UNIVERSIDAD CARLOS III DE MADRID

IMPLEMENTACIÓN DE UNA ARQUITECTURA DE VOD CON MULTIDESCRIPCIÓN Y SEÑALIZACIÓN SIP PARA ENTORNOS DE REDES DE SIGUIENTE GENERACIÓN

PROYECTO FIN DE CARRERA

TUTOR: JAIME JOSÉ GARCÍA REINOSO

SEPTIEMBRE 2008

IGNACIO ALONSO MUÑOZ

INGENIERÍA DE TELECOMUNICACIONES

Agradecimientos

Quiero dar mi más sincero agradecimiento a Jaime por haberme dirigido este proyecto final de carrera, por todo el tiempo que me ha dedicado y especialmente por todas las facilidades que me ha dado para desarrollarlo.

En estos agradecimientos no pueden faltar ni Mara, ni mi familia, que siempre han estado para apoyarme y aguantarme.

Este proyecto pone punto y final a mi etapa universitaria. Han sido unos años en los que he disfrutado de esta carrera, de muchas de sus asignaturas y del ambiente de esta universidad. Hoy comienza el recuerdo de unos años en lo que he conocido a muchas personas, entre ellas a Maxi y a Víctor, con los que sin duda las clases y laboratorios acabaron siendo más entretenidos.

ÍNDICE

Índice de Figuras

Índice de Tablas

RESUMEN

Las NGN son redes multiservicio capaces de manejar voz, datos y vídeo. Al ser independiente la capa de servicio de la tecnología de transporte, las NGN aportan muchas ventajas a aplicaciones sensibles a retardos como pueden ser IPTV, VoD o VoIP. Este proyecto se centra en aplicaciones de VoD donde la transmisión es unicast, en tiempo real y donde es más difícil garantizar la calidad de servicio. ETSI-TISPAN ha seleccionado SIP para la señalización en las NGN, por lo que el establecimiento de la conexión entre el cliente y los servidores debe usar este protocolo.

Este proyecto tiene como objetivo desarrollar una aplicación en Java para la provisión del servicio de VoD utilizando codificación con descripción múltiple (MDC). La arquitectura propuesta está formada por un cliente que solicita un fichero de vídeo con unas determinadas prestaciones, un servidor de vídeo que sirve dicho fichero y un servidor de control que gestiona las peticiones de los clientes, redirigiéndolos al servidor de vídeo que mejor les pueda atender. Del lado del cliente parten las peticiones de establecimiento, modificación y liberación de la sesión multimedia, que se basan en los protocolos SIP y SDP.

La codificación de vídeo con descripción múltiple (MDC), codifica una señal en dos o más flujos llamados descriptores. Cada uno de los descriptores puede ser decodificado de forma independiente brindando una reproducción útil de la señal original. Cuantos más descriptores se reciban, mejor será la calidad de la señal recibida. Al finalizar este proyecto no existe un conjunto de códecs MDC, por lo que no ha sido posible integrarlos en nuestra aplicación de VoD. Sin embargo, se han realizado pruebas para validar la propuesta y conocer los parámetros que la condicionan: el tipo de conexión y el retardo de playout.

PALABRAS CLAVE

Redes de Próxima Generación (NGN), Video bajo Demanda (VoD), SIP, SDP, Codificación con Multidescripción (MDC)

Capítulo 1
Introducción

En este primer capítulo se ofrece una descripción detallada del proyecto. En primer lugar se explica cuáles han sido las razones que motivaron la realización de este proyecto y posteriormente se presentan los objetivos que se quieren alcanzar a la vista de las mejoras y nuevas aportaciones que puede introducir.

Otra parte importante de este capítulo introductorio es la exposición de los tres principales puntos en los que se estructura el proyecto: Vídeo Bajo Demanda (VoD), Redes de Próxima Generación (NGN) y Codificación con Multidescripción (MDC). Cada uno de estos puntos ha condicionado las características de la aplicación que se ha implementado dentro de este proyecto.

1.1. Motivación y Objetivos

Internet, en sus primeras fases, nunca fue pensada para el transporte de aplicaciones en tiempo real como puede ser el audio o vídeo. Sin embargo, hoy en día se pretende transportar por Internet cualquier tipo de servicio, unificando así una red común, lo cual ahorraría a los proveedores de red costes de despliegue y mantenimiento.

Existen algunas iniciativas que permiten tener una solución intermedia en donde, aún no teniendo unas garantías de entrega, se pueda obtener la mejor calidad posible en transmisiones de audio y vídeo. Las NGN proporcionan altos niveles de QoS donde se requiera, a esto hay añadirle la utilización en este proyecto de protocolos como RTP y RTCP que incrementan la calidad de servicio.

Nuestra aplicación de VoD está orientada a un entorno de NGN. Este tipo de red obliga a la separación del plano de control y el de usuario. El protocolo preferido para utilizar en plano de control es SIP. Actualmente hay soluciones para la petición de vídeo con SIP. Se partirá de estas propuestas, se depurará su funcionalidad y se añadirán fundamentalmente tres nuevas funciones:

- Petición de vídeo con multidescripción.
- Modificación de la sesión establecida.
- Negociación entre un cliente y un servidor de control.

Un apartado importante de este proyecto es la multidescripción. Consiste en el envío de diferentes descriptores que forman un único flujo de vídeo en el Servidor de Vídeo. En función del número de descriptores que recibe el cliente, se podrá reconstruir la señal de vídeo con mejor o peor calidad.

Los objetivos de este proyecto son principalmente dos. En primer lugar la implementación de una aplicación en Java que realice el intercambio de mensajes SIP entre los diferentes actores de la aplicación de VoD. Estos mensajes permitirán a un cliente solicitar un fichero de vídeo a un servidor para que este se lo sirva en tiempo real. Con la sesión multimedia ya establecida, el cliente podrá enviar nuevas peticiones SIP al servidor para modificar las condiciones de la sesión. Loa capítulos 3 y 4 tratan sobre el diseño y la implementación del sistema respectivamente.

El segundo objetivo inicial de este proyecto era integrar esta aplicación de intercambio de mensajes SIP con los codificadores de multidescripción. Sin embargo estos codificadores se encuentran en una primera fase de desarrollo, todavía no es posible codificar un flujo de vídeo en varios descriptores en el servidor y volver a decodificarlos en única señal de vídeo en el cliente de forma eficiente.

Por esta razón se ha incorporado un capítulo de validación de la propuesta, que pretende obtener resultados reales sobre las ventajas de utilizar codificación con multidescripción en una aplicación de VoD. También sirve para fijar los puntos que condicionan la calidad final de la señal de vídeo como son el retardo de Playout, el buffer PoB y las calidades de las conexiones de cada descriptor. Dichos puntos se explicarán más adelante.

1.2. VoD

El VoD es un sistema de visionado de imágenes de vídeo sobre IP que se caracteriza por dar acceso a contenidos multimedia de forma personalizada en tiempo real.

En referencia al VoD en NGN, la principal ventaja es la garantía de calidad del vídeo. Un servidor de vídeo distribuirá al cliente el mejor códec que tenga disponible y que permita la carga del sistema. Desplegar VoD junto con una amplia cartera de servicios permite la reducción de costes debido a la infraestructura única, una mayor rapidez de integración en el mercado gracias a la preponderancia de Internet, además de ofrecer una gestión, operación y mantenimiento unificada de los servicios.

La arquitectura de red propuesta en este proyecto se refleja en la figura 1. En el capítulo 2, "Estado del arte", se describe la base teórica de los principales protocolos citados a continuación.

- **Protocolo de transporte: UDP**
  Aunque UDP no ofrece control, ni garantías de QoS, es la capa de transporte que utilizaremos porque es más importante recibir la información en el momento adecuado que la fiabilidad del transporte. Además RTCP ya aporta control al stream de datos RTP.
- **Transporte de datos: RTP**
  RTP se encarga del transporte de información multimedia en tiempo real.
- **Control de datos: RTCP**
  Asociado al flujo de datos de RTP, RTCP proporciona información de control, sin transportar ningún tipo de datos por sí mismo. En nuestro proyecto también se emplearán los mensajes RTCP para "funciones de control remoto".
- **Señalización: SIP**
  Al igual que en el núcleo de las NGN, el protocolo de señalización preferido es SIP.
- **Descripción de sesión: SDP**
  SIP actúa como envoltura al SDP, que describe los parámetros de inicialización de los flujos multimedia.

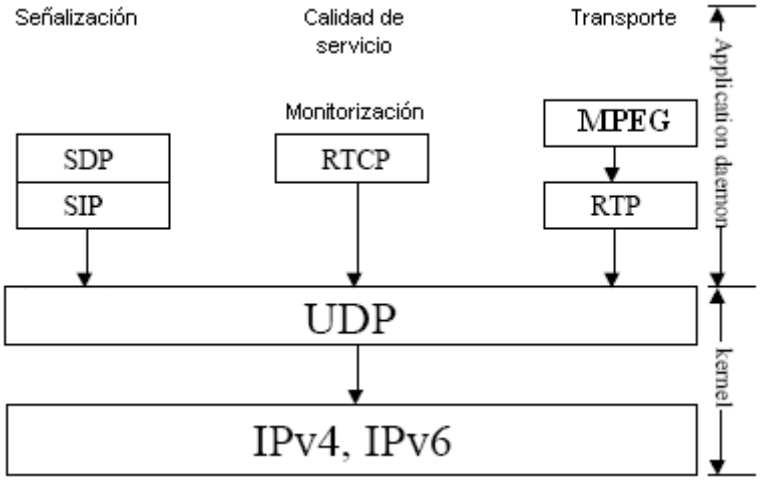

Figura 1: Arquitectura de protocolo de la aplicación de VoD

En cuanto a la estructura de componentes de la aplicación se basa en el draft “An extension of RTP and RTCP protocols For Video-on-Demand systems” [1] en el que fundamentalmente se propone una extensión de RTP y RTCP. Además especifica los diferentes actores de esta arquitectura, así como los procesos que intervienen en la aplicación. En este draft se propone enviar las peticiones mediante paquetes RTCP, partiendo de que las conexiones RTP y RTCP ya están establecidas entre los servidores y el cliente. Esta solución no será válida para nuestro proyecto ya que buscamos realizar las peticiones de vídeo utilizando SIP/SDP.

En el draft [1]se establecen los tres componentes fundamentales en una aplicación de VoD:

- **Cliente:** Envía peticiones de un archivo de vídeo al Servidor de Control. Se compone de un software para reproducir el vídeo solicitado y un buffer que almacena los datos recibidos del Servidor de Vídeo.
- **Servidor de Control:** Lleva a cabo operaciones de control en la transferencia de datos, es el punto de conexión entre el cliente y el Servidor de Vídeo. Cuando recibe una petición de un cliente, ejecuta una aplicación que estima el Servidor de Vídeo con menor carga para atender al cliente denominada "Load Balancing Tool" (LBT).
- **Servidor de Vídeo:** Almacena los archivos de vídeo y mantiene un listado de estos en una base de datos a la que accede el Servidor de Control. Ante una petición de vídeo de un cliente, transfiere el archivo solicitado. El cliente habrá obtenido su dirección a través del Servidor de Control, al haber evaluado este que es el servidor con menor carga.

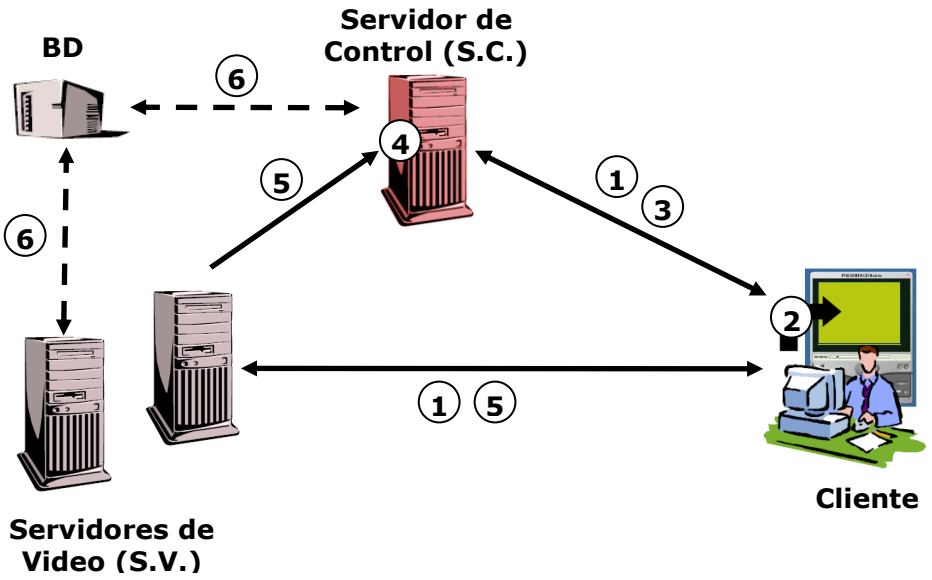

Figura 2: Procesos de la aplicación de VoD

También se indican los procesos que intervienen en los diferentes equipos que hacen referencia a la figura 2.

1. **Cliente:** Este proceso se encuentra en el equipo del cliente, se encarga de enviar peticiones de ficheros de vídeo y obtener las respuestas.
2. **Player:** Se encuentra en el equipo del cliente, es el software que permite reproducir los ficheros de vídeo.
3. **Servidor de Control:** Se encarga de todas las funciones de control en el S.C. atendiendo las peticiones de los clientes y accediendo a la base de datos.
4. **Load Balancing Tool (LBT):** Como se ha comentado anteriormente, se ejecuta en el S.C. evaluando el servidor con menor carga para que atienda la petición del cliente.
5. **Servidor de Vídeo:** Hilo que se encarga de enviar información de control al S.C. periódicamente y servir el vídeo al cliente cuando lo solicita.
6. **Meta Data Manager (MDM):** Se ejecuta en el S.V. y en el S.C. cuando hay interactuar con la base de datos.

Respecto al encapsulado de datos de vídeo, los estándares de compresión de vídeo nos brindan un gran número de beneficios entre los que destacan el aseguramiento de la interoperabilidad y la comunicación entre codificadores y decodificadores realizados por personas o compañías diferentes. Nuestra aplicación de VoD debe soportar una amplia variedad de códecs de vídeo, por esta razón se empleará el reproductor multimedia "VLC media player" [2].

VideoLAN es una solución de software completa, no es solo reproductor multimedia, también puede ser utilizado como servidor en unicast o multicast, en IPv4 o IPv6 , en una red de banda ancha para la transmisión de vídeo. Por consiguiente, como se muestra en la figura 3, VLC se ejecutará en el cliente para la recepción y reproducción del flujo de vídeo, y en el servidor de vídeo para su volcado a la red.

Figura 3: Arquitectura VideoLAN

1.3. NGN

Este apartado solo pretende dar una sencilla descripción de las NGN para entender el entorno de ejecución de la aplicación de VoD de este proyecto. Esta explicación parte fundamentalmente del artículo [3].

Las redes de próxima generación NGN (Next Generation Networks) se basan en el despliegue de redes multiservicio con conmutación de paquetes IP. También proporcionan soporte de calidad de servicio (QoS) en el transporte de la información, garantizando la independencia entre las funciones de control de los servicios y las tecnologías de transporte, soportando al mismo tiempo la movilidad de usuarios y servicios.

En la universidad se centraron en un escenario residencial con acceso fijo a una red de banda ancha, donde el nodo de pasarela residencial, RGW (Residencial Gateway), es un elemento clave a la hora de proveer QoS extremo a extremo a los usuarios residenciales.

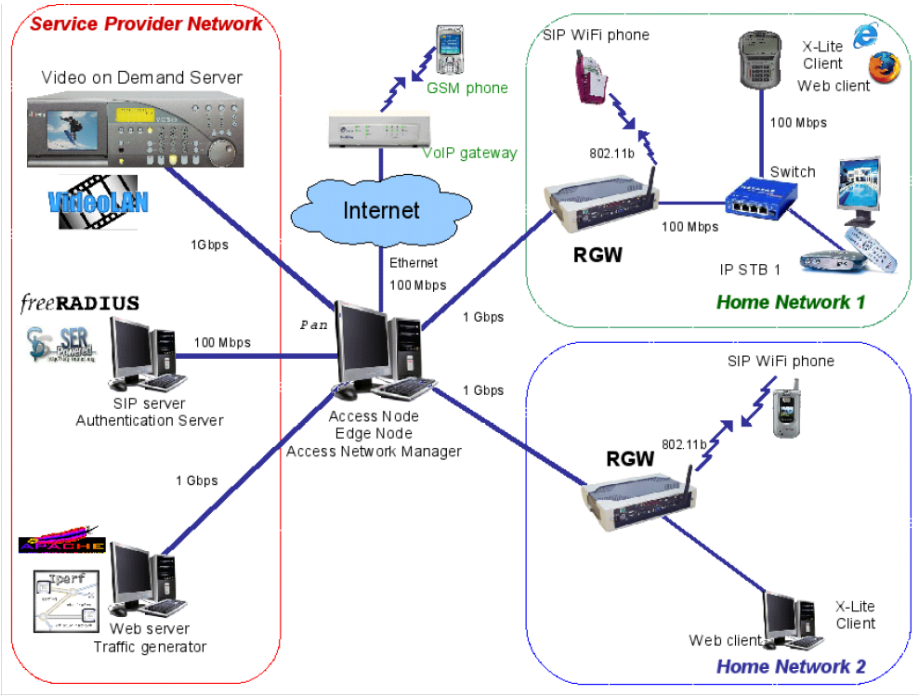

Figura 4: Escenario de NGN

La versatilidad que proporciona el protocolo de señalización SIP a la hora de invocar peticiones de servicio con soporte de QoS y el masivo uso de este protocolo en una gran variedad de terminales permite considerar el modelo de QoS basado en la señalización estándar de SIP como arquitectura para el de RGW.

Hay una serie de inconvenientes en el despliegue de una aplicación de VoD en un escenario NGN, ya que es necesario proporcionar al cliente la mejor calidad de vídeo en una transmisión tan exigente y costosa como es la unicast. Una arquitectura P2P permitiría que los usuarios finales pudieran distribuirse porciones de vídeo que hayan recibido previamente y se hayan almacenado en el disco duro del RGW. De esta forma el tráfico de vídeo se acercaría más a los usuarios finales descongestionando el nodo de red.

Una solución P2P presenta problemas de seguridad, espacio en el disco duro del RGW, pero sobretodo de ancho de banda. El empleo de codificación escalable (LC) o con multidescripción (MDC), reduciría el tamaño de los flujos de vídeo y por consiguiente, del ancho de banda. Emplearíamos tantos descriptores como de ancho de banda dispusiésemos, lo que fijaría la calidad final de la señal de vídeo.

Las NGN pueden proporcionar conexiones con diferentes calidades de servicio de tal forma que se garantice la recepción de al menos un descriptor con la mejor conexión, para poder así reconstruir la señal de vídeo. El resto de descriptores irían por conexiones con peores propiedades al tener mayor retardo, jitter y probabilidad de pérdidas.

En el capítulo 6 sobre la validación de la propuesta, se trabaja en la importancia de las características de la conexión para la correcta recepción de los descriptores en el cliente. Se realizan pruebas con cuatro tipos de conexiones que se pueden garantizar en NGN:

- De alta prioridad: High Priority (HP) o Low Latency.
- De media prioridad: Medium Priority (MP) o Real Time.
- De baja prioridad: Low Priority (LP) o Elastic.
- De mejor esfuerzo: Best Effort.

Resumiendo, las principales características de este proyecto relacionadas con las NGN son:

- La señalización SIP: Las peticiones del cliente al servidor se realizan con esta arquitectura, la única señalización admisible en el nodo de red de las NGN.
- Codificación MDC y LC: El empleo de esta codificación permitiría a la NGN ofrecer VoD con la máxima calidad posible sin comprometer a otras aplicaciones.

1.4. Codificación con Multidescripción

El método ideal para enviar vídeos sería generar un flujo de vídeo (stream) desde el servidor al cliente en respuesta a una solicitud del mismo. El cliente reproduciría el flujo entrante en tiempo real a medida que va recibiendo los datos. Sin embargo, el vídeo transmitido sobre redes de paquetes B*est Effort* tiene complicaciones debidas a factores como el desconocimiento, la variación en el tiempo del ancho de banda, demoras, pérdidas de datos y otros factores como compartir recursos de la red eficientemente y la realización correcta de la comunicación entre un punto y varios puntos.

Los métodos de codificación basados en el contenido y en la forma de onda buscan optimizar la eficiencia del proceso de codificación para una tasa de bit fija. Esto es un problema cuando múltiples usuarios intentan acceder al mismo material de vídeo a través de diferentes conexiones. La escalabilidad se refiere a la capacidad de recuperar imágenes o vídeos que tengan una secuencia de bits con información parcial.

La codificación escalable (Layered Coding – LC) genera una orden de prioridad para los datos de vídeo que permiten un descarte inteligente. Las diferentes prioridades pueden ser utilizadas para enviar vídeo confiable mediante el uso de protección de error desigual. A pesar de que el vídeo escalable con establecimiento de prioridad para los datos de vídeo permite coincidir la transmisión con el rendimiento de la red, no es muy común que las redes soporten un sistema de QoS. Este problema y las pérdidas del canal imprevisibles motivaron el desarrollo de la codificación con descripción múltiple (MDC) en la que se basará este proyecto.

La MDC codifica una señal en dos o más flujos independientes llamados descriptores múltiples (MD). La codificación MDC tiene propiedades importantes:

- Cada una de las descripciones puede ser decodificada de manera independiente de forma que la calidad en la recepción mejore en función del número de capas del que se disponga.
- Las descripciones múltiples contienen información complementaria que va mejorando la calidad de la señal recibida cuanto más MD se reciban correctamente.
- Permite aprovechar la existencia de múltiples caminos entre un único origen y el destino, o la posibilidad de que haya varios nodos sirviendo datos. [8][4]
- A diferencia del vídeo escalable (LC) común, la capa base no es imprescindible, ya que cualquiera de los MD puede generar vídeo.

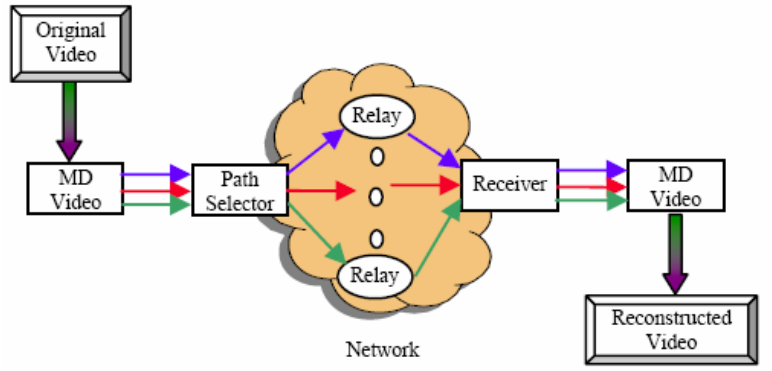

Figura 5: Ejemplo de distribución multicamino con MDC

En cuanto a la codificación y decodificación específica en multidescripción hay pocos artículos al respecto, los más destacados son [5], [6], [7] y [8]. Aunque el objetivo de este proyecto no es investigar las diferentes técnicas de codificación con multidescripción, es importante conocer en que se basan para hacernos una idea de la dificultad y los costes de estas técnicas.

Para entender bien los diferentes métodos de codificación, es necesario explicar antes como se forma un flujo de vídeo. En primer lugar se debe comprender la diferencia entre un códec y un formato contenedor:

- Un códec es un algoritmo de compresión, utilizado para reducir el tamaño de un flujo. Existen códecs de audio y códecs de vídeo como MPEG-1, MPEG-2, MPEG-4, Vorbis o DivX.
- Un formato contenedor contiene uno o varios flujos ya codificados por códecs. A menudo, hay un flujo de audio y uno de vídeo. Los principales tipos de formato contenedor son AVI, Ogg, MOV y ASF.

Existe un caso particular en MPEG. Pero MPEG es también un formato contenedor de varios tipos: ES, PS, y TS. El encapsulado de transporte de vídeo más utilizado es MPEG-2 Transport Stream, estándar ISO 13818. Es más, el streaming de VLC UDP/RTP requiere encapsulado MPEG-TS, no permite otro encapsulado.

MPEG-2 es por lo general usado para codificar audio y vídeo para señales de transmisión que incluyen televisión digital terrestre, por satélite o cable. Un archivo MPEG está compuesto de unas secuencias cíclicas llamadas GOP (Group Of Pictures - grupo de imágenes) que, como su nombre indica, engloban cierto número de fotogramas, normalmente 15. Los GOP's están formados por tres grupos distintos de fotogramas:

- **I-picture (imágen-I, cuadros internos):** Son los únicos estrictamente necesarios. Cada cuadro es comprimido con un tipo de compresión llamada "Intra-frame DCT coding" (codificación interna de cuadros DCT por transformación discreta de coseno). Si nuestro GOP tan sólo contara con cuadros-I tendríamos una secuencia de JPG's, el llamado Motion JPEG o MJPG.

  Como se refleja en la figura 6, la imagen se divide en cuadros NxN (en el caso de MPEG de 8x8) pero la distribución de la cantidad de información no se realiza de forma equitativa asignando la misma cantidad de información por cada píxel, sino que la cantidad de luminosidad y color son analizadas y los valores cercanos a cero se desprecian, asignando la cantidad despreciada a otros píxeles con mayor cantidad de información y, por tanto, más importantes.

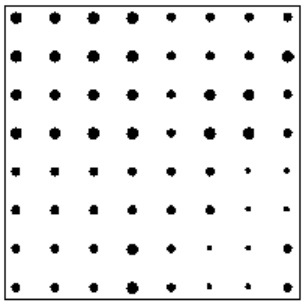

Figura 6: Representación proporcional de un I-picture

- **P-picture y B-picture (imágen-P e imágen-B - cuadros de predicción y cuadros de predicción bidireccionales):** Estas imágenes, en lugar de estar compuestas por los 101.376 píxeles (en el caso de tener un vídeo de 352x288), lo están tan sólo por los 1.376 cambiantes. Los cuadros-P analizan los cambios con respecto a cuadros I u otros cuadros P anteriores, mientras que los cuadros-B pueden analizar los cambios de cuadros-P anteriores y posteriores ("B" de bidireccionales) alcanzando los mayores grados de compresión.

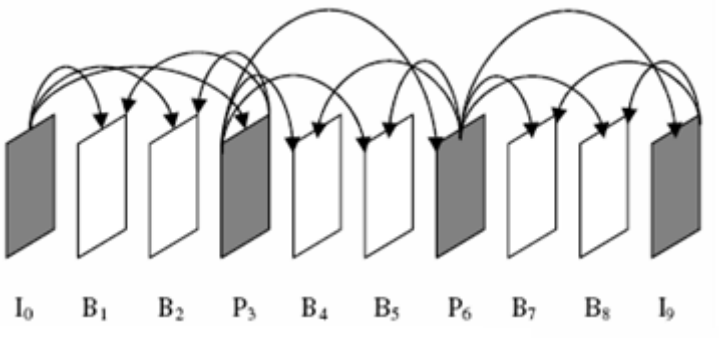

Figura 7: Secuencia GOP

Las técnicas de codificación de los artículos [5], [6], [7] y [8] se basan en la manipulación de los frames I, P y B (ej: Escoger solo los frames pares o impares de una secuencia) o en diferentes técnicas de compresión de los cuadros DCT. Este hecho traslada una importante dificultad técnica en el lado del servidor a la hora de codificar los diferentes descriptores, pero especialmente en el lado del cliente donde debe decodificarlos y reconstruir la señal de vídeo en función de los descriptores recibidos. En la figura 8 procedente del artículo [5], se detalla la secuencia de pasos que debe producirse.

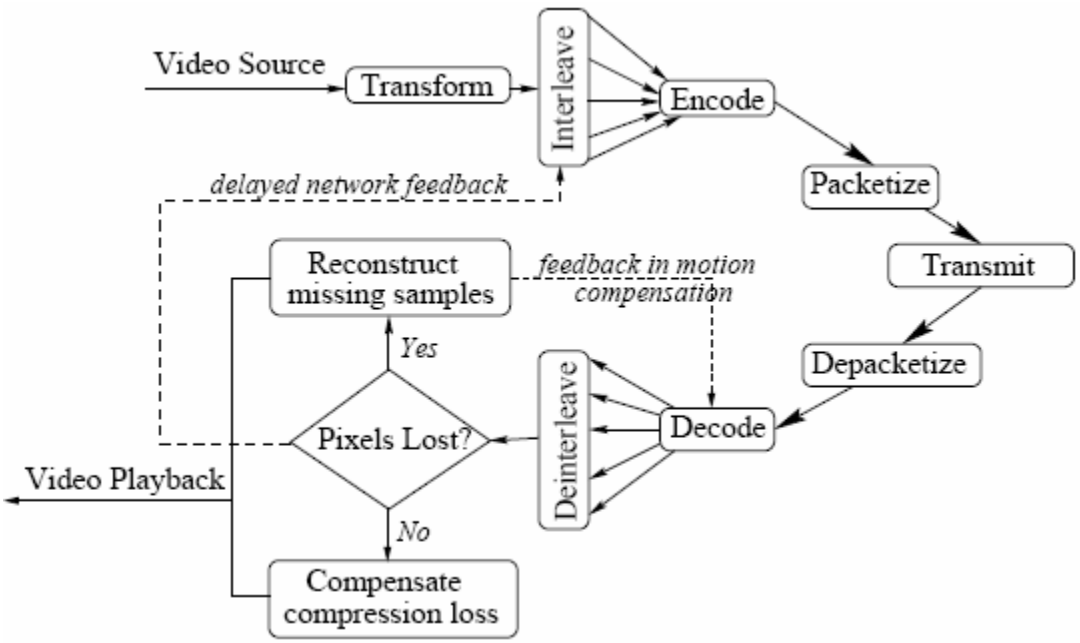

Figura 8: Prototipo de Video Streaming
[http://manip.crhc.uiuc.edu/Wah/papers/C131/C131pres.pdf]

Estas técnicas de codificación con multidescripción para vídeo se encuentran en fase de desarrollo. Actualmente se trabaja en implementaciones de los codificadores más rápidas, por lo que su utilización en este proyecto no es posible. De todas maneras, como se ha dicho anteriormente, este proyecto no va a tratar el tema de la codificación con multidescripción. Se centrará en la señalización de la transmisión de los diferentes descriptores multimedia entre el servidor de vídeo y el cliente con señalización SIP a nivel de SDP.

