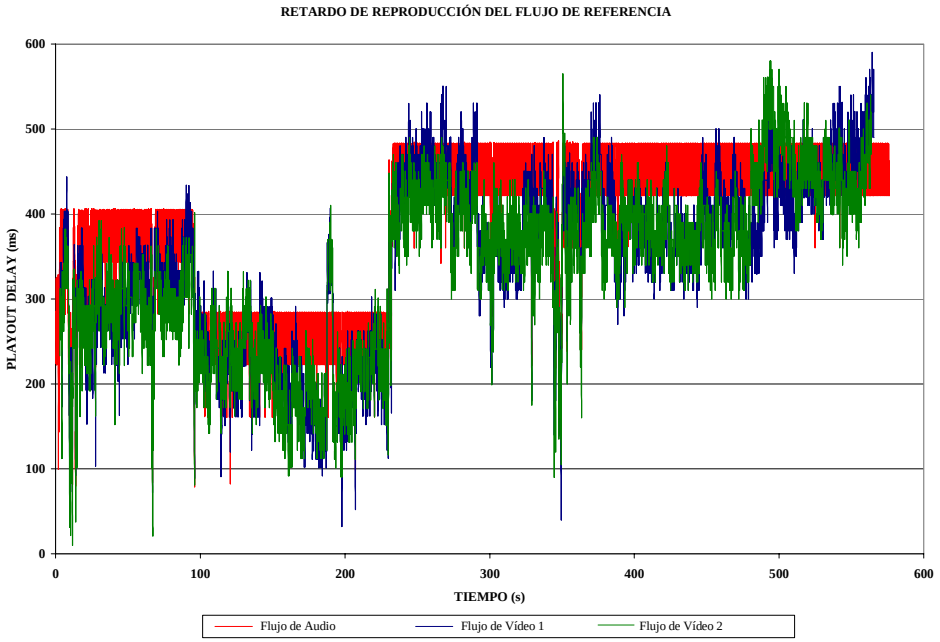
El único parámetro que nos puede servir para saber si la reproducción de los tres flujos (el de audio y los dos de vídeo) están siendo reproducidos de forma sincronizada es mediante el análisis del valor del *playout delay* de cada uno de ellos en cada instante. Como se ha comentado, si están sincronizados, dichos valores deberían ser similares en todo momento.

- ***Retardo de reproducción (playout delay) de los procesos reproductores de los tres flujos: el de audio (maestro) y los dos de vídeo (esclavos).***

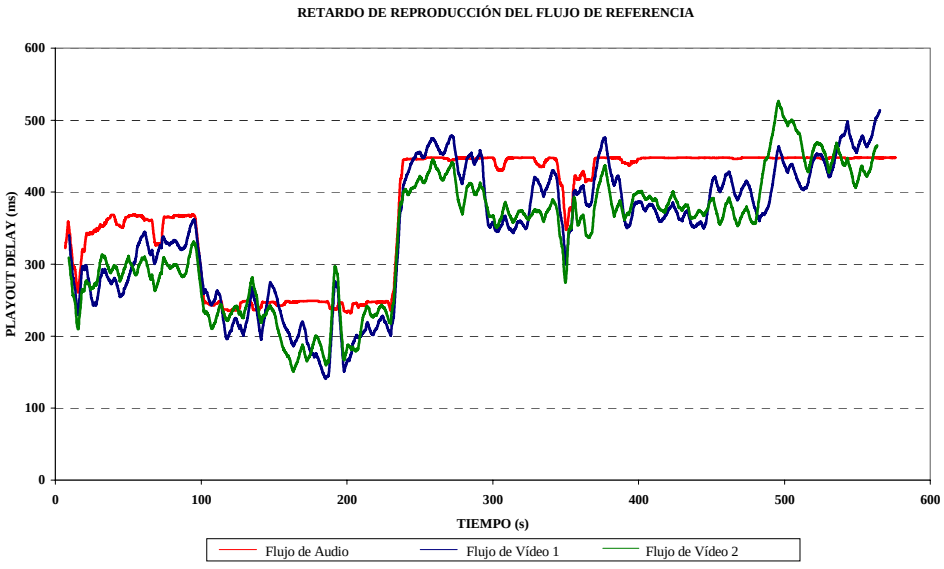
A continuación, se muestra la figura 6.70 que representa el valor del *playout delay* (valores reales y media móvil) de los tres flujos, calculados por cada uno de los procesos reproductores del flujo principal de cada receptor.

Los retardos de reproducción de los flujos de vídeo de los receptores *esclavos* se van adaptando al retardo de reproducción del flujo de audio del receptor *maestro* (en rojo), tomado como referencia para la sincronización de grupo propuesta. Esto ha sido posible gracias a los dos procesos evaluados anteriormente: la sincronización de grupo (distribuida) y la sincronización inter-flujo (local en cada receptor).

En la figura 6.71 se ha representado la gráfica anterior pero separando el *playout delay* del flujo de vídeo de cada receptor *esclavo*.

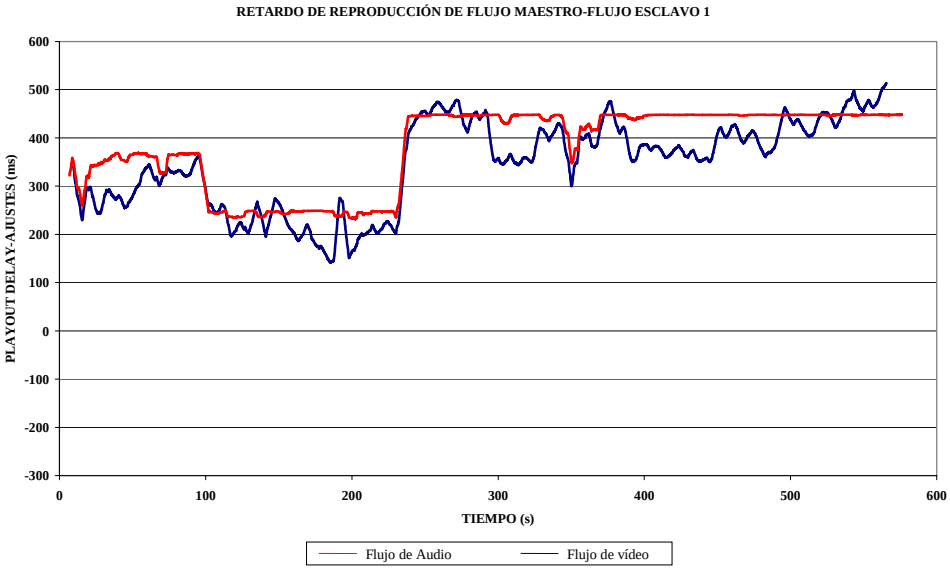


a) Valores reales

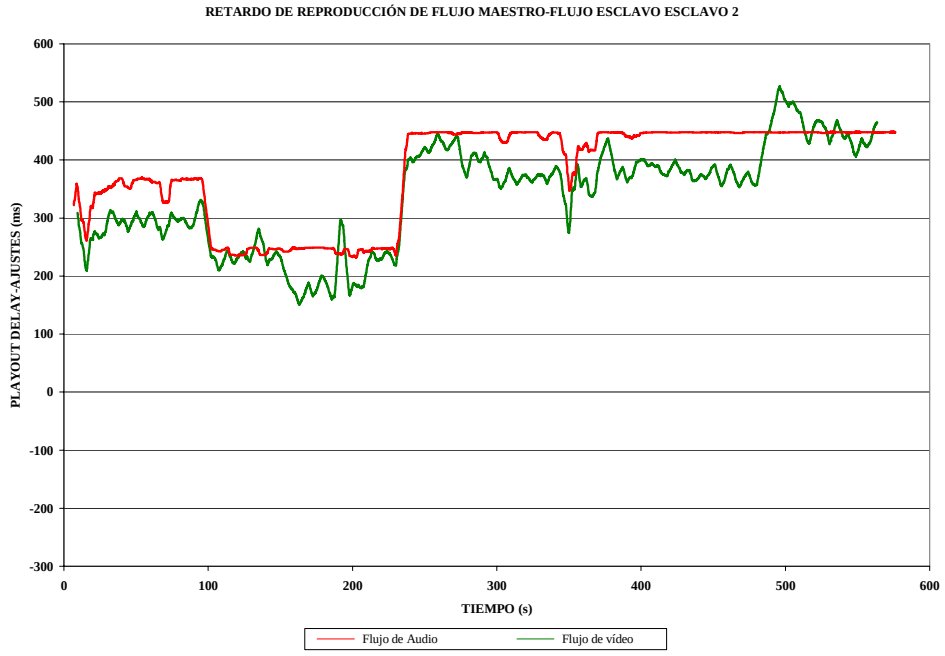


b) Media móvil (conjuntos de 100 muestras)

Figura 6.70. *Playout delay* de los tres flujos involucrados en la sesión (televigilancia - CAMPUS - con algoritmo)



a) Flujo esclavo vídeo 1



b) Flujo esclavo vídeo 2

Figura 6.71. *Playout Delay* de los flujos *maestro* y *esclavo*
en los receptores de vídeo
(televigilancia - CAMPUS - con algoritmo)

6.4. Conclusiones

En el presente capítulo se ha analizado la validez del algoritmo propuesto, mediante la *evaluación objetiva* del mismo, en dos aplicaciones reales y en dos entornos muy similares en cuanto a características (con retardos controlados), como son los entornos LAN y de CAMPUS.

En ambos entornos, se han ejecutado dos aplicaciones en las que se ha integrado el algoritmo propuesto y ha quedado probada su validez tanto para el caso de aplicaciones con una única fuente de flujos (como en la *aplicación de aprendizaje a distancia*), como para el caso de varias fuentes transmitiendo flujos de forma independiente (*aplicación de televigilancia*), incluso con varios receptores independientes para cada flujo.

En primer lugar, se ha comprobado que, *en ausencia de algoritmo de sincronización*, en ambas aplicaciones y en ambos entornos, los estados de los procesos reproductores de los diferentes flujos involucrados en las aplicaciones diseñadas sufren asincronías desde el inicio de la reproducción que, durante la misma, llegan incluso a sobrepasar los valores máximos admisibles, tomados de la referencia [STE96].

Posteriormente, se ha comprobado, mediante el análisis objetivo de ciertos parámetros típicos en la evaluación de la sincronización de flujos multimedia, que con el algoritmo propuesto, en ambas aplicaciones y en ambos entornos, se consiguen las siguientes mejoras:

1. Inicio de la reproducción sincronizado en todos los receptores, con lo que se consigue el mismo *Instante Inicial de Consumo* en todos ellos, del flujo considerado como *maestro*.

2. Mediante las acciones de resincronización del algoritmo se consigue la sincronización de grupo buscada por el mismo, es decir, una reproducción sincronizada del flujo *maestro* en todos los receptores. Se mantienen los valores de asincronía entre los procesos reproductores de dicho flujo, prácticamente la totalidad del tiempo de las sesiones de evaluación, por de debajo de los valores fijados de referencia.

3. Además, localmente, en cada uno de los receptores, en caso de reproducir varios flujos, se mantiene la reproducción de todos ellos de forma sincronizada (sincronización inter-flujo), con unos niveles de asincronía que no sobrepasan los umbrales tomados como referencia (salvo en ocasiones puntuales), corrigiéndose inmediatamente las situaciones de asincronía.

4. Se consigue, por tanto, según los dos puntos anteriores, una reproducción sincronizada de todos los flujos en todos los receptores a la vez.

En general, el protocolo propuesto garantiza que la asincronía entre receptores y entre flujos no supere los valores máximos de asincronía permitidos, excepto en determinadas situaciones puntuales que son rápidamente corregidas.

Por tanto, y para terminar el capítulo, podemos concluir, a partir de los resultados obtenidos en la evaluación objetiva, que, en las aplicaciones y entornos empleados, *el algoritmo de sincronización de grupo de flujos multimedia ha funcionado correctamente a lo largo de las diferentes sesiones de evaluación*, consiguiendo un mismos instante de inicio de la reproducción en todos los receptores y, a partir de éste y durante toda la sesión, una reproducción sincronizada de todos los flujos involucrados, corrigiendo las situaciones de asincronía puntuales aparecidas.

Capítulo 7

Evaluación Subjetiva y Resultados

7.1. Introducción

En las aplicaciones multimedia, en general, con flujos de vídeo, audio, texto, imágenes, etc., existe una representación de los diferentes flujos dirigida a un observador humano. Por tanto, para nosotros, los aspectos más importantes de dicha representación serán aquellos que sean perceptibles por el mismo, mientras que podemos ignorar aquellos aspectos que no lo sean, en la mayoría de las situaciones. Como consecuencia directa de la consideración anterior, la importancia de cada uno de los aspectos de la representación dependerá, de forma directa, de la trascendencia que le otorgue el sistema sensorial humano a los mismos.

Para asegurarnos del correcto funcionamiento del algoritmo de sincronización de grupo, sería conveniente complementar la evaluación objetiva presentada en el capítulo anterior con una *evaluación subjetiva*. Tal y como se comentó en dicho capítulo, la *Calidad Subjetiva* se refiere a la percepción de la

calidad de la presentación (en cuanto a la calidad de la imagen, tasa de reproducción, nivel de sincronización, etc.) por parte de un observador humano. Es la que determina, en último término, la aceptación, por parte del usuario, de un sistema o servicio multimedia y es por ello por lo que en el diseño de los sistemas multimedia tiene una gran importancia el estudio de las cualidades funcionales del *sistema audiovisual humano* [GIL95]. Dicho estudio permitirá determinar un *intervalo de calidad*, cuyos límites inferior y superior representan, respectivamente, el mínimo imprescindible para alcanzar una calidad aceptable y la frontera a partir de la cual se proporciona más calidad de la que el ser humano puede percibir.

En este capítulo nos vamos a centrar en la *evaluación subjetiva* consistente en estudiar cómo percibe el ser humano el efecto de sincronización del algoritmo desde el punto de vista psicofísico. Mediante dicho análisis se puede evaluar subjetivamente la calidad de una presentación multimedia en la que se haya implementado el algoritmo de sincronización propuesto.

La evaluación subjetiva, en este caso, sólo se ha realizado para la aplicación de la aplicación de aprendizaje a distancia descrita en el capítulo anterior en los dos entornos considerados en la evaluación objetiva (LAN y CAMPUS), en la que se considera más importante el factor del observador humano.

7.2. Evaluación Subjetiva

En este apartado se van a utilizar métodos de evaluación subjetiva para determinar la calidad del funcionamiento del algoritmo propuesto a través de mediciones y experimentos que anticipen, de manera más directa, las reacciones de aquellos usuarios que puedan utilizar los sistemas multimedia donde se implemente dicho algoritmo.

Para realizar la evaluación de la calidad subjetiva de nuestro algoritmo, nos hemos basado, principalmente, en el trabajo presentado por Steinmetz en [STE96], uno de los escasos trabajos y, sin duda, el más completo encontrado en la literatura, que se centra en el estudio de los requerimientos de las aplicaciones multimedia y del estudio de la percepción humana frente al efecto de la asincronía. Otros trabajos en los que nos hemos basado, que, a su vez, se han apoyado en el trabajo anterior, han sido el trabajo presentado por Guerri en [GUE97], donde se evalúa subjetivamente el comportamiento del protocolo Feedback-Global, y el trabajo presentado por Kaladji en [KAL99], donde se comparan los resultados de una evaluación subjetiva sistemática con los de la evaluación objetiva del algoritmo VTR propuesto por Ishibashi en [ISH95]. Para obtener la metodología de la evaluación también se ha seguido el método utilizado por Gili [GIL95] para evaluar subjetivamente la calidad de imágenes MPEG. Otros trabajos encontrados

sobre evaluación subjetiva, que han sido también consultados, son [WAT96], [WIL00], [ISH01b] y [ISH03], presentados en el capítulo 2.

En el capítulo anterior se ha presentado una evaluación objetiva del algoritmo propuesto basado en la asincronía entre flujos y el mantenimiento de ésta dentro de los límites fijados por [STE96], presentados en el capítulo 3. Sin embargo, no se ha considerado el efecto que produce sobre la percepción humana la ejecución, por ejemplo, de las acciones de ajuste mediante '*saltos*' y/o '*pausas*' en los procesos reproductores propias del algoritmo para corregir la asincronía entre flujos, la desviación en la tasa de consumo producida por las desviaciones de los relojes de los dispositivos involucrados, o incluso el efecto de no utilizar sistemas operativos de tiempo real.

Teniendo en cuenta todos estos aspectos, para completar el estudio de la calidad del resultado de nuestro algoritmo, se ha realizado también una evaluación del algoritmo propuesto mediante la *medida subjetiva de la calidad de sincronización a través del análisis de la percepción humana*. Ya que la percepción depende del individuo bajo estudio, los experimentos y las medidas se han realizado con una muestra formada por un número adecuado de individuos con el objetivo de poder obtener conclusiones significativas. Concretamente, la evaluación mediante el estudio de la percepción humana tiene como objetivo observar la valoración efectuada por una muestra representativa de población sobre la utilización del algoritmo propuesto como algoritmo de sincronización en una presentación multimedia distribuida.

Para ello, se ha medido la calidad y las prestaciones del algoritmo de sincronización propuesto mediante una escala de valores que permita a los individuos evaluadores puntuar cada una de las pruebas diseñadas a tal efecto y, posteriormente, se han analizado los resultados y obtenido las correspondientes conclusiones.

7.2.1. Diseño de los Tests de Evaluación de la Calidad Subjetiva de la Sincronización

7.2.1.1. Metodología

Como ya se ha comentado, los ejercicios de evaluación utilizados en la misma se han basado en los experimentos, medidas y resultados obtenidos en [GIL95], [STE96], [KOU96], [GUE97] y [KAL99].