Capítulo 2
Estado del Arte

Como se indicó en la introducción de este proyecto, en este capítulo se dará una descripción general de los principales protocolos que forman nuestra aplicación: SIP, SDP, RTP y RTCP. Los protocolos más importantes en este proyecto son los dos primeros, ya que son los que se han implementado en Java.

En este proyecto se emplea VLC [2] para la transmisión de vídeo. Este programa envía el flujo de vídeo mediante el protocolo RTP, sin embargo, no utiliza el protocolo RTCP. Se incluye el estado del arte de RTP y RTCP para tener una visión global y más completa de los protocolos que normalmente intervienen en una aplicación de VoD.

2.1. SIP

SIP [9] es un protocolo de señalización de nivel de aplicación desarrollado para el establecimiento, modificación y finalización de sesiones multimedia. Se desarrolló siguiendo los procedimientos del IETF, mientras que H.323 es una recomendación de la ITU-T.

Sus funciones básicas son:

- Establecer, modificar y finalizar llamadas/sesiones.
- Registro y localización de participantes. Movilidad.
- Gestión del conjunto de participantes y de los componentes del sistema.
- Descripción de características de las sesiones y negociación de capacidades de los participantes.

SIP no es un protocolo que proporcione de forma integrada comunicaciones multimedia, y por ello se utiliza en conjunción con otros protocolos. Los que utilizaremos en este proyecto son SDP para descripción de sesiones multimedia y RTP-RTCP para la transmisión de la información multimedia.

Un usuario SIP utiliza un Agente de Usuario para enviar y recibir mensajes SIP. Has dos tipos de UA: Agente de Usuario de Cliente (UAC) que crea peticiones SIP y Agente de Usuario Servidor (UAS) que interactúa con el usuario recibiendo una petición SIP y contestado a ella.

Principalmente hay tres tipos de servidores SIP:

- **Proxy:** Actúa encaminando *Requests* hacia usas, o encaminando respuestas hacia UACs. Están diseñados para ser relativamente transparentes y no pueden actuar sobre los mensajes SIP. Los proxies pueden ser con estado permanente (*Stateless proxies*) o sin estado (*Statefull proxies*).
- **Redirect:** Recibe *Requests* de clientes o servidores SIP, obtiene la asociación entre la dirección SIP del usuario que intenta contactar y sus direcciones de contacto actualizadas, y devuelve dichas direcciones de contacto del peticionario.
- **Registration:** Gestiona la información de localización de usuarios en un determinado dominio.

La estructura de las peticiones y respuestas SIP es similar a la de las peticiones y respuestas HTTP. Cada mensaje SIP contiene una cabecera y un cuerpo, separados por una línea en blanco, según se representa en la figura 9.

La primera línea de la cabecera indica el tipo de petición o respuesta. Las cinco cabeceras Via, From, To, Call-ID, y CSeq están presentes en todos los mensajes e identifican de forma unívoca cada mensaje dentro de cada llamada. Su significado es el siguiente:

- *Via:* Indica el transporte usado para el envío e identifica la ruta del request, por ello cada proxy añade una línea a este campo.
- *From:* Indica la dirección del origen de la petición.
- *To:* Indica la dirección del destinatario de la petición.
- *Call-Id:* Identificador único para cada llamada y contiene la dirección del host. Debe ser igual para todos los mensajes dentro de una transacción. En los mensajes de tipo OPTIONS, el campo Call-ID permite relacionar las peticiones y respuestas
- Cseq: Se inicia con un número aleatorio e identifica de forma secuencial cada petición.

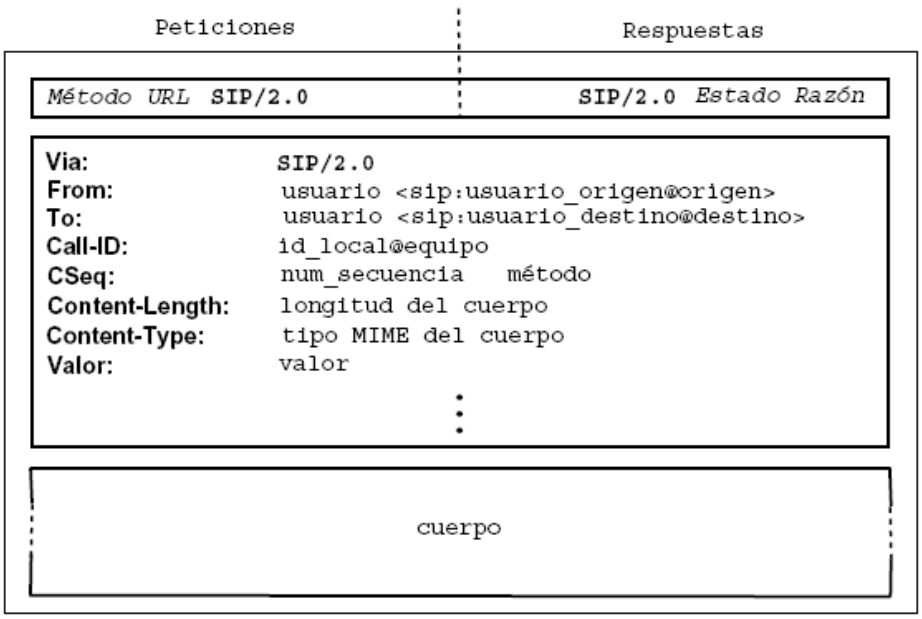

Figura 9: Cabecera de un mensaje SIP

En SIP hay cinco métodos básicos: INVITE, ACK, CANCEL, BYE, REGISTER y OPTIONS. SIP prevé mecanismos para añadir nuevos métodos de ampliación, así como para que los servidores que desconozcan estos nuevos métodos puedan responder con un código de error (código 501), indicando la lista completa de métodos admitidos en un campo, Allow, de la cabecera de la respuesta.

En la tabla 1 se indica la especificación de estos métodos y a continuación se describen.

- **Método INVITE.** Los campos de cabecera de las peticiones de inicio de llamada contienen, entre otros datos, la dirección de origen y destino de la llamada, el asunto, la prioridad, los equipos intermedios por los que se solicita que pase la llamada y las preferencias acerca de la localización del destinatario. El cuerpo contiene una descripción de la información multimedia que se pretende intercambiar en la sesión. Este componente, transparente para el protocolo SIP, se suele codificar siguiendo el formato SDP.

  Las respuestas a peticiones INVITE contienen en el cuerpo la descripción de la información multimedia que está dispuesto a intercambiar el destinatario de la petición inicial.

  Es posible modificar las características de una sesión multimedia posteriormente a su establecimiento. Las nuevas características se han de especificar en el cuerpo de un mensaje INVITE enviado con el mismo valor Call-ID, y nuevos valores para el cuerpo y posiblemente alguno de los campos de cabecera.

- **Método ACK**. Los mensajes de este tipo sirven de confirmación para intercambios fiables de mensajes de invitación. Los clientes deben generar mensajes ACK para confirmar que se ha recibido el mensaje final de aceptación correspondiente a una invitación. El cuerpo de los mensajes ACK puede contener la descripción de la sesión si el cliente decide realizar modificaciones.

- **Método BYE.** Estos mensajes indican a los servidores que un cliente desea finalizar la conexión entre dos participantes en una sesión. Se pueden generar tanto en los agentes que iniciaron la llamada como en los que recibieron la invitación.
- **Método CANCEL**. Empleado para cancelar una llamada pendiente, aunque su recepción por parte de un UAS no garantiza que éste no responda posteriormente, sino que simplemente constituye una sugerencia realizada por el remitente para optimizar el uso de la red.

  Estos mensajes deben contener el mismo valor para los campos Call-ID, To, From, y CSeq que el mensaje de invitación original. En todo caso, estos mensajes nunca finalizan una llamada ya establecida.
- **Método OPTIONS.** Útil para solicitar información acerca de las posibilidades de un servidor. Los servidores de redirección y los proxys simplemente los reenvían. Otros servidores pueden responder con un mensaje en el que indiquen sus capacidades o bien con la respuesta que hubiesen dado a una invitación.
- **Método REGISTER.** Los mensajes REGISTER proporcionan la localización de un agente de usuario a los servidores de registro. En ellos, los agentes de usuario clientes notifican a los proxys o los servidores de redirección la dirección o direcciones en las cuales se encuentra un usuario.

  En estos mensajes el campo To de la cabecera indica la dirección que se ha de registrar, mientras que el campo From indica la dirección del usuario responsable del registro. Para el registro de dispositivos durante su arranque se ha reservado una dirección multicast, sip.mcast.net, a la que pueden enviar mensajes REGISTER.

  Las respuestas a los mensajes de tipo REGISTER contienen información de configuración para los agentes de usuario, ampliable mediante las funciones de negociación de capacidades.
- **Método INFO**. El método INFO se ha definido para incorporar la posibilidad de enviar mensajes de control durante una sesión. Las peticiones de este tipo no cambian el estado de las llamadas ni los parámetros de la sesión, sino que se emplean para intercambiar cualquier tipo de información de control durante una llamada. Se trata de un método de carácter general, cuya cabecera y cuerpo se pueden emplear para enviar información de cualquier tipo.

  Este método se ha definido como un mecanismo abierto a la definición posterior de otros tipos de señalizaciones, incluyendo guías generales para ello. Este método juega un papel fundamental en la adaptación de la señalización SS7 de la red telefónica a redes basadas en SIP , permitiendo emular los mensajes que en la red telefónica se transmiten durante las llamadas.
- **Métodos SUBSCRIBE, UNSUBSCRIBE Y NOTIFY.** El protocolo PINT define ampliaciones tanto para SIP como para SDP. En particular define estos tres nuevos métodos SIP, mediante los cuales es posible solicitar, desde redes IP, la prestación de servicios telefónicos por parte de dispositivos situados en redes tradicionales, es decir, dotar a dispositivos SIP de capacidades de presencia.

  Los clientes SIP pueden enviar mensajes SUBSCRIBE a servidores PINT para suscribirse a sesiones de servicios telefónicos. La relación entre los métodos SUBSCRIBE y UNSUBSCRIBE es análoga a la existente entre los métodos INVITE y BYE. Esto es, tanto los agentes de usuario como los servidores PINT pueden enviar mensajes UNSUBSCRIBE con objeto de que cese la participación de un cliente en una sesión de servicios.

Los mensajes NOTIFY se envían durante una sesión de servicios telefónicos para comunicar a un suscriptor que se ha producido un cambio en el estado de la sesión; como puede ser, la finalización del envío de un fax.

- **Método PRACK.** Los agentes de usuario clientes envían peticiones de este tipo a los servidores para confirmar la recepción de respuestas provisionales. Se han introducido para aumentar la fiabilidad en los intercambios de peticiones y respuestas.
- **Método REFER.** Utilizado para que un UA pida a otro UA que establezca una sesión con un tercero. El ejemplo de uso más común es el de transferencia de llamadas entre usuarios.
- **Método MESSAGE.** Utilizado para transportar mensajes utilizando SIP, por ejemplo mensajes para servicios IM.
- **Método UPDATE.** Utilizado para modificar el estado de una sesión aún pendiente de establecer (sesiones en curso se modifican mediante un nuevo INVITE).

Especificación	Método	Descripción
RFC 3261	INVITE	Inicio de llamada/sesión
	ACK	Acuse de recibo final
	BYE	Terminar y transferir la llemada
	CANCEL	Cancelar búsqueda y llamada
	OPTIONS	Comprobación de capacidades del otro extremo
	REGISTER	Registro en servidores de localización
RFC 2976	INFO	Información durante la llamada
RFC 3265	SUBSCRIBE	Suscripción a una sesión de servicio
	UNSUBSCRIBE	Cancelación de suscripción
	NOTIFY	Notificación para suscriptores
RFC 3262	PRACK	Acuse de recibo provisional
RFC 3515	REFER	Llamada a tercero
RFC 3428	MESSAGE	Transporte de mensajes
RFC 3311	UPDATE	Modificación de estado de sesión

Tabla 1: Métodos básicos y de ampliación de SIP

La estructura de una respuesta es *<Sip-Version> <Status Code> <Reason-Phrase>.* En *Status Code* se indica el tipo de respuesta y en *Reason-Phrase* una explicación literal. Hay seis tipos de respuestas:

1xx - Mensajes provisionales.
2xx - Respuestas de éxito.
3xx - Respuestas de redirección.
4xx - Respuestas de fallo de método.
5xx - Respuestas de fallos de servidor.
6xx - Respuestas de fallos globales.

En la figura 10 se muestra un ejemplo básico de comunicación entre dos usuarios a través de un servidor SIP.

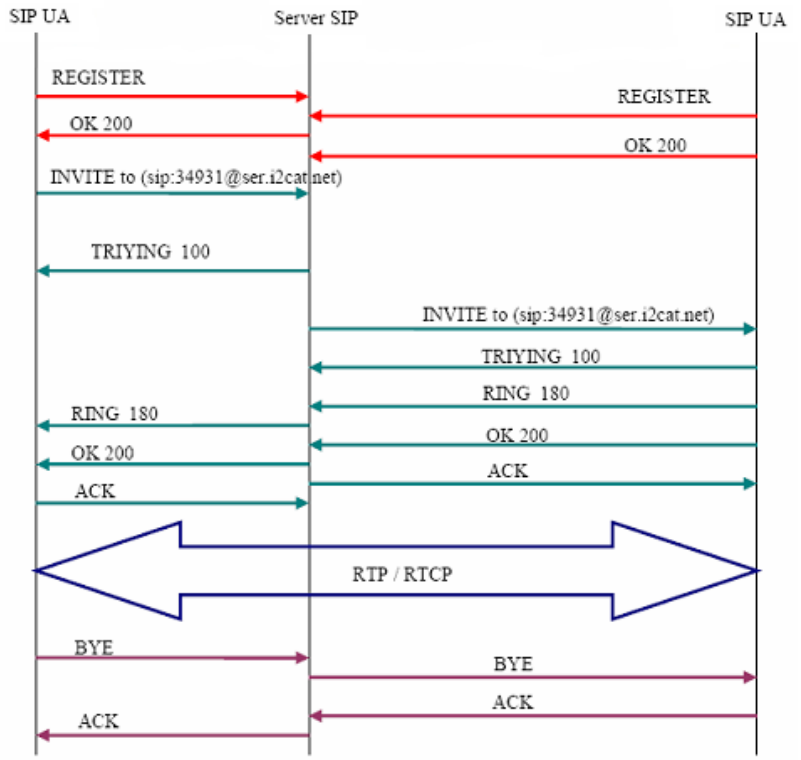

Figura 10: Establecimiento de sesión multimedia

Las dos primeras transacciones corresponden al registro de los usuarios. Los usuarios deben registrarse para poder ser encontrados por otros usuarios. En nuestra aplicación supondremos que los usuarios ya se han registrado en el correspondiente servidor SIP, por lo que directamente enviará peticiones INVITE a los servidores de vídeo.

La siguiente transacción corresponde a un establecimiento de sesión que comienza con una petición INVITE del usuario al proxy y finaliza con el mensaje de ACK de confirmación del cliente. En este momento la llamada está establecida, pasa a funcionar el protocolo de transporte RTP con los parámetros (puertos, direcciones, códecs, etc.) establecidos en la negociación mediante el protocolo SDP. La última transacción corresponde a una finalización de sesión.

2.2. SDP

Session Description Protocol (SDP) [10], es un protocolo para describir los parámetros de inicialización de los flujos multimedia. Fue publicado por el IETF en el RFC 2327. La última RFC se publicó en Julio del 2006, RFC 4566.

SDP comenzó como componente del SAP (Session Announcement Protocol), pero encontró otros usos en conjunto con RTP (Real-time Transport Protocol), SIP y como formato independiente para describir sesiones multicast. Se emplea actualmente para desempeñar dos funciones: descripción de sesiones y negociación de capacidades.

Cada descripción SDP proporciona la siguiente información:

- Nombre y propósito de la sesión.
- Tipo y formato de los medios que la componen.
- Intervalo(s) temporal(es) de desarrollo de la sesión.
- Parámetros necesarios para poder recibir e interpretar los datos: direcciones, puertos, formatos,etc.
- Información complementaria relativa a los recursos de red necesarios para participar en la sesión: ancho de banda e información de contacto del responsable de la sesión.

Una descripción de la sesión del SDP consiste en un número de líneas del texto de la forma:

<type>=<value>

donde <type> es un carácter específico y <value> es un texto estructurado cuyo formato depende de <type>. Los campos dentro de <value> están generalmente delimitados por un espacio.

La parte de nivel de sesión comienza con una línea con “v=” y continúa con la primera sección del nivel multimedia. Cada sección multimedia comienza con una línea “m=” y continúa con la siguiente sección o con la descripción entera de la sesión. Los valores del nivel de sesión son por defecto. Algunas líneas en cada descripción son obligatorias y en algunas son opcionales, pero todas deben aparecer exactamente en el siguiente orden. Las descripciones opcionales están marcados con “*”.

```
Descripción de sesión
  v = Versión del protocolo
  o = Identificador del autor y de la sesión
  s = Nombre de la sesión
  i = * Información de la sesión
  u = * URI de la descripción
  e = * Dirección de email
  p = * Número de teléfono
  c = * Información de la conexión
  b = * Información del ancho de banda
  z = * Zona horaria
  k = * Clave de cifrado
  a = * Atributos de sesión


Descripción del tiempo
  t = Tiempo de sesión activa
  r = * Tiempos de repetición
Descripción multimedia
   m = Nombre de los medios y dirección de transporte
   i = * Título multimedia
   c = * Información de la conexión (opcional si está incluido en el
         nivel de sesión)
   b=* Información del ancho de banda
   k=* Clave de cifrado
   a=* Atributos de sesión
```

Un ejemplo de la descripción de SDP es:

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/90000
```

Los mensajes SDP se encuentran en el cuerpo de los mensajes de SIP como se refleja en el siguiente ejemplo obtenido de la RFC 3261 de SIP.

```
************************************************************
* Content-Type: message/sip                                *
*                                                          *
* INVITE sip:bob@biloxi.com SIP/2.0                        *
* Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8  *
* To: Bob <bob@biloxi.com>                                 *
* From: Alice <alice@atlanta.com>;tag=1928301774           *
* Call-ID: a84b4c76e66710                                  *
* CSeq: 314159 INVITE                                      *
* Max-Forwards: 70                                         *
* Date: Thu, 21 Feb 2002 13:02:03 GMT                      *
* Contact: <sip:alice@pc33.atlanta.com>                    *
*                                                          *
* Content-Type: application/sdp                            *
*                                                          *
* v=0                                                      *
* o=alice 53655765 2353687637 IN IP4 pc33.atlanta.com      *
* s=Session SDP                                            *
* t=0 0                                                    *
* c=IN IP4 pc33.atlanta.com                                *
* m=audio 3456 RTP/AVP 0 1 3 99                            *
* a=rtpmap:0 PCMU/8000                                     *
************************************************************
```

La especificación original de SDP no contempla la función de negociación de capacidades, siendo por tanto poco adecuada para conferencias que se puedan iniciar de forma espontánea, como una llamada telefónica. La capacidad de negociar y seleccionar las características de cada sesión multimedia es una función básica de cualquier arquitectura de control, por lo que se requiere una solución específica para este problema.

El documento RFC 4317 [11] da ejemplos de los intercambios de oferta y respuesta de SDP. Los ejemplos incluyen la negociación y selección del códec, asentimiento y reanudación, adición y cancelación de streams. Los principales ejemplos son:

- **Negociación y selección del códec**

En este ejemplo se muestra una sesión de audio y vídeo en el cual se ofrecen múltiples códecs pero se acepta solamente uno. Para entender el ejemplo a continuación se describe el campo a=rtpmap:

a=rtpmap:<payload type> <encoding name>/<clock rate> [/<encoding parameters>]

[Offer]

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.example.com
s=
c=IN IP4 host.atlanta.example.com
t=0 0
m=audio 49170 RTP/AVP 0 8 97
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
m=video 51372 RTP/AVP 31 32
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
```

[Answer]

```
v=0
o=bob 2808844564 2808844564 IN IP4 host.biloxi.example.com
s=
c=IN IP4 host.biloxi.example.com
t=0 0
m=audio 49174 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 49170 RTP/AVP 32
a=rtpmap:32 MPV/90000
```

- **Asentimiento y reanudación**

Alice llama a Bob, pero cuando Bob contesta, coloca a Alice a la espera. Bob realiza posteriormente una segunda oferta. Alice cambia el puerto. La sesión Alice y Bob se activa después de la segunda respuesta de Alice.

[Offer]

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.example.com
s=
c=IN IP4 host.atlanta.example.com
t=0 0
m=audio 49170 RTP/AVP 0 97
a=rtpmap:0 PCMU/8000
a=rtpmap:97 iLBC/8000
```

[Answer]

```
v=0
o=bob 2808844564 2808844564 IN IP4 host.biloxi.example.com
s=
c=IN IP4 placeholder.biloxi.example.com
t=0 0
m=audio 49172 RTP/AVP 97
a=rtpmap:97 iLBC/8000
a=sendonly
```

[Second-Offer]

```
v=0
o=bob 2808844564 2808844565 IN IP4 host.biloxi.example.com
s=
c=IN IP4 host.biloxi.example.com
t=0 0
m=audio 49170 RTP/AVP 97
a=rtpmap:97 iLBC/8000
```

[Second-Answer]

```
v=0
o=alice 2890844526 2890844527 IN IP4 host.atlanta.example.com
s=
c=IN IP4 host.atlanta.example.com
t=0 0
m=audio 49178 RTP/AVP 97
a=rtpmap:97 iLBC/8000
```

- **Adición y eliminación de stream multimedia**

Solamente se establece una sesión de audio en el intercambio inicial entre Alice y Bob que usa códec de PCMU. Alicia agrega una stream de vídeo que es aceptado por Bob.

[Offer]

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.atlanta.example.com
s=
c=IN IP4 host.atlanta.example.com
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

[Answer]

```
v=0
o=bob 2808844564 2808844564 IN IP4 host.biloxi.example.com
s=
c=IN IP4 host.biloxi.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

[Second-Offer]

```
v=0
o=alice 2890844526 2890844527 IN IP4 host.atlanta.example.com
s=
c=IN IP4 host.atlanta.example.com
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 49172 RTP/AVP 31
a=rtpmap:31 H261/90000
```

[Second-Answer]

```
v=0
o=bob 2808844564 2808844565 IN IP4 host.biloxi.example.com
s=
c=IN IP4 host.biloxi.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 49168 RTP/AVP 31
a=rtpmap:31 H261/90000
```

2.3. RTP

Real-time Transport Protocol (RTP) [12], protocolo de nivel de aplicación utilizado para la transmisión de información en tiempo real de datos multimedia encapsulados dentro de paquetes del UDP.

Proporciona funciones de transporte de extremo a extremo, los nodos intermedios no interpretan RTP. Necesita de UDP para multiplexación, es más importante recibir la información en el momento adecuado que la fiabilidad del transporte. Al no existir garantías de calidad de servicio (QoS) va de la mano de RTCP utilizado para enviar periódicamente información de control asociada con el flujo de datos para que el emisor pueda ajustar su transmisión. RTP y RTCP se diseñan para ser independiente de las capas subyacentes del transporte y de red.

El formato de los mensajes RTP es el siguiente:

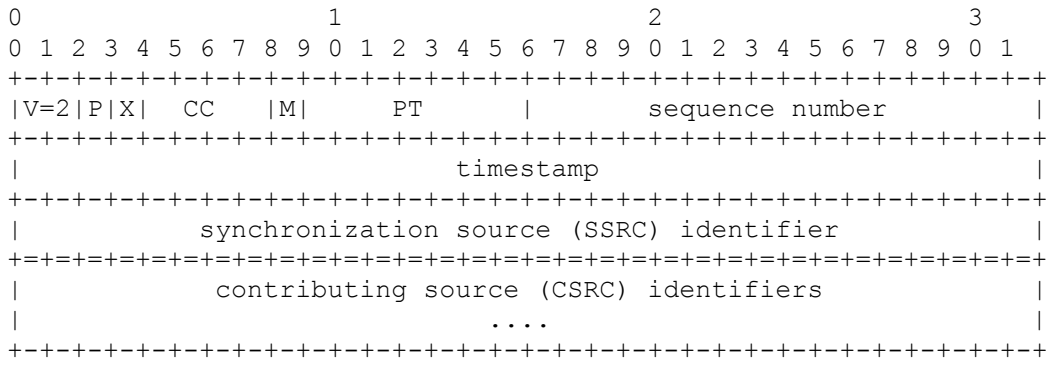

Figura 11: Formato de mensajes RTP

- V (Versión, 2 bits): Actualmente la 2.

- P (Bit de relleno, 1 bit): Indica si el paquete lleva relleno. Algunos algoritmos de cifrado trabajan con bloques de tamaño múltiplo de un número de bits dado. Si P=1, el último byte del relleno indica cuantos octetos hay que ignorar.

- X (Extensión, 1 bit): Si X=1, hay una cabecera de extensión al final.

- CC (Cuenta de contribuyentes, 4 bits): Los mezcladores indican de cuántas fuentes se están abasteciendo.

- M (Bit marcador, 1 bit): Puede marcar el comienzo de una ráfaga, instante ideal para ajustar el playout delay dinámicamente (al jitter de la red, compensar diferencias de los relojes en emisor-receptor, etc).

- PT (Tipo de datos, 7 bits): Indica el tipo de contenido (audio, vídeo) y su codificación (códec empleado, cifrado, etc). Los tipos de datos están definidos en la RFC 3551 [13].

- Sequence number (Número de secuencia, 16 bits): Cada fuente comienza con un número seleccionado aleatóriamente. Su objetivoo es detectar pérdidas

- Timestamp (Marca de tiempo, 32 bits): Indica el instante de generación del primer octeto de datos para la sincronización y cálculo de jitter de paquete. Se genera a partir de un reloj local a la fuente, cuya frecuencia del reloj es dependiente del formato de payload.

- SSRC (Synchronization Source Identifier, 32 bits): Identifica la fuente de sincronismo. El valor se escoge aleatoriamente de forma que dos orígenes en la misma sesión RTP nunca tengan el mismo SSRC.

- CSRC (Contributing Source Identifier, 32 bits): Matriz de 0 a 15 elementos que identifica las fuentes que contienen el payload del paquete.

En RTP, la multiplexación está proporcionada por la dirección de transporte de destino (dirección de red y número de acceso) que es diferente para cada sesión de RTP. Por ejemplo, en una teleconferencia formada por flujos de audio y vídeo codificados de forma independiente, cada uno pertenece a una sesión RTP con su propia dirección.

También se emplea típicamente para cada flujo en una sesión RTP, un identificador SSRC diferente. El origen establece este número asegurando que no se repita.

2.4. RTCP

RTP Control Protocol (RTCP) [12] tiene tres funciones principalmente: monitorización, identificación y control.

- *Monitorización de la congestión y del QoS:* Se envían informes sobre lo recibido a todos los participantes en una sesión. Se pueden tomar acciones correctivas (disminuir la calidad…).

- *Identificación:* Identificador de la fuente mediante una cadena de caracteres. Puede utilizarse para asociar flujos de diferentes sesiones. También permite enviar el nombre del usuario.

- *Control de sesión mínimo:* Abandono de sesión. Conocer el número de participantes.

Hay cinco tipos de mensajes RTCP:

I. ***Sender Report (SR):*** Indica lo que ha enviado (y deberían haber recibido los demás).

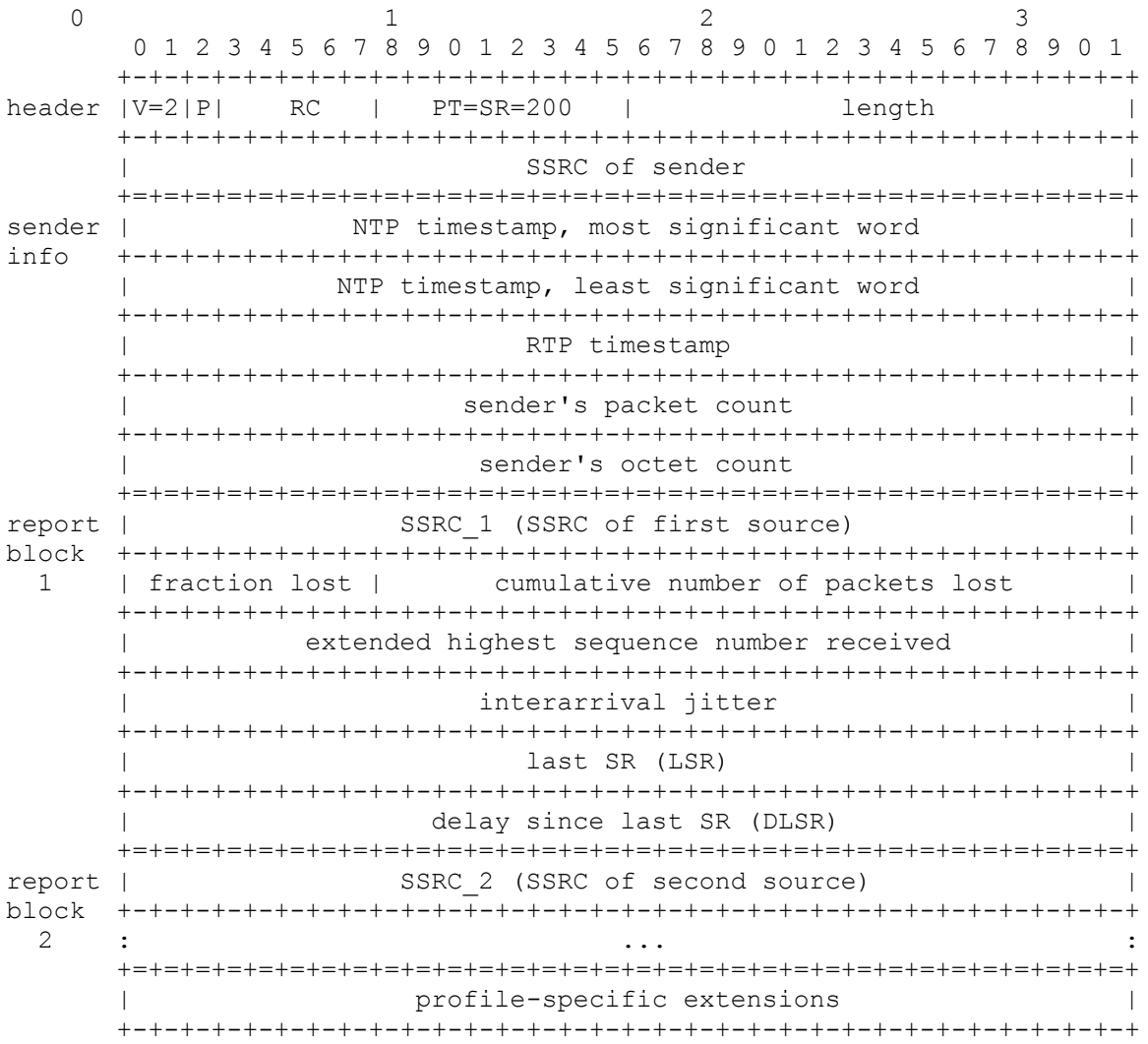

Figura 12: Formato de mensajes SR de RTCP

- Ve (Versión, 2 bits): Actualmente 2

- P (Relleno, 1 bit): Como en RTP

- RC (Receiver Count, 5 bits): Número de bloques contenidos en el paquete SR o RR, o el número de elementos listados en un paquete SDES o BYE.

- Tipo de paquete (8 bits): SR, RR, SDES, BYE y específico de aplicación.

- Longitud del paquete (16 bits): En múltiplos de palabras 32 bits, menos una palabra.

- SSRC (Synchronization Source Identifier, 32 bits): Identifica la fuente de sincronismo origen del paquete.

La segunda sección, la información del remitente, contiene 20 octetos, está presente en cada paquete del informe del remitente y contiene los datos transmisiones del remitente. Los campos tienen el siguiente significado.

- NTP timestamp (2 palabras de 32 bits): Este campo ocupa dos palabras de 32 bits. Indica el tiempo real en el que el paquete ha sido generado. Al recibir las respuestas de los receptores mediante los informes de recepción, se puede calcular el retardo de propagación de la red comparando.

- RTP timestamp (32 bits): Contiene la misma marca temporal, pero en las mismas unidades y con el mismo offset que en los paquetes de datos RTP.

- Contador de paquetes enviados (32 bits): Indica el total de paquetes RTP que el emisor ha enviado hasta la creación del informe.

- Contador de octetos enviados (32 bits): Indica el número total de octetos que el emisor ha enviado. Solamente se cuentan los bits del campo de payload del paquete RTP.

La tercera sección de estos informes es común para ambos. Se trata de un número variable de bloques, cada uno asociado a una sesión RTP activa. Cada uno de estos bloques contiene estadísticas de la recepción de la fuente que lo genera. Para ello se definen los siguientes campos en cada bloque:

- SSRC (32 bits): Identifica a la fuente a la cual se refieren las estadísticas del bloque.

- Fracción perdida (8 bits): Indica la fracción de paquetes RTP que han sido perdidos desde el envío del anterior informe. Este número solo hace referencia a la sesión RTP de la que se está informando, no al número de paquetes perdidos en la sesión multimedia.

- Número acumulativo de paquetes perdidos (24 bits): Este paquete indica el número total de paquetes RTP perdidos desde el establecimiento de la sesión RTP. Al igual que el campo anterior, solamente se hace referencia a la sesión de la que se informa. El hecho de poseer un contador acumulativo permite que se puedan hacer estimaciones de los paquetes perdidos entre diferentes periodos de tiempo, no solo entre dos informes consecutivos o en el total de la sesión.

- Extensión del número de secuencia más alto (32 bits): Este campo de 32 bits se divide en dos partes. En los últimos 16 bits se indican el número de secuencia del último paquete de datos RTP recibido de la fuente. Los primeros 16 bits son una extensión de ese número de secuencia. Esta extensión cumple funciones de control de errores.

- Intervalo de jitter (32 bits): Este campo contiene una estimación del jitter entre llegadas de paquetes de datos RTP. Este cálculo se realiza en función de la marca temporal de envío y el instante de tiempo en que se reciben los paquetes. El cálculo se hace en base a los dos últimos paquetes recibidos. La monitorización de este parámetro es importante, ya que puede indicar la existencia de principios de congestión en la red, antes de que dicha congestión se traduzca en un aumento de la tasa de pérdidas. De esta forma, si se toman las medidas preventivas necesarias ante el aumento del jitter se puede llegar a prevenir la pérdida de paquetes.

- Último informe de envío (LSR) (32 bits): Contiene el timestamp del último paquete RTCP de informe de envío recibido de la sesión RTP.

- Retardo desde el último informe de envío (DLSR) (32 bits): Este campo indica el tiempo que ha pasado desde la recepción del último informe de envío.

Por último, en la figura 12 se puede observar la existencia de una cuarta sección, que es opcional, en la que se permite a las aplicaciones extender las funcionalidades de los informes de envío y recepción, de forma que se puedan incluir más datos que permitan la creación de diferentes estadísticas.

Al procesar todos estos datos se pueden obtener una gran cantidad de parámetros útiles para monitorizar la calidad de servicio. Se pueden calcular tasas de pérdidas, desviaciones del retardo, tasa de envío de las fuentes y un largo etcétera, tomando intervalos de tiempo pequeños o sobre el tiempo total de la sesión multimedia.

II. ***Receiver Report (RR):*** Informa de lo que ha recibido.

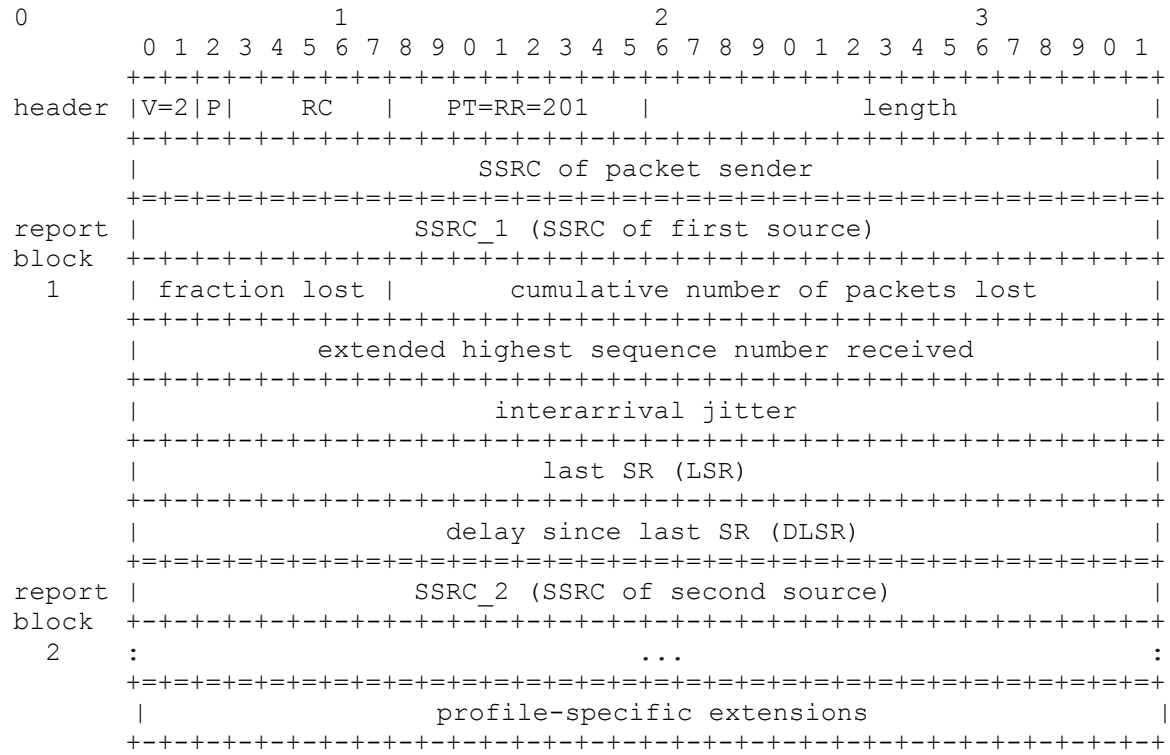

Figura 13: Formato de mensajes RR de RTCP

La única diferencia con SR, a parte del código de tipo de paquete, se encuentra en que el informe de envío tiene 20 bytes extras para ofrecer información sobre el emisor que ha generado el paquete.

III. ***Source Description (SDES):*** Descripción de la fuente.

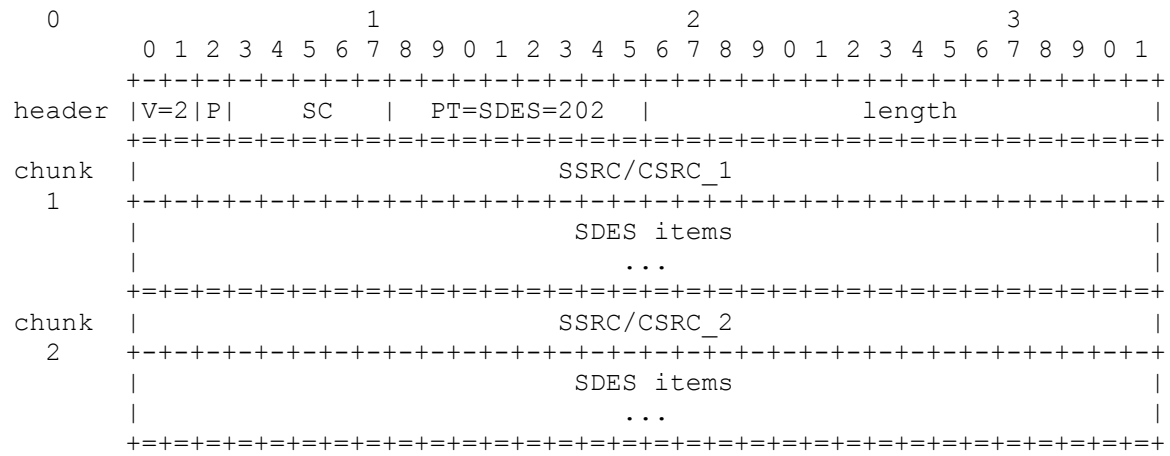

Figura 14: Formato de mensajes SDES de RTCP

Cada bloque consta de dos partes. En primer lugar se indica a que sesión o sesiones RTP hacen referencia las descripciones. La segunda parte es variable. Su contenido es diferente según el tipo de descripción que se haga de la fuente. Consta de los siguientes campos:

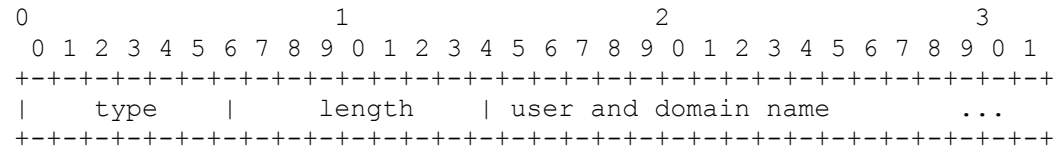

Figura 15: Formato de los descriptores en los mensajes SDES de RTCP

En primer lugar se utilizan 8 bits para identificar el tipo de descripción. La especificación del protocolo RTP define ocho tipos distintos de descripciones:

- CNAME (1): Contiene el identificador canónico de la fuente RTP. Este es el tipo de descripción más importante. Tanto es así que es el único tipo de descripción obligatorio en cualquier implementación de RTP.

- NAME (2): Nombre real del participante, utilizado para identificar a la fuente a través de la interfaz de las aplicaciones.

- EMAIL (3): Dirección de correo electrónico del participante.

- PHONE (4): Número de teléfono del participante.

- LOC (5): Indica la localización geográfica del participante.

- TOOL (6): Un string que indica la versión de la aplicación que genera el flujo RTP.

- NOTE (7): Esta descripción permite a los usuarios informar sobre su estado.

- PRIV (8): Este string se utiliza para que las aplicaciones puedan incluir sus propias descripciones.

A continuación se indica la longitud del tercer campo (no se incluyen estos dos octetos). Este tercer campo contiene la descripción de la fuente. Su contenido está codificado en formato de texto para poder albergar cualquier tipo de descripción, ya que sería imposible mapearlas en códigos.

IV. ***BYE:*** Puede indicar la razón por la que se sale de la sesión.

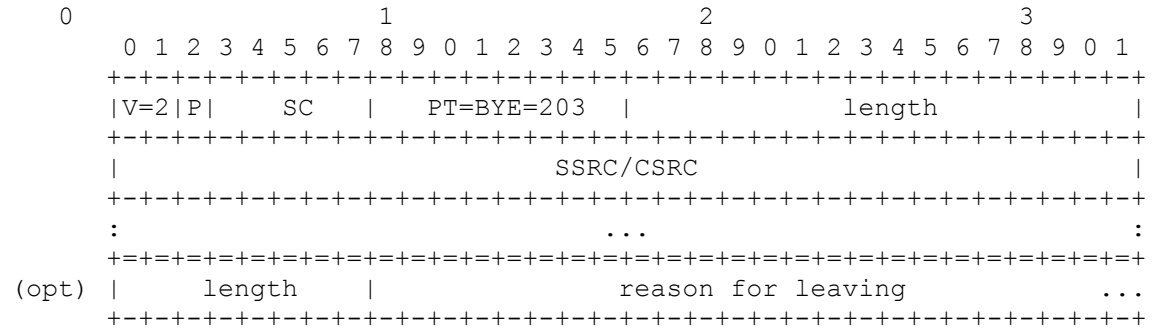

Figura 16: Formato de mensaje BYE de RTCP

El paquete BYE está formado por la cabecera RTCP estándar y el identificador SSRC del participante o participantes que abandonan la sesión. Opcionalmente se puede añadir un campo en el que se explican los motivos del abandono, precedido por un campo que indica la longitud del mensaje.

Es muy común que la mayoría de los usuarios abandonen la sesión de forma simultánea, por ejemplo al terminar una reunión o al finalizar de la visualización de contenidos en directo. Si el número de participantes es muy alto (más de 50) se corre el riesgo de sufrir una inundación de paquetes BYE en la red. Para evitar esto los participantes deben seguir un algoritmo de contención, que les permite ir abandonando la sesión multimedia de forma escalonada. Si el número de participantes es inferior a 50 se puede enviar directamente un paquete BYE, sin tener que seguir ningún algoritmo. Con este mecanismo se asegura que, en el peor de los casos, el porcentaje de ancho de banda de la sesión utilizado por paquetes RTCP no supere el 10% (5% para otros paquetes RTCP y 5% para paquetes BYE).

V. ***APP:*** Específico de aplicación. Experimental, sin necesidad de definir nuevos paquetes.

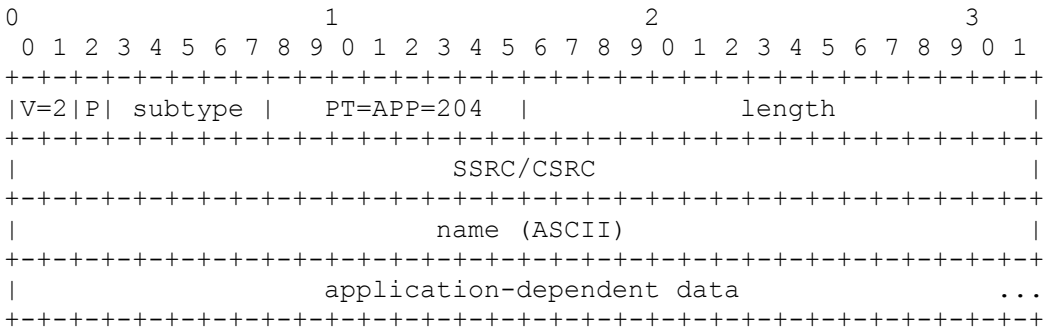

Figura 17: Formato de los mensajes APP de RTCP

El paquete APP contiene, además de la cabecera estándar, un campo con un texto codificado en ASCII, el cual identifica el tipo de paquete APP del que se trata. A continuación se utiliza un campo variable, en el que se insertarán los diferentes campos que formarán el paquete específico de la aplicación.

Capítulo 3
Diseño del Sistema

En este capítulo se describirá con detalle las diferentes funcionalidades que tiene nuestra aplicación de VoD. El diseño se ajusta a lo descrito en una serie de RFCs y drafts que se detallarán en los siguientes subapartados, incluyendo decisiones propias en aquellos puntos en los que existe un cierto vacío en las propuestas del IETF.

Comenzamos explicando los diferentes mensajes SIP que se intercambian entre el cliente, el Servidor de Vídeo y el Servidor de Control para la petición de vídeo. En esta explicación además de los mensajes SIP se describe el contenido de la carga SDP.

En el siguiente subapartado se añade funcionalidad a la petición básica de vídeo permitiendo la solicitud de vídeo con MDC o LC. En ambos casos es posible la petición con garantías de calidad de servicio por parte del cliente, lo cual requiere un envío de mensajes diferentes.

Finalmente se desarrolla el punto de modificación de la sesión, en el que se encuentra la petición o eliminación de un descriptor de vídeo con la sesión ya establecida, o el cambio de dependencia entre los descriptores en el caso de LC.

3.1. Petición de Vídeo

A continuación se muestran todos los pasos que se dan en la aplicación desde que el cliente pide un determinado fichero de vídeo hasta que el S.V. lo transmite. En el diagrama de la figura 18 se refleja el intercambio de mensajes.

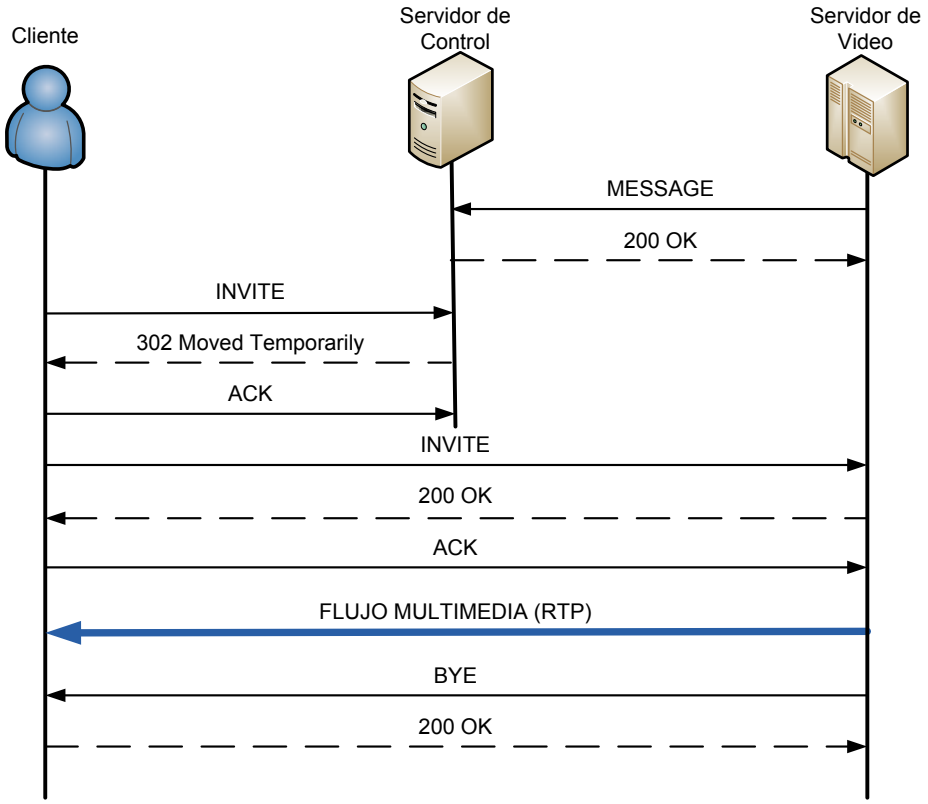

Figura 18: Diagrama de intercambio de mensajes

1. **Comunicación S.V. - S.C.:** El S.V. establece una sesión SIP con el S.C. Mediante la petición MESSAGE, se envía periódicamente la información de control sobre su carga. Estos mensajes hacen posible que el S.V. conozca la distribución de carga del sistema.

2. **Comunicación Cliente - S.C.:** El cliente pide un archivo de vídeo específico al S.C. Esta petición se realiza con el método INVITE de SIP solicitando al servidor el establecimiento de una sesión. El S.C. consulta en la base de datos los diferentes servidores de vídeo que almacenan la película que solicita el cliente.

   Analizando los mensajes de control MESSAGE que le envían los servidores de vídeo, elige el servidor con menor carga que pueda atender la petición del cliente. En la respuesta 302, mediante la cabecera "Contact", se especifica el S.V.

3. **Comunicación Cliente - S.V.:** El cliente solicita de nuevo el establecimiento de una sesión para un determinado archivo de vídeo pero esta vez al S.V. El S.V. acepta la petición INVITE del cliente respondiendo afirmativamente con un ACK, posteriormente comienza la retransmisión del archivo de vídeo. Este intercambio de mensajes es diferente en el caso de que el cliente solicite requisitos de calidad de servicio.

En los siguientes subapartados se profundiza en el contenido de los pasos descritos anteriormente, dando una descripción más detallada.

3.1.1. Comunicación S.V. - S.C.

En el draft [1] se plantea la generación de unos mensajes específicos de RTCP (SERVSTAT) para enviar la información de control del S.V. Sin embargo para este proyecto se busca una solución más sencilla, se pretende evitar propuestas que supongan la creación de nuevos mensajes y que no estén estandarizadas.

Para esta fase del proceso se propone el envío de mensajería instantánea SIP entre los dos tipos de servidores. Se entiende por mensajería instantánea a la transferencia de mensajes entre usuarios casi en tiempo real. Estos mensajes generalmente suelen ser cortos. La mensajería instantánea se utiliza a menudo en un modo conversacional, es decir, la transferencia de mensajes de un lado al otro y viceversa es lo suficientemente rápida como para que los participantes puedan mantener una conversación en forma interactiva.

El método MESSAGE, es una extensión del protocolo SIP que permite la transferencia de mensajes instantáneos. Puesto que la petición MESSAGE es una extensión de SIP, hereda todas las características de enrutamiento y de seguridad de ese protocolo. Las peticiones MESSAGE no inician un diálogo SIP por sí mismas, bajo uso normal cada mensaje instantáneo es independiente. Estas peticiones se pueden enviar dentro del contexto de un diálogo iniciado por otra petición SIP.

En la RFC 3428 [14] se describe este método ampliado de SIP. El funcionamiento de la mensajería instantánea en SIP es muy simple, solo se envían mensajes con el método MESSAGE y se responde con un mensaje cuyo status code es 200 OK.

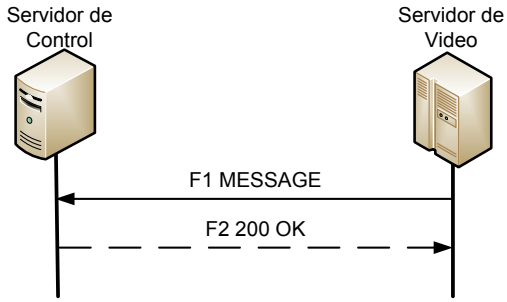

Figura 19: Mensajes SIP de comunicación S.V. y S.C.

El contenido de los mensajes puede ser el siguiente:

[F1 MESSAGE]

```
MESSAGE sip:s.possion@dist.probability.org SIP/2.0
Via SIP/2.0/TCP lecturehall21.academy.ru:5060;branch=z9hG4bK3gtr2
Max-Forwards: 70
To: M. Poisson <sip:s.possion@dist.probability.org>
From: P. L. Chebychev <sip:chebychev@academy.ru>;tag=4542
Call-ID: 9dkei93vjq1ei3
CSeq: 15 MESSAGE
Allow: ACK, INVITE, CANCEL, BYE, NOTIFY, SUBSCRIBE, MESSAGE
Content-Type: text/plain
Content-Length: 9

Hi There!
```

[F2 200 OK]

```
SIP/2.0 200 OK
Via SIP/2.0/TCP
lecturehall21.academy.ru:5060;branch=z9hG4bK3gtr2;received=19.34.3.1
To: M. Poisson <sip:s.possion@dist.probability.org>;tag=2321
From: P. L. Chebychev <sip:chebychev@academy.ru>;tag=4542
Call-ID: 9dkei93vjq1ei3
CSeq: 15 MESSAGE
Content-Length: 0
```

El tamaño de las peticiones MESSAGE no debe exceder los 1300 bytes. Para prevenir replays de viejas peticiones SIP, todas las peticiones MESSAGE y las respuestas deben contener un campo de fecha cubierto por la firma del mensaje. Cualquier mensaje con una fecha más vieja que esta varios minutos en el pasado, o que esta varios minutos en el futuro, debe ser contestado con un mensaje 400 (fecha u hora incorrecta).

Este mensaje de control debe tener los datos necesarios para que el S.C. pueda evaluar correctamente la carga de todos los servidores de vídeo sin llegar a tener una sobre información. Basándonos en el draft [1] fijamos los atributos básicos.

- **Capacidad del servidor (CS):** Medida en MBytes sobre la capacidad de carga total que tiene el servidor de vídeo.
- **Nº de clientes (NC):** El número total de clientes que en ese momento sirve el servidor.
- **Identificador del cliente (IC):** Número entero que identifica a cada uno de los clientes.
- **Identificador del stream (IS):** Identificador único del stream entre todos los clientes.
- **Tamaño del fichero (TF):** Tamaño en KBytes del fichero total que se sirve en el stream IS.
- **Offset del fichero (OF):** Indica en Bytes la cantidad del fichero que ya ha sido servida al cliente.

Teniendo en cuenta que cada S.V. sirve a `n´ clientes y que cada cliente puede tener asociados varios streams de vídeo, los mensajes de descripción de carga deben almacenar toda la siguiente información:

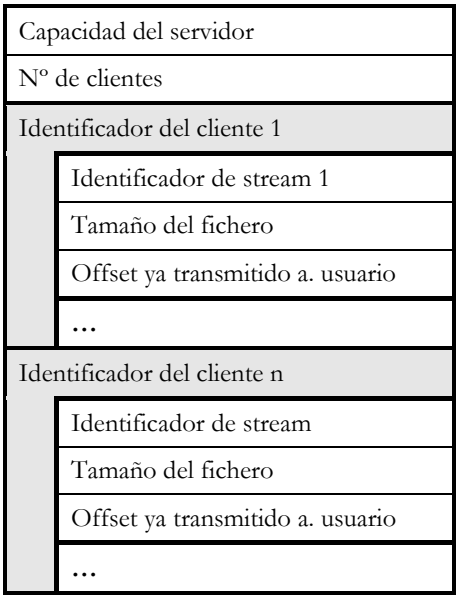

Figura 20: Contenido de mensaje de control de un S.V.

3.1.2. Comunicación Cliente - S.C.

En primer lugar expondremos las diferentes opciones que hay a la hora de especificar el nombre del archivo de vídeo que el cliente se desea descargar para poder evaluarlas conjuntamente. Estas soluciones son diferentes propuestas que pretender solventar una situación que no está reflejada en ningún draft, ni RFC. Por las características de las NGN buscamos peticiones de vídeo mediante SIP/SDP.

- **Propuesta SIP**

A nivel SIP podemos especificar el nombre del fichero mediante su URL. En la sección 25.1 de la RFC de SIP [9] ("Basic Rules"), encontramos el formato de la SIP-URI.

```
SIP-URI = "sip:" [userinfo] hostport uri-parameters [headers ]

userinfo = (user/telephone-subscriber) [":" password] "@"
user = 1*(unreserved/escaped/user-unreserved)
hostport = host [":" port]
uri-parameters = *(";" uri-parameter)
uri-parameter = transport-param/user-param/method-param
                /ttl-param/maddr-param/lr-param/other-par
other-param = pname ["=" pvalue]
```

Dentro de la URL podemos tener dos soluciones con la etiqueta *userinfo* ó con *uri-parameters*. La primera utiliza el identificador del usuario como un índice de películas y la segunda emplea una etiqueta que no tiene una descripción de uso precisa. Un posible ejemplo de estas soluciones sería:

INVITE sip: **rambo**@blockbuster.com:5060 SIP/2.0

INVITE sip: servidor_video_1@blockbuster.com:5060 ; **film=rambo** SIP/2.0

De estas soluciones la más correcta sería emplear *userinfo* para evitar utilizar etiquetas muy poco específicas y con poco uso en la práctica.