Tipo de Secuencias

En nuestra evaluación se han utilizado secuencias con flujos de audio y vídeo, de dos tipos:

1.- *Tipo Noticias* (busto parlante): caracterizada por un movimiento mínimo y un fondo de imagen homogéneo. Se ha utilizado el tipo de plano denominado *Primer Plano*, donde se muestra la cara de una persona como primer plano. Este tipo de plano permite apreciar, de forma muy exacta, el movimiento de los labios y detectar si existe sincronización labial o no. Representa el plano típico utilizado en las noticias, discursos, etc. No se han utilizado presentaciones de *plano medio* y *plano lejano* debido a que, según [STE96], a priori, se deben obtener prácticamente los mismos resultados con los tres tipos de planos. Además, dado el tamaño de la imagen, en estos tipos de planos resulta complicado apreciar correctamente el movimiento de los labios, a medida que aumenta la distancia de la persona representada. Tal y como recomienda Steinmetz en [STE96], se ha escogido un fondo de la imagen sin elementos que pudieran distraer a los individuos evaluadores y, así, conseguir que se concentrasen más en el movimiento de los labios, gestos y cambios de tono de la persona representada.

2.- *Tipo película de acción*: caracterizada por la aparición de movimientos bruscos, efectos especiales y fondos de imagen variando constantemente. En adelante, se le denominará plano *Tipo Película*.

Básicamente, el experimento consistió en la transmisión de los dos tipos de secuencia, de la siguiente manera:

A) Para el primer tipo de secuencia (tipo *Primer Plano*), se reprodujo en cada receptor una presentación multimedia formada por un flujo de audio *y* uno de vídeo, obtenidos de una grabación en directo mediante una cámara de vídeo doméstico y un micrófono. En la figura 7.1 aparece una secuencia de tipo primer plano.

Figura 7.1. Secuencia tipo *Primer plano*

Para poder repetir la misma secuencia a todos los sujetos encuestados, se grabó en una cinta de vídeo doméstico para poder reproducirse repetidas veces.

B) Para el segundo tipo (tipo *Película*) se reprodujeron dos flujos, uno de audio y otro de vídeo, obtenidos de la película española (para poder apreciar la sincronización labial) '*Fugitivas*', a partir de la cinta de video obtenida de un videoclub. Se trata de una secuencia de acción donde se produce un atraco a una administración de loterías.

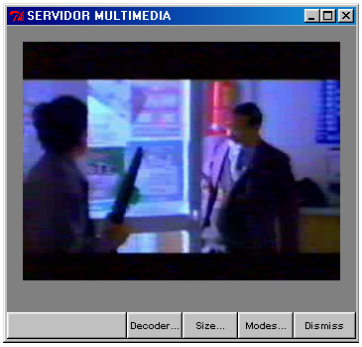


Figura 7.2. Secuencia tipo *Película*

Por tanto, en ambos casos la fuente de los flujos de audio y vídeo será el reproductor de vídeo doméstico.

Los sujetos encuestados visualizaron los dos tipos de secuencias pero en diferentes presentaciones, caracterizadas cada una de ellas por un *grado de sincronización* determinado.

Los grados de sincronización evaluados han sido 3:

- Presentación de los flujos de audio y vídeo *'fuera de sincronía' o 'sin algoritmo'*. En este caso, la secuencia percibida por el individuo será la forma final de la presentación multimedia cuando no se utilice ningún mecanismo de sincronización ni de grupo, ni de sincronización inter-flujo, para corregir la posible asincronía. Este grado de sincronización evaluado se denominará *'Sin Sincronización'*.

- Presentación con audio y vídeo *'en sincronía' o 'sincronizado'* de forma local, pero sin ejecutar el algoritmo de sincronización de grupo. Esta secuencia está caracterizada porque el audio y el vídeo se reproducen sincronizados localmente en cada receptor e independientemente de los demás receptores. Este grado de

sincronización evaluado se denominará *'Con Sincronización Inter-flujo'*.

- Presentación con audio y vídeo en varios receptores, '*sincronizados por el algoritmo de grupo propuesto'*. La secuencia percibida por el individuo será la forma final de la presentación multimedia una vez el algoritmo completo de sincronización propuesto actúe realizando las acciones oportunas para corregir las posibles asincronías entre receptores (de forma distribuida) y entre flujos (de forma local en cada receptor). Este grado de sincronización evaluado se denominará *'Con Sincronización de Grupo'*

Tasa de Reproducción

Para evaluar el efecto de la tasa de reproducción en la apreciación de la sincronización audio-vídeo, por parte de los usuarios encuestados, se han utilizado dos tasas de reproducción diferentes: 15 y 25 tramas por segundo. La elección de dichas tasas ha venido determinada por las condiciones del entorno, tanto hardware como software, en el que se han realizado las pruebas.

El uso de las dos tasas de reproducción nos ha permitido medir el efecto subjetivo de dicha tasa sobre la calidad percibida, aunque éste no es uno de los principales objetivos de nuestra evaluación.

Parámetros de Sincronización

En la siguiente tabla se resumen todos los parámetros que se han utilizado a la hora de realizar las medidas utilizando las secuencias según los tres *Grados de Sincronización* explicados anteriormente: *'Sin Sincronización'*, *'Con Sincronización Inter-flujo' y 'Con Sincronización de Grupo'*.

Los valores de asincronía máxima permitida entre los flujos de audio de los diferentes receptores y entre los flujos de audio-vídeo han sido seleccionados de los valores obtenidos por Steinmetz en [STE96]

Sin Sincronización	Tasa de reproducción del flujo de vídeo	15 tramas/s	25 tramas/s
	Tipo de plano	Primer Plano	Primer Plano
		Película	Película
Con Sincronización *Inter-flujo*	Tasa de reproducción vídeo	15 tramas/s	25 tramas/s
	Tipo de plano	Primer Plano	Primer Plano
		Película	Película
	Asincronía máxima permitida entre audio y vídeo	±80 ms	
Con Sincronización de Grupo	Tasa de reproducción vídeo	15 tramas/s	25 tramas/s
	Tipo de plano	Primer Plano	Primer Plano
		Película	Película
	Asincronía máxima permitida entre receptores de audio	±120 msg.	
	Asincronía máxima permitida entre audio y vídeo	±80 ms	

Tabla 7.1. Parámetros de audio y vídeo de las pruebas

Evaluación de la calidad del Algoritmo de Sincronización

Al visualizar cada una de las secuencias, en primer lugar, el individuo deberá indicar si ha detectado o no algún efecto extraño en la presentación y, en caso de detectarlo, describirlo. Estos efectos detectados serán analizados para ver si es posible que sean causados por el mecanismo de sincronización propuesto o son inherentes al propio sistema de comunicaciones.

A continuación, en segundo lugar, el encuestado deberá realizar un juicio valorativo de la calidad de la sincronización percibida entre los flujos de la aplicación, por medio de la evaluación de las secuencias mediante una *Escala de Calidad, compuesta por 6 notas*, de 0 a 5, donde un valor de 5 supone una sincronización total y un valor de 1 una falta total de sincronización entre flujos. El valor 0 indica indecisión en el usuario a la hora de evaluar.

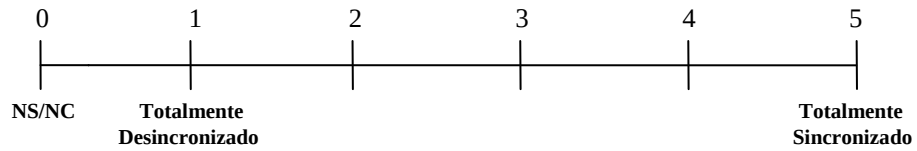


Figura 7.3. Escala de Calidad de la Sincronización

Por último, en tercer lugar, en caso de haber detectado algún tipo de error en el primer paso, el usuario deberá calificar dicho error empleando la siguiente *Escala de Degradación, compuesta por 6 notas*, de 0 a 5, indicando cómo se considera dicho error:

Grados	Efecto
0	Muy Molesto
1	Molesto
2	Ligeramente molesto
3	Perceptible pero no molesto
4	Imperceptible
5	No estoy seguro de si lo aceptaría o no

Tabla 7.2. Escala de degradación

Estas escalas de calidad y de degradación coinciden con las propuestas por la recomendación UIT-R BT. 500-7 ('*Metodología de evaluación subjetiva de la calidad de las imágenes de televisión*', en adelante Rec. UIT-R BT. 500-7), a las que se les ha añadido la nota con valor de 0 que indicará indecisión en el individuo que esté realizando la evaluación.

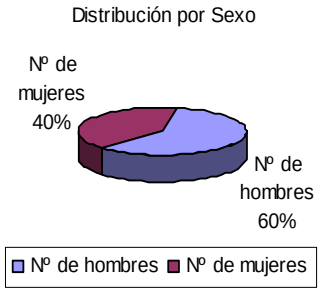
- *Muestra*

Según la Rec. UIT-R BT. 500-7, los individuos observadores no deben ser expertos en la materia, en el sentido de que no estén directamente familiarizados con la calidad de sincronización en sistemas multimedia en su trabajo normal, ni tampoco deben ser evaluadores experimentados. También se indica que el número de individuos necesarios depende de la sensibilidad y la fiabilidad del procedimiento de prueba adoptado y del tamaño previsto del efecto que se busca.

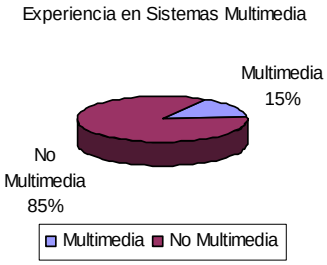
En los diferentes estudios consultados, el tamaño de la muestra poblacional variaba considerablemente. Mientras que para realizar las medidas de [STE96] se utilizaron 107 individuos, en [KOU96] se tomó una muestra de 8 individuos, en [GIL95] se realizaron diferentes pruebas con 12, 24 y 31 individuos y en [GUE97] y [KAL99] la muestra fue de 20 individuos. En [WAT96] la muestra fue de 21 sujetos y en [WAT97] y [WIL00] fue de 24 sujetos. En [ISH01b] y [ISH03] fueron

16 sujetos los evaluados. En la Rec. UIT-R BT. 500-7 se indica que la evaluación deben realizarla, al menos, 15 individuos.

Para la evaluación del algoritmo propuesto, en nuestro caso se ha escogido una muestra de 20 individuos, donde 8 son mujeres y 12 son hombres (figura 7.4a). Ninguno de los individuos participantes tenía experiencia en evaluación subjetiva ni tampoco tenía experiencia con técnicas de sincronización. Tan solo un 15 % de los mismos tenía experiencia como usuario de sistemas multimedia (figura 7.4b).



a) Por sexo



b) Por experiencia con sistemas multimedia

Figura 7.4. Distribución de la muestra de individuos

- *Equipamiento*

Para los experimentos realizados, la reproducción de los flujos de audio y vídeo se ha llevado a cabo sobre los mismos PCs utilizados para la evaluación objetiva, es decir, ordenadores personales con procesador AMD ATHLON 1600+ a 1’4 GHz, con 128 MB de memoria RAM y un monitor de 17 pulgadas. Las características de dichos PCs, tanto del hardware como del software de los sistemas de reproducción, son las típicas de un ordenador de sobremesa con su tarjeta de sonido y sus altavoces. A cada individuo se le ha entregado unos ‘cascos’ para poder aislar el sonido que escuchen de la transmisión de sonidos externos del laboratorio donde se han realizado las pruebas.

La distancia entre la pantalla del monitor y cada individuo, será la misma que si estuviera utilizando el PC, es decir, entre 50 centímetros y 1 metro.

La duración de cada sesión fue de 3 minutos. La transmisión del flujo de vídeo y la posterior reproducción del mismo se realizó con tasas de 15 tramas/s y 25 tramas/s, posibles en entornos hardware y software como los comentados.

El formato de captura del vídeo y del audio corresponde a los mismos formatos que los utilizados en la evaluación objetiva, es decir, formatos H.261 y

GSM, respectivamente. La tasa de reproducción de audio es de 1 LDU de audio cada 20 milisegundos y una LDU de vídeo cada 40 milisegundos.

Steinmetz [STE96] y Kaladji [KAL99] indican que bastaría con 30 y 40 segundos, respectivamente, de presentación para que los usuarios detectasen la existencia de asincronía entre el audio y el vídeo. Estas diferencias son debidas a que en [STE96] se realizaban las medidas añadiendo una asincronía fija (±ms) durante toda la presentación, y no tenía en cuenta otros factores que pudiesen afectar a la sincronía como pueden ser el jitter, la tasa de consumo, la desviación de los relojes, o las acciones de resincronización. Esto era debido a que su objetivo era evaluar los requerimientos de sincronización entre flujos de información. De forma análoga, en [GIL95] se utilizan secuencias de 20 segundos puesto que sólo se evalúa la calidad de la imagen MPEG. En [KAL99] para tener en cuenta el jitter se escogen muestras de mayor duración. En [GUE97], para ver cómo se corregía el efecto de la desviación de los relojes, por ejemplo que el reloj de un receptor se adelantase o se atrasase respecto a su tasa nominal, se tomaron secuencias de 2,5 minutos de duración. Dicho efecto aumentará conforme aumenta el número de tramas consumidas a no ser que actúe el algoritmo de sincronización ejecutando las acciones de resincronización. En [ISH01b] y [ISH03] las sesiones de evaluación duraron 40 minutos y en ellas las pruebas consistían en tests de un mínimo de 25 segundos de duración. En nuestro caso, se han elegido secuencias de 3 minutos, permitiendo que se alcance el régimen estable y se pueda comprobar el comportamiento del algoritmo.

- *Diseño de los Experimentos*

En primer lugar se tuvo que elegir el sistema de captación simultánea de los flujos de vídeo y audio. Como ya se ha comentado, para ambas secuencia (*tipo primer plano* y *tipo película*) la fuente de los flujos de audio y vídeo fue un reproductor de vídeo doméstico, del cual se conectaron las salidas de audio y vídeo a las entradas de las tarjetas capturadora de vídeo y de adquisición de sonido del servidor multimedia.

Para sincronizar los relojes de los dispositivos involucrados tanto del transmisor como de los receptores se utilizó el protocolo NTP.

En los experimentos se ha considerado el flujo de audio como flujo *maestro* y el de vídeo como flujo *esclavo*.

- *Diseño, Análisis Experimental y Procedimiento*

Respecto a las medidas, se corresponden con pruebas de tipo *intraindividuo,* en las que todos los individuos tienen el mismo tratamiento experimental. La tarea consistió en que cada sujeto visualizase las dos secuencias descritas con los tres grados de sincronización descritos (‘*Sin Sincronización’, ‘Con Sincronización Inter-flujo’* y ’*Con Sincronización de grupo’*) y con diferentes tasas de transmisión (‘*con tasa 25 tramas/s*’ y ‘*con tasa 15 tramas/s*’).

De todos los aspectos evaluados, el que realmente nos interesa será la calidad de sincronización del algoritmo propuesto. Se ha incluido la variación de la tasa de reproducción para constatar cómo afecta la tasa de reproducción a la percepción humana, aspecto que ya ha sido demostrado en los trabajos consultados ya mencionados, y así dejar clara la posibilidad de utilizar la misma plataforma para la evaluación de otros algoritmos de sincronización multimedia.

Antes de realizar los ejercicios de evaluación, se familiarizó a los individuos con el método de evaluación y se les explicó en qué consistía la prueba y cómo se debía evaluar rellenando el correspondiente formulario de respuesta para cada prueba. A continuación, a cada individuo se le presentaron las 6 secuencias del vídeo tipo *primer plano* y las 6 secuencias del vídeo tipo *película de acción*, descritas en la tabla 7.3.

PRUEBA	Tipo De secuencia	Tasa de reproducción del flujo de vídeo	Sincronización
A	Primer Plano	25 tramas/s	Sin Sincronización
B	Película	25 tramas/s	
C	Primer Plano	15 tramas/s	
D	Película	15 tramas/s	
E	Primer Plano	25 tramas/s	Con Sincronización Inter-flujo
F	Película	25 tramas/s	
G	Primer Plano	15 tramas/s	
H	Película	15 tramas/s	
I	Primer Plano	25 tramas/s	Con Sincronización de Grupo
J	Película	25 tramas/s	
K	Primer Plano	15 tramas/s	
L	Película	15 tramas/s	

Tabla 7.3. Pruebas realizadas

Para cada una de las presentaciones, el individuo debía contestar un *Cuestionario de Evaluación* (figura 7.5) donde realizaba el juicio valorativo de la calidad de la presentación de cada una de las pruebas.

CUESTIONARIO DE EVALUACIÓN SUBJETIVA

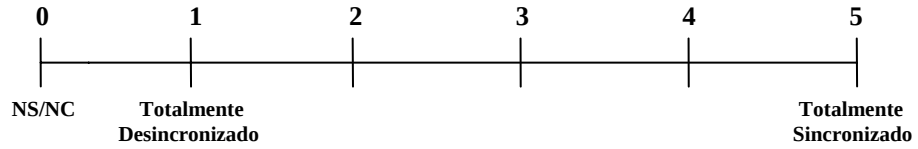
Hombre ( ) Mujer ( )

1.- Mientras visualizaba el vídeo, ¿ha detectado algún efecto que consideres **artificial** o **extraño**? (Contestar Sí o No)_____. En el caso de haber contestado que Sí, por favor, intente describirlo brevemente en el siguiente espacio:

2.- Evalúe la presentación, en función de la sincronización entre el audio y el vídeo, utilizando la siguiente escala de valoración:

0	1	2	3	4	5

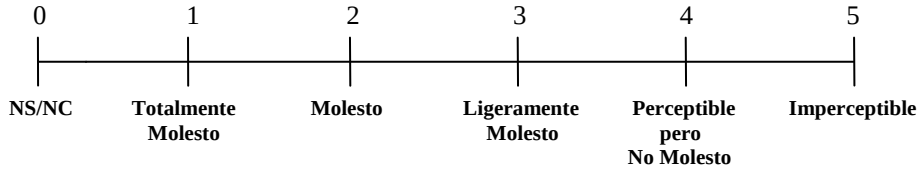
Escala de valoración



3.- Si ha notado un error en la sincronización, ¿Cómo calificarías dicho error si tuviera que ver todos los vídeos con dicho error?_____

0	1	2	3	4	5

Escala de valoración



(Por favor, visualice la siguiente presentación y rellene otro formulario)

Figura 7.5. Cuestionario de Evaluación Subjetiva

7.2.2. Evaluación y Resultados del Algoritmo de Sincronización obtenidos en la aplicación de Aprendizaje a Distancia

A continuación se muestran los resultados de la evaluación subjetiva de la aplicación en los dos entornos propuestos.

De los resultados obtenidos, que se muestran en los siguientes subapartados, se ha verificado su coherencia, obteniendo los valores medios, máximo y mínimo. De este modo se expresa la calidad subjetiva de las muestras de test mediante datos MOS (*Mean Opinion Score*).

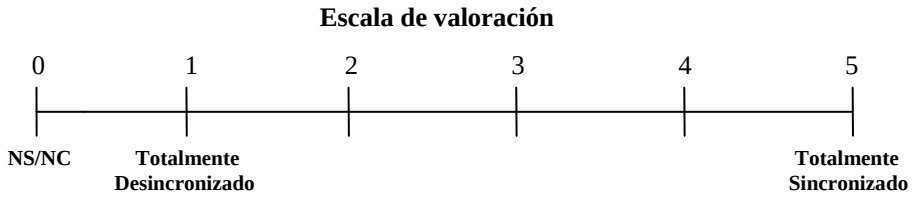
7.2.2.1. Evaluación en entorno LAN

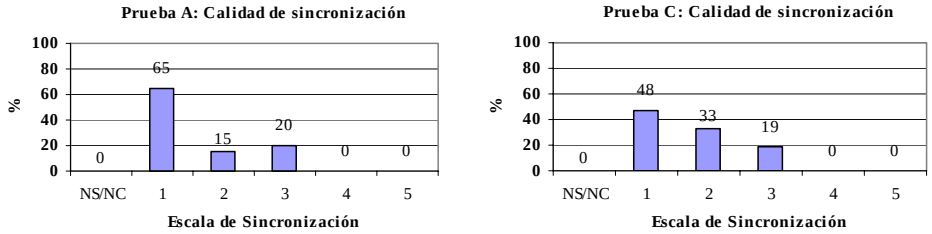
- ***Secuencia tipo Primer Plano***

Los resultados que se muestran a continuación son los obtenidos de las pruebas A, C (*Sin Sincronización*), E, G (*Con Sincronización Inter-flujo*), I y K (*Con Sincronización de Grupo*), correspondientes a las sesiones en las que se distribuía vídeo y audio correspondientes a una secuencia de tipo *Primer Plano*.

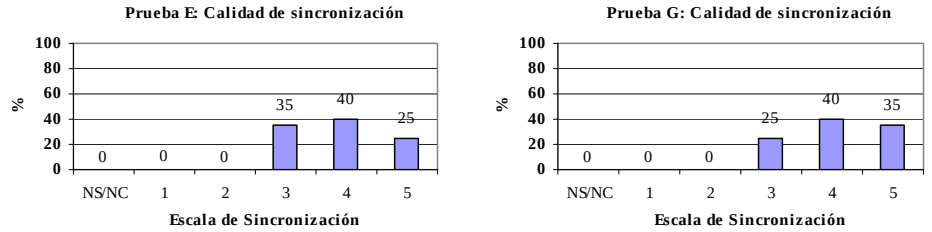
Calidad de la Sincronización

A continuación se presentan los resultados obtenidos de la evaluación de cada una de las pruebas anteriores. En el eje y de las gráficas presentadas en las mismas se muestra el porcentaje de individuos que han otorgado cada una de las calificaciones. En el eje *x*, aparece la escala de evaluación mostrada en la figura.

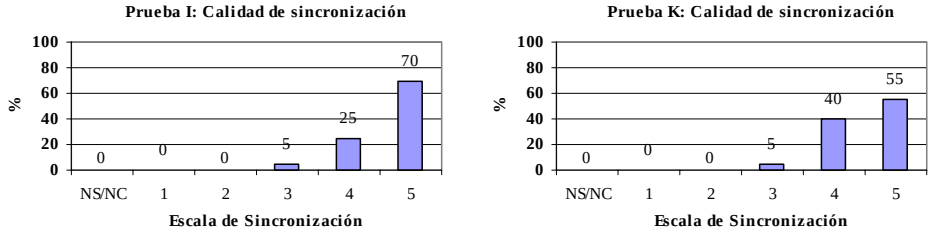




a) *Sin Sincronización*, 25 y 15 tramas/s



b) *Con Sincronización Inter-flujo*, 25 y 15 tramas/s



c) *Con Sincronización de Grupo*, 25 y 15 tramas/s

Figura 7.6. Valoración de la Calidad de la Sincronización
(Primer plano - LAN)

Para comparar las tres situaciones, si ahora superponemos las gráficas, utilizando líneas en vez de barras, obtenemos las siguientes gráficas:

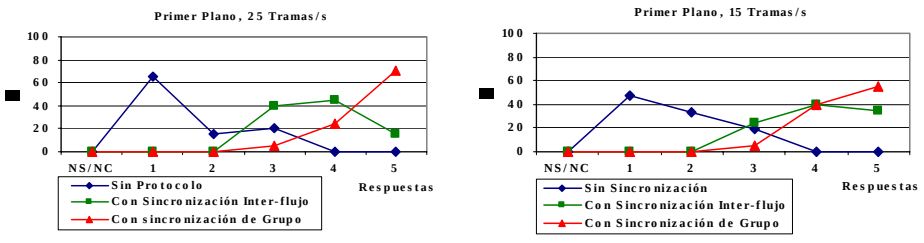


Figura 7.7. Valoración de la Calidad de la Sincronización
Según la tasa de transmisión
(Primer plano - LAN)

En primer lugar, se puede destacar la clara diferencia de los resultados cuando no se utiliza ningún algoritmo de sincronización que corrija las desviaciones entre los procesos reproductores de los flujos de audio y el vídeo. Como era de esperar, los individuos han valorado peor la secuencia en la que no se ejecutaba algoritmo de sincronización alguno.

Cabe destacar la notable valoración recibida respecto a los resultados obtenidos con el algoritmo de sincronización de grupo propuesto en la Tesis, lo cual indica que las acciones que el algoritmo ha ejecutado han evitado que se supere una asincronía claramente perceptible por los individuos.

Además, también se puede señalar el hecho de que los resultados obtenidos en las pruebas con el algoritmo de sincronización de grupo han sido mejores que las obtenidas con la sincronización inter-flujo únicamente. Esto puede ser debido a que en nuestro algoritmo sólo se realizan acciones de resincroniación cuando se superan ciertos umbrales, con lo que se producen menos '*saltos*' y '*pausas*'. Cuando se utiliza sólo sincronización inter-flujo, el proceso reproductor de audio envía mensajes *mbus* de sincronización al reproductor de vídeo cada ráfaga de audio (*talkspurt*) que recibe con lo que se producen muchos más '*saltos'* y '*pausas'*.

A continuación se muestran los valores medios de las evaluaciones anteriores junto con los valores mínimos y máximos.

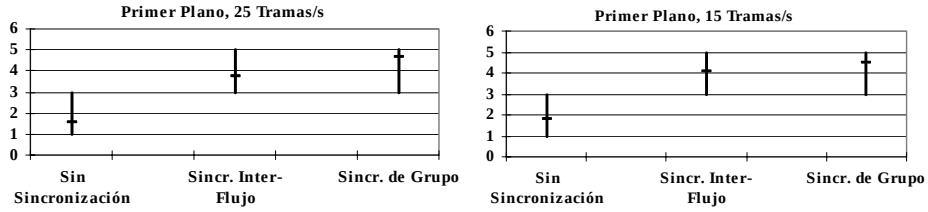


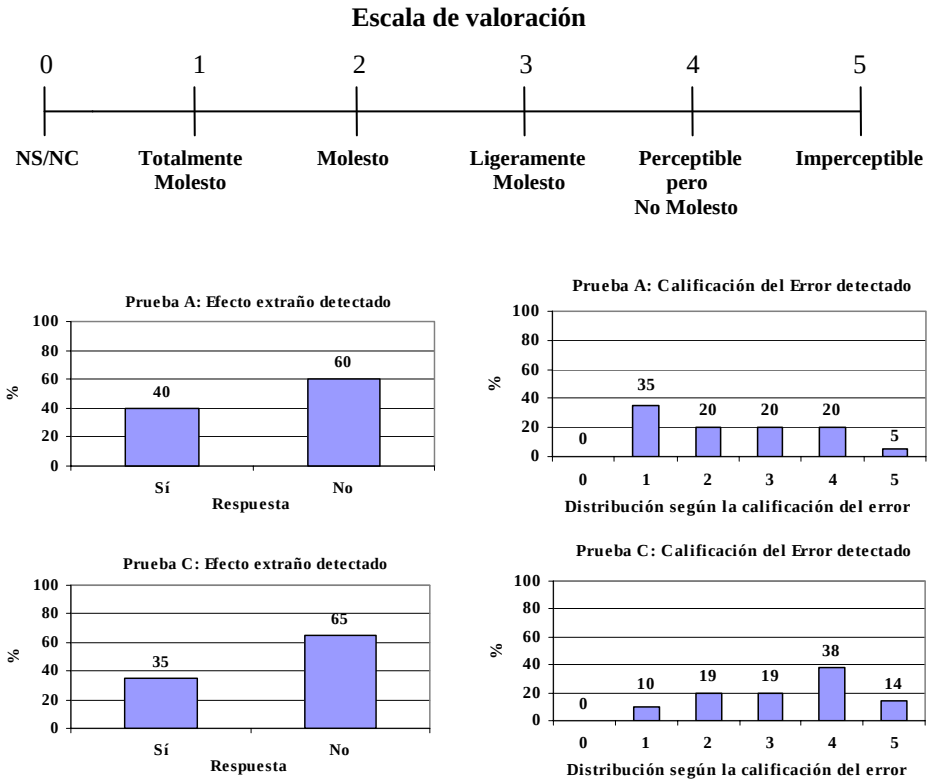
Figura 7.8. Valoración media, máxima y mínima de las secuencias de tipo Primer Plano (LAN)

Se puede observar cómo las secuencias con el algoritmo de sincronización de grupo poseen unos márgenes de valoración similares a las secuencias con sólo sincronización inter-flujo, pero se han alcanzado valores de puntuación mayores por lo que la media que se obtiene es incluso mayor. También se puede apreciar, la tendencia evidente a mejorar la consideración sobre la calidad de la sincronización cuando se corrige la asincronía mediante un algoritmo de sincronización, y quizás lo más interesante, una tendencia similar en cuanto a la consideración de la calidad

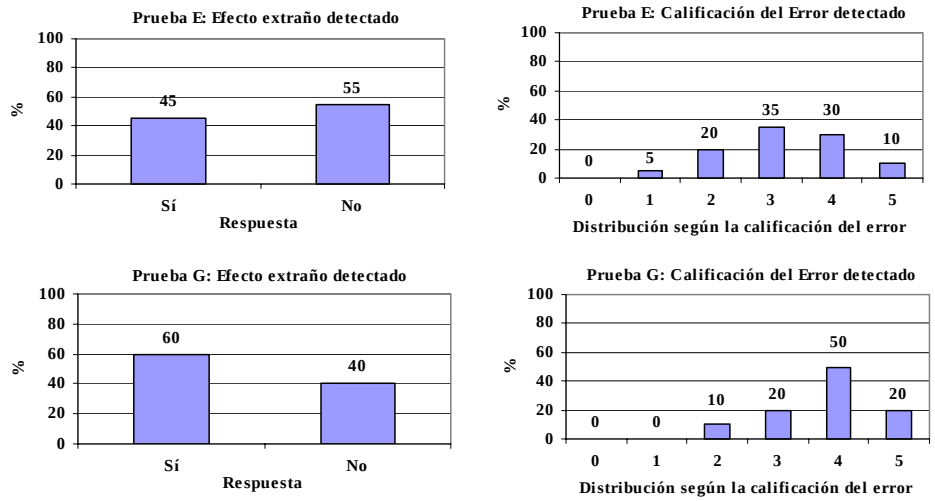
de la sincronización cuando se ejecuta el algoritmo de sincronización inter-flujo solamente y cuando se ejecuta el algoritmo de sincronización de grupo propuesto.

Efectos Extraños detectados por los usuarios

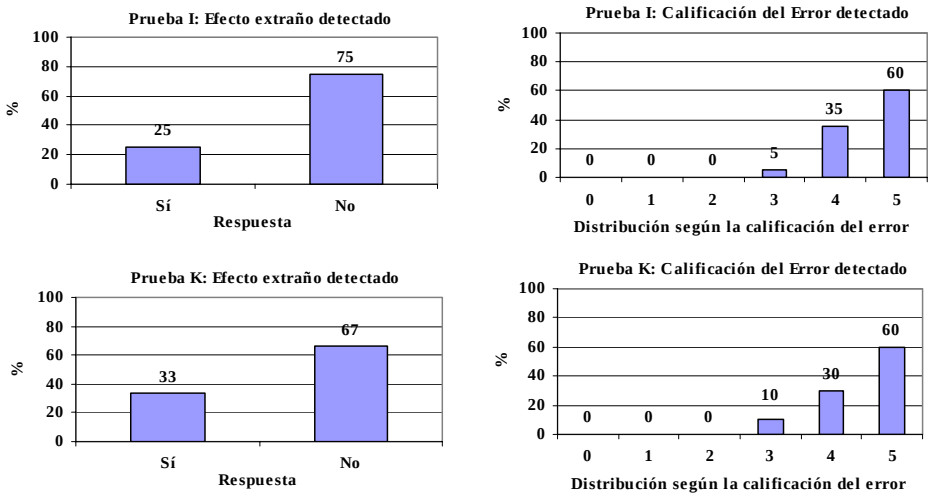
A continuación, se procede a analizar los efectos extraños en la reproducción de las secuencias detectados por los sujetos. En la siguiente figura se representan los porcentajes de sujetos que han detectado algún efecto extraño o artificial en las secuencias visualizadas y la calificación de dicho efecto realizada por los mismos. En el eje *x*, aparece la escala de evaluación incluida en la figura.



a) *Sin sincronización*, 25 y 15 tramas/s



b) *Con Sincronización Inter-flujo*, 25 y 15 tramas/s



c) *Con Sincronización de Grupo*, 25 y 15 tramas/s

Figura 7.9. Efectos extraños detectados por los usuarios
(Primer plano - LAN)

Como efecto extraño, y tomando como base los comentarios realizados en el test por los individuos, se puede concluir que son debidos básicamente a:

- Las asincronías detectadas.
- En los casos b) y c), a las propias acciones de resincronización que producen '*saltos*' y '*pausas*' en la reproducción.

- A la baja calidad de la reproducción en el propio PC, es decir, al efecto que produce la reproducción del flujo de vídeo a unas tasas menores que las que los usuarios están acostumbrados a ver en televisión.

A pesar de los usuarios que han detectado efectos artificiales, en los casos *Con Sincronización Inter-flujo* y *Con Sincronización de Grupo*¸ la mayoría los han calificado como imperceptibles o poco molestos.

En el caso del algoritmo de grupo propuesto, pruebas I y K, Los resultados demuestran que el algoritmo utiliza un mecanismo de resincronización eficiente sin generar acciones que provoquen grandes '*saltos*' y '*pausas*' que detecten los sujetos como efectos extraños. En todo caso, los bajos porcentajes de sujetos que detectan efectos extraños los califican como imperceptibles o perceptibles pero no molestos, mientras que sólo una minoría de ellos los considera ligeramente molestos.

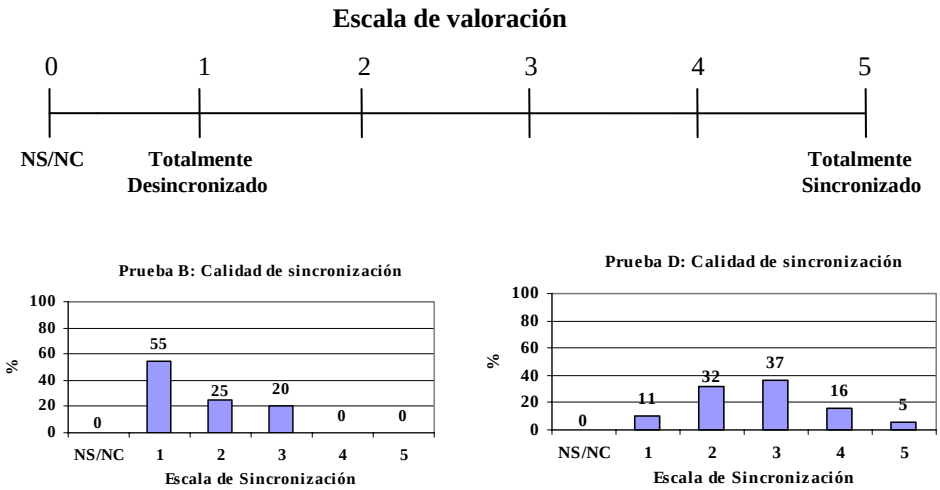
Una *primera conclusión fundamental* que se puede obtener de los resultados obtenidos es que *desde el punto de vista subjetivo, la calidad de sincronización del algoritmo de sincronización de grupo, se ha valorado incluso mejor que las secuencias Con Sincronización Inter-flujo. Por otra parte, aunque un porcentaje, aproximado, del 30% de los sujetos ha detectado algún efecto extraño o artificial, dicho efecto se ha calificado como aceptable o imperceptible, de forma que no modificaría la opinión afirmativa de visualizar las secuencias de vídeo y audio con dicha calidad de sincronización.*

- ***Secuencia tipo Película***

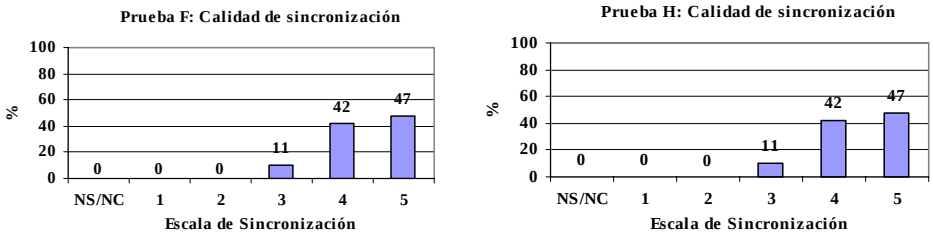
En este caso se muestran los resultados obtenidos de las pruebas B, D (*Sin Sincronización*), F, H (*Con Sincronización Inter-flujo*), J y L (*Con Sincronización de Grupo*), correspondientes a las sesiones en las que se distribuía vídeo y audio correspondientes a una secuencia de tipo *Película*.