- **Propuesta SDP**

Otra solución que aporta un planteamiento diferente sería a nivel SDP. En la parte de descripción multimedia tenemos estas opciones según la RFC 4566 de SDP [10]:

Descripción multimedia

```
m = Nombre de los medios y dirección de transporte
i = * Título multimedia
c = * Información de la conexión (opcional si está incluido en el
       nivel de sesión)
b=* Información del ancho de banda
k=* Clave de cifrado
a=* Atributos de sesión
```

Utilizando el parámetro "i" podemos indicar el nombre del fichero de vídeo. Un ejemplo del mensaje SDP sería:

```
v=0
o=alice 53655765 2353687637 IN IP4 pc33.atlanta.com
s=Session SDP
t=0 0
c=IN IP4 pc33.atlanta.com
m=audio 3456 RTP/AVP 0 1 3 99
a=rtpmap:0 PCMU/8000
i='rambo'
```

- **Propuesta elegida**

Una vez planteadas las soluciones habría que discutir las ventajas y desventajas de cada una. Si nos fijamos en la descripción teórica de SIP y SDP,

SIP: Protocolo para la inicialización, modificación y finalización de sesiones multimedia.

SDP: Protocolo para la DESCRIPCIÓN de flujos multimedia.

el nombre de la película encaja más en la descripción del flujo, por lo que la solución planteada con SDP sería la más correcta. Al igual que se solicitan determinados códecs, se podría indicar el título del fichero de vídeo.

Mediante la etiqueta *userinfo* de SIP se identifica el índice de la película, tratando a los ficheros de vídeo del S.V. como usuarios de este, solución teóricamente muy correcta. Además si pensamos en la arquitectura de nuestra aplicación, la propuesta con SIP al dejar libre de uso el atributo "i" de SDP, tiene las siguientes ventajas:

- ✓ En una arquitectura P2P podemos describir el origen de los diferentes flujos que recibe el usuario (de un determinado servidor de vídeo, de otros usuarios…).
- ✓ En multidescripción podemos utilizar las líneas "i" para identificar los streams que vamos a enviar (stream con "tramas par", "tramas impar"…).

Por estas razones, nos decantamos finalmente por la propuesta con SIP-URI con la etiqueta *userinfo***.**

Tras especificar cómo realizar la petición de un fichero de vídeo en concreto, ahora nos centraremos en el intercambio de mensajes entre el Cliente y el S.C. La solución propuesta en este proyecto se basa en la RFC 3665 [15][14] "SIP Basic Call Flow Examples", en concreto en el punto 3.6 "Session via Redirect and Proxy Servers with SDP in ACK".

El cliente envía un mensaje INVITE al Servidor de Control (S.C.) o Servidor de Redirección. El S.C. envía una respuesta 302 "Moved Temporarily" donde la cabecera Contact contiene la dirección del Servidor de Vídeo (S.V.) que puede atender al cliente. Finalmente el cliente envía mensaje ACK de confirmación al S.C.

Un ejemplo del intercambio de mensajes entre el S.C. y el cliente puede ser el siguiente:

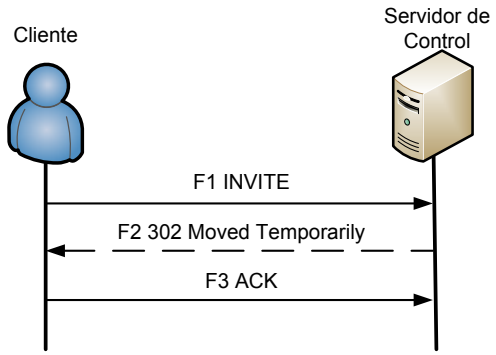

Figura 21: Mensajes SIP de asignación de S.V.

[F1 INVITE]

```
INVITE sip: rambo@server_ctrl.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKbf9f44
Max-Forwards: 70
From: <sip: cliente@atlanta.example.com>;tag=4992881234
To: <sip: rambo@server_ctrl.com>;tag=193402342
Call-ID: 898234234@agenta.atlanta.example.com
CSeq: 1 INVITE
Contact: sip: cliente@atlanta.example.com
Content-Length: 0
```

[F2 302 Moved Temporarily Redirect]

```
SIP/2.0 302 Moved Temporarily
Via: SIP/2.0/UDP agenta.atlanta.example.com;branch=z9hG4bK2293940223
To: <sip: rambo@server_ctrl.com>;tag=193402342
From: <sip:cliente@atlanta.example.com>;tag=4992881234
Call-ID: 898234234@agenta.atlanta.example.com
CSeq: 1 INVITE
Contact: <sip: rambo@serv_video1.com;transport=udp>
Content-Length: 0
```

[F3 ACK]

```
ACK sip: rambo@server_ctrl.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKbf9f44
Max-Forwards: 70
From: <sip: cliente@atlanta.example.com>;tag=4992881234
To: <sip: rambo@server_ctrl.com>;tag=193402342
Call-ID: 898234234@agenta.atlanta.example.com
CSeq: 1 ACK
Content-Length: 0
```

3.1.3. Comunicación Cliente - S.V.

A continuación se muestra la petición por parte del cliente del archivo de vídeo al S.V. El intercambio de mensajes es diferente en el caso de que el cliente solicite requisitos de calidad de servicio con reserva de recursos.

- **Petición básica**

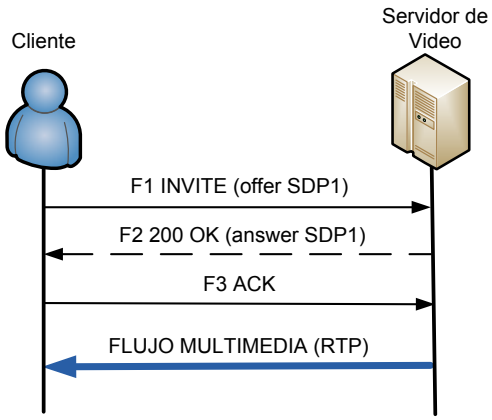

Figura 22: Mensajes SIP de petición de archivo de vídeo al cliente

En el escenario del ejemplo, el cliente solicita una serie de códecs que él puede reproducir, el S.V. debe aceptar aquellos códecs con mayor calidad que sea capaz de servir. En el siguiente apartado se profundizará más en este punto del proceso de la aplicación teniendo en cuenta el proceso de multidescripción del vídeo.

[F1 INVITE]

```
INVITE sip: rambo@serv_video1.com SIP/2.0
Via: SIP/2.0/UDP serv_video1.com;branch=z9hG4bK2293940223
To: <sip: rambo@serv_video1.com>
From: <sip:cliente@atlanta.example.com>;tag=193402342
Call-ID: 898234234@serv_video1.com;
CSeq: 1 INVITE
Contact: sip: cliente@atlanta.example.com
Content-Type: application/sdp

v=0
o=cliente 2890844526 2890844526 IN IP4 host.atlanta.example.com
s=
c=IN IP4 host.atlanta.example.com
t=0 0
m=video 51372 RTP/AVP 31 32
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
```

El cliente solicita dos tipos diferentes de códecs para el vídeo. La petición de una película en concreto se hace con la etiqueta *userinfo*, como se explico en el punto anterior.

[F2 200 OK]

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP serv_video1.com;branch=z9hG4bK2293940223
To: <sip: rambo@serv_video1.com >
From: <sip:cliente@atlanta.example.com>;tag=193402342
Call-ID: 898234234@serv_video1.com;
CSeq: 1 INVITE
Contact: sip: rambo@serv_video1.com
Content-Type: application/sdp

v=0
o=servidor_video_1 2808844564 2808844564 IN IP4 host.bilo.example.com
s=
c=IN IP4 host.biloxi.example.com
t=0 0
m=video 49170 RTP/AVP 32
a=rtpmap:32 MPV/90000
```

En este ejemplo es necesario destacar que en el mensaje INVITE se envía la petición SDP del cliente al S.V. El servidor responde a esta oferta en el mensaje OK con una nueva carga SDP.

Entre las diferentes cargas hay varias modificaciones:

- El S.V. solo deja el códec que va a servir al cliente. Comprueba si posee alguno de los códecs solicitados por el cliente, empezando por el primero de la petición SDP al ser el orden de preferencia deseado por cliente, hasta que lo encuentre y elimina el resto.
- Se modifica la línea “o=”. La línea que identifica el autor y la sesión es diferente si el mensaje procede del servidor o del cliente. El campo “name” identifica al autor de la carga SDP, “Session ID” y “Session Version” se inicializan independientemente

cuando se establece la sesión al enviar el mensaje INVITE en el cliente y cuando se acepta en el servidor, por eso tienen valores diferentes. Mientras "Session ID" no cambia durante toda la sesión, "Session Version" se incrementa cada vez que se modifican las condiciones de la sesión.

- También se modifica la etiqueta "m=" de descripción del flujo multimedia y la etiqueta "c=" de información de la conexión. En ellas cada actor de la sesión identifica la dirección IP y el puerto de su lado del flujo RTP.

- **Petición con calidad de servicio**

La RFC 3312 [16] especifica el establecimiento de una sesión con calidad de servicio. Esta condición requiere que los participantes reserven recursos de la red antes de continuar la sesión. Cómo se observa en la figura 23, el S.V. no responde con un OK al INVITE del cliente, sino que se envía una respuesta provisional para indicar que se está tratando de establecer la sesión. En este punto comienza el intercambio de mensajes de SIP en los que se van actualizando la descripción de la sesión que permite finalmente la reserva de recursos por parte del cliente y el S.V.

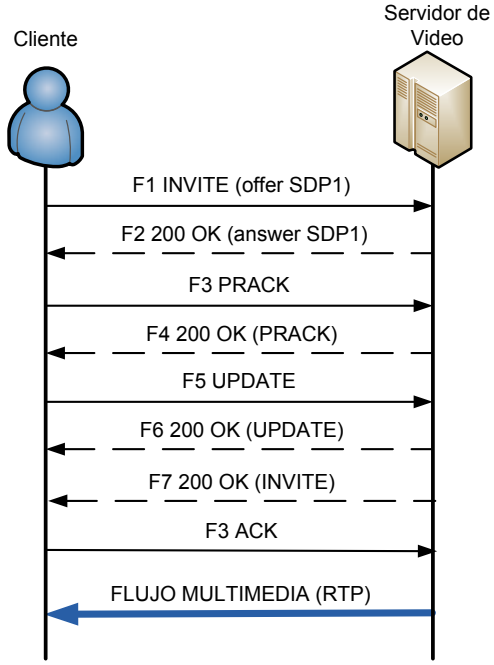

Figura 23: Mensajes SIP de petición de archivo de vídeo al cliente con QoS

Para este tipo de petición de vídeo aparecen un nuevo tipo atributos SDP, los atributos de precondiciones. Utilizaremos los atributos de estado actual (*current-status*) y de estado deseado (*desired-status*). En la RFC 3312 [16] encontramos la siguiente descripción de estos atributos:

```
current-status     =  "a=curr:" precondition-type
                      SP status-type SP direction-tag
desired-status     =  "a=des:" precondition-type
                      SP strength-tag SP status-type
                      SP direction-tag
precondition-type  =  "qos" | token
strength-tag       =  ("mandatory" | "optional" | "none"
                   =  | "failure" | "unknown")
status-type        =  ("e2e" | "local" | "remote")
direction-tag      =  ("none" | "send" | "recv" | "sendrecv")
```

A continuación se muestra la actualización de estos atributos SDP en el intercambio de mensajes de la figura 23.

[F1 INVITE (offer SDP1)]

```
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local recv
a=des:qos none remote sendrecv
```

Las dos primeras líneas indican que actualmente (curr) no hay calidad de servicio garantizada (none) tanto en local (cliente) como en remoto (S.V.). Las dos siguientes manifiestan que se desea (des) calidad de servicio obligatoria para recibir en local (cliente) y todavía sin especificar en remoto (qos none remote sendrecv).

[F2 183 (answer SDP1)]

```
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local send
a=des:qos mandatory remote recv
```

En este mensaje ahora local es el S.V. y remoto el cliente. En la tercera línea se manifiesta que se desea calidad de servicio para enviar.

En los siguientes mensajes [F3 PRACK] y [F4 200 OK (PRACK)] no hay cambio en estos atributos ya que es el intante en el que se reservan recursos en ambos lados.

[F5 UPDATE]

```
a=curr:qos local recv
a=curr:qos remote none
a=des:qos mandatory local recv
a=des:qos mandatory remote send
```

Con la actualización de la primera línea se indica que ya hay calidad de servicio para recibir en el cliente.

[F6 200 OK (UPDATE)]

```
a=curr:qos local send
a=curr:qos remote recv
a=des:qos mandatory local send
a=des:qos mandatory remote recv
```

La calidad de servicio está garantizada con este último mensaje al confirmarse la reserva de recursos para enviar por parte del S.V.

Este intercambio de mensajes también se especifica en los artículos [17] y [18], en lo que se habla de la reserva de calidad de servicio en las NGN por parte del RGW.

3.2. Vídeos con Multidescripción

En este apartado se describirán las propuestas de dos drafts del IETF para la señalización de flujos de vídeo con MDC. Al final del apartado se expondrán las ventajas e inconvenientes de cada una y cual se ha implementado finalmente en este proyecto.

- **Propuesta 1 - [20]**

En el artículo [19] basado en el draft "Standard-compatible Multiple-Description Coding (MDC) and Layered Coding (LC) of Audio/Video Streams" [20] se propone una implementación de un esquema conjunto MDC/LC. La creación de "descriptions/layers" es independiente de los códecs de los flujos multimedia y se lleva a cabo en una fase de preprocesado. Los "descriptions/layers" son enviados en diferentes flujos RTP independientes, en los que se utiliza SDP para enviar información específica de cada uno de ellos. Hay dos grupos de parámetros:

- Parámetros de preprocesado: Describen la creación de cada flujo "descriptions/layers" para que los receptores inteligentes puedan combinarlos independientemente del modelo de pérdidas.
- Parámetros de postprocesado: Describen la combinación de cada flujo para aquellos servidores básicos que no pueden calcular como combinarlos.

Uno de estos parámetros se envía mediante SDP mediante el atributo de descripción multimedia "a". En el draft [20] se propone utilizar la etiqueta "X-attribute:value" que es definida mediante la metasintaxis BNF e ignorada por el receptor, permitiendo añadir cualquier información adicional. Las diferentes sentencias son:

- **`"a=X-mdclc-tag:" payload-type mdclc-tag`**

  Todos los descriptores empleados en MDC/LC son identificados mediante esta etiqueta. Por ejemplo, "a=X-mdclc-tag: 97 D1" indica que la carga 97 es etiqueta como D1 (Primer descriptor). Hay etiquetas (mdclc-tag) que tienen un significado espacial: S: Flujo multimedia origen, E: Flujo final reconstruido, D: Descriptor independiente…

- **`"a=X-mdclc-group:" group-tag +(space mdclc-tag)`**

  Los grupos de "descriptions/layers" en MDC/LC son identificados con una nueva etiqueta. Se emplea en los parámetros de postprocesado. Por ejemplo, "a=X-mdclc-group: lp0 D1 D2" indica que el grupo lp0 está formado por los descriptores D1 y D2.

- **`"a=X-mdclc-pre:" *(tag"="expression";")`**

  Sentencia empleada para representar los parámetros de preprocesado. Además de poder utilizar funciones lógicas para la creación de los diferentes "descriptions/layers" partiendo de los flujos origen, se emplean funciones de sobremuestreo, submuestreo y filtros FIR. Por ejemplo, "a=X-mdclc-pre: D1=dn(2, 0, Y, fir([0, 0.75;0.25], S));" indica que el descriptor D1 se obtiene en la fase de preprocesado, filtrando la secuencia origen S mediante un filtro FIR y después realizar un submuestreo a la salida del filtro con un factor 2:1.

- **`"a=X-mdclc-post:" group-tag *(tag"="expression";")`**

  Representa los parámetros de postprocesado. Muy parecida a la sentencia anterior, la principal diferencia es que las expresiones van asociadas a un grupo. El ejemplo de esta sentencia relacionada con la de group es, "a=X-mdclc-post: lp0 E = up(2,0,Y,D1)+up(2,1,Y,D2);", que indica que la secuencia reconstruida E se obtiene

por el sobremuestreo de los descriptores del grupo lp0 (D1 y D2) y sumando el resultado.

Para explicar la propuesta del draft [20], partimos del siguiente ejemplo para un esquema MDC con dos descriptores balanceados.

```
m=video 49170 RTP/AVP 97 98
a=rtpmap:97 H264/90000
a=fmtp:97 ...
a=X-mdclc-tag: 97 D1
a=X-mdclc-pre: D1= dn(2,0,Y, S);
a=rtpmap:98 H264/90000
a=fmtp:98 ...
a=X-mdclc-tag: 98 D2
a=X-mdclc-pre: D2= dn(2,1,Y, S);
```

La cantidad de datos codificados es 2*1/2. En un receptor básico se debe especificar el comportamiento para cada patrón de pérdidas (lpN). En nuestro caso, con dos descriptores D1 y D2, tenemos cuatro tipos de patrones de pérdidas:

- cuando se reciben los descriptores que se mezclan con normalidad,
- cuando se recibe solo uno de ellos y las líneas perdidas se obtienen mediante la media de las otras líneas,
- y cuando no se recibe ningún descriptor y la imagen recibida es de color gris (128). El código SDP explicado sería el siguiente:

```
a=X-mdclc-group: lp0 D1 D2
a=X-mdclc-post:  lp0 E= up(2,0,Y,D1) + up(2,1,Y,D2);
a=X-mdclc-group: lp1 D1
a=X-mdclc-post:  lp1 E= fir([0.5;1;0.5], up(2,0,Y,D1));
a=X-mdclc-group: lp2 D2
a=X-mdclc-post:  lp2 E= fir([0.5;1;0.5], up(2,1,Y,D2));
a=X-mdclc-group: lp3
a=X-mdclc-post:  lp3 E= 128;
```

- **Propuesta 2 - [21]**

En el segundo draft [21] se propone una solución mucho más sencilla y no por ello menos completa. Se basa en los atributos SDP descritos en la RFC 3388 [22] *"group"* y *"mid"*.

- **"group":** Agrupa diferentes streams multimedia.

```
group-attribute   = "a=group:" semantics
semantics         = "LS" (Lip Synchronization)  |
                    "FID" (Flow Identification)
```

- **"mid":** Identifica un stream multimedia.

```
mid-attribute         = "a=mid:" identification-tag
identification-tag    = token
```

El draft [21] propone crear en el atributo *"group"* una nueva etiqueta *semantics* llamada "DDP" (Decoding Dependency) y un nuevo atributo "depend".

- **"depend":** Describe la dependencia en la decodificación.

```
depend-attribute      = "a" "=" "depend" ":"
                        dependency-type-tag
dependency-type-tag   = "lay" / "mdc"
```

A continuación se exponen los ejemplos de codificación escalable y de multidescripción que se encuentran en el draft [21][22][21].

```
v=0
o=mdcsrv 289083124 289083124 IN IP4 host.example.com
s=MULTI DESCRIPTION VIDEO SIGNALING Seminar
t=0 0
c=IN IP4 192.0.2.1/127
a=group:DDP 1 2 3

m=video 40000 RTP/AVP 94
a=mid:1
a=depend:mdc

m=video 40002 RTP/AVP 95
a=mid:2
a=depend:mdc

m=video 40004 RTP/AVP 96
c=IN IP4 192.0.2.2/127
a=mid:3
a=depend:mdc
```

```
v=0
o=svcsrv 289083124 289083124 IN IP4 host.example.com
s=LAYERED VIDEO SIGNALING Seminar
t=0 0
c=IN IP4 192.0.2.1/127
a=group:DDP 1 2 3 4

m=video 40000 RTP/AVP 94
b=AS:96
a=framerate:15
a=rtpmap:94 h264/90000
a=mid:1

m=video 40002 RTP/AVP 95
b=AS:64
a=framerate:15
a=rtpmap:95 svc1/90000
a=mid:2
a=depend:lay 1

m=video 40004 RTP/AVP 96
b=AS:128
a=framerate:30
a=rtpmap:96 svc1/90000
a=mid:3
a=depend:lay 1

m=video 40004 RTP/SAVP 100
c=IN IP4 192.0.2.2/127
b=AS:512
k=uri:conditional-access-server.example.com
a=framerate:30
a=rtpmap:100 svc1/90000
a=mid:4
a=depend:lay 1 3
```

- **Propuesta elegida**

La propuesta 1 del draft [20] es las más detallada y compleja, especificando parámetros de preprocesado y postprocesado, mientras que la propuesta 2 [21] es la más simple, identificando únicamente la dependencia entre los flujos multimedia. En la tabla 2 se comparan ambas propuestas.

Propuesta 1 - [20]
✓ Descripción detallada del pre-procesado y del post-procesado de los diferentes flujos. ✓ Permite diferentes formas de codificar y decodificar los flujos.
× Compleja. × Necesita 4 nuevos atributos SDP no definidos en ninguna RFC.
Propuesta 2 - [21]
✓ Sencilla. ✓ Solo necesita 1 nuevo atributo SDP. ✓ Permite emplear codificación escalable o con multidescripción.
× Solo permite una forma de codificar y decodificar los flujos. × Esta forma de codificar debe ser conocía por cliente y S.V.

Tabla 2: Propuesta 1 vs propuesta 2 de multidescripción

Finalmente en este proyecto se tomó la decisión de implementar la propuesta 2 [21] del atributo SDP "depend" por las ventajas expuestas en la tabla 2. Se consideró que era mejor implementar la solución más sencilla y menos compleja. Tener una solución a nivel SDP con demasiada descripción aumentaría el tiempo de procesado de las tramas.

3.3. Modificación de la Sesión

Falta por explicar cómo y qué parámetros se pueden modificar en una sesión. El cliente podrá modificar una sesión, previamente establecida, en la que haya varios descriptores, es decir solo podrá modificar los parámetros de una sesión con codificación escalable (LC) o con multidescripción (MDC).

Existen iniciativas en relación a las redes de próxima generación, entre las que se encuentra el proyecto europeo MUSE (Multi Service Access Everywhere). En MUSE, el ámbito de aplicación de la calidad de servicio cubre el camino completo de comunicación extremo a extremo entre usuarios finales. En concreto, en el entorno residencial del usuario, la calidad de servicio se proporciona mediante una figura clave, la pasarela residencial.

La pasarela residencial (RGW) es parte de la red del usuario, encontrándose en el límite entre el entorno residencial del usuario y la red de acceso. Desde el RGW parten las peticiones de modificación de la sesión, permitiendo configurar automáticamente la calidad de servicio. El RGW analiza el estado actual de la sesión para saber que parámetros puede modificar para mejorar o hacer más eficiente la transmisión del vídeo.

La renegociación de los parámetros de la sesión se realiza a nivel SDP como se explicará a continuación, pero previamente habrá que fijar los mensajes SIP que se intercambian entre el cliente y el S.V., aquellos que contienen la carga SDP.

El método UPDATE de SIP (RFC 3311) [23] permite actualizar los parámetros de una sesión multimedia. En la figura 24 se indica que el mensaje UPDATE que parte del cliente contiene la carga SDP2, que es similar a la SDP1 con la que se estableció la sesión, excepto por los parámetros modificados. La respuesta OK del S.V. confirma al cliente esta modificación en la sesión y marca el instante de cambio en las propiedades del flujo de vídeo. El S.V. conoce los cambios de la sesión al comparar SDP1 con SDP2.

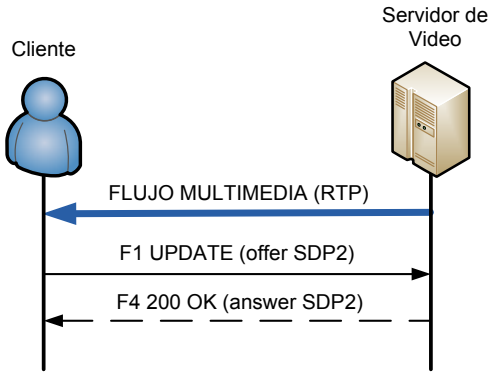

Figura 24: Mensajes SIP de modificación de la sesión básica

Sin embargo, por las características de este proyecto, había que tener en cuenta el caso en el que se solicitase añadir un nuevo descriptor cuando en el establecimiento de la sesión se negociase la calidad de servicio de esta. Es necesario hacer una nueva reserva de recursos que garantice la calidad de servicio del nuevo descriptor multimedia, sin que esta afecte a los anteriores descriptores ya establecidos. Por esta razón tendremos un intercambio de mensajes diferentes (figura 25), volviéndonos a basar en la RFC 3312 [16].

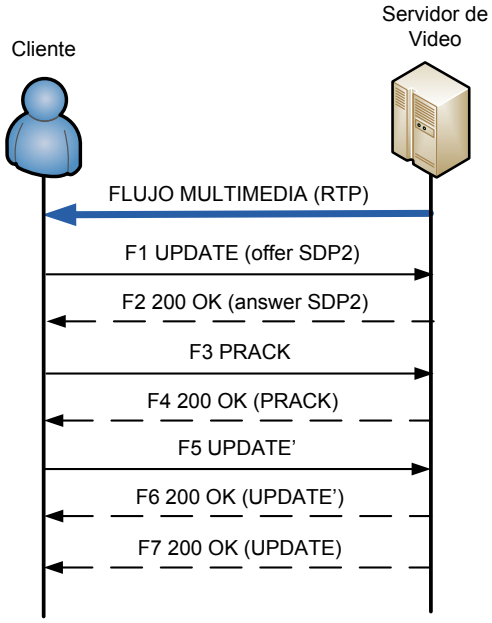

Figura 25: Mensajes SIP de modificación de la sesión con QoS

Respecto al contenido de la carga SDP, lo dividiremos en dos puntos en función de los parámetros que se pueden modificar:

1. **Añadir o eliminar un descriptor de vídeo:** Aumentar el número de descriptores que se envían desde el S.V. hacia el cliente mejora la calidad del stream de vídeo final, pero también puede ser necesario disminuir esa calidad eliminando un descriptor en el caso que empeoren las condiciones de la sesión.
2. **Modificar la dependencia de los descriptores de vídeo:** Esto solo sería posible en el caso de una sesión con codificación escalable (LC) donde existe una dependencia entre los diferentes descriptores.

3.3.1. Añadir o eliminar un descriptor de vídeo

Este punto se basa en la RFC 4317 [11], en la que aparecen ejemplos de flujos de audio o vídeo que se añaden o eliminan de la sesión. Partiendo de estos ejemplos se diseñaron los mensajes SDP para añadir o eliminar descriptores. Estos mensajes serán diferentes si se añade un descriptor de forma básica o si es con calidad de servicio.

- **Añadir un descriptor de vídeo básico**

Este es el caso más sencillo. Los cambios respecto a la carga SDP1 son:

- Un nuevo atributo multimedia.
- Incremento del "Session Version".
- El atributo "group" añade un descriptor.

Como en apartados anteriores, se aporta un ejemplo de la carga SDP para los dos casos posibles, con codificación escalable (LC) o con multidescripción (MDC).

Ejemplo1 con codificación escalable (LC)
[Offer SDP1]

```
v=0
o=cliente 2890844526 2890844526 IN IP4 cliente.biloxi.example.com
s=
c=IN IP4 cliente.biloxi.example.com
t=0 0
a=group:DDP 1

m=video 40000 RTP/AVP 94
b=AS:96
a=framerate:15
a=rtpmap:94 h264/90000
a=mid:1
```

[Answer SDP1]

```
v=0
o=srvVideo 2808844564 2808844564 IN IP4 srvVideo.biloxi.example.com
s=
c=IN IP4 srVideo.biloxi.example.com
t=0 0
a=group:DDP 1

m=video 40000 RTP/AVP 94
b=AS:96
a=framerate:15
a=rtpmap:94 h264/90000
a=mid:1
```

[Offer SDP2]

```
v=0
o=cliente 2890844526 2890844527 IN IP4 cliente.biloxi.example.com
s=
c=IN IP4 cliente.biloxi.example.com
t=0 0
a=group:DDP 1 2

m=video 40000 RTP/AVP 94
b=AS:96
a=framerate:15
a=rtpmap:94 h264/90000
a=mid:1

m=video 40002 RTP/AVP 95
b=AS:64
a=framerate:15
a=rtpmap:95 svc1/90000
a=mid:2
a=depend:lay 1
```

[Answer SDP2]

```
v=0
o=srVideo 2808844564 2808844565 IN IP4 srVideo.biloxi.example.com
s=
c=IN IP4 srVideo.biloxi.example.com
t=0 0
a=group:DDP 1 2

m=video 40000 RTP/AVP 94
b=AS:96
a=framerate:15
a=rtpmap:94 h264/90000
a=mid:1

m=video 40002 RTP/AVP 95
b=AS:64
a=framerate:15
a=rtpmap:95 svc1/90000
a=mid:2
a=depend:lay 1
```

Ejemplo2 con multidescripción (MDC)

[Offer SDP1]

```
v=0
o=cliente 2890844526 2890844526 IN IP4 cliente.biloxi.example.com
s=
c=IN IP4 cliente.biloxi.example.com
t=0 0
a=group:DDP 1

m=video 40000 RTP/AVP 94
a=mid:1
a=depend:mdc
```

[Answer SDP1]

```
v=0
o=srvVideo 2808844564 2808844564 IN IP4 srvVideo.biloxi.example.com
s=
c=IN IP4 srVideo.biloxi.example.com
t=0 0
a=group:DDP 1

m=video 40000 RTP/AVP 94
a=mid:1
a=depend:mdc
```

[Offer SDP2]

```
v=0
o=cliente 2890844526 2890844527 IN IP4 cliente.biloxi.example.com
s=
c=IN IP4 cliente.biloxi.example.com
t=0 0
a=group:DDP 1 2

m=video 40000 RTP/AVP 94
a=mid:1
a=depend:mdc

m=video 40002 RTP/AVP 95
a=mid:2
a=depend:mdc
```

[Answer SDP2]