Calidad de Sincronización

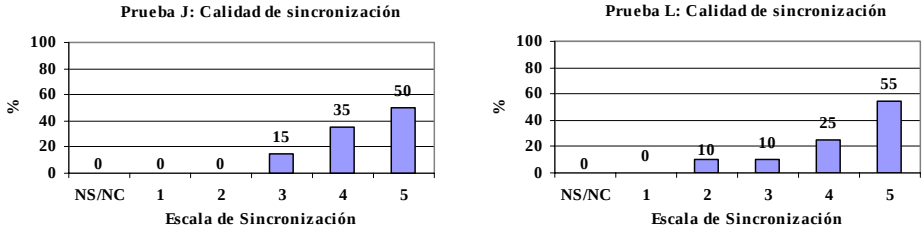
Las gráficas de la siguiente figura presentan los resultados obtenidos de la evaluación de cada una de las pruebas anteriores. En el eje *y* de las gráficas se muestra el porcentaje de individuos que han otorgado cada una de las calificaciones, mientras que en el eje *x*, aparece la escala de evaluación que también se muestra en la figura.



a) *Sin Sincronización*, 25 y 15 tramas/s



b) *Con Sincronización Inter-flujo*, 25 y 15 tramas/s



c) *Con Sincronización de Grupo*, 25 y 15 tramas/s

Figura 7.10. Valoración de la Calidad de la Sincronización (Película - LAN)

Si se superponen los resultados correspondientes a la misma tasa de transmisión, se obtienen las gráficas de la figura 7.11.

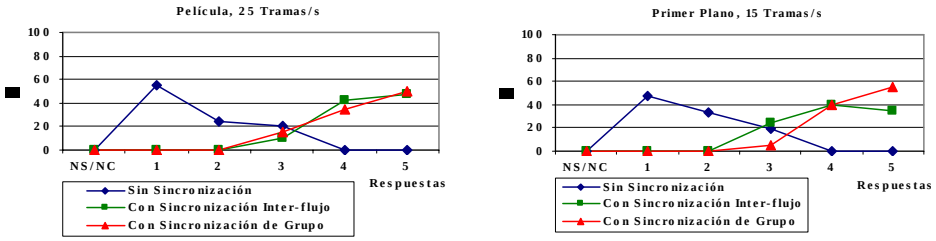


Figura 7.11. Valoración de la Calidad de la Sincronización según la tasa de transmisión (Película - LAN)

En este caso también se puede observar una clara diferencia de los resultados cuando no se utiliza ningún algoritmo de sincronización que corrija las desviaciones entre los procesos reproductores de los flujos de audio y el vídeo. Además, igual que en el caso anterior, los resultados obtenidos con las secuencias *Con Sincronización de Grupo* propuesto son similares a los obtenidos con las secuencias *Con Sincronización Inter-flujo*. Esta valoración favorable otorgada por los individuos indica que las acciones de resincronización que el algoritmo ha ejecutado, han evitado que se supere una asincronía claramente perceptible por los individuos.

En la siguiente figura se muestra los valores medio, máximo y mínimo de las calificaciones obtenidas.

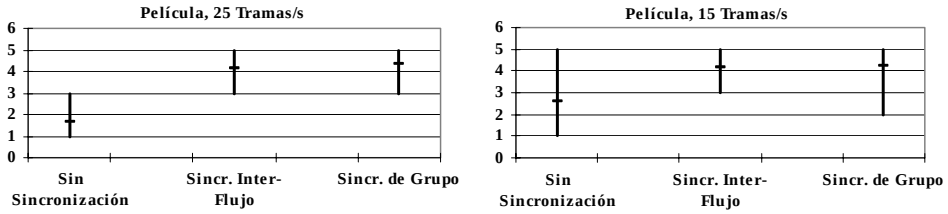


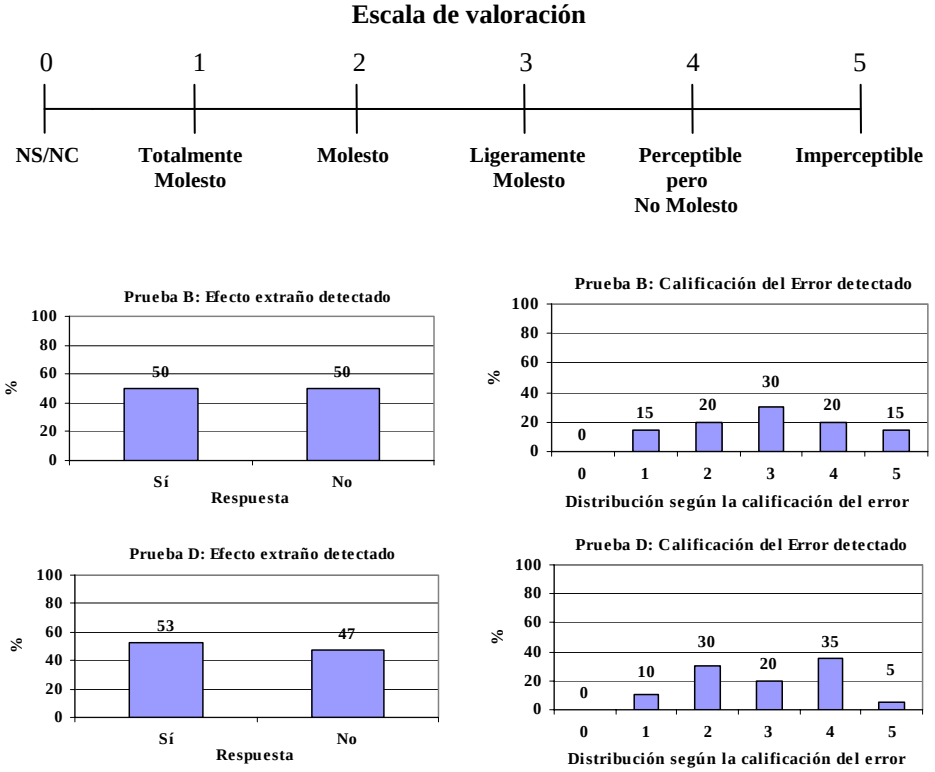
Figura 7.12. Valoración media, máxima y mínima de las secuencias de Tipo Película (LAN)

En este caso también se observa que las secuencias *Con Sincronización de Grupo* poseen unos márgenes de valoración similares a las secuencias *Con Sincronización Inter-flujo*, pero ligeramente mayores por lo que la media obtenida es ligeramente más alta (los motivos de esto último son los mismos que han sido comentados para el caso de secuencias de tipo Primer Plano).

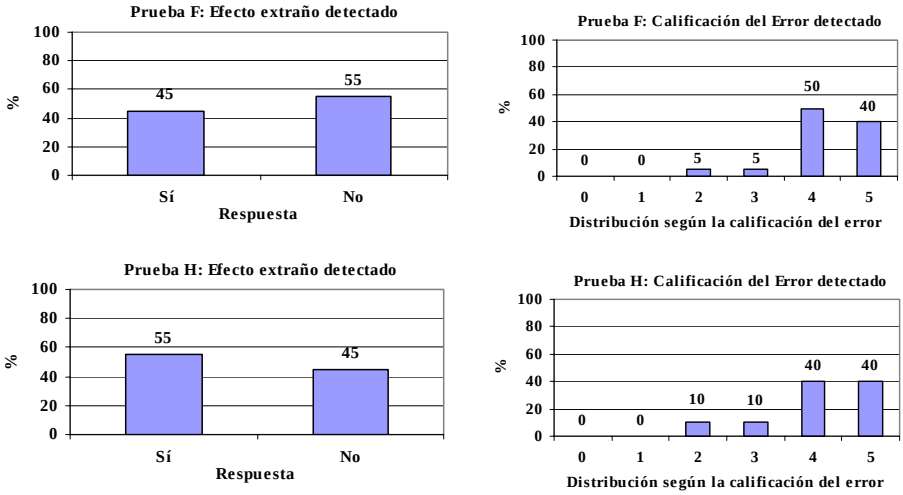
También queda patente en el caso de secuencias tipo Película la mejora de la calidad de la sincronización cuando la asincronía es corregida mediante un algoritmo de sincronización.

Efectos Extraños detectados por los usuarios

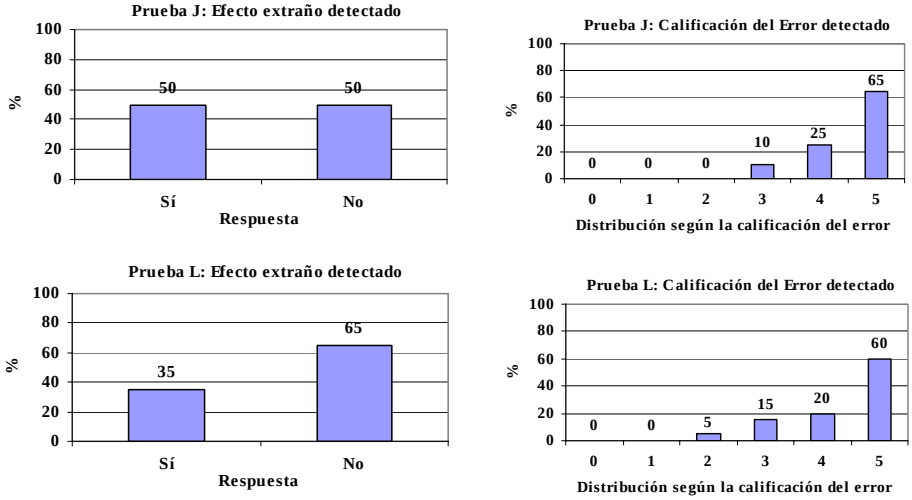
Veamos ahora si se han detectado efectos extraños en la reproducción de la secuencias de tipo Película. Como en el caso anterior, en la siguiente figura se representan los porcentajes de sujetos que han detectado algún efecto extraño o artificial en las secuencias de *tipo Película* visualizadas y la calificación de dicho efecto realizada por los mismos.



a) *Sin sincronización*, 25 y 15 tramas/s



b) *Con Sincronización Inter-flujo*, 25 y 15 tramas/s



c) *Con Sincronización de Grupo*, 25 y 15 tramas/s

Figura 7.13. Efectos Extraños detectados por los usuarios (Película - LAN)

Como efectos extraños, y tomando como base los comentarios realizados en el test por los individuos, se puede concluir que son debidos básicamente a las mismas causas que en el caso de las secuencias de Primer Plano, aunque, en este caso, gran parte de los mismos tienen origen en la dificultad de apreciar bien los

movimientos de la película debido a la tasa de transmisión (15 y 25 tramas por segundo) y al sistema de codificación (H.261) utilizados.

Para este tipo de secuencias, a pesar de que ciertos usuarios han detectado efectos artificiales en los casos *Con Sincronización Inter-flujo* y *Con Sincronización de Grupo*¸ la mayoría los han calificado como imperceptibles o poco molestos.

En el caso del algoritmo de grupo propuesto, pruebas J y L, los resultados demuestran que, en este caso, también, el algoritmo utiliza un mecanismo de resincronización eficiente sin generar acciones que provoquen grandes '*saltos*' y '*pausas*' en la reproducción detectables por los sujetos evaluados. En todo caso, los sujetos que han detectado algún efecto extraño los califican, en su mayoría, como imperceptibles o perceptibles pero no molestos.

Para este tipo de secuencias también se puede concluir que d*esde el punto de vista subjetivo, la calidad de sincronización del algoritmo de sincronización de grupo, se ha valorado incluso mejor que las secuencias Con Sincronización Inter-flujo. Por otra parte, aunque un cierto porcentaje de los usuarios hayan detectado algún efecto extraño, lo han calificado como aceptable o imperceptible, de forma que no modificaría la opinión afirmativa de visualizar las secuencias de vídeo y audio con dicha calidad de sincronización.*

7.2.2.2. Evaluación en entorno CAMPUS

A continuación se van a mostrar los resultados de la evaluación de la aplicación en entorno CAMPUS, con el servidor de flujos multimedia ubicado en el Campus de Valencia y los receptores (y, por tanto, los sujetos evaluadores) en el campus de Gandia.

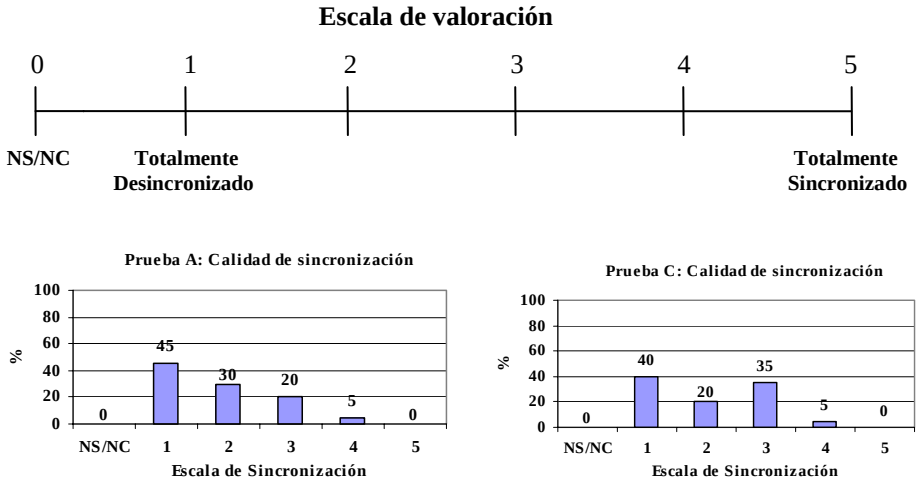
- ***Secuencia Tipo Primer Plano***

Los resultados presentados en este apartado se corresponden con los obtenidos de las pruebas A, C (*Sin Sincronización*), E, G (*Con Sincronización Inter-flujo*), I y K (*Con Sincronización de Grupo*), en las que se distribuía vídeo y audio correspondientes a una secuencia de tipo *Primer Plano*.

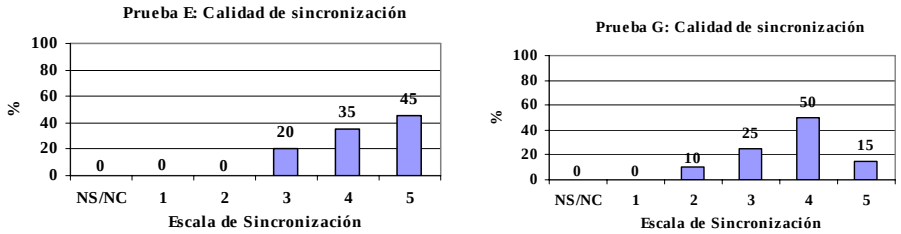
Calidad de Sincronización

En la figura 7.14 se muestran las gráficas con la valoración de la calidad de la sincronización en cada una de las secuencias evaluadas. En la figura 7.15 se han

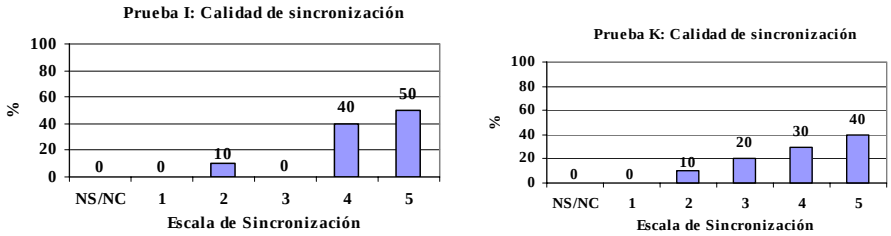
superpuesto los resultados para las secuencias con tasas de 15 y 25 tramas por segundo.



a) *Sin sincronización*, 25 y 15 tramas/s



b) *Con Sincronización Inter-flujo*, 25 y 15 tramas/s



c) *Con Sincronización de Grupo*, 25 y 15 tramas/s

Figura 7.14. Valoración de la Calidad de la Sincronización (Primer Plano - CAMPUS)

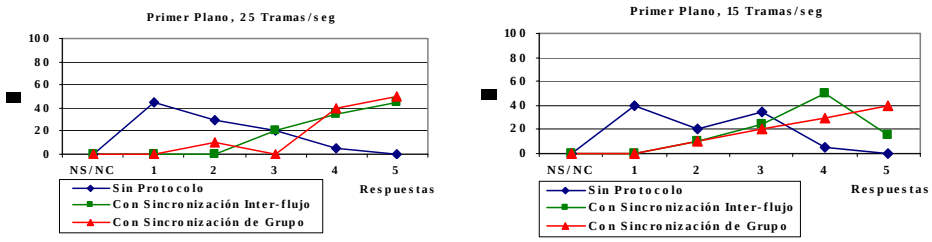


Figura 7.15. Valoración de la Calidad de la Sincronización según la tasa de transmisión (Primer Plano - CAMPUS)

Como es lógico en este caso también presentan una clara diferencia los resultados cuando no se utiliza ningún algoritmo de sincronización. En este caso, también los individuos han valorado peor la secuencia 'Sin Sincronización'.

También se ha obtenido un buen resultado en cuanto a la valoración de la calidad de la sincronización de las secuencias donde se ejecutaba el algoritmo de sincronización de grupo propuesto, incluso mejorando el obtenido en las secuencias donde sólo se ejecutaba la sincronización inter-flujo, por los motivos ya expuestos anteriormente.

En la figura 7.16 se muestran los valores medio, máximo y mínimo obtenidos en dichas pruebas.

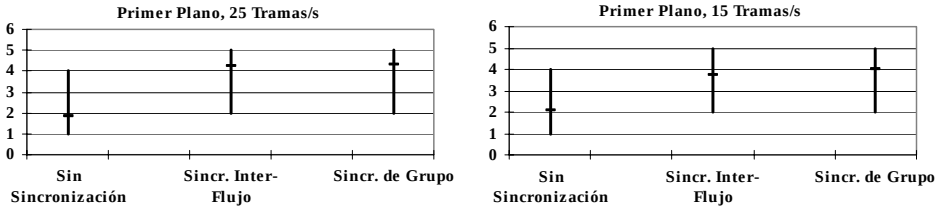
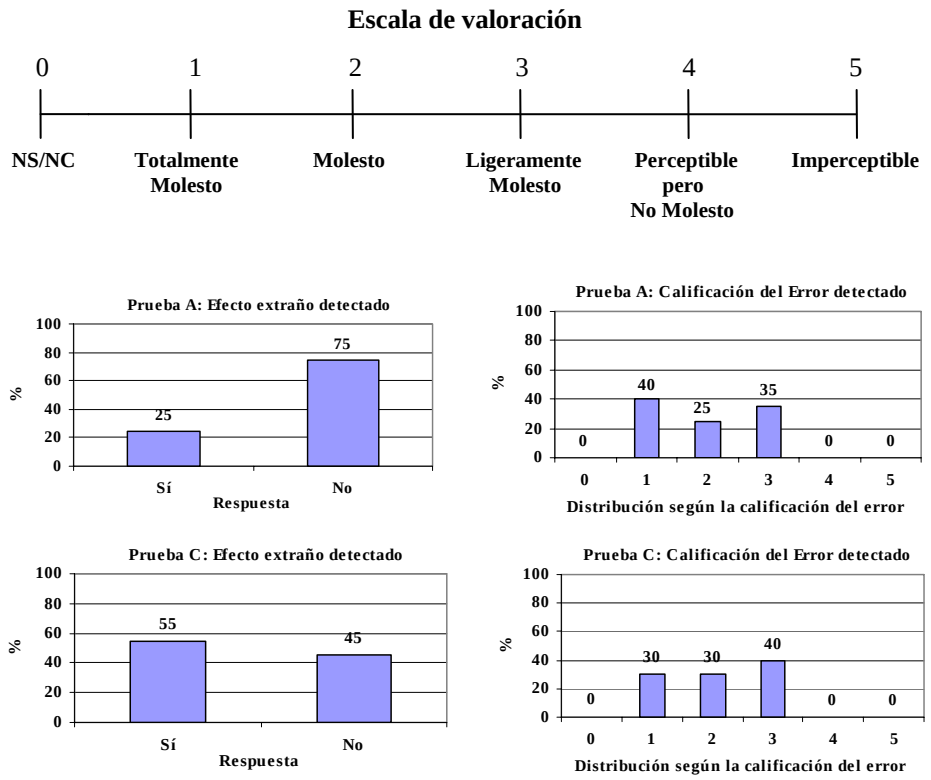


Figura 7.16. Valoración media, máxima y mínima de las secuencias de tipo Primer Plano (CAMPUS)

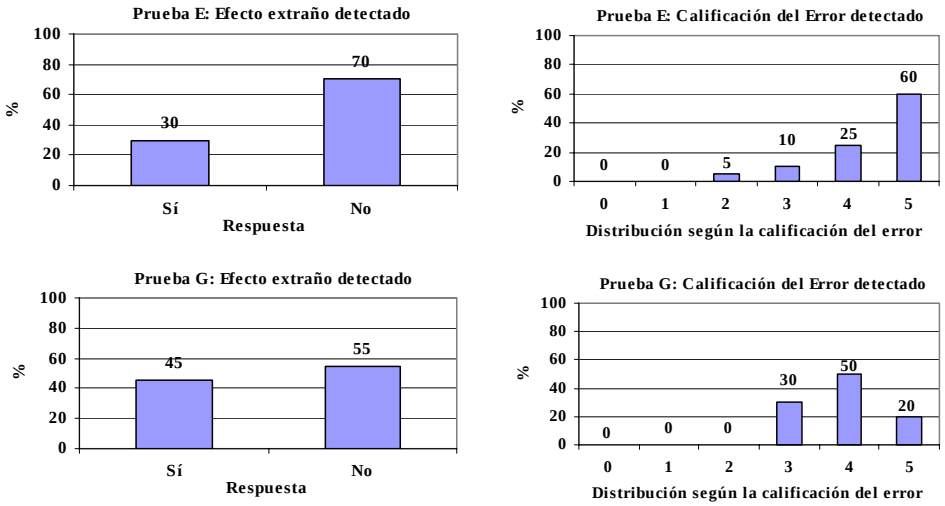
En este caso el margen dinámico de respuestas ha sido un poco más amplio pero prácticamente se obtienen valores medios similares a la evaluación en entorno LAN, apreciándose también la tendencia evidente a mejorar la consideración sobre la calidad de la sincronización cuando se corrige la asincronía mediante un algoritmo de sincronización.

Efectos Extraños detectados por los usuarios

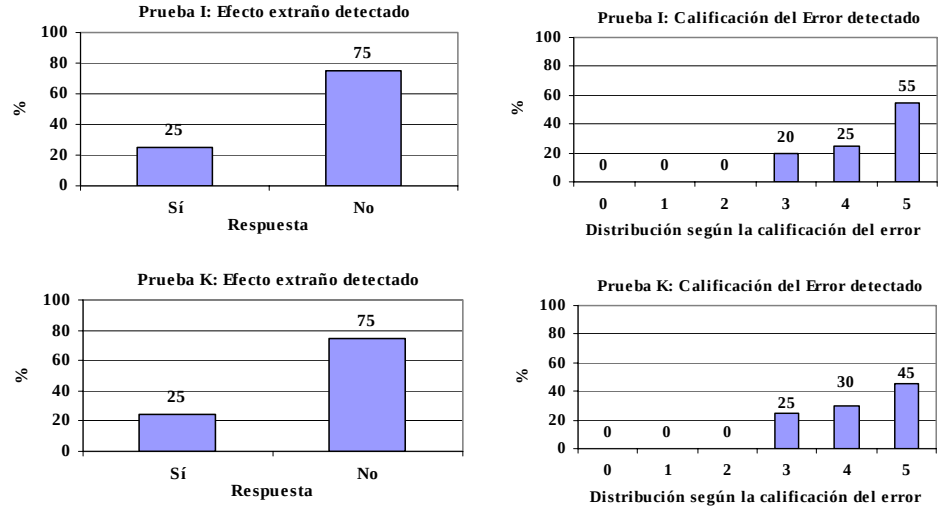
En la siguiente figura se presentan los porcentajes de sujetos que han detectado efectos extraños o artificiales en las secuencias visualizadas y la calificación que les han otorgado a los mismos.



a) *Sin Sincronización*, 25 y 15 tramas/s



b) *Con Sincronización Inter-flujo*, 25 y 15 tramas/s



c) *Con Sincronización de Grupo*, 25 y 15 tramas/s

Figura 7.17. Efectos extraños detectados por los usuarios (Primer Plano - CAMPUS)

En este caso los usuarios detectan algunos efectos extraños debidos a los mismos motivos explicados en la evaluación en el entorno anterior. Se puede apreciar que en este entorno también se obtienen peores resultados en la calificación de las secuencias con tasa de transmisión de 15 tramas por segundo.

A pesar haber detectado efectos artificiales o extraños en las secuencias sincronizadas (*Con Sincronización Inter-flujo* y *Con Sincronización de Grupo*), la mayoría los han calificado como imperceptibles o poco molestos. Se aprecia que también se ha valorado mejor las secuencias *Con Sincronización de Grupo*, bajando el porcentaje de usuarios que detectaron efectos extraños con respecto a las secuencias *Con Sincronización Inter-flujo*.

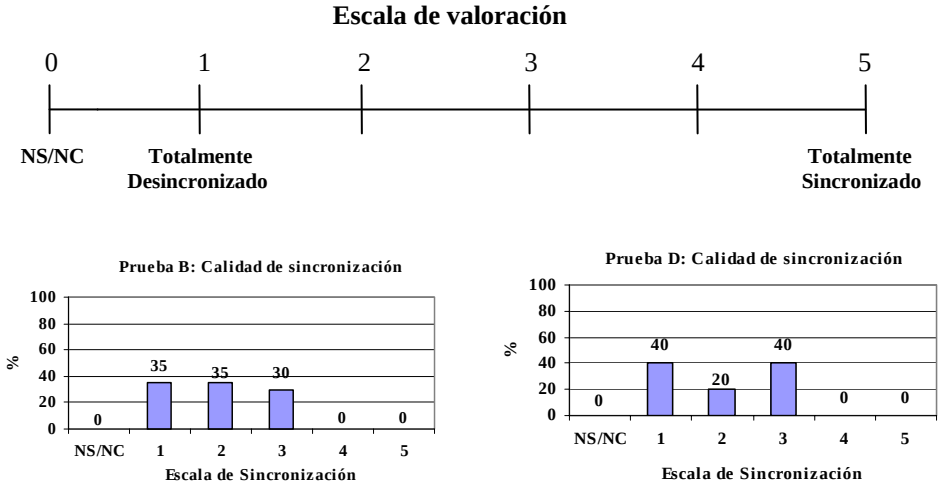
Los resultados demuestran, también en entorno CAMPUS, que el algoritmo utiliza un mecanismo de resincronización eficiente y, por tanto, se puede concluir que d*esde el punto de vista subjetivo, la calidad de sincronización del algoritmo de sincronización de grupo obtenida en este entorno garantiza la aplicabilidad del algoritmo en este tipo de aplicación en el mismo.*

- ***Secuencia Tipo Película***

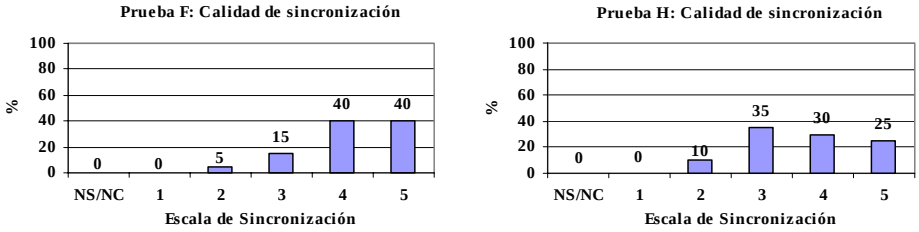
Las pruebas en las que se transmitieron secuencias de tipo *Película* son los las pruebas B, D (*Sin Sincronización*), F, H (*Con Sincronización Inter-flujo*), J y L (*Con Sincronización de Grupo*). A continuación se muestran los resultados obtenidos en las mismas.

Calidad de Sincronización

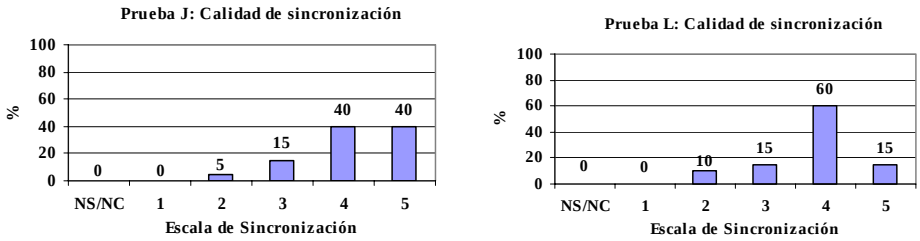
En cuanto a la valoración de la calidad de sincronización obtenida de los usuarios se muestran las gráficas correspondientes en la figura 7.18. En la figura 7.19 se han superpuesto los 3 resultados correspondientes a cada tasa de transmisión.



a) *Sin sincronización*, 25 y 15 tramas/s



b) *Con Sincronización Inter-flujo*, 25 y 15 tramas/s



c) *Con Sincronización de Grupo*, 25 y 15 tramas/s

Figura 7.18. Valoración de la Calidad de la Sincronización
(Película - CAMPUS)

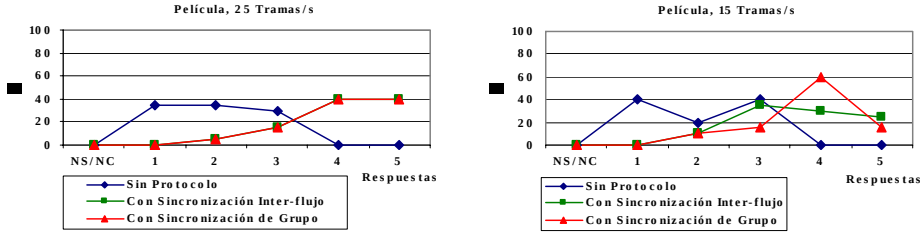


Figura 7.19. Valoración de la Calidad de la Sincronización
según la tasa de reproducción
(Película - CAMPUS)

Los resultados son similares a los obtenidos en entorno LAN. Se puede observar la mejora de la apreciación de la calidad de sincronización cuando se utiliza algún algoritmo de sincronización ya sea inter-flujo o de grupo.

Si observamos los valores medio, máximo y mínimo de las valoraciones (figura 7.20) observamos que los resultados son también similares a los obtenidos en un entorno LAN para el mismo tipo de secuencia. En este caso, el margen dinámico de las respuestas es un poco más amplio en las referentes a las secuencias

Con Sincronización de Grupo con respecto a las secuencias *Con Sincronización Inter-flujo*.

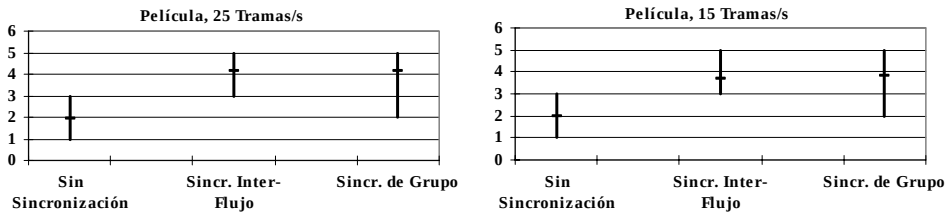
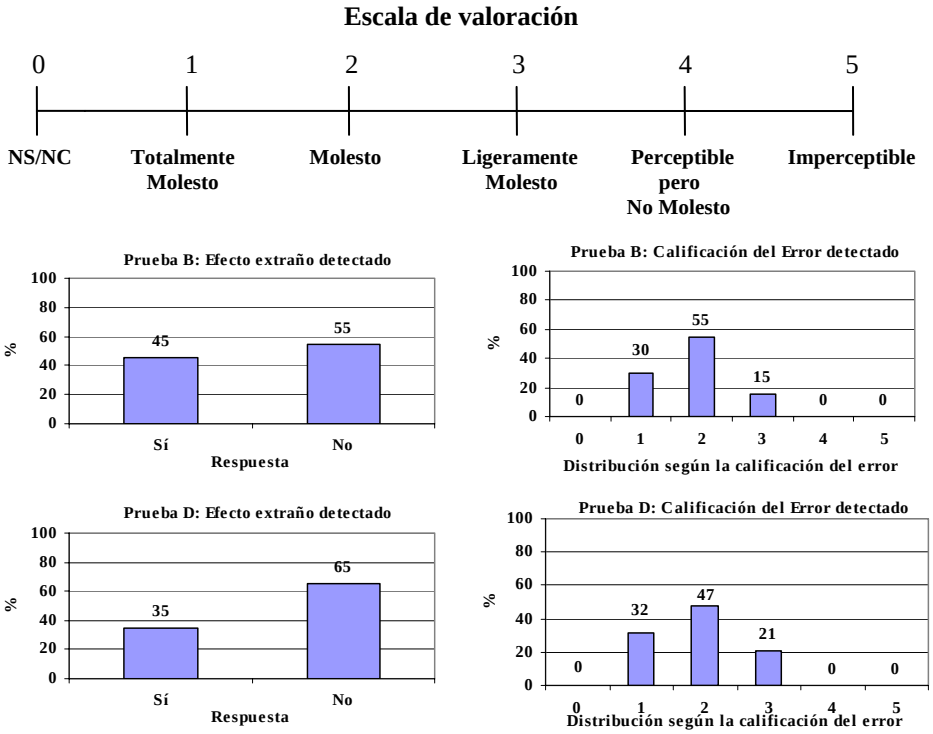


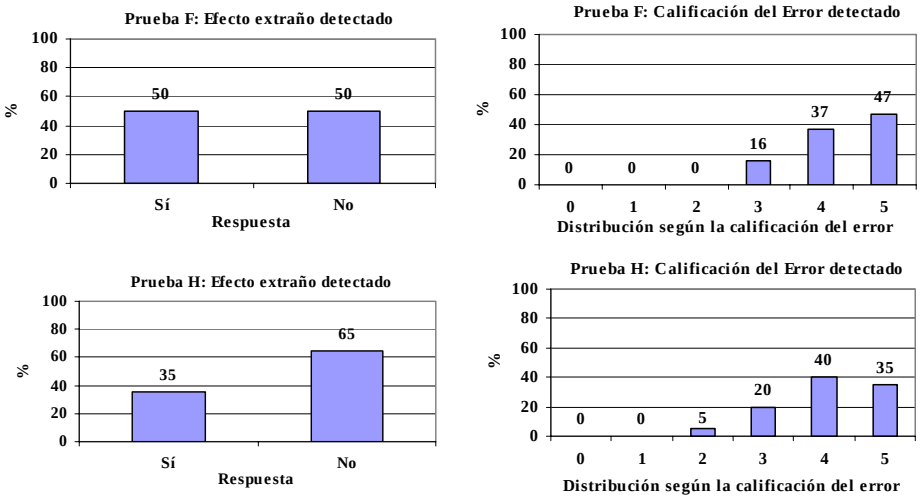
Figura 7.20. Valoración media, máxima y mínima de las secuencias de Tipo Película (CAMPUS)