```
v=0
o=srVideo 2808844564 2808844565 IN IP4 srVideo.biloxi.example.com
s=
c=IN IP4 srVideo.biloxi.example.com
t=0 0
a=group:DDP 1 2

m=video 40000 RTP/AVP 94
a=mid:1
a=depend:mdc

m=video 40002 RTP/AVP 95
a=mid:2
a=depend:mdc
```

- **Añadir un descriptor de vídeo con calidad de servicio**

Como en el caso básico, se modifican los parámetros “Session Version” y “group”. El contenido del mensaje SDP es igual al del subapartado 3.3.3 como se puede ver en el siguiente ejemplo para codificación con multidescripción (MDC). Para codificación escalable (LC) sería similar.

Lo más destacable de este ejemplo es que solo se modifican los atributos “curr” y “des” del descriptor multimedia que se añade. El anterior descriptor mantiene su calidad de servicio.

[F1 UPDATE (offer SDP2)]

```
v=0
o=user 3420221709 3420221710 IN IP4 cliente.biloxi.example.com
s=-
c=IN IP4 cliente.biloxi.example.com
t=0 0
a=group:DDP 1 2

m=video 6060 RTP/AVP 94
a=curr:qos local recv
a=curr:qos remote send
a=des:qos mandatory local recv
a=des:qos mandatory remote send
a=mid:1
a=depend:mdc
a=recvonly

m=video 6062 RTP/AVP 95
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local recv
a=des:qos none remote sendrecv
a=mid:2
a=depend:mdc
a=recvonly
```

[F2 183 (answer SDP2)]

```
v=0
o=videoServer 3420221895 3420221896 IN IP4 srVideo.biloxi.example.com
s=-
c=IN IP4 srVideo.biloxi.example.com
t=0 0
a=group:DDP 1 2

m=video 6060 RTP/AVP 94
a=curr:qos local send
a=curr:qos remote recv
a=des:qos mandatory local send
a=des:qos mandatory remote recv
a=conf:qos remote recv
a=mid:1
a=depend:mdc
a=recvonly

m=video 6062 RTP/AVP 95
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local send
a=des:qos mandatory remote recv
a=conf:qos remote recv
a=mid:2
a=depend:mdc
a=recvonly
```

[F5 UPDATE']

```
v=0
o=user 3420221709 3420221710 IN IP4 cliente.biloxi.example.com
s=-
c=IN IP4 cliente.biloxi.example.com
t=0 0
a=group:DDP 1 2

m=video 6060 RTP/AVP 94
a=curr:qos local recv
a=curr:qos remote send
a=des:qos mandatory local recv
a=des:qos mandatory remote send
a=mid:1
a=depend:mdc
a=recvonly

m=video 6062 RTP/AVP 95
a=curr:qos local recv
a=curr:qos remote none
a=des:qos mandatory local recv
a=des:qos mandatory remote send
a=mid:2
a=depend:mdc
a=recvonly
```

[F6 200 OK(UPDATE')]

```
v=0
o=videoServer 3420221895 3420221896 IN srVideo.biloxi.example.com
s=-
c=IN IP4 srVideo.biloxi.example.com
t=0 0
a=group:DDP 1 2

m=video 6060 RTP/AVP 94
a=curr:qos local send
a=curr:qos remote recv
a=des:qos mandatory local send
a=des:qos mandatory remote recv
a=mid:1
a=depend:mdc
a=recvonly

m=video 6062 RTP/AVP 95
a=curr:qos local send
a=curr:qos remote recv
a=des:qos mandatory local send
a=des:qos mandatory remote recv
a=mid:2
a=depend:mdc
a=recvonly
```

- **Eliminar un descriptor de vídeo**

La RFC 4317 [11] indica que para eliminar un descriptor multimedia, hay fijar su puerto a cero. Después de la respuesta OK al UPDATE, se elimina este descriptor de la carga SDP, por lo que en posteriores mensajes no aparecerá. Los cambios respecto a la carga SDP1 son:

- El puerto del descriptor multimedia a cero.
- Incremento del "Session Version".
- El atributo "group" elimina un descriptor.

Este ejemplo, al igual que el anterior, es con codificación con multidescripción, pero esta vez sin reserva de recursos, porque este caso no influye en la eliminación de un descriptor.

[Offer SDP2]

```
v=0
o=user 3420221709 3420221710 IN IP4 cliente.biloxi.example.com
s=-
c=IN IP4 cliente.biloxi.example.com
t=0 0
a=group:DDP 1

m=video 6060 RTP/AVP 94
a=mid:1
a=depend:mdc
a=recvonly

m=video 0 RTP/AVP 95
a=mid:2
a=depend:mdc
a=recvonly
```

[Answer SDP2]

```
v=0
o=videoServer 3420223517 3420223518 IN IP4 srVideo.biloxi.example.com
s=-
c=IN IP4 192.168.1.3
t=0 0
a=group:DDP 1

m=video 6060 RTP/AVP 94
a=mid:1
a=depend:mdc
a=recvonly

m=video 0 RTP/AVP 95
a=mid:2
a=depend:mdc
a=recvonly
```

3.3.2. Modificar la dependencia de los descriptores de video

Como en el punto anterior, partimos de la RFC 4317 [11] y particularmente de sus ejemplos en los que se modifican los códecs de un descriptor multimedia para fijar el contenido de los mensajes SDP en el caso de cambiar la dependencia. Simplemente se modifica el atributo de dependencia "depend". El S.V. al compara el mensaje SDP2 con SDP1, identifica la modificación.

El siguiente ejemplo parte de tres descriptores. El tercer descriptor (lay3) inicialmente depende del segundo (lay2). La sesión se modifica para que pase a depender del primero (lay1). De nuevo es necesario incrementar el "Session Version".

[Offer & Answer SDP1]

```
m=video 6064 RTP/AVP 96
a=rtpmap:96 svc1/90000
a=mid:3
a=depend:lay 2
a=recvonly
```

[Offer & Answer SDP2]