Efectos Extraños detectados por los usuarios

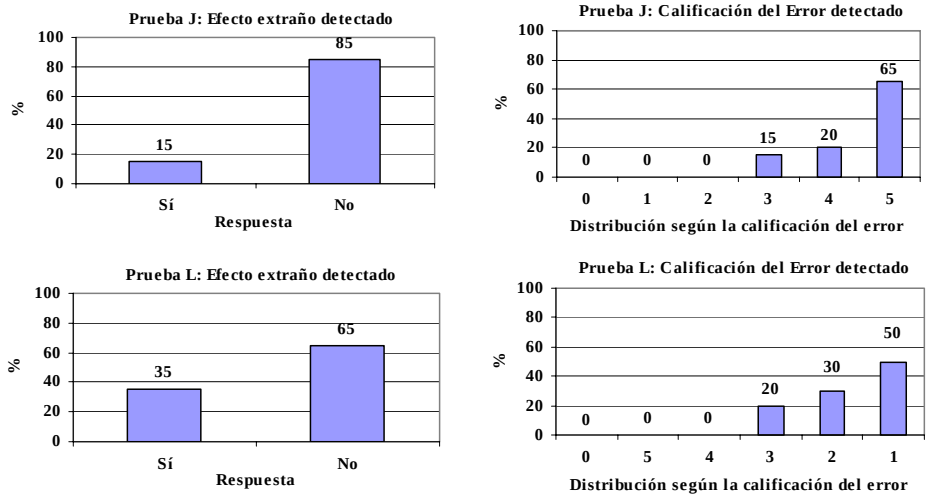
En este caso los individuos también han detectado algunos efectos extraños en la reproducción, la mayoría de ellos debidos a las mismas causas que en un entorno LAN, sobre todo a la mala imagen visualizada de la película. En la figura 7.21 se muestran los resultados obtenidos para la secuencia de tipo Película.



a) *Sin sincronización*, 25 y 15 tramas/s



b) *Con Sincronización Inter-flujo*, 25 y 15 tramas/s



c) *Con Sincronización de Grupo*, 25 y 15 tramas/s

Figura 7.21. Efectos extraños detectados por los usuarios (Película - CAMPUS)

Los resultados también muestran en este caso, como en los anteriores, que los usuarios que han detectado efectos artificiales, al visualizar las secuencias sincronizadas, es decir, *Con Sincronización Inter-flujo* y *Con Sincronización de Grupo*¸ la mayoría los han calificado como imperceptibles o poco molestos.

También podemos concluir en este caso que d*esde el punto de vista subjetivo, la calidad de sincronización del algoritmo de sincronización de grupo,*

se ha valorado favorablemente. Por otra parte, aunque un porcentaje de usuarios hayann detectado algunos efectos extraños o ratifícales, los han calificado como tolerables o imperceptibles, de forma que no les modificaría la opinión afirmativa de visualizar las secuencias de vídeo y audio con dicha calidad de sincronización.

7.2.2.3. Comparación de la influencia del Tipo de Secuencia

A la vista de los resultados de las pruebas realizadas con los dos tipos de secuencia (tipo Primer Plano y tipo Película) podemos concluir que la calidad de la sincronización es más fácil de detectarse en una secuencia de tipo Primer Plano, donde se puede apreciar perfectamente el movimiento de los labios, por lo que los usuarios detectarán con más facilidad las asincronías entre ambos flujos. Por otro lado, en una secuencia de tipo película las asincronías son más difíciles de detectar debido a que el usuario no aprecia tan fácilmente los movimientos de los labios de los personajes como, en las secuencias de tipo Primer Plano, y, además hay otros elementos en la imagen que 'despistan' al mismo y hacen que no le asigne tanta atención a la sincronización entre flujos.

En la secuencia de tipo Película, los cambios de plano se suceden constantemente, por lo que las acciones de resincronización traducidas en '*saltos*' y las '*pausas*' del proceso de reproducción son menos perceptibles para los usuarios.

Además, los usuarios están acostumbrados a ver películas extranjeras donde se incluyen doblajes con ciertos niveles de asincronía debido a los mismos y se muestran más tolerantes a los mismos.

Puede apreciarse en la figura 7.10 que ha habido un porcentaje bajo de sujetos que han evaluado como 'totalmente sincronizadas' las secuencias donde no se ejecutaba algoritmo de sincronización alguno.

7.2.2.4. Comparación de la influencia de la tasa de reproducción

A partir de las pruebas realizadas y las respuestas de los sujetos evaluados, también se puede aportar alguna consideración respecto a la calidad de la presentación cuando se modifica la tasa de reproducción. En nuestra evaluación hemos utilizado secuencias reproducidas con tasas de 15 y 25 tramas por segundo.

Desde el punto de vista de la calidad, como es lógico, a mayor tasa de reproducción mayor será la calidad apreciada por los individuos. Sin embargo, desde el punto de vista de la percepción de la calidad de sincronización, en las secuencias donde se ha utilizado algún algoritmo de sincronización, se han observado dos aspectos característicos:

- Valores muy parecidos es cuanto a la valoración general de la calidad de sincronización.

- Se aprecia que, generalmente, cuando las secuencias se reproducen con una tasa de 15 tramas por segundo, se detectan más efectos extraños que, por los comentarios de los sujetos evaluados, consisten en una sensación de '*saltos*' entre las diferentes imágenes y movimientos entrecortados. Sin embargo, a pesar de ser apreciable para los usuarios no les ha influido en la consideración general sobre la calidad de la sincronización.

7.3. Conclusiones

La mayor parte de los protocolos de sincronización presentados en el capítulo 2 de estado del arte, han sido evaluados midiendo, únicamente de forma objetiva, la precisión respecto a la sincronización entre los diferentes flujos de información. A partir de este tipo de medidas, se puede afirmar que un algoritmo ofrece buenas prestaciones si la asincronía se mantiene dentro de un determinado margen. Los límites de dicho intervalo dependen de la aplicación y entorno, y determinan la calidad de la sincronización requerida. Típicamente, en la mayoría de los trabajos estudiados, los valores que se han utilizado como referencia son los proporcionados por los experimentos realizados en [STE96]. No obstante, a pesar de los esfuerzos por encontrar los límites permitidos en los valores de los parámetros que miden la calidad de una presentación multimedia, debemos tener claro que los usuarios van a exigir, cada vez más, una mayor calidad en todos los aspectos de las aplicaciones, incluida la sincronización multimedia. Posiblemente, todo ello repercutirá en que las especificaciones actuales de los requerimientos mínimos para las diferentes aplicaciones multimedia sean mucho más estrictas en un futuro no muy lejano.

A pesar de que la medida objetiva de la calidad de sincronización nos ha permitido comprobar y validar las prestaciones del algoritmo, se ha considerado interesante corroborar y validar los resultados obtenidos en la misma mediante una evaluación subjetiva de la calidad de la sincronización obtenida con el mismo. Se ha presentado una *evaluación subjetiva* del algoritmo de sincronización propuesto sobre la *aplicación de aprendizaje a distancia* en los *entornos LAN y CAMPUS* descritos en el capítulo anterior.

La evaluación subjetiva ha consistido en la valoración por un observador humano de varias secuencias (con flujos de audio y vídeo) con tres grados de sincronización diferentes: en ausencia de sincronización alguna y sincronizadas, bien con sincronización inter-flujo, o bien con el algoritmo de sincronización de grupo propuesto.

La principal característica del estudio llevado a cabo en este capítulo, ha sido la posibilidad de incorporar en la evaluación el efecto que introduce desde el punto de vista de la percepción humana, el propio mecanismo y funcionamiento del algoritmo propuesto. De esta manera se ha conseguido conocer cuál es el efecto producido por las acciones de resincronización del algoritmo sobre un observador humano.

Las conclusiones más importantes de la evaluación se pueden concretar en los siguientes puntos:

1. Se ha comprobado como el algoritmo propuesto muestra una buena aceptación y valoración en el tipo de aplicación y los entornos evaluados, desde el punto de vista subjetivo.

2. Se comprueba que los resultados obtenidos para el algoritmo incluso mejoran los obtenidos cuando se utiliza un mecanismo de sincronización inter-flujo típico.

3. En caso de no utilizar un algoritmo de sincronización la presentación puede degradarse hasta unos mínimos que la hacen inaceptable.

4. Como único efecto detectado, los evaluadores han detectado una cierta *'ralentización'* o *'saltos'* en determinados instantes, efecto que en ningún caso se ha calificado de molesto.

Hay que tener en cuenta que, como es evidente, la importancia de este tipo de evaluación es relativa a los requerimientos de la aplicación donde se implemente el algoritmo. En concreto, la evaluación de la sincronización en aplicaciones multimedia de las consideradas *críticas* (por ejemplo, aplicaciones médicas de reproducción de imágenes y datos) se deberá utilizar métodos objetivos y deterministas. Sin embargo, en la mayoría de las aplicaciones multimedia, como, por ejemplo, la de aprendizaje a distancia propuesta, una evaluación subjetiva determinará en último término la aceptación de la misma por parte del usuario.

Capítulo 8

Conclusiones, Aportaciones y Líneas Futuras de Investigación

8.1. Introducción

Tras haber expuesto el contenido teórico de la presente Tesis, en los dos capítulos anteriores se han mostrado y discutido los resultados obtenidos tanto en la evaluación objetiva como subjetiva del algoritmo propuesto.

En el presente capítulo se resumen las principales conclusiones de la Tesis, se detallan las principales aportaciones relacionadas con la misma y se destacan los posibles campos de aplicación y líneas de investigación abiertas, que podrán ser abordadas en el futuro, bien como una continuación de este trabajo o bien como profundización en algunos aspectos aquí tratados.

8.2. Conclusiones

Las conclusiones particulares de cada capítulo ya se han resumido al final de los mismos, a modo de corolario. Es por ello que en este capítulo se presentan las conclusiones globales de la Tesis, destacando de modo particular los aspectos más importantes de la misma.

Uno de los principales requerimientos que caracterizan a los sistemas distribuidos multimedia es la necesidad de garantizar y mantener las relaciones temporales entre los flujos de información en el momento de la presentación de los mismos. La mayoría de las referencias estudiadas y presentadas en el capítulo 2 de estado del arte, se centran en la descripción de conceptos básicos y en la aportación de soluciones particulares a determinados aspectos que constituyen el problema de la sincronización.

Podemos definir tres tipos de sincronización multimedia: *sincronización intra-flujo, sincronización inter-flujo* y *sincronización de grupo*. El trabajo desarrollado en esta Tesis Doctoral se ha centrado en el estudio y diseño de un algoritmo de sincronización de grupo de flujos multimedia englobando los dos últimos tipos, cumpliendo con los objetivos de diseño especificados.

Entre las diferentes soluciones estudiadas, se ha tomado como base para el desarrollo del algoritmo las ideas presentadas por los protocolos *Feedback* ([RAN92 y RAM93]) y *Feedback Global* ([GUE97]), implementándose sobre los protocolos RTP/RTCP ([SCH96]), para aprovechar los mensajes RTCP de realimentación (*feedback*), y se propone la utilización del protocolo NTP para la obtención de una referencia de tiempo global en todos los dispositivos involucrados en la sesión multimedia.

En el capítulo 4 se ha presentado una tabla comparativa de los principales algoritmos que tratan la sincronización inter-flujo y de grupo, incluyendo el propuesto en la Tesis.

Una vez descrito el algoritmo, implementado y evaluado (capítulos 7 y 8) en dos entornos diferentes (LAN y CAMPUS) se pueden destacar las siguientes características:

- Sirve tanto para entornos *unicast* como *multicast*, con una o varias fuentes y uno o varios receptores.

- Las fuentes pueden ser de flujos de datos almacenados o de flujos de datos capturados en directo (como, por ejemplo, de audio y vídeo en tiempo real, como en las aplicaciones de videoconferencia). Los flujos pueden ser tanto dependientes como independientes del tiempo.

- No se necesitan conexiones adicionales para transmitir los mensajes de realimentación (o *feedback*) de los receptores a las fuentes, ya que se utiliza, para ello, los mensajes de informes del protocolo RTCP que los receptores envían a las fuentes, sea cual sea la configuración del sistema de transmisión (*unicast* o *multicast*).

- Tiene una respuesta rápida y precisa a las asincronías detectadas.

- En las aplicaciones de tiempo real, como, por ejemplo, las de aprendizaje a distancia, los usuarios son libres de entrar y salir de la sesión sin que afecte al resto de usuarios. La entrada o salida de usuarios a la sesión multicast no afecta al funcionamiento del algoritmo propuesto. Por tanto, no afecta a la escalabilidad de la aplicación.

- Gracias al uso de NTP, ampliamente utilizado, se soluciona el problema asociado con el *drift* de los relojes de los sistemas implicados en las aplicaciones multimedia. No es necesario conocer a priori la distribución del *drift* de los relojes.

- Aunque no es necesario conocer a priori los valores máximo y mínimo del retardo extremo a extremo entre las fuentes y los receptores, en los entornos de aplicación del algoritmo propuesto, el retardo debe estar limitado para evitar situaciones de '*overflow*' y '*underflow*' en los búffers de recepción.

- Se reduce la carga de mensajes de control en la red ya que se utilizan extensiones en los paquetes RTCP utilizados por las aplicaciones donde se ha implementado el algoritmo de sincronización.

- En las pruebas realizadas, en la aplicación de aprendizaje a distancia se ha garantizado una sincronización de grupo e inter-flujo (en las medidas de sincronización de grupo e inter-flujo se han tomado los valores de ±120 milisegundos y de ±80 milisegundos, respectivamente, como valores de referencia de las asincronía máxima permitida en ambos casos) en un entorno LAN y de CAMPUS.

- El algoritmo también ha sido evaluado de forma subjetiva, en una aplicación de aprendizaje a distancia, tanto en entorno LAN como en entorno CAMPUS, obteniendo resultados satisfactorios.

- Con las modificaciones realizadas en las herramientas *vic* y *rat*, se dispone de una plataforma de medidas adecuada para la evaluación de otros

algoritmos de sincronización sobre RTP/RTCP dada la flexibilidad de la aplicación e ideas básicas de funcionamiento.

Lógicamente, el éxito del algoritmo propuesto depende, en gran medida, de los protocolos de encaminamiento multicast empleados, de la precisión obtenida por el protocolo NTP en los sistemas involucrados en la aplicación donde se integre, del soporte de aplicaciones de tiempo real que ofrezcan los sistemas operativos empleados y de las propia aplicación multimedia.

Por otra parte, hay que tener en cuenta que en las aplicaciones para las que está pensada la posible utilización del algoritmo propuesto (de tiempo real no críticas), pequeños errores de sincronización son tolerables. Como es lógico, debido a la gran cantidad de factores aleatorios que influyen en el proceso de sincronización, que se produzcan algunos errores a lo largo de la sesión multimedia es inevitable.

Por último señalar la importancia de los capítulos 2 y 3, donde se ha presentado el estado del arte y los requerimientos y características de los sistemas multimedia, resumiendo las aportaciones de las principales contribuciones estudiadas y exponiendo los aspectos más relevantes sobre sincronización multimedia.

8.3. Aportaciones

Además de la principal aportación, que ha constituido el objetivo principal de la presente Tesis, consistente en haber especificado y evaluado un algoritmo de sincronización de grupo de flujos multimedia con resultados satisfactorios, a continuación se presentan las aportaciones y logros adicionales que se han desprendido del presente trabajo investigador.

8.3.1. Publicaciones

A continuación se detallan las principales publicaciones relacionadas con el contenido de la Tesis:

- La propuesta de implementación del algoritmo propuesto en la presente tesis doctoral haciendo uso del protocolo RTP/RTCP se publicó en el congreso *JITEL'99*, celebrado en Leganés (Madrid) ([BOR99]).

- En el año 2001, se presentó en el congreso *EUROMEDIA'01* ([BOR01]), realizado en Valencia (España), en un artículo extenso que describía la propuesta del algoritmo un poco más evolucionado.

- En el año 2002, se presentó en Módena (Italia) en el congreso *EUROMEDIA'02* ([BOR02a]), una implementación de dicho algoritmo sobre herramientas desarrolladas en el seno de la red MBone mostrando su validez. El artículo presentado obtuvo el *Premio al Mejor Artículo del Congreso (Best Paper Award)*

- Durante los dos últimos años, se han desarrollado herramientas software que facilitaran la realización de las pruebas de evaluación del algoritmo propuesto. Entre los resultados obtenidos cabe destacar el desarrollo de una *herramienta de configuración y análisis del servicio de tiempo global ofrecido por NTP* que, además de ser utilizada en la presente Tesis, se ha utilizado para realizar una comparativa de prestaciones del servicio de tiempos en diversos sistemas operativos, incluso tomando como referencia receptores GPS comerciales de bajo coste, que se presentó en el congreso *CITEL'02*, celebrado en la Ciudad de la Habana (Cuba) ([BOR02b]) y cuya descripción se presentó en *JITEL'03*, en Las Palmas de Gran Canaria ([BOR03b]).

- Además de la herramienta anterior, también se ha desarrollado una *herramienta de monitorización del servicio RTP/RTCP en una sesión multimedia, denominada MonitorRTP* que fue presentada en el Congreso *EUROMEDIA'03*, que tuvo lugar en Plymouth (Inglaterra) ([BOR93a]).

8.3.2. Programas y Algoritmos desarrollados

Aparte de la integración del algoritmo propuesto en las herramientas *vic* y *rat*, se han desarrollado tres herramientas, descritas en los artículos referenciados en el apartado anterior, que son las siguientes:

- *Herramienta de transmisión de texto mediante RTP/RTCP*, siguiendo las recomendaciones de la RFC 2793 ([HEL00]).

- *Herramienta de configuración y análisis del servicio de tiempo global ofrecido por NTP*

- *Herramienta de monitorización del servicio RTP/RTCP en una sesión multimedia, denominada MonitorRTP*.

Las dos primeras están programadas en lenguaje *Visual C++* mientras que la tercera está desarrollada en *java*, utilizando la API *Java Media Framework*.

Adicionalmente, para la implementación de la aplicación de transmisión de texto sobre RTP, se han implementado en lenguaje *C++* los *algoritmos de codificación y decodificación UTF-8*.

8.4. Posibles Líneas Futuras de Investigación

A continuación se van a exponer algunas de las líneas que suponen una continuación directa de la propuesta realizada en la Tesis:

- El algoritmo de sincronización desarrollado se basa en un esquema *maestro/esclavo*, donde uno de los receptores es tomado como referencia (*maestro*) por el resto para sincronizar su estado de reproducción. A priori, el trabajar con una arquitectura *maestro/esclavo* podría tener el inconveniente lógico de la debilidad que supone un único receptor *maestro*, ya que éste puede fallar en un determinado momento y provocar un mal funcionamiento del algoritmo. Este inconveniente no es tal en nuestro caso ya que se ha propuesto un mecanismo de selección de receptor *maestro* que supondría una recuperación del algoritmo pasado un cierto tiempo sin recibir respuesta de éste. No obstante, se podrían estudiar *nuevos algoritmos de selección dinámica del receptor considerado como referencia* en cada momento.

- También se podría considerar la posibilidad de *reducir la sobrecarga de control introducido por el algoritmo*, aunque ésta sea muy poca. Esto se podría conseguir de las siguientes dos formas

  - Comprimiendo las cabeceras de las unidades de datos encapsuladas en UDP/RTP/IP tal y como se indica en la RFC 2508 [CAS99]. Dicha compresión se puede realizar enlace a enlace (*on a link-by-link basis*) para transportar sólo aquellos cambios de la cabecera que cambien de forma inesperada de un paquete a otro. Ello puede reducir la combinación de cabeceras RTP/UDP/IP de 40-60 a 2-4 octetos.

  - Mediante algún mecanismo (como los descritos en [RAM93] y [GUE97]) con el cual la fuente sincronizadora, en el algoritmo propuesto, indicara a los receptores cuándo deben enviar la información añadida al mensaje RR extendido, para no enviarla en todos los mensajes RR.

- Evaluar la calidad de la sincronización conseguida con el algoritmo propuesto entre flujos de información tanto dependientes (como audio y vídeo) como independientes del tiempo (como, por ejemplo, datos o texto)

- Evaluar el algoritmo propuesto en una red con servicio *best-effort* y reserva de recursos.

Referencias Bibliográficas

[AKY96] I. F. Akyildiz, W. Yen, “Multimedia Group Synchronisation Protocols for Integrated Services Networks”, IEEE Journal Selected Areas in Comm., vol. 14, pp. 162 - 173, January 1996.

[ALM99] G. Almes, S. Kalidindi, M. Zekauskas, “A One-way Delay Metric for IPPM”, RFC 2679, September 1999.

[ALV93] F. Alvarez-Cuevas, M. Bertran, F. Oller y J.M. Selga, “Voice synchronisation in packet switching networks”, IEEE Networks Magazine 7, 5 (September 1993), 20-25.

[AND91] David P. Anderson, George Homsy, “A Continuous Media I/O Server and Its Synchronisation Mechanism”. IEEE Computer 24(10): 51-57 (1991).

[APP95] W. Appelt et alt., ‘BSCW project, *Basic Support for Cooperative Work*’. http://orgwis.gmd.de/projects/BSCW/, http://bscw.gmd.de/

[BAD00] G. Baddeley “Glenn Baddeley - GPS - NMEA sentence information”, January 2000.

[BAL00] M. Baldi, Y. Ofek. “End-to-End Analysis of Videoconfernecing over Packet-Switched networks”. IEEE/ACM Transactions on Networking, vol. 8, no. 4, pp. 479-492, August 2000.

[BAQ96] S. Baqai, M. F. Khan, M. Woo, S. Shinkai, A. A. Khokar, and A. Ghafoor, “Quality-based evaluation of multimedia synchronisation protocols for distributed multimedia information systems”, IEEE J. Selected Areas Comm. 14, 7 (Sep. 1996), 1388-1403.

[BER00] Berthaud, J-M. “Time synchronisation Over Networks Using Convex Closures”. IEEE/ACM Transactions on Networking, vol. 8, no. 2, pp.265-273, April 2000.

[BIE96] Biersack, E., Geyer, W., Bernhardt, C.: "Intra- and Inter-Stream Synchronisation for Stored Multimedia Streams", in: Proc. ICMCS'97, IEEE International Conference on Multimedia Computing and Systems, Hiroshima, Japan, June 1996. (Outstanding Paper Award).

[BLA96] G. Blakowski, R. Steinmetz, “A Media Synchronisation Survey: Reference Model, Specification and Case Studies”, IEEE Journal on Selected Areas in Communications, vol. 14, no. 1, pp. 5-35, January 1996.

[BOL94] J-C. Bolot, T. Turletti, I. Wakeman, “Scalable feedback control for multicast video distribution in the Internet”, Proc. ACM Sigcomm '94, pp. 58-67, London, UK, Sept. 1994.

[BOL98] J.C. Bolot, S. Fosse-Parisis. “Adding voice to a distributed game on the Internet”, Proc. Infocom '98, San Fransisco, CA, April 1998.

[BOR99] F. Boronat, J.C. Guerri, M. Esteve, C. Palau. “Propuesta de un protocolo de Sincronización de flujos Multimedia sobre RTP”. Jornadas de Ingeniería Telemática (JITEL'99). Año 1999.

[BOR01] F. Boronat, “Multimedia Streams Synchronisation Protocol based in RTP/RTCP (Real-time Transport Protocol): RTP-based Feedback Global Protocol”, Euromedia 2001, 18-20 abril, Valencia, España.

[BOR02a] F. Boronat, J.C. Guerri, M. Esteve, J.M. Murillo, “RTP-based Feedback Global Protocol integration in Mbone tools”, Euromedia 2002, 15-17 abril, Módena, Italia. (*BEST PAPER AWARD*).

[BOR02b] F. Boronat, M. Esteve y J. Carlos Guerri, “Sincronización de una Subred utilizando tiempo global vía NTP”, II Congreso Internacional de Telemática 2002 (Citel 02), 25 a 29 noviembre, Ciudad de la Habana (Cuba).

[BOR03a] F. Boronat, S. Llopis, J. Carlos Guerri, M. Esteve, “Graphical RTP Session Monitor using Java Mediaframework (JMF)”, Euromedia 2003, 14-16 abril, Plymouth, UK.

[BOR03b] F. Boronat, J. Carlos Guerri, J. Ramón Mengual, “Herramienta gráfica de configuración y análisis del servicio de tiempo global ofrecido por el protocolo NTP”, Jitel 2003, 15-17 septiembre, Palmas de Gran Canaria, España.

[BOU98] A. Bouch, A.Watson, M.A. Sasse, “QUASS - A Tool for Measuring the Subjective Quality of Real-Time Multimedia Audio and Video”. HCI '98, 1-4 September 1998, Sheffield, England.

[BRA97] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, “Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification”. September 1997. RFC 2205.

[BRA98] K. Obraczka. “Multicast Transport Protocols: A Survey and Taxonomy” Communications Magazine, Enero 1998.

[BRA02] J. Brassil, H. Schlzrinne, “Structuring Internet Media Streams with Cueing Protocols”, IEEE/ACM Transaction on Networking, vol. 10, nº 4, August 2002.

[CAN92] S. McCanne, “A distributed Whiteboard for Network Conferencing”. May 1992, UC Berkeley CS 268 Computer Networks term project.

[CAN95] S. McCanne and Van Jacobson, “vic: A Flexible Framework for Packet Video”. ACM Multimedia, November 1995, San Francisco, CA, pp. 511-522.

[CAS99] S. Casner y V. Jacobson, “Compressing IP/UDP/RTP Headers for Low-Speed Serial Links”, Network Working Group, RFC: 2508, February 1999.

[CAS98] V. Castelo, “MBone: El camino hacia una Internet multimedia.” Noviembre de 1998.

[CHE96] H.Y Chen, J.L. Wu, “MultiSync: A Synchronisation Model for Multimedia System”, *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 1, pp. 238-248, January 1996.

[CHE96a] Zhigang Chen, See-Mong Tan, Roy H. Campbell, Yongcheng Li, "Real Time Video and Audio in the World Wide Web", http://choices.cs.uiuc.edu/Papers/New/vosaic/vosaic.html, 1996.

[CIV00] M. Civanlar y G. Cash, “RTP Payload Format for Real-Time Pointers”, Network Working Group, RFC 2862, June 2000.

[CRO98] J. Crowcroft, L. Vicisano, Z. Wang, A. Ghosh, M. Fuchs, C. Diot and T. Turletti. “RMFP: A Reliable Multicast Framing Protocol”, Internet Draft: draft-crowcroft-rmfp-02.txt, March 1998.

[CRO99] J. Crowcroft, M. Handley, I. Wakeman, “Internetworking multimedia”, Morgan Kaufmann Publishers, San Francisco, California, 1999. ISBN: 0-7484-0808-8.

[DEC89] “Digital time Service Functional Specification version T.1.0.5”, Digital Equipment Corporation, December 1995.

[DEE89] S.E. Deering. “Host extensions for IP -ing. RFC 1112. Aug-01-1989.

[DIG89] Digital Equipment Corp., “Digital Time Service Functional Specification, version T.1.0.5”, 1989.

[DIO99] Christophe Diot and Laurent Gautier "A Distributed Architecture for Multiplayer Interactive Applications on the Internet", IEEE Network, vol. 13, pp. 6 - 15, July/August 1999.

[EHL94] L. Ehley, B. Furht, M. Ilyas: Evaluation of Multimedia Synchronisation techniques. ICMCS 1994, pp 514-519.

[ESC94] Julio Escobar, Craig Partridge, Debra Deutsch, "Flow synchronisation protocol", IEEE/ACM Transactions on Networking (TON), Volume 2, Issue 2 (April 1994), Pages: 111 – 121.

[EST95] M. Esteve, C. Palau y J.C. Guerri, "A Synchronisation Service Framework for XTP", in proc. 20th Annual Conference on Local Computer Networks, Mineapolis, pp. 209-218, October 1995.

[FER90] D. Ferrari, "A Scheme for Real-Time Channel Establishment in Wide Area Networks", *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 3, pp. 368-379, April 1990.

[FRE94] R. Frederic, "Experiences with real-time video compresión", Xerox Parc, July 1992. ftp://ftp.parc.xerox.com/pub/net-research/

[FUR94] B. Furth, "Multimedia Systems: An Overview", IEEE Multimedia, Spring 1994, pp. 47-59.

[GAR98] J. A. García "MBone: Arquitectura y Aplicaciones" Versión electrónica del boletín N. 43 de RedIRIS. Junio 1998. http://www.rediris.es/rediris/boletin/43/enfoque3.html

[GEY96] Geyer, W., Bernhardt, C., Biersack, E.: "A Synchronisation Scheme for Stored Multimedia Streams", Proc. IDMS'96, International Workshop on Interactive Distributed Multimedia Systems and Services, Berlin, Germany, Springer, Heidelberg, LNCS 1045, March 1996, pp. 277-296.

[GIL95] J. Gili, M. Roser, A. Vicente, "Algoritmo de codificación de vídeo MPEG: Evaluación de la calidad subjetiva de la imagen", Comunicaciones de Telefónica I+D, Vol 6, Nº 1, Enero-junio 1995.

[GON00] A. J. González and H. Abdel-Wahab, " Light-Weight Stream Synchronisation Framework for Multimedia Collaborative Applications," in proceedings of The Fifth IEEE Symposium on Computers and Communications (ISCC'2000), pp. 398-403, Antibes-Juan Les Pins, France, July 2000.

[GUE97] J.C. Guerri. "Especificación y evaluación de prestaciones de un Protocolo de Sincronización Multimedia Adaptativo con Control de Flujo, basado en un Tiempo Global y Técnicas de Realimentación" Tesis doctoral, U.P.V. Junio 1997.

[GUE99] Juan C. Guerri, Carlos Palau, Manuel Esteve, Fernando Boronat. "Evaluación del protocolo feedback-global para sincronización de Flujos Multimedia". URSI 99. XIV SIMPOSIUM NACIONAL.

[GUS85] R. Gusella y S. Zatti, "The Berkeley UNIX 4.3BSD time synchronisation protocol: Protocol specification", Univ. de California, Berkeley, Tech. Rep. UCB/CSD 85/250, june 1985.

[HAN95] M. Handley, I. Wakeman and J. Crowcroft, "The Conference Control Channel Protocol (CCCP): A scalable base for building conference control applications" SIGCOMM. Pp. 275-287, Cambridge, Massachusetts, September 1995.

[HAN95a] M. Handley, P. Kirstein, M.A. Sasse, "Multimedia Integrated Conferencing for European Researchers (MICE): piloting activities", Computer Networks and ISDN Systems, 26 (3), 275-290, 1995

[HAN96] M. Hnadley, H. Schulzrinne, E. Schooler. 'SIP: Session Initiation Protocol'. Internet draft, IETF. Dec. 1996. Work in progress.

[HAN98] M. Handley, V. Jacobson. "SDP: Session Description Protocol". RFC 2327 April 998.

[HAN00] M. Handley, C. Perkins, E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.

[HAR95] V. Hardman, A. Sasse, "Multi-way Multicast Speech for Multimedia Conferencing over Heterogenous Shared Packet Networks (RAT – Robust Audio Tool)", EPSRC MNA Project #GR/K772780, Department of Computer Science, UCL.

[HAR96] V. Hardman & M. Iken, “Enhanced Reality Audio in Interactive Network Environments”, in Proceedings of the FIVE Technical Conference, Pisa, Italy, December 1996.

[HEL00] G. Hellstrom, “RTP Payload for Text Conversation”, Omnitor AB, RFC 2793, may 2000.

[HOO92] B. Hoogeboom, W.A. Halang, “The Concept of Time in the Specification of Real Time Systems”, Technical Report, Vakgroep Informatica, 1992.

[ISH95] Y. Ishibashi and S. Tasaka, "A synchronization mechanism for continuous media in multimedia communications," Proc. IEEE INFOCOM'95, pp.1010--1019, April 1995.

[ISH97] Y. Ishibashi, A. Tsuji, S. Tasaka. “A Group Synchronisation Mechanism for Stored Media in Multicast Communications”. IEEE INFOCOM 1997, pp.693-701, April 1997.

[ISH99] Y. Ishibashi, S. Tasaka. “A Group Synchronisation Mechanism for Stored Media and Its Measured Performance in a Heterogeneous Network”. IEICE TRANS. ON COMMUNICATIONS 1999, Vol. E82-B, nº 7, pp.1009-1018, July 1999.

[ISH00] Y. Ishibashi, S. Tasaka, “A Comparative Survey of Synchronization Algorithms for Continuous Media in Network Environments, IEEE Conference on Local Computer Networks (LCN) 2000, November.

[ISH01a] Y. Ishibashi, S. Tasaka, Y. Tachibana, “Media Synchronisation and Causality Control for Distributed Multimedia Applications”, IEICE Transactions on Communications, Vol.E84-B, No.3, p.667, Multimedia Systems, March 2001.

[ISH01b] Y. Ishibashi, S. Tasaka, H. Ogawa; “A comparison of media synchronization quality among reactive control schemes”, Proc. IEEE INFOCOM, 2001, pp. 77 - 84, April 2001.

[ISH01c] Y. Ishibashi, S. Tasaka, Y. Tachibana; “Adaptive causality and media synchronization control for networked multimedia applications, Proc. IEEE ICC, 2001, pp. 952-958, June 2001.

[ISH02] Y. Ishibashi, S. Tasaka, H. Miyamoto, “Joint Synchronisation between Stored Media with Interactive Control and Live Media in Multicast Communications”, Multimedia Systems, Vol.E85-B, No.4, p.812,

[ITO02] K. Ito, S. Tasaka, and Y. Ishibashi, "Media synchronisation quality of packet scheduling algorithms", IEICE Trans. on Commun., vol. E85-B, no. 1, pp. 52-62, Jan. 2002.

[JAC92] V. Jacobson, “VAT Manual pages”, Lawrence Berkeley Laboratory (LBL), February 1992.

[JEO02] Won J. Jeon, Kyung-Joon Park, Klara Nahrstedt, “Robust Playout Mechanism for Internet Audio Applications”, in Proc. of IEEE Conference on Local Computer Networks (LCN), Tampa, FL, November, 2002.

[KAL99] F. Kaladji, Y. Ishibashi, S. Tasaka, “Subjective Assessment of Stored Media Synchronization Quality in the VTR Algorithm”, IEICE Transactions on Communications, Vol. E82-B, nº. 1, pp. 24-33, 1999.

[KAN01] A. Kansal, A. Karandikar, “Adaptive Delay Estimation for Low Jitter Audio over Internet”, IEEE Globecom’01, 2001.

[KÖL94] D. Köhler, H. Müller, “Multimedia Playout Synchronization Using Buffer Level Control”, in Proc. 2nd International Workshop on Advanced Teleservices and High-Speed Communication Architectures (IWACA’94), pp. 167-180, September 1994.