```
m=video 6064 RTP/AVP 96
a=rtpmap:96 svc1/90000
a=mid:3
a=depend:lay 1
a=recvonly
```

Capítulo 4
Implementación del Sistema

La implementación del sistema en Java se basa en el capítulo anterior que trata sobre su diseño. Partiendo de la aplicación del proyecto "Plataforma Genérica de Provisión de Servicios en Redes con Plano de Control IMS" [24], se ha creado una aplicación nueva que tiene una arquitectura diferente respecto a la del proyecto de partida al incluir el S.C. y que permite la petición de vídeo con multidescripción, así como la modificación de la sesión.

En primer lugar se explicará en qué consiste la arquitectura completa de la aplicación. En este punto se definirán las librerías Java empleadas en este proyecto y cuál es su modo de funcionamiento. Posteriormente se ofrecerá una descripción de cada interfaz de los componentes de la arquitectura y un análisis detallado de su funcionamiento.

La plataforma empleada para desarrollar esta aplicación ha sido Netbeans 5.5 [25] por las facilidades que ofrece en el diseño de interfaces gráficos y por su jerarquía modular. En el CD que se adjunta en esta memoria se encuentran los tres proyectos de Netbeans pertenecientes al S.V., al S.C. y al cliente. También se incluyen los APIs generados mediante Javadoc de cada uno ellos, que contienen el conjunto de funciones de cada clase.

4.1. Arquitectura Completa

Para gestionar la señalización SIP en Java, existe una interfaz denominada JAIN SIP. Esconde toda la complejidad de la sintaxis de los mensajes SIP y del transporte de los mismos, a través de una interfaz con una pila de protocolo SIP, dejando al usuario gestionar los mecanismos a nivel de transacción.

El método de funcionamiento se basa en eventos/mensajes y la implementación de SIP Listener y SIP Provider (figura 26). SIP Provider transforma mensajes SIP que recibe de la red en eventos para pasárselos a la aplicación, mientras que SIP Listener escucha estos eventos y los encapsula dentro de mensajes para que se gestione la comunicación pertinente.

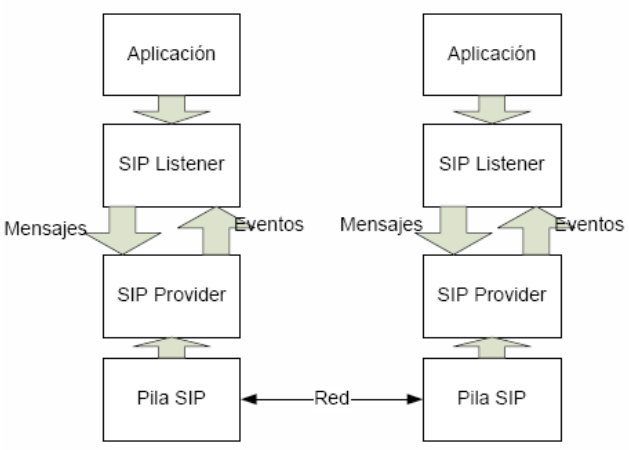

Figura 26: Modo de funcionamiento de JAIN SIP

La colección de librerías empleadas en este proyecto que permiten implementar tanto la descripción de sesiones mediante SDP como la gestión de la señalización de SIP es la siguiente:

- **JainSipApi1.1.jar - JAIN SIP 1.1 [26]:** Contiene la implementación completa de la especificación correspondiente al protocolo SIP. Se trata de un API estándar de Java para el acceso a bajo nivel a una pila de protocolo SIP.
- **nist-sip-1.2.jar - NIST SIP 1.2:** Implementación de referencia del protocolo SIP que utiliza la pila del NIST.
- **nist-sdp-1.0.jar – NIST SDP 1.0:** Implementación de referencia del protocolo SDP del NIST.

En la figura 27 se representa el diagrama del sistema completo. Muestra las clases de las que está compuesto cada componente (S.V., S.C. y cliente) y la relación entre dichas clases. En los siguientes apartados se desarrollará cada componente por separado, explicando su interfaz gráfica y su funcionalidad.

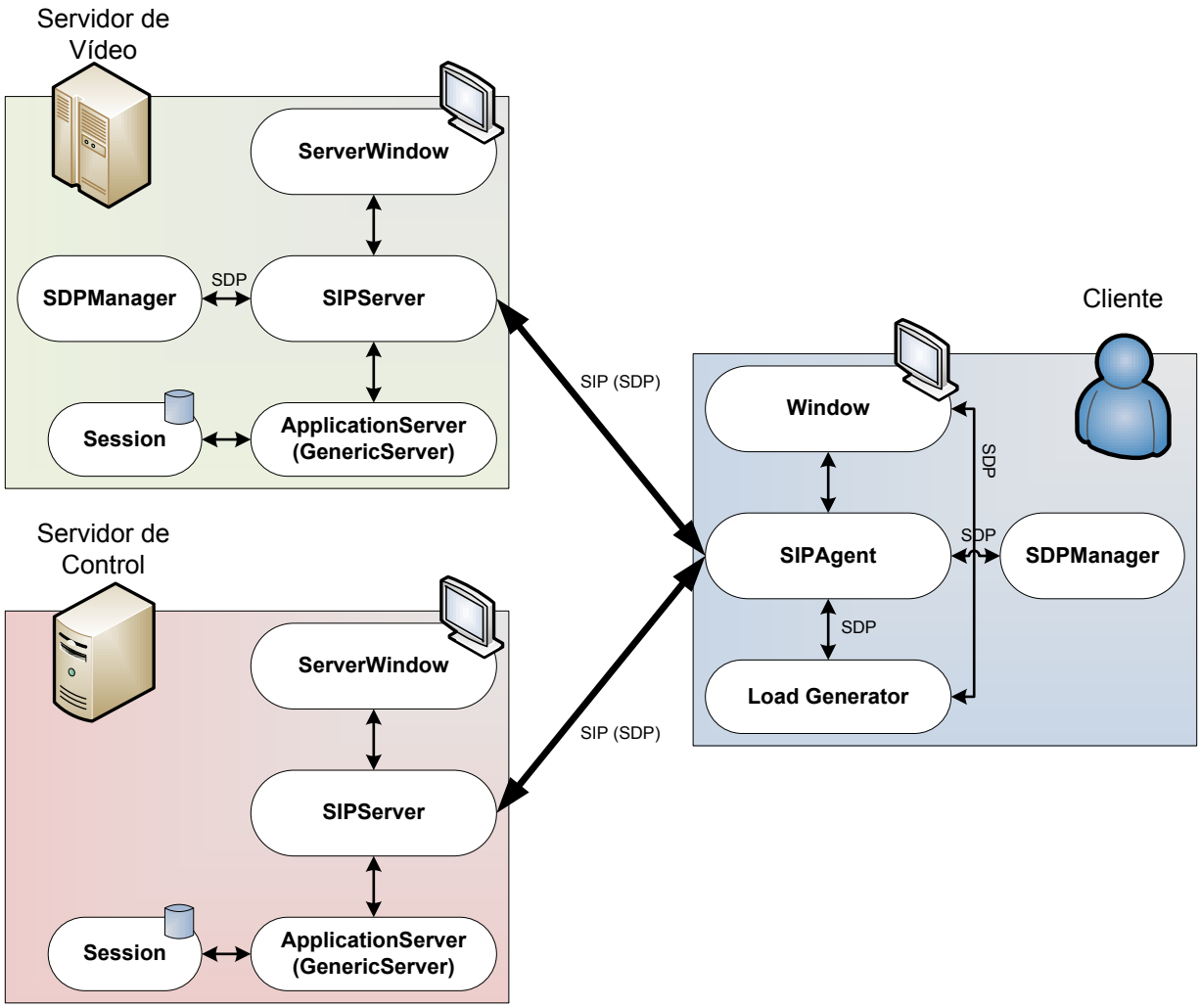

Figura 27: Sistema completo de la aplicación

4.2. Servidor de Control

El S.C. es el sistema con la funcionalidad más limitada. Simplemente atiende las peticiones de los clientes, redireccionándolos al S.V. que pueda servirles el fichero de vídeo con las mejores garantías de calidad y servicio. Sin embargo su papel es muy importante a la hora de centralizar el control de la aplicación, haciéndola más eficiente y evitando cargar innecesariamente el lado del S.V.

En el capítulo 3 se diseño una fase de comunicación con el S.V. para que este le facilitara datos sobre su estado de ocupación y sobre los ficheros de vídeo que puede servir. En la implementación se prescindió de esta parte al ser innecesaria en un proyecto con una arquitectura tan pequeña que solo tiene un S.V.

La aplicación de este servidor contiene cinco clases Java que se detallan a continuación:

- **SIPServer.java:** Clase main de la aplicación del servidor de control. Contiene todo el manejo de solicitudes SIP correspondientes al servidor genérico.
- **GenericServer.java:** Interfaz con los métodos de comunicación del S.C.
- **ApplicationServer.java:** Clase que hereda del interfaz GenericServer.java y cuyos métodos implementan la funcionalidad de comunicación del S.C.
- **Session.java:** Clase que encapsula la variable filePos que contiene la posición en la tabla hash del fichero de vídeo solicitado por el cliente.
- **ServerWindow.java:** Clase encargada de la interfaz gráfica en el S.C.

4.1.1. Interfaz gráfica del Servidor de Control

Este interfaz solo permite configurar la dirección IP y el puerto del servidor SIP. El inicio de la sesión parte del cliente, por lo que el S.C. solo atiende las peticiones que este le envía.

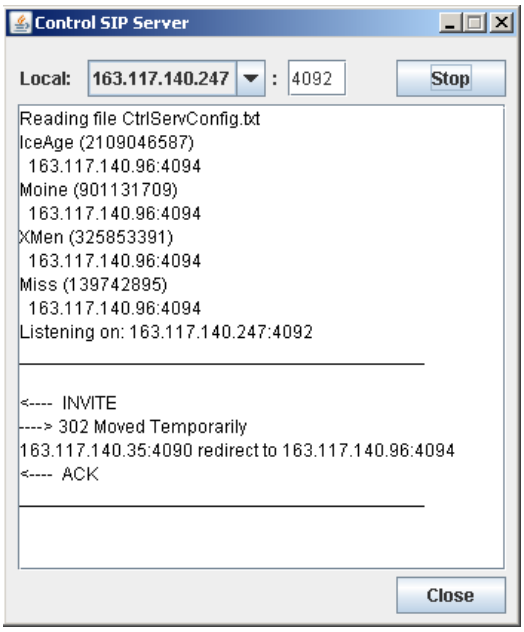

Figura 28: Interfaz gráfica del Servidor de Control

Al ejecutar esta aplicación se procesa el fichero *CtrlSerConfig.txt* que contiene la dirección del correspondiente S.V. que tiene almacenado cada fichero de vídeo. Aunque en esta aplicación se ha empleado el fichero *CtrlSerConfig.txt,* la información que se encuentra en el sería el resultado de acceder a la base de datos que comparte con los servidores de vídeo (figura 2) y ejecutar el proceso Load Balancing Tool para saber cuál es el servidor más apropiado. El contenido de este fichero es:

"nombre fichero" "descripción" códec
dirección IP del S.V.: puerto del S.V.

Con la información del fichero se construye una tabla hash, cuyo identificador es igual a la etiqueta *userinfo* de la petición INVITE del cliente.

Al pulsar el botón "Start" se inicia la escucha de las peticiones del cliente en la dirección SIP introducida. Desde este instante, como se puede ver en la figura 28, se mostrarán por pantalla las diferentes peticiones recibidas, así como las respuestas Moved Temporarily, indicando el S.V. al que se redirecciona al cliente.

4.1.2. Funcionalidad del Servidor de Control

La figura 29 contiene el diagrama de flujo que describe como se procesan las peticiones procedentes del cliente.

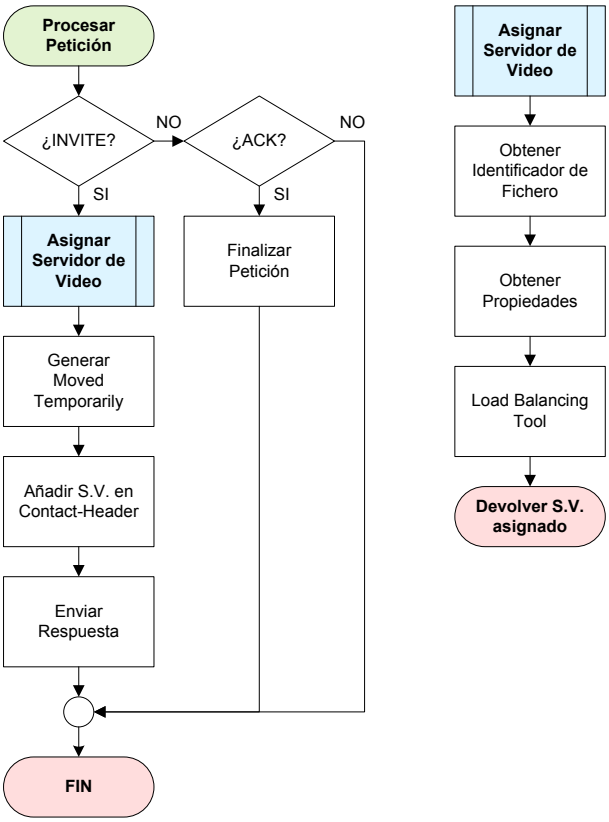

Figura 29: Procesamiento de petición del Servidor de Control

Los puntos a destacar de este diagrama son:

- Al S.C. solo le llegan dos tipos de peticiones del cliente: INVITE y ACK.
- El S.C. solo envía respuestas SIP 302 *Moved Temporarily*.
- El proceso predefinido "Asignar Servidor de Vídeo" extrae el identificador *userinfo* de la petición INVITE y busca el S.V. que pueda atender esta petición.
- Entre el S.C. y el cliente nunca llega a establecerse una sesión SIP, entre ellos solo hay un simple intercambio de mensajes SIP.

4.3. Servidor de Vídeo

A diferencia del S.C., el S.V. tiene una funcionalidad más compleja. Recibe peticiones SIP procedentes del cliente para establecer una sesión multimedia simple o con multidescripción. También atiende las peticiones de modificación de sesión para añadir o eliminar un determinado descriptor, o para cambiar la dependencia de estos descriptores.

Las clases java de esta aplicación son prácticamente las mismas que las del S.C. al compartir funcionalidad, a excepción de la clase SDPManager.java. Las principales diferencias entre ambos servidores son:

- **SIPServer.java:** Se extiende esta clase para que el S.V. atiende más peticiones.
- **Session.java:** Encapsula la posición del recurso en la memoria del servidor (filePos) y la carga de la sesión SDP (sd).
- **SDPManager.java:** Nueva clase que se encarga de la gestión y manipulación de la carga SDP.

4.2.1. Interfaz gráfica del Servidor de Vídeo

Cuando se ejecuta la aplicación, se procesa el fichero *ASconfig.txt* que contiene los ficheros de vídeo que tiene almacenado el S.V. y una descripción de estos. Sería como acceder a la base de datos del servidor. El contenido de este fichero es:

"ruta donde se encuentra el fichero" "nombre del fichero" "descripción" códec

Al igual que con el S.C., con la información del fichero se construye una tabla hash. A esta tabla se accederá a través del identificador *userinfo* de la petición INVITE del cliente. Aunque está contemplado que el S.V. responda negativamente al cliente en el caso de que este solicite al S.V. un fichero de vídeo del que el S.V. no disponga, el servidor siempre debería tener el fichero reclamado, ya que anteriormente el cliente ha sido redireccionado por el S.C. después de acceder este a la base de datos común que tiene con los servidores de vídeo.

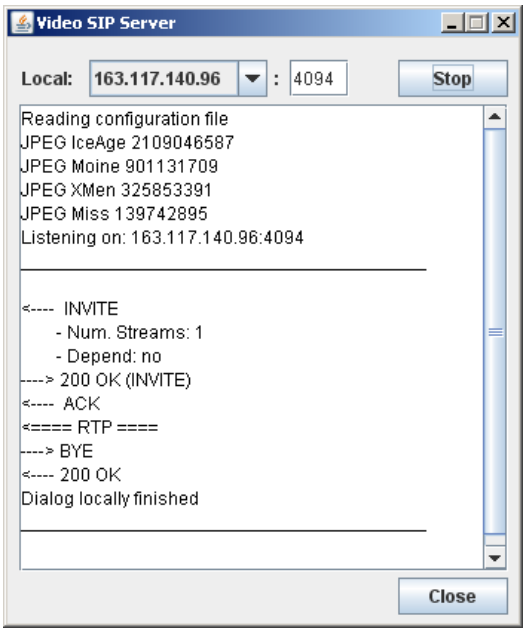

Figura 30: Interfaz gráfica del Servidor de Vídeo

Después de iniciar la escucha del S.V., en su interfaz gráfica (figura 30) se refleja las peticiones recibidas, las respuestas enviadas y el establecimiento de la sesión multimedia. Cuando se recibe la petición del cliente de establecimiento de la sesión, por pantalla se muestran las características del tipo de dependencia de descriptores y número de descriptores solicitados por el cliente. Con la sesión establecida, cuando el S.V. recibe un mensaje UPDATE para modificar las características de la sesión, también se muestra por pantalla los cambios solicitados.

4.2.2. Funcionalidad del Servidor de Vídeo

El diagrama de flujo del S.V. está formado por el diagrama general de la figura 31 y por los diagramas de los procesos predefinidos “Procesar SDP”, “Obtener Modificación” y “Actualizar SIP” de la figura 32.

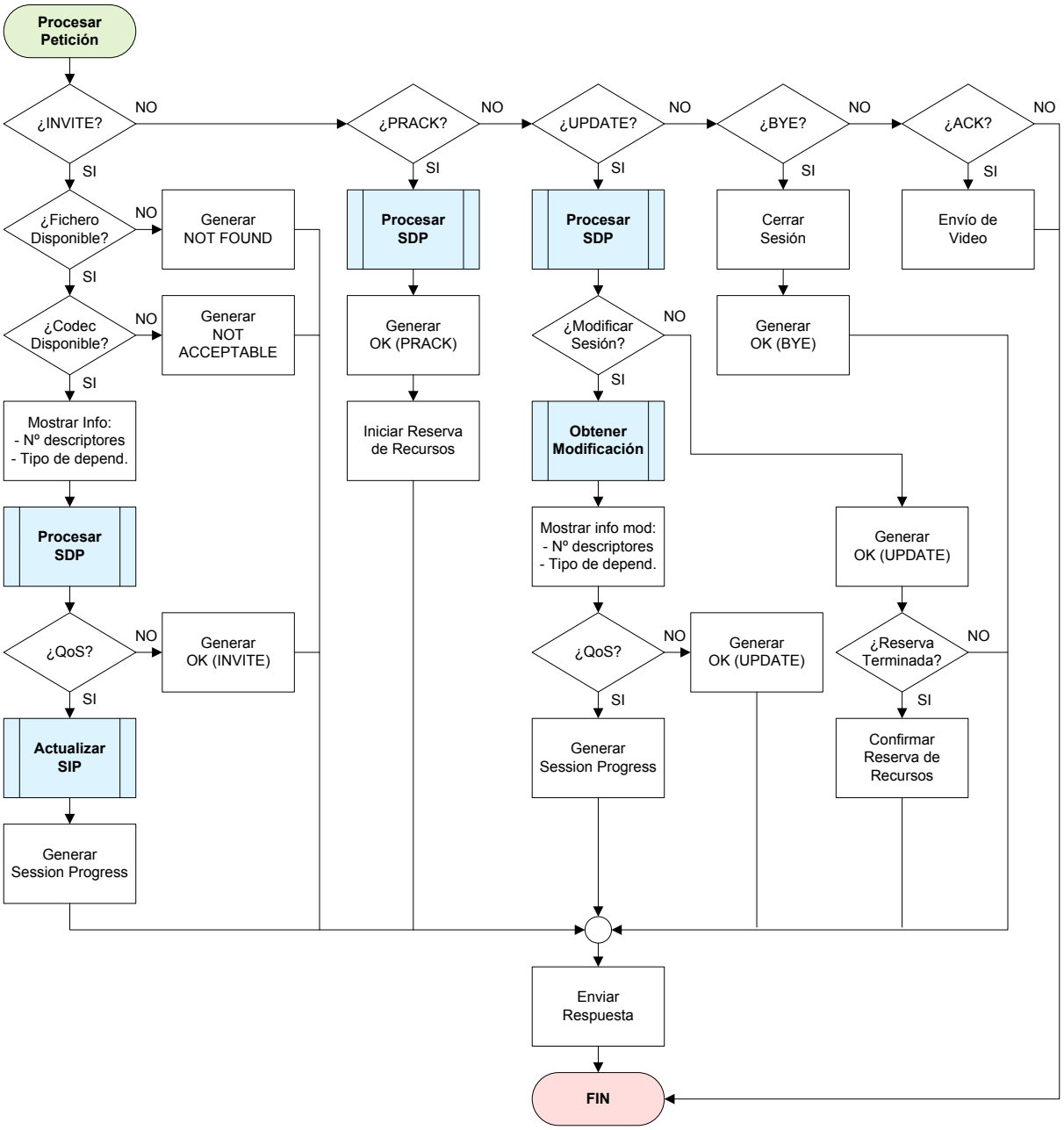

Figura 31: *Procesamiento de petición 1/2 del Servidor de Vídeo*

El procesado de la petición depende de qué tipo sea. En nuestra aplicación solo puede haber cinco tipos de peticiones: INVITE, PRACK, UPDATE, ACK y BYE. En el caso concreto de UPDATE, su naturaleza puede ser distinta.

En el diagrama de la figura 31 la petición UPDATE se clasifica según la caja de decisión "¿Modificar sesión?". Una respuesta negativa indica que este mensaje procede del instante en el que se está estableciendo una sesión con reserva de recursos. Una respuesta afirmativa indica que es una petición de modificación de la sesión. En la implementación, a la hora de diferenciar este mensaje por parte del S.V., se basa en comprobar si la sesión ya está establecida o modificada, o si está en proceso.

La caja de decisión "¿QoS?" indica si la petición de establecimiento o modificación de la sesión es simple o con garantías de calidad de servicio.

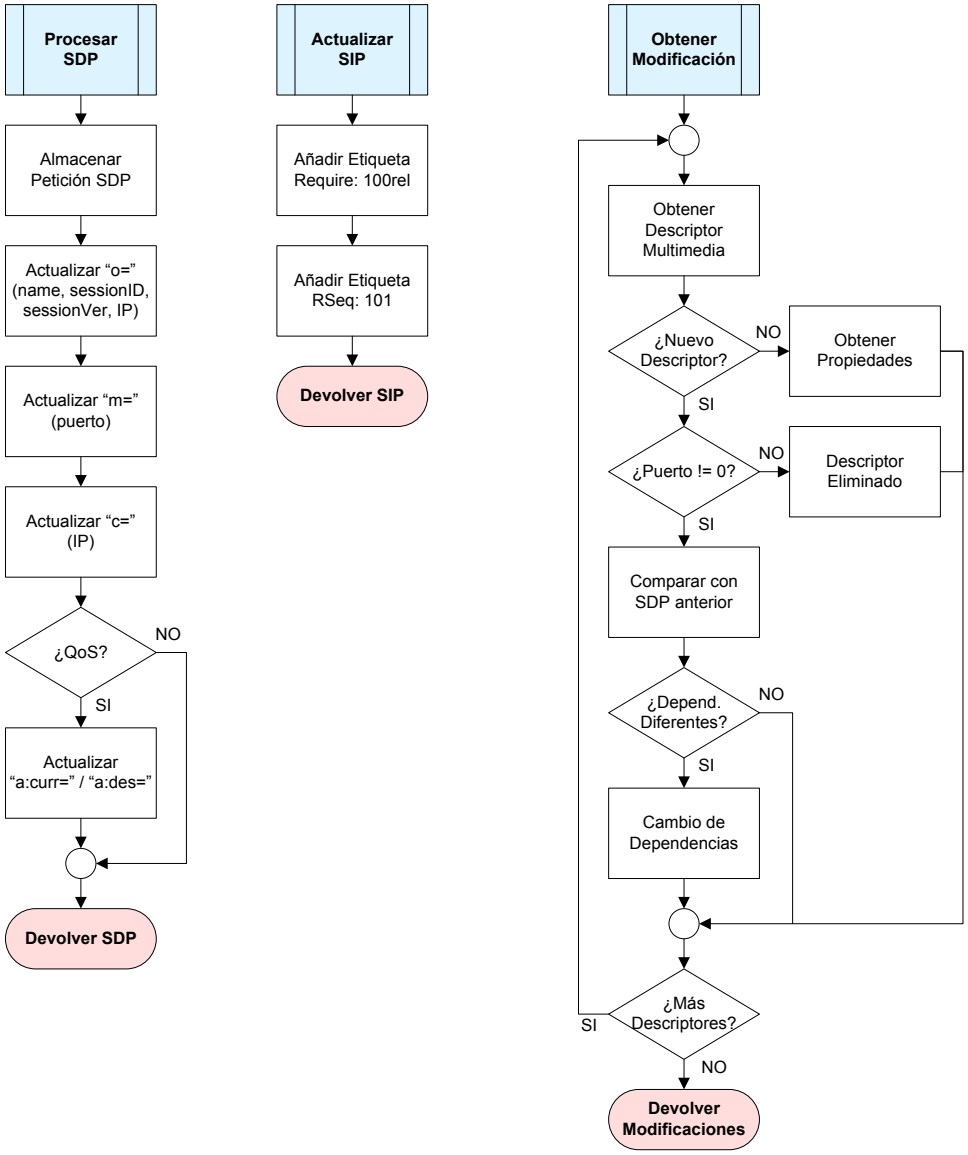

Figura 32: *Procesamiento de petición 2/2 del Servidor de Vídeo*

La figura 32 contiene los tres procesos más importantes de la funcionalidad del S.V. El proceso "Procesar SDP" realiza los siguientes pasos:

- Se modifica el atributo "o=" de cada descriptor multimedia cambiando los campos *name, Session ID*, *Session Version* y dirección IP del flujo RTP por los propios del S.V.
- El puerto de la etiqueta multimedia "m=" se cambia por el puerto RTP del S.V.
- En la etiqueta "c=" se fija la dirección IP del flujo RTPdel lado del S.V.
- En el caso de que sea una petición con garantías de calidad de servicio, habrá que invertir los atributos "a:curr=" y "a:des=" de local a remote y de remote a local.

El proceso de “Actualizar SIP” solo es para el caso concreto de solicitar QoS en el que hay que modificar las entradas Require y RSeq de la cabecera SIP.

Cuando se recibe un mensaje UPDATE solicitando la modificación de la sesión, es necesario procesarlo para saber las propiedades que exige el cliente. Los pasos a dar se reflejan en el proceso “Obtener Modificación”, que trata uno a uno cada descriptor multimedia, comprobando si se ha eliminado un descriptor, se ha añadido un descriptor o si se ha modificado la dependencia de un descriptor comparando con la carga SDP almacenada en la clase Session del S.V.

Cuando llega el ACK al cliente, el S.V. envía el video mediante VLC para que pueda ser reproducido en el cliente. En el caso de que el cliente haya solicitado el vídeo con MDC, el S.V. envía el vídeo de la misma manera que en la petición simple, ya que no es posible el envío de varios descriptores al no disponer de codificadores de multidescripción. La diferencia con el caso simple se encontrará en la información de la petición del cliente que se muestra por pantalla y en que estará permitida la modificación de la sesión.

4.4. Cliente

En el cliente reside el peso de la aplicación de VoD, por eso tiene la interfaz más compleja y con más parámetros configurables. Esta interfaz permite construir la cabecera del mensaje INVITE de SIP que inicia la sesión, así como generar la carga de descripción multimedia (SDP).

También implementa toda la comunicación necesaria a nivel SIP de peticiones y respuestas que se intercambia con el S.C. y el S.V. para el establecimiento y modificación de la sesión. Las clases que forman la aplicación del cliente son:

- **SIPAgent.java:** Gestiones el envío, recepción y procesamiento de mensajes SIP.
- **LoadGenerator.java:** Clase encargada de generar todas las descripciones multimedia y de sesión que manda el cliente.
- **SDPManager.java:** Se encarga de gestionar las modificaciones de los parámetros SDP partiendo de una oferta SDP anterior.
- **Window.java:** Clase que implementan SDPWindow, SIPWindow y SendWindow.
- **SIPWindow.java:** Interfaz gráfica que corresponde a la ventana SIP (Paso 1)
- **SDPWindow.java:** Interfaz gráfica que corresponde a la ventana SDP (Paso 2).
- **SendWindow.java:** Interfaz gráfica que corresponde a la ventana de envío y modificación de la sesión (Paso 3)
- **MessageWindow.java:** Interfaz gráfica que muestra el contenido de los mensajes SIP.

4.3.1. Interfaz gráfica del Cliente

La interfaz tiene tres ventanas que clasifican los tres pasos que tiene que dar el cliente para solicitar la transmisión de un fichero de vídeo.

Los primeros parámetros que debe introducir el cliente se encuentran en la ventana de la figura 33 y son los referentes a SIP. Se clasifican en tres grupos:

- **Video File:** Contiene un listado de los ficheros de vídeo que puede solicitar el cliente.
- **Client:** Permite introducir los datos del cliente referentes a su nombre, IP y puerto. La dirección que se incluye es para recibir los mensajes SIP y nada tiene que ver con la dirección del flujo RTP.
- **Control Server:** En este panel se agrupa la dirección final del S.C. con el que debe contactar el cliente antes de establecer la sesión y sus direcciones de enrutamiento.

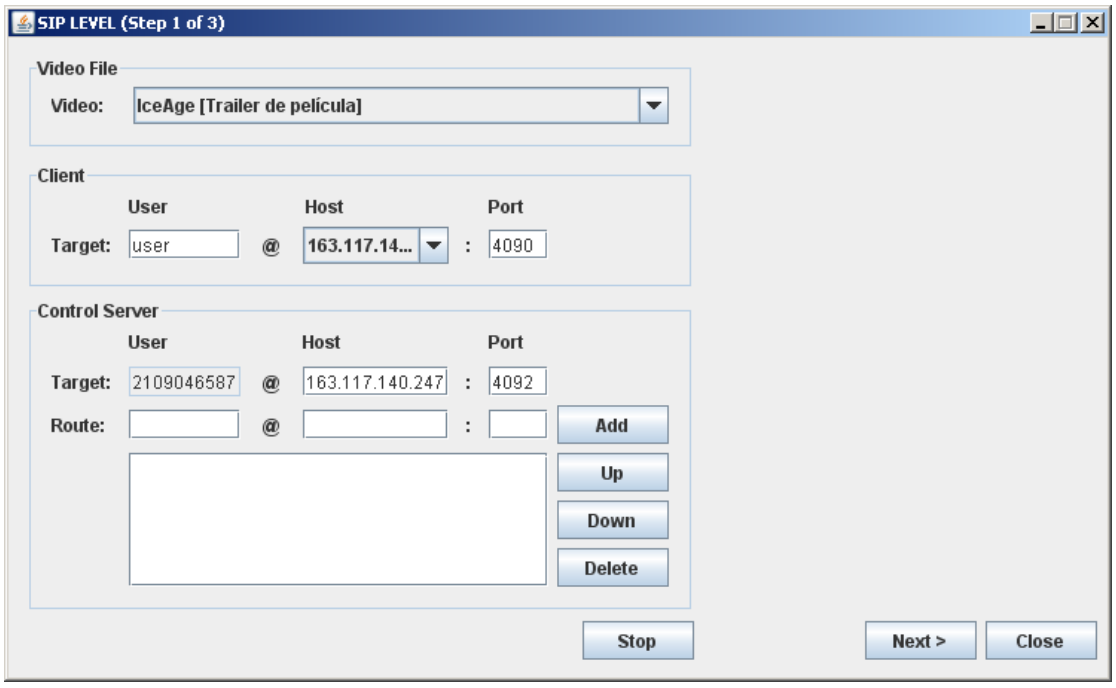

Figura 33: *Interfaz gráfica del Servidor de Vídeo. Paso 1/3*

Es necesario introducir todos los parámetros antes de pulsar “Start” para que el cliente comience a escuchar en la dirección introducida. Un mensaje de información confirmará que se ha iniciado la escucha permitiendo pulsar el botón “Next” para pasar a la siguiente ventana de la aplicación.

De configurar los parámetros SIP, pasamos a configurar los parámetros de la carga SDP en la ventana de la figura 34, la cual se divide en tres nuevos paneles:

- **Session Description:** Permite configurar la dirección del flujo RTP, su orientación (que en nuestra aplicación en el cliente es por defecto *Receive)* y si solicitamos calidad de servicio para ese flujo.

  En el caso de de tener varios flujos multimedia, la dirección del primer flujo sería el *host* y el puerto introducido. Los sucesivos flujos tendrían el mismo *host*, con el puerto incrementado dos unidades, ya que el puerto de RTP siempre debe ser par.

- **Multimedia Description:** Contiene toda la información referente a los flujos multimedia. Dependiendo del tipo de dependencia (*Dependency Type*), tendremos habilitadas unas u otras propiedades como refleja la tabla 3.

	no	mdc	Lc
Nº Media Streams	×	✓	✓
Dependency	×	×	✓
Codecs	✓	×	×

Tabla 3: Propiedades habilitadas en función de Dependency Type

- **Request SDP Message:** Panel que muestra el contenido de la carga SDP. Cada vez que pinchamos en "Create" generamos una nueva carga SDP. El contenido de este panel será el que se incluirá en le petición SIP inicial del cliente.

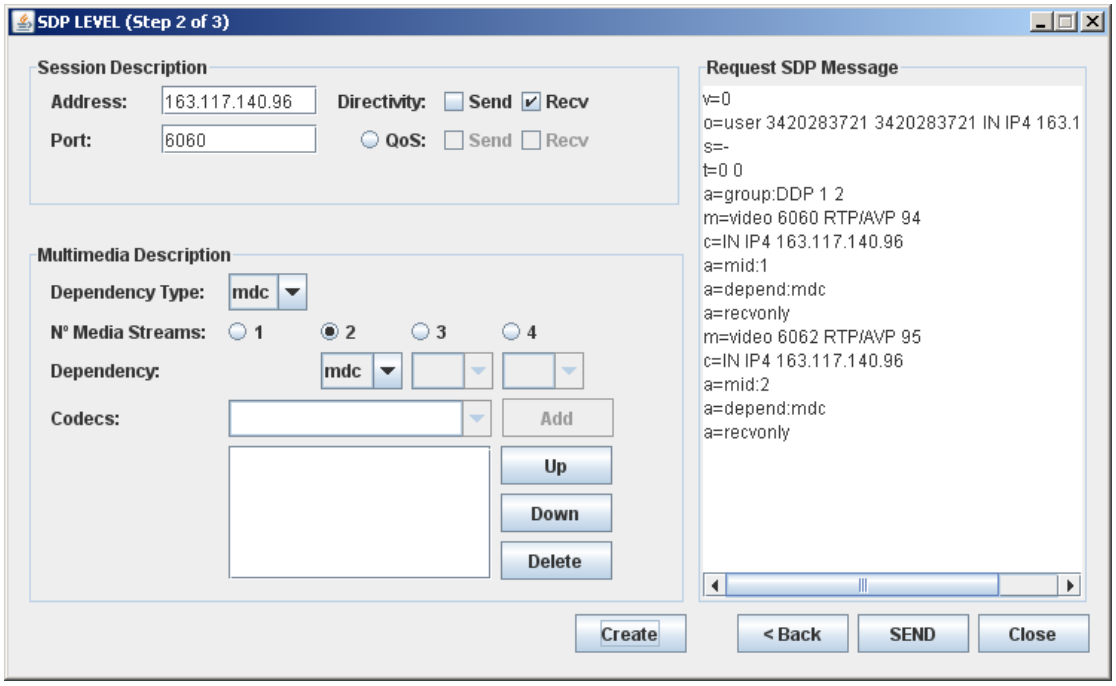

Figura 34: *Interfaz gráfica del Servidor de Vídeo. Paso 2/3*

Al pulsar el botón "SEND" iniciamos el proceso de petición del fichero de vídeo contactando con el S.C. y pasamos a la última ventana de la aplicación del cliente. En esta nueva ventana se puede seguir el estado actual de la sesión y permite la modificación de los parámetros de esta en el caso de haber elegido dependencia MDC o LC.

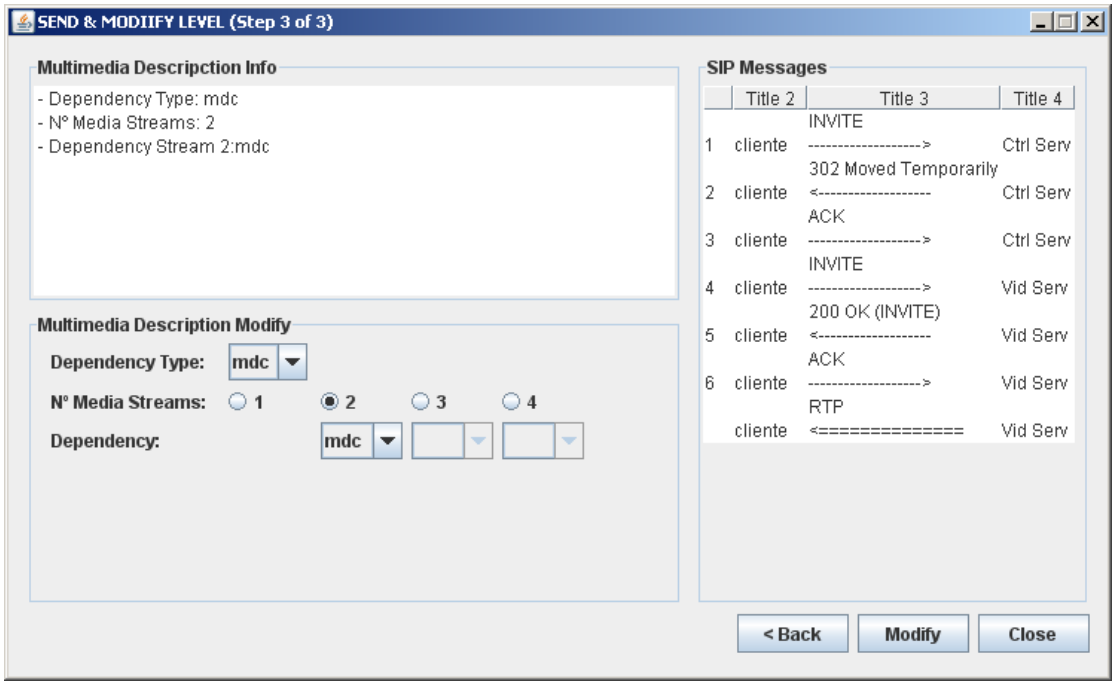

Figura 35: *Interfaz gráfica del Servidor de Vídeo. Paso 3/*

La nueva ventana de la figura 35 es sobretodo informativa y los únicos parámetros configurables son para la modificación de la sesión. Los paneles que se encuentran en esta ventana son:

- **Multimedia Description Info:** Muestra los datos sobre los flujos multimedia (tipo de dependencia, el número de streams y la dependencia que hay entre ellos).
- **Multimedia Description Modify:** Este panel solo se habilita en el caso de tener dependencia MDC o LC, cuando se puede modificar la sesión. En el caso de MDC podemos cambiar el número de flujos multimedia, en LC también es posible cambiar el tipo de dependencia entre los streams.
- **SIP Messages:** Recoge todos los mensajes que envían el cliente con el S.V. y con el S.C. Al pinchar en cada mensaje se abre una nueva ventana (figura 36) con el contenido de los mensajes.

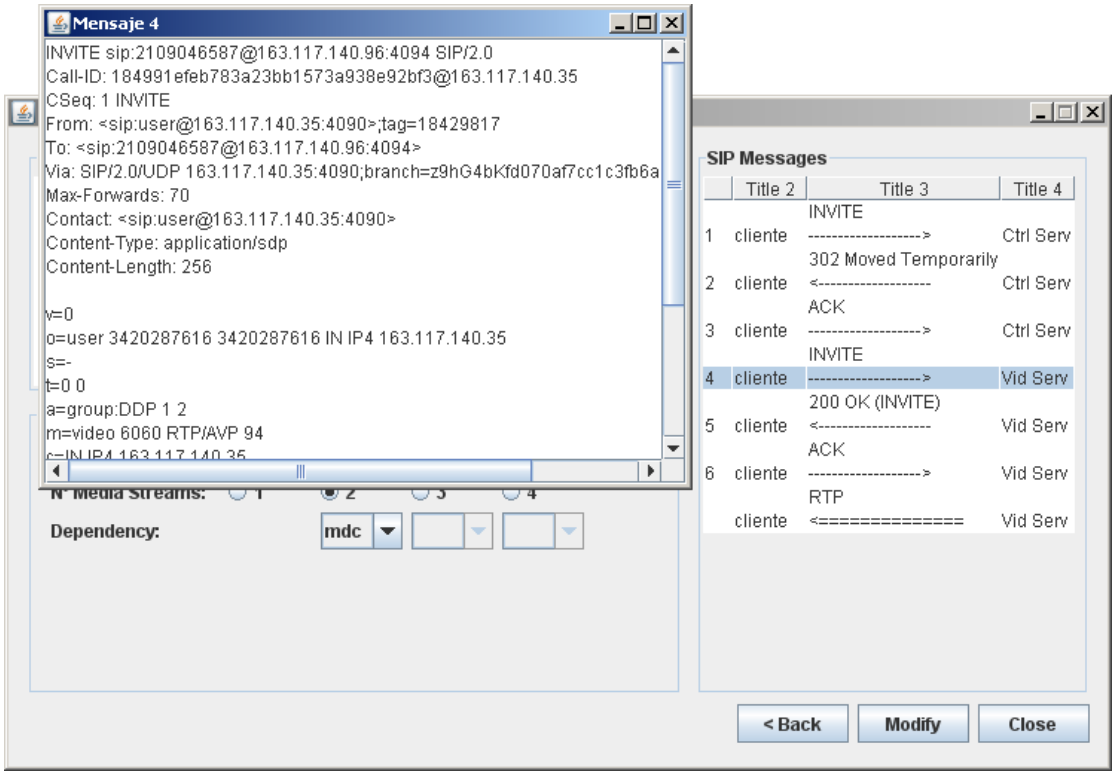

Figura 36: *Interfaz gráfica del Servidor de Vídeo. Detalle de paso 3/3*

4.3.2. Funcionalidad del Cliente

La figura 37 recoge el diagrama de flujo sobre el procesado de las respuestas procedentes del S.V. y del S.C. que realiza el cliente. Emplea el mismo proceso predefinido “Procesar SDP” de la figura 32 que modifica la carga SDP. Hay tres tipos de respuesta:

- OK: Respuesta afirmativa sobre las peticiones que envía el cliente a los servidores. El único caso que hay que destacar es que cuando se recibe un OK (INVITE), se envía un ACK al S.V. para que comience la transmisión de vídeo y a su vez se abre el reproductor VLC para recibirla.
- Session Progress: Inicia el proceso de reserva de recursos.
- Moved Temporarily: Contiene el S.V. al que es redirigido el cliente por parte del S.C.

El diagrama de “Iniciar Sesión” (figura 38) contiene los procesos predefinidos “Obtener parámetros SIP” y “Obtener Carga SDP” que corresponden a los datos que se recogen de las dos primeras ventanas (figura 33 y 34).

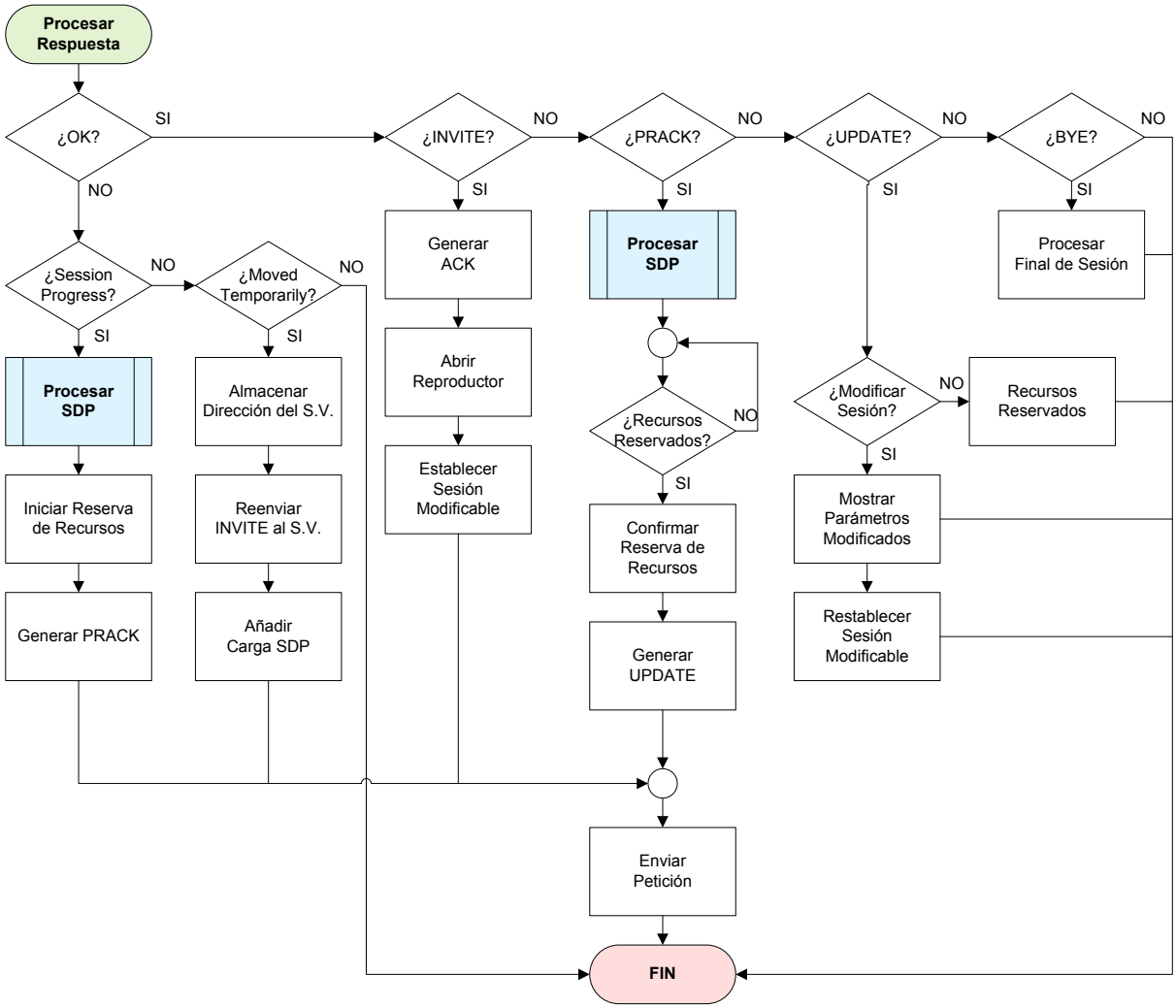

Figura 37: *Procesamiento de respuesta del Cliente*

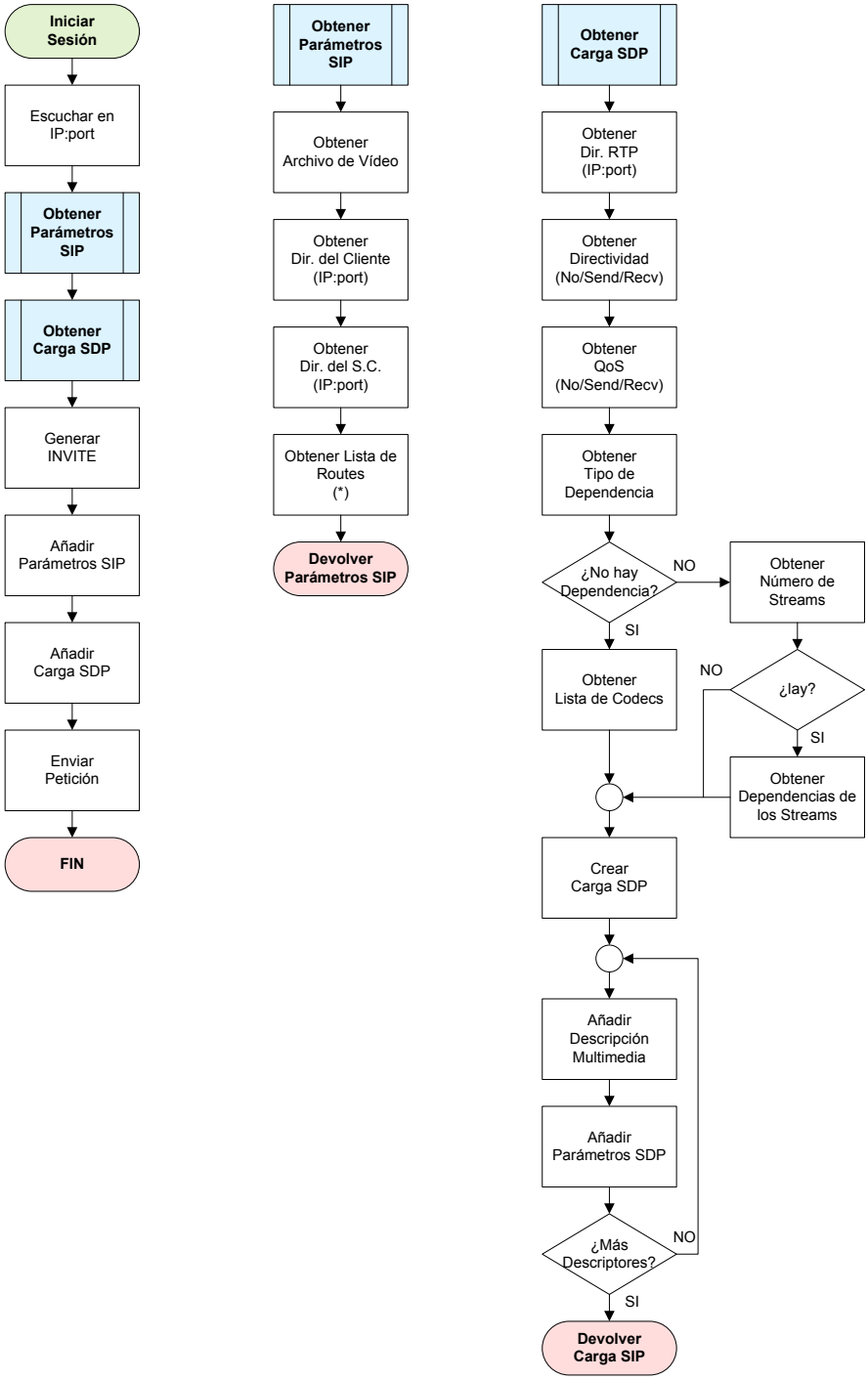

Figura 38: *Inicio de sesión del Cliente*

Capítulo 5
Validación de la Propuesta

El principal objetivo de este proyecto es establecer el escenario y fijar los mensajes de señalización SIP necesarios para una aplicación de VoD en la que se puede emplear codificación con multidescripción. La parte de señalización, tanto de establecimiento de la sesión, como de modificación de esta, ha sido implementada en Java. Al no haber podido integrar en ella los bloques de codificación y decodificación de los descriptores de vídeo por encontrarse estos en una fase de desarrollo que impide su utilización actualmente en una aplicación de estas características, especialmente si es en tiempo real, era necesario un estudio sobre las prestaciones que aporta la multidescripción.