[KOP87] H. Kopetz, W. Oschsenreiter, “Clock Synchronization in Distributed Real-Time Systems”, IEEE Transactions on Computer, C-38, pp. 933-939, August 1987.

[KOU96] I. Kouvelas , V. Hardman , A. Watson. “Lip Synchronisation for Use Over the Internet: Analysis and Implementation”. Globecom’96, London, November 1996.

[KUO98] F. Kuo, W. Effelsberg y J.J. García-Luna-Aceves, “Multimedia Communications Protocols and Applications”, Prentice Hall, 1998, ISBN 0-13-856923-1.

[KUO01] C.-C. Kuo, M.-S. Chen and J.-C. Chen, "An Adaptive Transmission Scheme for Audio and Video Synchronisation based on Real-time Transport Protocol," Proceedings of IEEE International Conference on Multimedia and Expo (ICME-01), August 22-25, 2001.

[KUT00] D. Kutscher, J. Ott, “The Message Bus. A Communication ¬ integration Infraestructure for Component-based Systems”, White paper, January 2000. http://www.mbus.org/mbuswp.pdf.

[LAM96] L. Lamont, L. Li, R. Brimont, D. Georganas, “Synchronisation of Multimedia Data for a Multimedia News on demand Application”, IEEE J. S. A. C., Vol. 14, nº 1, enero 1996.

[LAO01a] N. Laoutaris, I. Stavrakakis, ""An Analytical Design of Optimal Playout Schedulers for Packet Video Receivers"", Computer Communications Journal, and 2nd Intl. Workshop Quality Future Internet Services, Coimbra, Portugal, 2001.

[LAO01b] N. Laoutaris y I. Stavrakakis. “Adaptive Playout Strategies for Packet Video Receivers with finite buffer capacity”. Proc. IEEE ICC, Helsinki, Finland, june 2001.

[LAO02] N. Laoutaris y I. Stavrakakis. “Intrastream Sunchronization for Continuous Media Streams: A Survey of Playout Schedulers”. IEEE Network Magazine, vol. 16, no. 3, pp. 30-40, May/June 2002.

[LI99] X. Li, M. H. Ammar, S. Paul. “Video Multicast over the Internet”. IEEE Network Magazine. Marzo/abril 1999.

[LIT90] D.C.Little, A.Ghafoor. “Synchronisation and storage Models for Multimedia Objetcs”. IEEE Journal on Selected Areas in Communications, Abril 1990.

[LIT91] T.D.C. Little, A. Ghafoor, “Multimedia Synchronization”, *IEEE Data Engineering Bulletin*, vol. 14, no. 3, pp. 26-35, September 1991.

[LIT91a] T.D.C. Little, A. Ghafoor, “Multimedia Synchronisation Protocols for Broadband Integrated Services”, IEEE Journal on Selected Areas in Communications, vol. 9, no. 9, pp. 1368-1382, December 1991.

[LIT92] T.D.C. Little, F. Kao, “An Intermedia Skew Control System for Multimedia Data Presentation”, in *Proc. 3rd. Intl. Workshop and Operating System Support for Digital Audio and Video*, pp. 121-132, November 1992.

[LIT93] T.D.C. Little, “A Framework for Synchronous Delivery of Time-Dependent Multimedia Data”, *Multimedia Systems*, vol. 1, no. 2, pp. 87-94, 1993.

[LIU96] C. Liu, Y. Xie, M.j. Lee, T.N. Saadawi, “Multipoint Multimedia Teleconference System with Adaptative Synchronisation”, IEEE Journal on Selected Areas in Communications, vol. 14, no. 7, pp. 1422-1435, September 1996.

[MAC94] Macedonia, Michael R. and Brutzman, Donald P., "MBone Provides Audio and Video Across the Internet," IEEE COMPUTER, vol. 27 no. 4, April 1994, pp. 30-36.

[MAR96] R. El-Marakby and D. Hutchison. “Evaluation of the Real-time Transport Protocol (RTP) for Continuous Media Communications”, Proc. of the 3rd Comm. Networks Symposium, Manchester, U.K., July 1996.

[MAR96a] R. El-Marakby, D. Hutchison, "Integrating RTP into the World Wide Web" , Proc. of W3C Workshop, Sophia-Antipolis, France, October 1996.

[MAU99] M. Mauve, Volker Hilt, Christoph Kuhmünch, Wolfgang Effelsberg. “RTP/I - An Application layer Protocol for the Transmission of Interactive Media with Real-Time Characteristics”. Proc. of IEEE Multimedia Systems '99 (ICMCS'99), Florence, Italy, June 1999.

[MEL02] H. Melvin y L. Murphy. “Time Synchronisation for VoIP Quality of Service”. IEEE Internet Computing, pp. 57-63, may/june 2002.

[MEY98] D. Meyer, “Administratively Scoped IP Multicast”, RFC 2365, University of Oregon, July 1998.

[MIL85a] D. L. Mills, “Experiments in network clock synchronisation”, RFC 957, September 1985.

[MIL85b] D. L. Mills, "Network Time Protocol", RFC 958, September 1985.

[MIL88] D. L. Mills, “The Fuzzball”, ACM SIGXOMM 88 Symposium, Palo Alto, CA, pp. 115-122, August 1988.

[MIL91a] D. L. Mills, “Internet time synchronisation: the Network Time Protocol”. IEEE Trans. Communications, vol. 39, nº 10, pp. 1482-1493, October 1991.

[MIL91b] D. L. Mills, “On the Cronology and metrology of computer network timescales and their application to the Network Time Protocol”, ACM Computer Communications Review 21, pp. 8-17, October 1991.

[MIL92] D. L. Mills, "Network Time Protocol", RFC 1305, 1992.

[MIL94] D. L. Mills y A. Thyagarajan. “Network time protocol version 4 proposed changes”. Electrical Engineering Department Report 94-10-2, University of Delaware, October 1994.

[MOO98] S. B. Moon, Jim Kurose, and Don Towsley. "Packet Audio Playout Delay Adjustment: Performance Bounds and Algorithms," ACM/Springer Multimedia Systems, Vol. 6, pp. 17-28, January, 1998.

[NIC90] C. Nicolau. “An Architecture for Real-Time Multimedia Communication Systems”. IEEE Journal on Selected Areas in Communications, vol. 8, no. 3, pp. 391-400, Abril 1990.

[OTT99a] J Ott, Colin Perkins D.Kuscher “The Message Bus: Messages and Procedures” Diciembre 1999. draft-ietf-mmusic-mbus-semantics-00.txt.

[OTT99b] J. Ott, C. Perkins, D. Kutscher, Internet-Draft: “Requirements for Local Conference Control” Diciembre 1999. draft-ietf-mmusic-mbus-req-00.txt.

[OTT00] J Ott, Colin Perkins D. Kutscher “A Message Bus for Local Coordination”. November 2000. draft-ietf-mmusic-mbus-transport-03.txt.

[PAX98] V. Paxson, G. Almes, J. Mahdavi, M. Mathis, “Framework for IP Performance Metrics”, RFC 2330, May 1998.

[PER97] C. Perkins, I. Kouvelas, O. Hodson, et al. “RTP Payload for Redundant Audio Data” RFC 2198, Network Working Group, INRIA Sophia Antipolis, September 1997.

[PER98] C. Perkins. “Using the Mbus library”. January 1998.

[PER00] C. S. Perkins, J. Crowcroft, “Notes on the use of RTP for shared workspace applications”, ACM Computer Communication Review, Volume 30, Number 2, April 2000.

[QIA97] L. Qiao and K. Nahrstedt, "Lip Synchronisation within an Adaptive VOD", International Conference on Multimedia Computing and Networking, February 1997, San Jose

[RAM92] S. Ramanathan and P. V. Rangan. “Continuous Media Synchronisation in Distributed Multimedia Systems”. In Proceedings of the 3rd International Workshop on Network and Operating System Support for Digital Audio and Video, pages 289-296, November 1992.

[RAM93] S. Ramanathan, P. V. Rangan “Adaptive Feedback Techniques for Synchronized Multimedia Retrieval over Integrated Networks”, IEEE/ACM Transactions on Networking, vol. 1, no. 2, pp. 246-259, April 1993.

[RAM93b] S. Ramanathan, P. V. Rangan, “Feedback Techniques for Intra-Media Continuity and Inter-Media Synchronisation in Distributed Multimedia Systems”, The Computer Journal, 36(1):19--31, 1993.

[RAM94] R. Ramjee, J. Kurose, D. Towsley, H Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks," in Proceedings of the Conference on Computer Communications (IEEE INFOCOM), (Toronto, Canada), pp. 680-688, IEEE Computer Society Press, Los Alamitos, California, June 1994.

[RAM97] Suchitra Raman, Angela Schuett, "On-demand Remote Playback". University of California, Berkeley.

[RAN92] P. V. Rangan, S. Ramanathan, “Designing an On-Demand Multimedia Service”, *IEEE Communications Magazine*, vol. 30, no. 7, pp. 56-64, July 1992.

[RAN95a] P. V. Rangan, S. Ramanathan, “Performance of Inter-media Synchronisation in Distributed and Heterogeneous Multimedia Systems”, Computer Networks and ISDN Systems, 1993, Vol. 27, Issue 4, pp 549-565, 1995.

[RAN95b] P. V. Rangan, S. Ramanathan, S. Sampathkumar, “Feedback techniques for continuity and synchronisation in multimedia information retrieval” ACM Transactions on Information Systems (TOIS), Vol. 13, Issue 2, April 1995, pp 145 – 176, ISSN:1046-8188.

[RAN96] P. V. Rangan, S. Srihari, S. Rajan, “Continuity and Synchronisation in MPEG”, IEEE Journal on Selected Areas in Communications, vol. 14, no. 1, pp. 52-60, January 1996.

[RAY90] M. Raynal, J.M. Helary, “Synchronisation and Control of Distributed Systems and Programs”, John Wiley and Sons, Wiley Series in Parallel computing, 1990.

[RIV92] R. Rivest, “The MD5 Message-Digest Algorithm”, RFC 1321, Network Working Group, MIT Laboratory for Computer Science and RSA Data Security, Inc., April 1992.

[ROC01] M. Roccetti et al., “Design and Experimental Evaluation of an Adaptive Palyout Delay Control Mechanism for Packetized Audio for Use over the Internet”, Multimedia Tools and Applications, vol. 14, nº 1, may 2001.

[ROS01] E. Rosen, A. Viswanathan, R. Callon, “Multiprotocol Label Switching Architecture”, RFC: 3031, IETF Network Working Group, January 2001.

[ROS98] J. Rosenberg, H. Schulzrinne, "Timer Reconsideration for Enhanced RTP Scalability", Proceedings of IEEE Infocom’98, San Francisco, CA, USA.

[ROT95] Kurt Rothermel and Tobias Helbig, “An Adaptive Stream Synchronisation Protocol”. Proc. 5th Intl. Workshop on Network and Operating System Support for Digital Audio and Video, Durham, New Hampshire, April 18-21, 1995, pp. 189-202.

[ROT97] Rothermel, Kurt; Helbig, Tobias, “An Adaptive Protocol for Synchronizing Media Streams”, In: ACM/Springer Multimedia Systems. Vol. 5(5). pp. 324-336, Springer, January 1997. University of Stuttgart, Faculty of Computer Science, Article in Journal.

[RUI99] Pedro Ruíz Miguel Martínez, “La cara oculta de MBONE, Una arquitectura para el control de sesiones en Mbone”. Mayo 1999, http://ants.dif.um.es/staff/pedrom/pfc/pfc.ps

[SAR99] Sarac, K. and Almeroth, K., “Sdr global session monitoring effort”, 1999. http://imj.ucsb.edu/sdr-monitor/global/index.html

[SCH92] H. Schulzrinne, “Voice communication across the Internet: A Network Voice Terminal”, University of Massachusetts, Technical Report, June 1992.

[SCH93] E. Schooler, Distributed Music: A Foray into Networked Performance, International Network Music Festival, Santa Monica, CA (Sept 1993).

[SCH96] H. Schulzrinne, S. Casner, R. Frederick & V. Jacobson. “RTP: A Transport Protocol for Real-Time Applications”. Audio-Video Transport Working Group. RFC 1889. January 1996.

[SCH96a] H. Schulzrinne “RTP Profile for Audio and Video Conferences with Minimal Control. Audio-Video Transport Working Group, RFC 1890. January 1996.

[SCH98] H. Schulzrinne, A. Rao, R. Lanphier. “Real Time Streaming Protocol (RTSP)”. April 1998. RFC 2326.

[SCH00] H. Schulzrinne y S. Petrack, “RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals”, RFC 2833, Network Working Group, May 2000.

[SHI95] N. Shivakumar,C. Sreenan,B. Narendran,P. Agrawal: The Concord Algorithm for Synchronisation of Networked Multimedia Streams. 2nd IEEE International Conference on Multi-media Computing and Systems'95 (ICMCS'95), pp. 31 - 40.

[SHI98] Myung-Ki Shin, Jae-Yong Lee, “A Web-Based Real-time Multimedia Application for the Mbone”, INET'98, Geneva, Switzerland, 1998.

[SHI98a] Myung-Ki Shin, Jae-Yong Lee, Jung-Sook Bae, Jin-Ho Hahm, “The RTMW Application: Bringing Multicast Audio/Video to the Web”, Computer Networks and ISDN Systems, Vol. 30, No. 1~7, pp. 685~687 (7th International WWW Conference Proceedings), 1998.

[SHI99] Myung-Ki Shin, Jin-Ho Hahm, “Applying QoS Guaranteed Multicast Audio and Video to the Web”, IEEE Multimedia System'99, Florence, Italy, 1999.

[SON96] S.H. Son, N. Agarwal, “A Model for Specification and Synchronisation of Data for Distributed Multimedia Applications”, Multimedia Tools and Applications, vol. 3, no. 2, September 1996.

[SRE96] C.J. Sreenan, B. Narendran, P. Agrawal, and N. Shivakumar. “Internet Stream Synchronisation using Concord”. InIS&T/SPIE, editor, Conference on Multimedia Computing and Networking, 1996.

[SRE00] C.J. Sreenan, Jyh-Cheng Chen, P Agrawal, and B Narendran, "Delay reduction techniques for playout buffering", IEEE Transactions on Multimedia, vol. 2, no. 2, June 2000.

[STA88] John A. Stankovic. “Misconceptions About Real-Time Computing”. IEEE Computer, Octubre 1988.

[STE90] Ralf Steinmetz. “Synchronisation Properties in Multimedia Systems”. IEEE Journal on Selected Areas in Communications, Abril 1990.

[STE95] R. Steinmetz, “Analyzing the Multimedia Operating System”, IEEE Multimedia” vol, 2, nº 1, pp.68-84, Spring 1995.

[STE96] R. Steimetz, “Human Perception of Jitter and Media Skew”, IEEE Journal Selected Areas in Comm., Vol. 14, nº 1, january 1996.

[STO95] D.L. Stone and K. Jeffay, "An Empirical Study of a Jitter Management Scheme for Video Teleconferencing", ACM Multimedia Systems, Volume 2, Number 6, (January 1995), pages 267-279.

[TAS98a] S. Tasaka, Y. Ishibashi, “Media Synchronisation in Heterogeneous Networks: Stored Media Case”, Vol.E81-B No.8 pp.1624-1636 August 1998.

[TAS98b] S. Tasaka, Y. Ishibashi, “A performance Comparison of single-Stream and Multi-Stream Approaches to Live Media Synchronisation”, Vol.E81-B No.11 pp.1988-1997, November 1998.

[TAS00] S. Tasaka, T. Nunome, Y. Ishibashi, “Live media synchronization quality of a retransmission-based error recovery scheme”, Proc. IEEE ICC, 2000, pp. 1535 - 1541, June 2000.

[TUR94] T. Turletti, “The INRIA videoconferencing System (IVS). ConneXions 8, 10 (October 1994), 20-24.

[WAT98] A. Watson & M. A. Sasse, "Measuring perceived quality of speech and video in multimedia conferencing applications", In Proceedings of ACM Multimedia'98, Bristol, UK, September 1998.

[WHI97] Paul P. White, “RSVP and Integrated Services in the Internet: A Tutorial”, IEEE Communications magazine, pp. 100-106, May 1997.

[WIL00] Wilson, G., “Multimedia Conferencing: What Cost to Users?”, In A Pras (eds.) Proceedings of Sixth EUNICE (European Network of Universities and companies in Information and Communication Engineering) Open European Summer School: Innovative Internet Applications, pp.173-180, September 13th - 15th, University of Twente, Enschede, the Netherlands. ISBN 90-3651-4983. HTML/pdf. UCL Research Note RN/00/35. (2000).

[XIE99] Y. Xie, C. Liu, M. J. Lee, T. N. Saadawi, “Adaptive multimedia synchronisation in a teleconference system” Multimedia Systems, Volume 7 Issue 4 (1999) pp 326-337.

[YAN96] C.C. Yang, J.H. Huang, “A Multimedia Synchronisation Model and Its Implementation in Transport Protocols”, IEEE Journal on Selected Areas in Communications, vol. 14, no. 1, pp. 212-225, January 1996.

[YAV92] R. Yavatkar, "MCP: A Protocol for Coordination and Temporal Synchronisation in ollaborative Applications", Proc. of the ICDCS 92, pp.606-613.

[YAV94] R. Yavatkar and K. Lakshman. Communication support for distributed collaborative applications. Multimedia Systems, 2(4), 1994. 16.

[YER98] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

[ZHA93] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. IEEE Network, 7(5):8-18, September 1993.

[ZAR96] P. N. Zarros, Myung J. Lee, Tarek N. Saadawi "Interparticipant synchronisation in real-time Multimedia conferencing using feedback" IEEE/ACM Transactions on Networking (TON) Volume 4, Issue 2 (April 1996) Pages: 173 – 180.

Lista de Figuras

Lista de Tablas