Durante este capítulo se expondrán las medidas y los datos obtenidos en un escenario de multidescripción ficticio. El escenario sobre el que se quiere realizar el estudio está formado por tres actores: el servidor de vídeo, el cliente y la red. En este escenario se ha eliminado al servidor de control ya que solo interviene a nivel de señalización.

En los siguientes apartados, además de detallar el escenario de la simulación, se explicarán los datos y las estadísticas obtenidas, presentando finalmente conclusiones de estas.

5.1. Escenario de las pruebas

En la figura 39 se representa el escenario de la simulación con los tres actores mencionados en la introducción del capítulo: el servidor de vídeo, el cliente y la red. Se han empleado tres ordenadores para cada uno de ellos, todos conectados al mismo switch. Para tener unas medidas reales era necesario introducir los efectos de la red en los flujos de vídeo, esa es la razón de utilizar una red emulada donde se pueden configurar parámetros como delay, jitter y probabilidad de pérdidas.

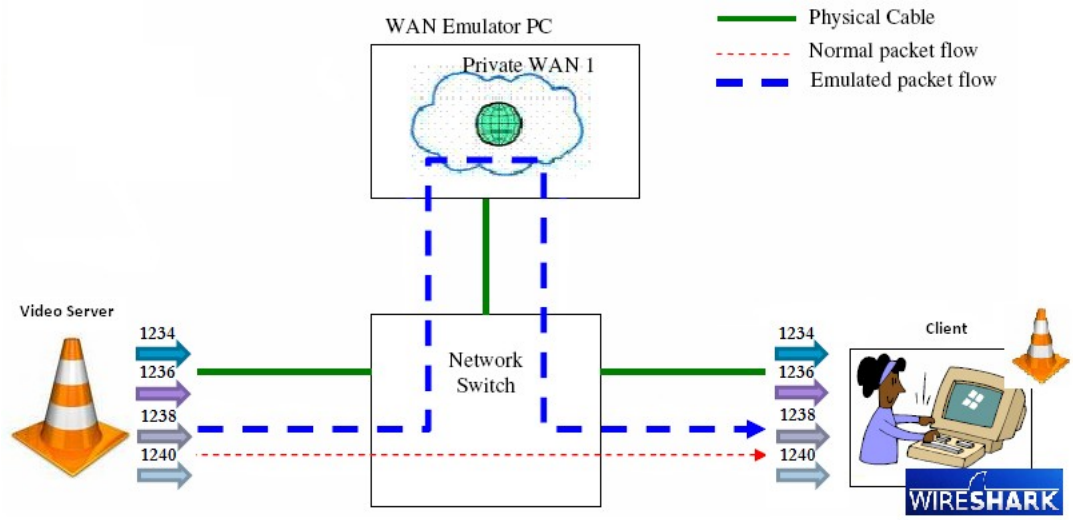

Figura 39: Escenario de la simulación

Las características de cada uno de los actores de este escenario son:

- **Red Emulada**

La función del ordenador que simula la red es la de recibir los paquetes del S.V., someterlos a las propiedades fijadas en su configuración inicial de retardos y pérdidas, y reenviárselos al cliente. Los emuladores que cumplían los requisitos que necesitábamos eran WANem [27] y NetEm [28]. Nos decantamos por el primero por su sencillez y por su fácil control a través una interfaz gráfica.

WANem permite tener diferentes reglas según el puerto destino de los paquetes, lo que nos permite fijar diferentes propiedades sobre los cuatro flujos de vídeo. De esta forma se puede emularr con más precisión una red en la que los descriptores recorrerán diferentes caminos.

El S.V. envía cuatro flujos de vídeo a cuatro puertos diferentes del cliente. Cada flujo tendrá unas características diferentes de retardo, jitter y probabilidad de pérdidas simulando las diferentes prioridades que se pueden fijar en la red. Los parámetros que caracterizan a cada flujo se obtuvieron del artículo del ITU-T [29] y son los siguientes:

	Low Latency (HP)	**Real Time (MP)**	**Elastic (LP)**	**Best Effort**
IPTD Transfer Delay	≤ 100 ms	≤ 400 ms	≤ 100 ms	-
IPDV Delay Variation	≤ 50 ms	≤ 50 ms	-	-
IPLR Packet Loss Ratio	$\leq 10^{-3}$ ms	$\leq 10^{-3}$ ms	$\leq 10^{-3}$ ms	-

Tabla 4: Parámetros recomendados para la simulación según[29]

Finalmente las propiedades que se configuraron en WANem se resumen en la tabla 5. La distribución de los retardos (delay y jitter) es Normal y con correlación del 0 %.

	Low Latency (HP)	Real Time (MP)	Elastic (LP)	Best Effort
Nº Stream	1	2	3	4
Puerto UDP	1234	1236	1238	1240
Delay (ms)	50	350	350	500
Jitter (ms)	50	50	100	200
Prob. Pérdidas (%)	0.1	0.1	1	5

Tabla 5: Parámetros fijados en la simulación

- **Servidor de Vídeo (S.V.)**

Lo ideal sería que el S.V. codificase el flujo de vídeo en varios descriptores y los enviase de forma independiente a través de la red al cliente. Como esto no es posible enviaremos diferentes flujos de datos que simulen los descriptores. Podríamos utilizar herramientas como IPerf para enviar flujos de datos de un ordenador a otro, pero preferimos utilizar el software VLC [2]con vídeos reales, para que el flujo de datos tenga un patrón de vídeo y no sea constante.

Mediante la opción de "Volcado de Red" de VLC [2] se envían desde el S.V. cuatro flujos de diferentes ficheros de vídeo a cuatro puertos del cliente. La razón de utilizar varios ficheros de vídeo es para simular correctamente los descriptores, ya que cada uno tiene un perfil de transmisión (nº frame/time) diferente. Mediante un archivo ejecutable .bat se envían los cuatro flujos al mismo tiempo, para que haya un cierto sincronismo en ellos.

Los vídeos que envía el S.V. forman parte de la colección de Media.Xiph.org [32]. Estos vídeos se emplean normalmente en las pruebas de proyectos relacionados con imágenes de vídeo. Las características de los ficheros enviados se encuentran en la tabla 6.

Vídeo	Stream	Descripción
	Stream 1 Puerto UDP: 1234	4:3 1065 frames
	Stream 2 Puerto UDP: 1236	4:3 1374 frames
	Stream 3 Puerto UDP: 1238	4:3 870 frames
	Stream 4 Puerto UDP: 1240	4:3 961 frames

Tabla 6: Vídeos empleados en la simulación

- **Cliente**

El cliente ejecuta varios procesos VLC [2] en modo "Abrir Volcado de Red", escuchando en los cuatro puertos diferentes desde los que emite el S.V. También debe lanzar antes de la recepción de los ficheros de vídeo Wireshark [33], que permite la captura de tráfico para su posterior procesado.

Una vez finalizada la captura del tráfico, se analizaron los diferentes flujos RTP y los datos obtenidos de cada uno se almacenaron en variables que se procesaron en Matlab. Los parámetros de cada flujo pertenecen desde el instante en el que comienza la retransmisión, hasta que finaliza el flujo de menor duración.

5.2. Procesado de Datos

Antes de profundizar en la fase de procesamiento de datos, es necesario asentar una serie de conocimientos como son el playout delay, el buffer en recepción y el proceso de reproducción, basándonos en los artículos [30] y [31].

Las transmisiones de vídeo tienen una tasa de bit variable (VBR), con esta compresión se consigue una mayor calidad en ficheros de menor tamaño. En estas circunstancias, el buffer de playout es un punto crítico en el lado del cliente. Las variaciones de retardo en la red (jitter) eliminan el sincronismo entre las tramas multimedia (MUs) que constituyen el flujo de vídeo, por eso es necesario un mecanismo de almacenamiento y reordenamiento para garantizar que los paquetes tengan la misma tasa con la que fueron transmitidos en el S.V.

Partiendo del esquema de la figura 40, en el S.V. se generan paquetes de igual longitud con un intervalo entre paquetes de H_n. Cuando los paquetes llegan al cliente, con un retardo de red de D_n, son almacenados temporalmente en el buffer de playout (PoB). El tiempo T que transcurre desde que llega el primer paquete al buffer, hasta que sale para ser reproducido, se denomina *dejitter time* o *playout delay*.

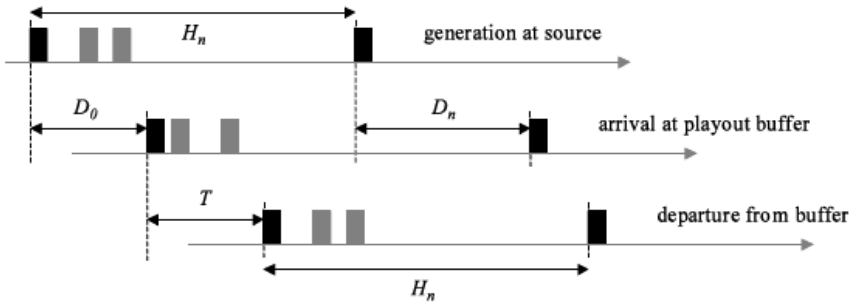

Figura 40: Dejittering del flujo de vídeo

La calidad de la imagen se puede degradar por dos razones diferentes:

- Si un paquete llega al PoB demasiado tarde, la imagen a la que corresponda este paquete se perderá, provocando así una distorsión en el vídeo que se reproduce.
- Si llegan demasiados paquetes al PoB antes que el paquete correspondiente a la imagen anterior, el PoB se desbordará afectando a la calidad del vídeo.

Las probabilidades correspondientes a estos eventos son P_{under} y P_{over} respectivamente, cuyos límites vienen fijados por ε y δ. Para calcular estas probabilidades, partimos de las siguientes ecuaciones, siendo Q el tamaño en paquetes de PoB y R_{max} la tasa de transmisión.

$$P_{under} = Pr[H_n + D_n > H_n + D_0 + T] = Pr[D_n > D_0 + T] \leq \varepsilon$$

$$P_{over} = Pr\left[H_n + D_n \leq H_{n-Q} + D_0 + T\right] = Pr\left[D_n \leq D_0 + T - \left(H_n - H_{n-Q}\right)\right] =$$
$$= Pr[D_n \leq D_0 + T - Q/R_{max}] \leq \delta$$

Con los límites ε y δ fijados, podemos dimensionar el tiempo T como muestran las gráficas de la figura 41.

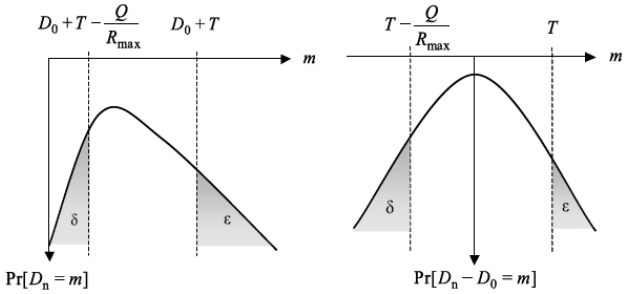

Figura 41: Dimensionamiento de T cuando D_0 es conocido y desconocido

Lo explicado hasta ahora nos indica que los datos de entrada en el procesamiento de datos son T (playout delay) y Q (número de paquetes máximo en el PoB). Estos datos van a condicionar las salidas de nuestras pruebas que serán las probabilidades P_{under} y P_{over}. En el cálculo de P_{under} se han incluido los paquetes que no llegan a tiempo a PoB y los paquetes que se han perdido en la red.

Con el procesamiento de los datos capturados buscamos obtener resultados como los de la figura 42. La gráfica de c*lient video reception* se representa con los paquetes que llegan al cliente. El proceso de *playout* en el cliente de tasa constante nos servirá para calcular P_{under} y P_{over}. Si un paquete llega al cliente después de su instante de reproducción por este proceso, se descarta. Y si el tiempo entre la llegada de un paquete y su reproducción por este proceso es lo suficientemente grande como para desbordar al PoB, también se descarta.

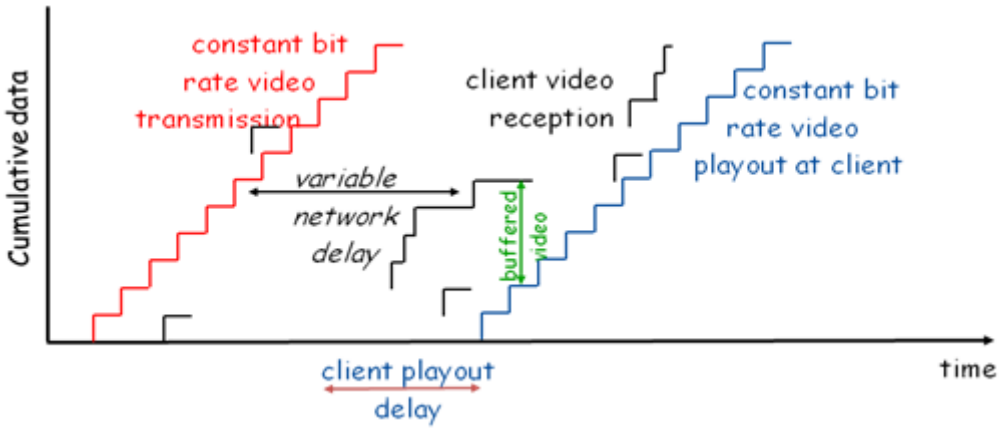

Figura 42: Gráfica característica de Playout Buffer

Sin embargo nosotros no tenemos una tasa constante de bit, sino que es variable (VBR), por lo que no podemos definir el proceso de *playout* como una recta de pendiente constante. El problema que surgió en esta etapa era como definir este proceso. Al ser este proceso una barrera que marca el límite temporal de llegada de cada paquete, lo que se hizo fue tomar el perfil paquete/tiempo de llegada en una situación ideal, es decir, como llegan los paquetes al cliente en una red sin retardos y sin pérdidas, donde la calidad de la reproducción del vídeo es la máxima posible. Para conseguir este perfil, se enviaron los cuatro flujos de vídeo al cliente sin utilizar la red simulada y con los datos capturados construimos este proceso de *playout.*

En este punto tenemos almacenados en cuatro matrices los parámetros de número de secuencia, delta (ms), jitter (ms) y tiempo de llegada (ms) de los paquetes RTP de cada flujo. Posteriormente, en función de los datos de entrada T (Playout Delay) y Q (Tamaña de PoB), se procesan estas variables con Matlab.

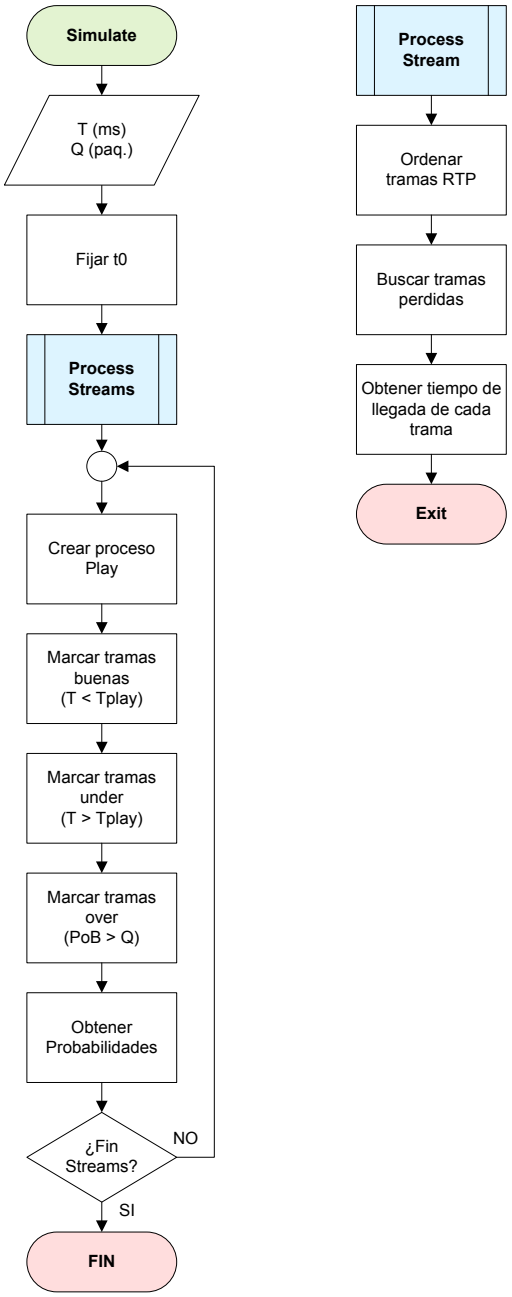

Figura 43: Diagrama de flujo de "simulate.m"

El diagrama de flujo de la figura 43 pertenece al fichero "simulate.m" que realiza esta función. Esta función comienza fijando t0, el instante en el que llega el primer flujo de vídeo al cliente. El proceso de reproducción comenzará en este instante sumado el retardo T. Como se ha comentado anteriormente el perfil de número de paquete e instante de tiempo se ha obtenido en una captura de paquetes sin emplear la red de retardos.

El proceso de reproducción solicita las tramas de forma ordenada. Si en este instante esta trama no se encuentra en el buffer PoB (si la trama no ha llegado al cliente), se marcará como trama "under" para el posterior cálculo de P_{UNDER}. De igual modo para P_{OVER}, se marcará la trama como "over" si cuando llega al cliente el PoB está desbordado.

La probabilidad de pérdidas la calcula la función "processStream.m" ordenando de menor a mayor las tramas RTP en función de CSeq. Si hay algún salto en CSeq significa que la trama se ha perdido.

5.3. Resultados

Este apartado se dividirá en cuatro puntos, cuatro resultados que nos darán diferente información de las pruebas realizadas:

- Variación de T en función de la P_{UNDER} exigida a las diferentes conexiones de los flujos.
- Progresión de la capacidad del buffer PoB.
- Probabilidad de que las tramas no lleguen a tiempo (P_{UNDER}), de pérdidas y de desbordamiento (P_{OVER}) de cada flujo.
- Número de flujos de vídeo a lo largo del tiempo que indica la máxima calidad posible de la imagen en cada instante.

5.3.1. Tiempo T en función de la P_{UNDER}

Antes de calcular las probabilidades de cada flujo, era necesario conocer la influencia de las variables de entrada T y Q. Con este subapartado se quiere conocer el *playout delay* (T) que hay que fijar para tener una máxima P_{UNDER} (ε) en las diferentes conexiones de la simulación. Las figuras de la 44 a la 47 son el resultado de procesar los datos con un rango de valores T comprendido entre los 0 y 400 ms, y obtener la correspondiente probabilidad P_{UNDER} para cada caso.

Con las mejores conexiones (Low Latency y Real Time) se obtienen gráficas similares al tener el mismo jitter de 50 ms. Para obtener una P_{UNDER} baja, debe retrasarse más el inicio de la reproducción (T), pero llega un punto en el que no podremos tener mejores probabilidades aunque se siga aumentando T. La cota máxima de T se encuentra entre 85 y 75 ms para una probabilidad aproximada de P_{UNDER} de 10^{-3}.

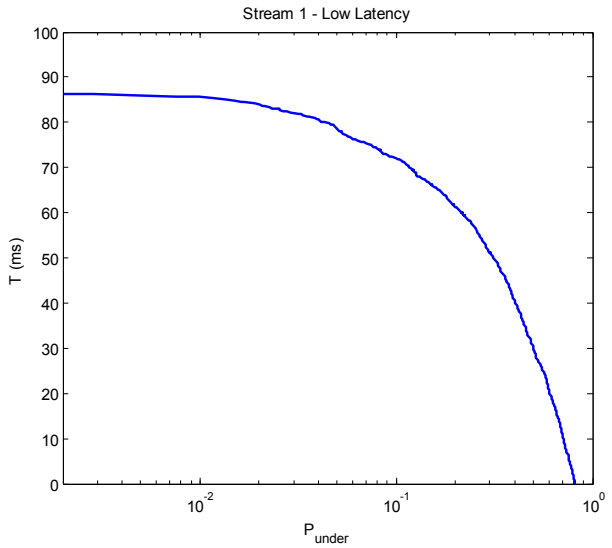

Figura 44: T(P_{UNDER}) para la conexión Low Latency

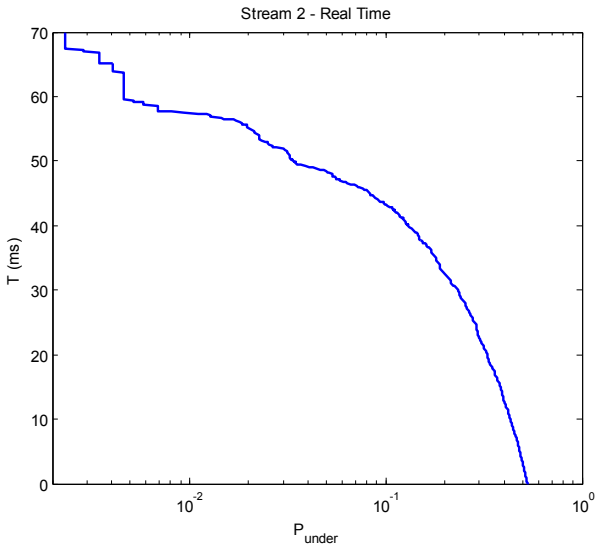

Figura 45: T(P_{UNDER}) para la conexión Real Time

Recordemos que en la conexión Elastic se fijo un jitter de 100 ms y en la Best Effort de 200 ms, lo que justifica las diferencias entre las gráficas de las figuras 32 y 33. Para una probabilidad aproximada de P_{UNDER} de 10^{-2} tenemos una cota máxima de T de 180 ms para la conexión Elastic y de 380 ms para la conexión Best Effort.

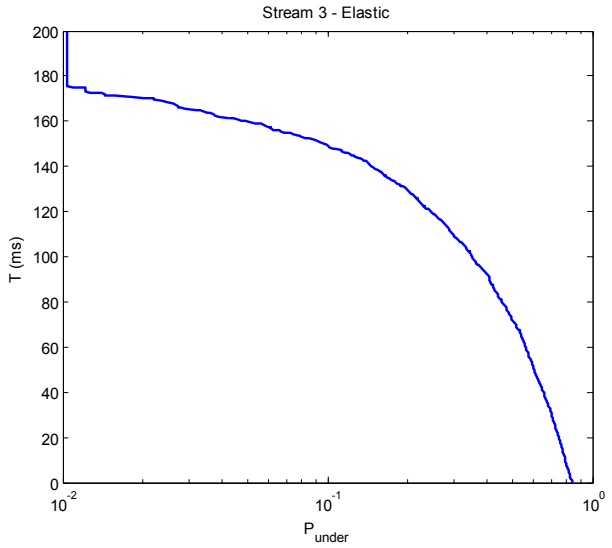

Figura 46: T(P_{UNDER}) para la conexión Elastic

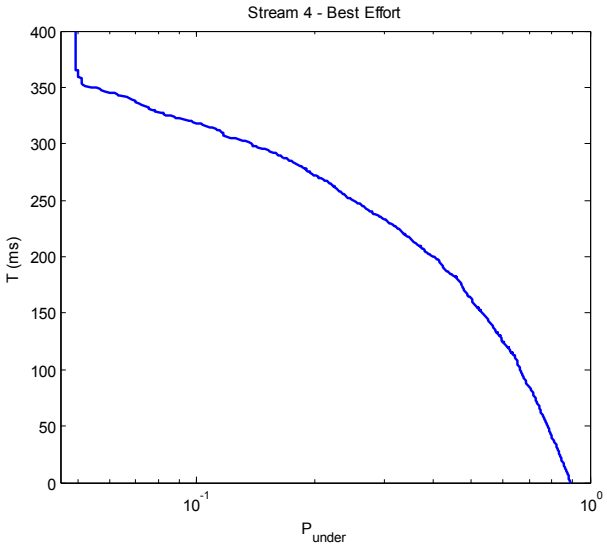

Figura 47: T(P_{UNDER}) para la conexión Best Effort

La principal conclusión que debemos obtener de estas gráficas, es que el máximo tiempo T que podemos fijar en cada conexión para tener la menor probabilidad P_{UNDER} posible, tratando cada conexión de forma independiente, es aproximadamente el doble del jitter de la conexión.

Pero en nuestro caso no podemos considerar las conexiones independientes, ya que las cuatro fijan la calidad final de la señal de vídeo. El tiempo T que se establezca en la aplicación, será el mismo para cada conexión. La elección de este intervalo vendrá condicionada por las necesidades de la aplicación. Si no nos importa tener un considerable retardo en el inicio de la reproducción, fijaremos un tiempo T elevado, lo que nos permitirá disponer del máximo número de descriptores, teniendo por consiguiente la máxima calidad

de vídeo posible. Si necesitamos reproducir el vídeo lo antes posible, escogeremos el tiempo T lo más bajo posible, para una calidad mínima exigida.

5.3.2. Ocupación de PoB

Al igual que en el subapartado anterior, en este se quiere conocer entre que valores se encuentra el segundo dato de entrada (Q). Por eso en la figura 48 se representa el número de tramas que se acumulan en el buffer de PoB, las cuales se van consumiendo a medida que avanza el proceso de reproducción. Se muestra el número de paquetes acumulado en las cuatro conexiones en el PoB, en una prueba en la que se ha fijado T a 100 ms y Q a 200 paquetes, un valor lo suficientemente elevado para que no se descarten paquetes por desbordamiento. El tamaño del buffer Q, corresponde al número máximo de paquetes acumulados que se permiten en cada stream.

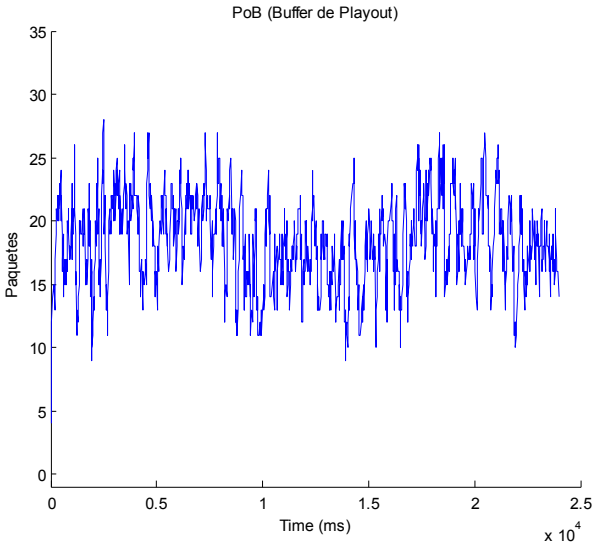

Figura 48: Ocupación de PoB a lo largo del tiempo

En las peores conexiones (Streams 3 y 4) no se acumulan prácticamente las tramas para un tiempo T tan bajo ya que se consumen en el mismo instante de llegada al cliente, sin embargo la ocupación del PoB en las mejores conexiones se encuentra entre 8 y 18 paquetes.

Si VLC envía tramas UDP de 1370 bytes, tener un buffer total con una capacidad de 30 paquetes para que P_{OVER} sea cero, implica que el tamaño del buffer debe ser aproximadamente de 40.2 KB. Con un tamaño de buffer exigido tan pequeño podemos despreocuparnos de la probabilidad P_{OVER}, no va a condicionar la calidad de la imagen de vídeo.

Esta conclusión la realizamos para el caso concreto de nuestras pruebas, en las cuales la señal de vídeo tiene una calidad media. Si transmitimos vídeos de alta definición (HD), el número de paquetes acumulado en el PoB aumentará, necesitando por tanto un buffer de mayor tamaño de bytes. En este caso, el tamaño del buffer exigido puede ser lo suficientemente grande para no poder descartar P_{OVER}.

5.3.3. Probabilidades de cada flujo

Las probabilidades de la tabla 7 se han obtenido fijando un tamaño de PoB (Q) lo suficientemente grande para que P_{OVER} sea cero y un *playout delay* (T) de 100 ms. Los datos de entrada fijados se basan en los resultados obtenidos anteriormente.

Stream	P_{OK} (%)	P_{UNDER} (%)		
		Time	Loss	Total
Stream 1 Low Latency	**99.88**	0	0.12	**0.12**
Stream 2 Real Time	**99.77**	0	0.23	**0.23**
Stream 3 Elastic	**64.85**	34.11	1.04	**35.15**
Stream 4 Best Effort	**33.58**	61.49	4.92	**66.42**

Tabla 7: Probabilidades de cada flujo

La calidad del flujo de vídeo en cada conexión viene expresada por la probabilidad P_{OK}. Como es evidente, en las mejores conexiones tendremos mayor número de tramas buenas, es decir, tramas de un descriptor que llegan a tiempo para que se añadan al flujo de video que se va a reproducir. La probabilidad P_{UNDER} es la suma del porcentaje de tramas perdidas y de las tramas que no llegan tarde al PoB.

Las figuras de la 49 a la 52 representan el instante de llegada de cada paquete en el cliente junto con la curva de reproducción *Play*, que es diferente para cada prioridad. Se han dibujado las tramas en un corto espacio de tiempo para su mejor visualización. Si el paquete llega antes que el proceso *Play,* se marca como bueno representándolo en azul. Pero si no llega a tiempo, se descartará el paquete, no se añadirá al flujo de video que reproduce *Play* y se representará en rojo.

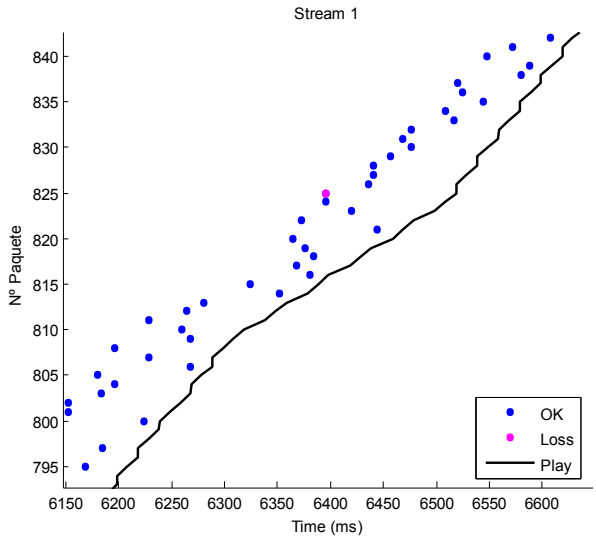

Figura 49: Nº paquete vs. Tiempo en Stream 1

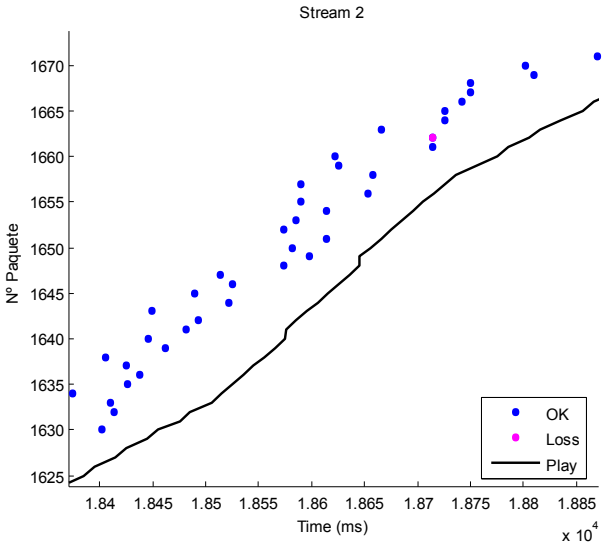

Figura 50: Nº paquete vs. Tiempo en Stream 2

En las gráficas del stream 1 (figura 49) y del stream 2 (figura 50) se puede ver como la mayaría de las tramas se encuentran por encima del proceso *Play*, por eso se marcan como buenas y se representan en azul. Estas conexiones tienen una probabilidad de pérdidas del 0.1 %, por eso prácticamente no hay tramas perdidas, las cuales se representan en magenta.

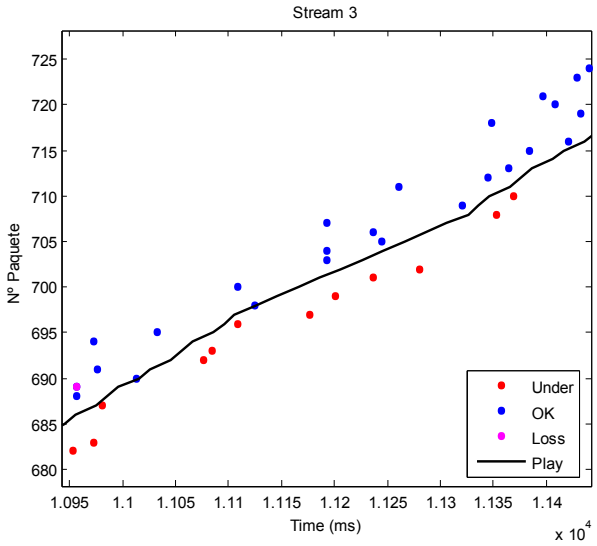

Figura 51: Nº paquete vs. Tiempo en Stream 3

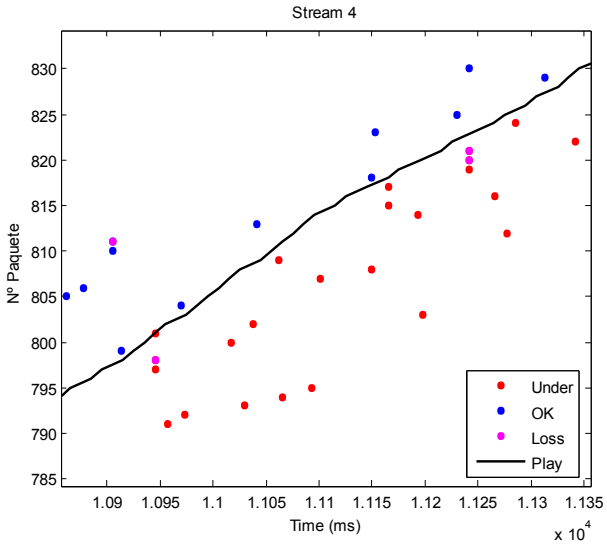

Figura 52: Nº paquete vs. Tiempo en Stream 4

El stream 3 (figura 51) ya comienza a tener tramas que no llegan a tiempo, tramas que se sitúan por debajo del proceso *Play* y se representan en rojo. Las probabilidades del stream 4 son P_{OK} = 33.58 % y P_{OVER} = 66.42 %, el número de tramas malas en este caso es mayor que el de la buenas, por eso en la figura 52 el número de tramas en rojo es mayor que en azul. También esta conexión tiene la probabilidad de pérdidas más alta, del 5 %, por lo que el número de tramas en magenta ha aumentado.

Todas las figuras reflejan como llegan las tramas de desordenadas, cuanto peor es la conexión, más desordenadas llegan. La diferencia de tiempos entre paquetes (jitter), también varía y va aumentando con el empeoramiento de la conexión. En la figura 49 las tramas están agrupadas y pegadas al proceso *Play*, en la figura 52 están más desagrupadas y alejadas de *Play*. Es necesario recalcar que este proceso no tiene una pendiente constante como el de la figura 42 y que su perfil es diferente para cada conexión.

5.3.4. Nº de flujos a lo largo del tiempo

Finalmente, en este subapartado se va a indicar el número de streams que llegan al cliente a lo largo del tiempo. Cuantos más streams lleguen al cliente y por extensión al decodificador de descriptores, mejor será el flujo de vídeo final. Los datos de la tabla 8 corresponden a una simulación con T fijado a 100ms y Q a 30 paquetes.

Tendremos una máxima calidad (4 streams) el 22.81 % del tiempo y una calidad media (2 y 3 streams) el 76.14 % del tiempo. Para MDC esta afirmación es cierta, pero para LC viene condicionada a que el base stream sea el stream 1, aquel que tiene la mejor conexión (Low Latency).

Número de Streams	P (%)
Ninguno	**0.00**
1 Stream	**1.05**
2 Streams	**22.64**
3 Streams	**53.50**
4 Streams	**22.81**

Tabla 8: Porcentaje de nº de streams en el tiempo

De la tabla 7 obtenemos que la probabilidad P_{OK} del stream 1 es del 99.88 % y para el stream 2 del 99.77 %, por lo que si el base stream de LC se transmite por cualquiera de estas dos conexiones, garantiza la señal de video en el cliente la mayor del tiempo. Con MDC siempre podremos reproducir la señal al no tener nunca ausencia de descriptores.

En la figura 53 se representa el número de streams (o descriptores) del que dispone el cliente a lo largo del tiempo. Haciendo un zoom a esta gráfica (figura 54) se puede observar como lo que se dibuja es el número de paquetes acumulados en cada instante. Por ejemplo, en el instante t podemos tener el paquete 850 de dos descriptores, por lo que tendremos la calidad del 50 % en la señal de video final en ese instante.

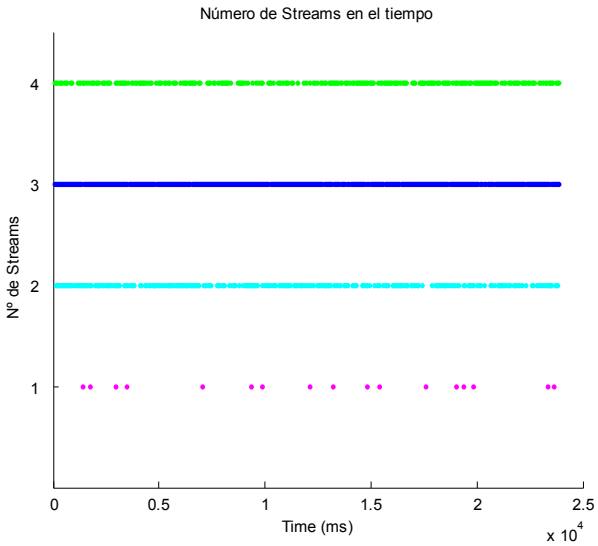

Figura 53: Nº de streams a lo largo del tiempo

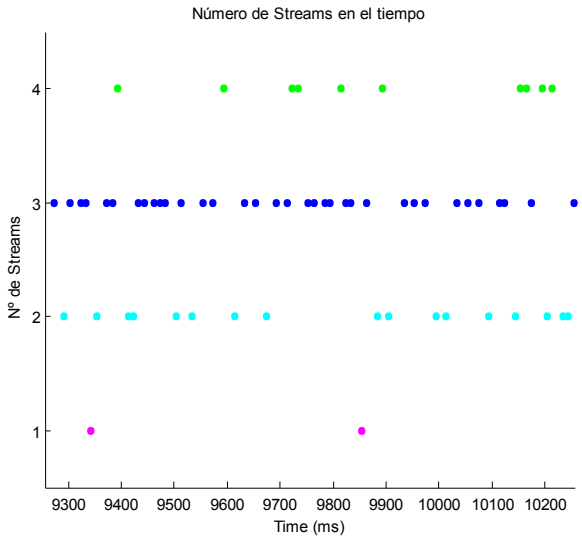

Figura 54: Detalle de la figura 53

De esta gráfica hay que destacar que el número de streams, o lo que es lo mismo, la calidad de la señal, está distribuida a lo largo del tiempo. No tenemos un 1.05 % un stream y a continuación un 22.64 % dos streams. Aunque la mayor parte del tiempo, un 53.5 % del tiempo, se tienen 3 streams, pero el número de streams varía constantemente.

Como línea de trabajo futura quedaría por comprobar cómo afecta a la señal de vídeo final los cambios bruscos en el número de descriptores recibidos para realizar la decodificación. Sería interesante saber si el cliente es capaz de apreciar cambios en la calidad de la imagen cuando se pasa de decodificar una señal con un determinado número de descriptores, a decodificar con más o menos descriptores, para volver finalmente a decodificar con el mismo número de decodificadores anterior.

Capítulo 6
Conclusiones y Trabajo Futuro

Se ha diseñado el sistema de señalización SIP/SDP para una plataforma de VoD que permite el establecimiento de una sesión normal, de una sesión con codificación con multidescripción y la modificación de dicha sesión por parte del cliente. También se ha establecido la arquitectura de esta plataforma formada por tres tipos de nodos: S.C., S.V. y cliente. Este tipo de arquitectura permite la formación de redes más complejas de tipo P2P, cuya finalidad es reducir la carga en los servidores vídeo y mejorar la calidad de servicio en el cliente.

El diseño de esta aplicación, refiriéndonos a los diferentes mensajes SIP/SDP que en ella se envían, se basa en varias RFCs y drafts al no existir un único documento que resolviese todas nuestras especificaciones. De esta manera, la propuesta de este proyecto tiene una validez mayor y podría servir como referencia para una propuesta oficial que agrupe la señalización SIP/SDP de una aplicación de VoD.

Las tres aplicaciones implementadas facilitarán el despliegue de este servicio en escenarios reales con mayor número de nodos y más complejos. La sencillez de los interfaces, especialmente la del cliente, elimina la complejidad que guarda la generación de las cargas SDP y de la cabecera de los mensajes SIP, mostrando únicamente los parámetros necesarios para la petición de un fichero de vídeo.

Referente a esta parte del proyecto, como línea de trabajo futura quedaría realizar pruebas de peticiones en paralelo en los servidores. También habría que desplegar esta aplicación en un entorno basado en IMS, donde en vez de utilizar simplemente el cliente y los servidores en una misma subred, se utilizaría un servidor SIP en configuración proxy con el objetivo de verificar su validez.

El capítulo de “Validación de la Propuesta” ha reflejado cuales son los parámetros más importantes de una aplicación multimedia con codificación con multidescripción y ha fijado los valores aproximados que deben tener. Estos parámetros están condicionados por el tipo de conexión, dependen de las características de retardo, jitter y probabilidad de pérdidas del enlace por el que se transmiten los descriptores.

El primero de estos parámetros es el *playout delay* (T) que tomará valores entre 80 ms y 380 ms dependiendo del tipo de conexión. Este retardo establece el valor del segundo parámetro, la probabilidad P_{under}, que condiciona la calidad de la señal de vídeo final. El

último parámetro, que también limita la calidad de la señal, es el tamaño del PoB que fija la probabilidad P_{over}.

Las principales líneas de trabajo futuro para este apartado son:

- Realizar pruebas con descriptores reales. De esta manera habría dependencia entre los diferentes flujos multimedia y se podría comprobar cuál es la calidad real que hay en la señal de vídeo final.
- Como se indica en el subapartado 5.3.2, habría que volver a emular este escenario de multidescripción con vídeo de alta calidad para ver la influencia de P_{over}. En las pruebas realizadas en este proyecto se emplearon vídeos de calidad media que no requerían una capacidad grande del PoB, por lo que siempre se obtenía un probabilidad de P_{over} del 0%.
- En el subapartado 5.3.4 se ha dejado una línea de investigación abierta para saber la influencia que tiene en la calidad de la imagen los cambios rápidos al pasar de decodificar una señal de vídeo de 3 descriptores a 4 descriptores. Sería interesante saber si el cliente percibe estos cambios bruscos y si son molestos.

ACRÓNIMOS

ACK	Acknowledgement
API	Application Programming Interface
DCT	Discrete Cosine Transform
DDP	Decoding Dependency
FID	Flow Identification
GOP	Group Of Pictures
HP	High Priority
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem
IP	internet Protocol
IPDV	IP Delay Variation
IPLAR	IP Packet Loss Ratio
IPTD	IP Transfer Delay
ITU	International Telecommunication Union
LBT	Load Balancing Tool
LC	Layered Coding
LP	Low Priority
LPN	Lost Podcasting Network
LS	Lip Synchronization
MD	Multimedia Descriptors
MDC	Multiple Description Coding
MDM	Meta Data Manager
MID	Multimedia Identifier
MP	Medium Priority
MPEG	Motion Pictures (Coding) Experts Group
MU	Multimedia Unit
MUSE	Multi Service Access Everywhere
NGN	Next Generation Network
P2P	Peer To Peer
PoB	Playout Buffer
PSNR	Peak Signal Noise Ratio
QoS	Quality of Service
RFC	Request For Comments
RGW	Residential Gateway
RR	Receiver Report

RTCP	Real-Time Transport Control Protocol
RTP	Real-Time Transport Protocol
RTSP	Real Time Streaming Protocol
RTSPD	RTSP Server
S.C.	Servidor de Control
S.V.	Servidor de Vídeo
SDES	Source Description Items
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SNR	Signal Noise Ratio
SR	Sender Report
STB	Set-Top Box
TCP	Transmission Control Protocol
UA	User Agent
UAC	User Agent Client
UAS	User Agent Servidor
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VBR	Variable Bit Rate
VoD	Vídeo on Demand

BIBLIOGRAFÍA

[1] Rahul Banerjee - draft-banerjee-rtp-vod-00.txt: “An extension of RTP and RTCP protocols For Video-on-Demand systems” (2002)

[2] VideoLAN - VLC media player [http://www.videolan.org/]

[3] Jaime Garcia, Francisco Valera, Ivain Vidal, Arturo Azcorra: ‘‘A broadcasting enabled Residential Gateway for Next Generation Networks” (2006)

[4] Isaac Seoane Pujol, David Larrabeiti López – Universidad Carlos III de Madrid: “Distribución de Video de Alta Calidad en Redes Multicamino”

[5] Xiao Su and Benjamin W. Wah - IEEE Transactions On Multimedia, Vol. 3, No. 1, March 2001 “Multidescription Video Streaming with Optimized Reconstruction-Based DCT and Neural-Network Compensations”

[6] J. Chakareski, S. Han and B. Girod - Stanford University: “Layered Coding vs. Multiple Descriptions for Video Streaming over Multiple Paths” (2003)

[7] P. Seeling and M. Reisslein – Arizona State University: “Video Coding Multiple Descriptors and Spatial for Device Diversity in Wireless Multi-hop Networks” (2004)

[8] Gabriella Olmo - Politecnico di TORINO: “Multiple Description Coding of visual” (2006)

[9] SIP: Session Initiation Protocol (RFC 3261)

[10]SDP: Session Description Protocol (RFC 4566)

[11]Session Description Protocol (SDP) Offer/Answer Examples (RFC 4317)

[12]RTP: A Transport Protocol for Real-Time Applications (RFC 3550)

[13]RTP Profile for Audio and Video Conferences with Minimal Control (RFC 3551)

[14]Session Initiation Protocol (SIP) Extension for Instant Messaging (RFC 3428)

[15]Session Initiation Protocol (SIP) Basic Call Flow Examples (RFC 3665)

[16]Real Integration of Resource Management (RFC 3312)

[17]I. Vidal, J. Garcia, F. Valera, I. Soto, A. Azcorra: “Integration of a QoS aware end user network within the TISPAN NGN solutions “ (2007)

[18]Iván Vidal, Jaime García, Francisco Valera, Ignacio Soto, Arturo Azcorra: ”Adaptive Quality of Service Management for Next Generation Residential Gateways ” (2006)

[19]VITALI Andrea L., BORNEO Antonio, FUMAGALLI Marco, RINALDO Roberto: “Video over IP using standard-compatible multiple description coding: an IETF proposal” (2006)

[20]A. Vitali, M. Fumagalli - draft-vitali-ietf-avt-mdc-lc-00.txt: “Standard-compatible Multiple-Description Coding (MDC) and Layered Coding (LC) of Audio/Video Streams” (2005)

[21]S. Wenger - draft-schierl-mmusic-layered-codec-04: “Signaling media decoding dependency in Session Description Protocol (SDP)” (2007)

[22]Grouping of Media Lines in the Session Description Protocol (SDP) (RFC 3388)

[23]The Session Initiation Protocol (SIP) UPDATE Method (RFC 3311)

[24]José Luis Cantarero Rodríguez: “Plataforma Genérica de Provisión de Servicios en Redes con Plano ce Control IMS” (2007)

[25]Netbeans IDE [http://www.netbeans.org/]

[26] JAIN-SIP 1.2 API [http://snad.ncsl.nist.gov/proj/iptel/jain-sip-1.2/javadoc/]

[27] WANEM: The Wide Area Network Emulator [http://wanem.sourceforge.net/]

[28] NetEm: Network Emulation [http://www.linux-foundation.org/en/Net:Netem]

[29]ITU-T: “Recommendation Y.1541: Network Performance Objectives for IP-based services” (2002)

[30]B. Steyaert, K. Laevens, D. De Vleeschauwer and H. Bruneel: “Analysis and design of a playout buffer for VBR streaming video” (2008)

[31]Nikolaos Laoutaris and Ioannis Stavrakakis - University of Athens: “Intrastream Synchronization for Continuous Media Streams: A Survey of Playout Schedulers” (2002)

[32]Xiph.Org Test Media, a repository for freely-redistributable test sequences [http://media.xiph.org/video/derf/]

[33]Wireshark [http://www.wireshark.org/]

ANEXO A
MANUAL DE LA APLICACIÓN

La finalidad de este anexo es establecer los pasos que se deben dar en el cliente y en los diferentes servidores para ejecutar la aplicación. También se recogen los diferentes mensajes de error que pueden aparecer en el cliente y cuál es su significado.

A.1. Ejecución de la aplicación

- **Servidor de Control (S.C.) y Servidor de Vídeo (S.V.)**

I. Seleccionar la conexión IP de SIP.
II. Introducir el puerto SIP.
III. Pinchar en "Start" para comenzar la escucha en el servidor.

- **Cliente**

Ventana "SIP LEVEL (Step 1 of 3)":

I.I. Seleccionar la película deseada.
I.II. Introducir el nombre del cliente.
I.III. Seleccionar la conexión IP de SIP.
I.IV. Introducir el puerto SIP.
I.V. Introducir la dirección IP de SIP del S.C.
I.VI. Introducir el puerto SIP del S.C.
I.VII. Pinchar en "Start".
I.VIII. Pinchar "Aceptar" en el mensaje informativo que "Listening on…".
I.IX. Pinchar en "Next >".

Ventana "SDP LEVEL (Step 2 of 3)":

II.I. Introducir la dirección IP de RTP.
II.II. Introducir el puerto RTP.
II.III. Elegir el sentido del flujo RTP en el cliente (ninguno, send, receive o send/receive).
II.IV. Elegir exigencias de calidad servicio (ninguna, en el cliente, en el S.V. o en ambos).
II.V. Seleccionar tipo de dependencia del flujo multimedia (sin dependencia, mdc o lc).
II.VI. Si hemos seleccionado "no": Seleccionador el códec multimedia y pinchar "Add".
Si hemos seleccionado "mdc": Marcar el número de descriptores deseados.

Si hemos seleccionado “lc”: Marca el número de descriptores deseados y seleccionar la dependencia de cada descriptor.

II.VII. Pinchar en “Create”.

II.VIII. Comprobar el campo SDP generado, si es correcto pinchar en “SEND”.

Ventana “SEND & MODIFY LEVEL (Step 3 of 3)”:

III.I. Si hemos seleccionado “mdc: Modificar el número de descriptores deseados.

III.II. Si hemos seleccionado “lc: Modificar el número de descriptores deseados o la dependencia de al menos uno de ellos.

III.III. Pinchar en “Modify” y comprobar si se han efectuado los cambios.

A.2. Mensajes de error

En la aplicación del cliente pueden saltar diferentes mensajes de error (figura 55).

Figura 55: Ejemplo de mensajes de error

A continuación se listan los mensajes de error en función del código de error:

A.1. *Client not listening:* Al pulsar “Start” sin tener la IP del cliente seleccionada.

A.2. *Port must be numeric:* Al pulsar “Start” sin haber introducido el puerto del cliente o haber introducido un puerto inválido.

A.3. *Incomplete server name:* Al pulsar “Start” sin tener el fichero de vídeo (que será el nombre del S.C. en la dirección SIP) seleccionado.

A.4. *Incomplete S.C. host:* No haber introducido o haber introducido mal la dirección IP del S.C.

A.5. *Port must be numeric:* Al pulsar “Start” sin haber introducido el puerto del S.C. o haber introducido un puerto inválido.

A.6. *Wrong route address:* Al pulsar “Route” sin tener la dirección correcta.

A.7. *Client is not listening:* Al pulsar “Route” sin haber pulsado antes “Start”.

B.1. *Incomplete or wrong address:* Al pulsar “Create” la dir IP de RTP no es correcta.

B.2. *Incomplete or wrong port:* Al pulsar “Create” el puerto RTP no es correcto.

B.3. *Type of depend isn´t selected:* Al pulsar “Create” no se ha seleccionado el tipo de dependencia.

B.4. *No nº media stream:* Al pulsar “Create” con dependencia mdc o lc no está seleccionado el número de descriptores.

B.5. *Codec list empty:* Al pulsar “Create” con dependencia “no” no se ha introducido ningún eleccionado un códec.

C.1. *You can´t modify:* Al pulsar “Modify” no hay nuevos parámetros.

C.2. *Session isn´t established ori s finished:* Al pulsar “Modify” la sesión no se ha establecido o está terminada, por lo que no se permite la modificación de la sesión.

ANEXO B
MANUAL DE PRUEBAS

En este anexo se detallan los pasos que se han dado para configurar el escenario de las simulaciones y como se ha realizado el procesamiento de los datos. Pretende ser un manual de ayuda para pruebas posteriores que se realicen.

El manual se divide en dos partes: la configuración de los tres equipos que forman el escenario de la figura 39 y la explicación de las fases de procesado de datos. En los siguientes apartados hay que tener en cuenta que los parámetros de cada equipo son:

Equipos	Dirección IP
Servidor de Vídeo	163.117.140.35
Cliente	163.117.140.96
PC WANem	163.117.140.53
Switch (Gateway)	163.117.140.2

Tabla 9: Parámetros de la red emulada

B.1. Configuración

- **Red Emulada**

La distribución de WANem es en forma de disco de arranque, no requiere instalación. El ordenador arrancará en el SO Knoppix Linux. Los pasos a seguir son:

1. Aparece una pantalla de Knoppix con una aviso en la parte inferior izquierda que indica “boot:”. Pulsar “Enter”.
2. Después de que termine de arrancar Knoppix, nos pregunta si queremos configurar los interfaces de red usando dhcp. Pulsaremos “n” porque introduciremos manualmente la dirección IP de la Red Emulada.
3. Accedemos a la pantalla de configuración de la red. Debemos seleccionar la interfaz Ethernet deseada (ej. eth0) e introducir posteriormente la dirección IP, la máscara de red y la dirección de la puerta de enlace de la Red Emulada. Salvamos pulsando “y” y salimos pulsando “s”. (figura 56)
4. WANem nos pedirá que introduzcamos la contraseña para el usuario “perc”. Usando este identificador de usuario y esta contraseña podemos accede remotamente al PC WANem con programas como putty o ssh.

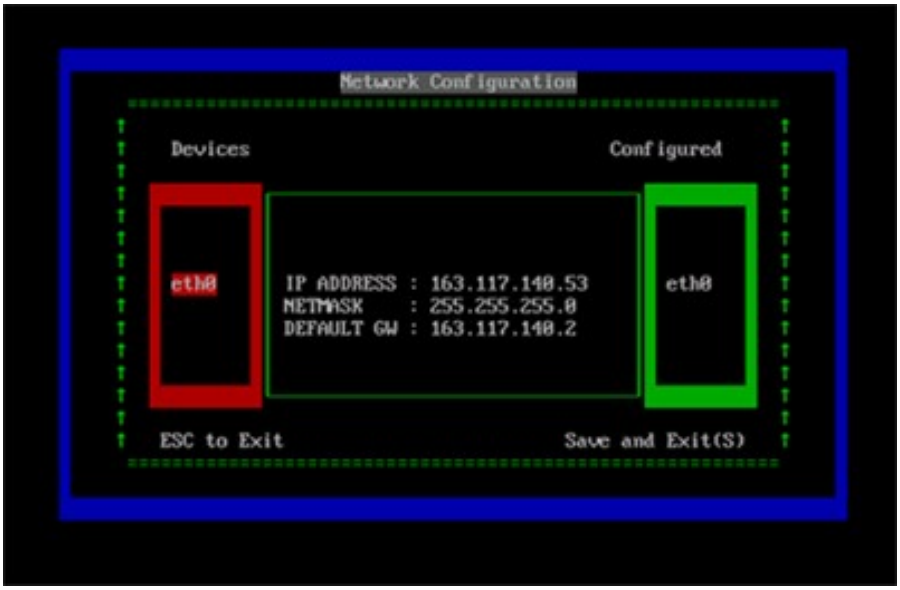

Figura 56: Pantalla de configuración de la red

La configuración básica de la Red Emulada termina en este punto. Ahora desde otro ordenador (ej. en el S.V.) accedemos al PC WANem a través de un navegador Web introduciendo la URL http://163.117.140.53/WANem. En la interfaz gráfica de usuario de WANem a la que accedemos aparecen dos modos de configuración: "Basic Mode" y "Advanced Mode". Pulsaremos en "Advanced Mode" donde podemos controlar más parámetros de la simulación.

La pantalla que aparece (figura 57) nos permite configurar los parámetros de la simulación para cada flujo de vídeo en función del puerto UDP destino. De esta forma emulamos varios caminos en la red con diferentes propiedades de retardos y de probabilidad de pérdidas. Para cada flujo de vídeo debemos introducir los parámetros, guardarlos pinchando en "Apply settings" y pinchando en "Add a rule set" para añadir los parámetros de un nuevo flujo.

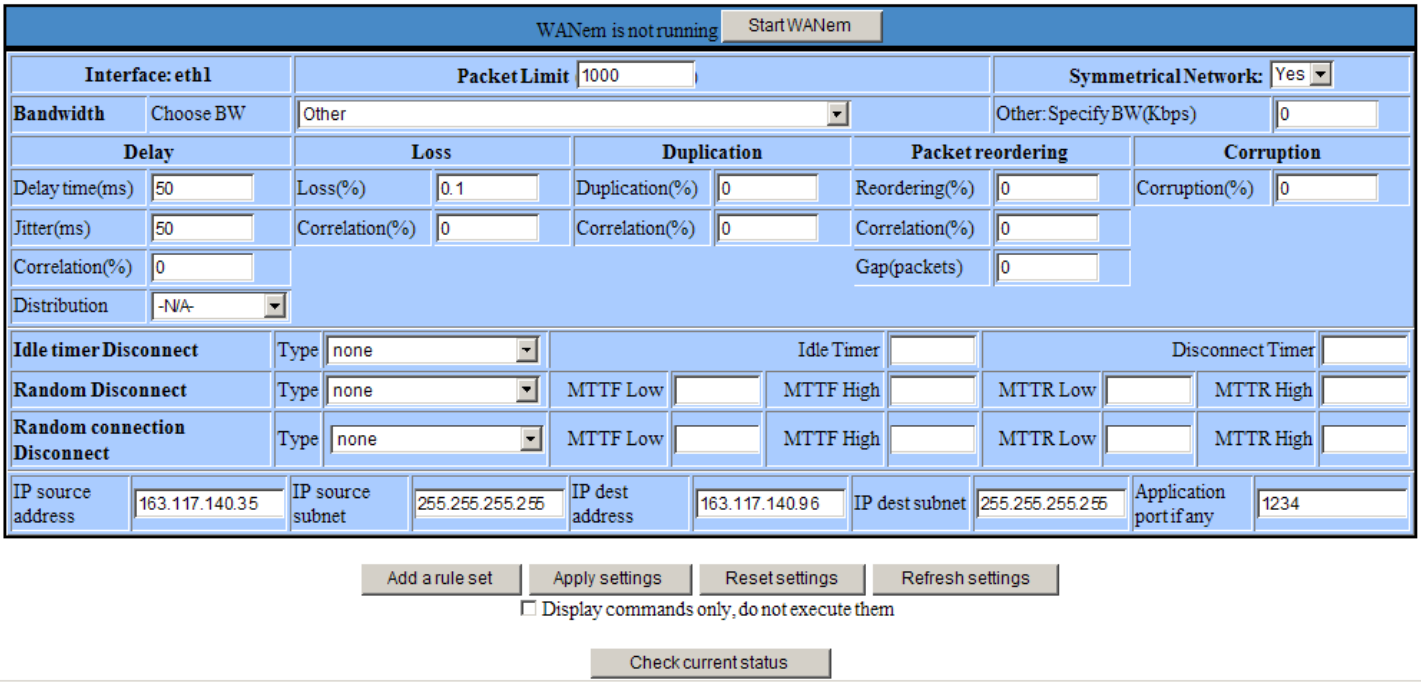

Figura 57: Pantalla de configuración de parámetros de la simulación

Si queremos tener propiedades de red Low Latency con IPTD < 100ms, IPDV < 50ms e IPLR < 10^{-3}, en el primer stream (1234), los parámetros que deberíamos introducir son:

Delay time (ms)	50
Jitter (ms)	50
Loss (%)	0.1
IP source address	163.117.140.35
IP source subnet	255.255.255.255
IP dest address	163.117.140.96
IP dest subnet	255.255.255.255
Application port if any	1234

Tabla 10: Parámetros de una red Low Latency

- **Servidor de Vídeo (S.V.)**

En el S.V. hay que cambiar la tabla de rutas para redirigir los paquetes del cliente al PC WANem. Para ello ejecutamos el comando:

Windows: *route add 163.117.140.96 mask 255.255.255.255 163.117.140.53*

Linux: *route add –host 163.117.140.96 netmask 0.0.0.0 gw 163.117.140.53*

Para comprobar que se ha introducido bien la entrada en la tabla de rutas ejecutamos *route print*.

- **Cliente**

En esta escenario de simulación no es necesario configurar el cliente, solo recibe el flujo de vídeo, por lo que no tiene que cambiar su tabla de rutas.

B.2. Captura de Datos

En esta fase solo intervienen el cliente y el S.V.

- **Cliente**

El cliente debe lanzar cuatro procesos de captura de VLC y arrancar el Wireshark. Para facilitar las cosas se ha creado un fichero ejecutable .bat para lanzar los procesos VLC.

```
c:
cd "C:\Archivos de programa\VideoLAN\VLC"
start vlc udp://@:1234 exit
start vlc udp://@:1236 exit
start vlc udp://@:1238 exit
start vlc udp://@:1240 exit
```

Antes de empezar a capturar tramas con Wireshark vamos a aplicar un filtro para quedarnos con las tramas que nos interesan. En la etiqueta "Filter:" introducimos *"frame.protocols contains "udp" and ip.dst == 163.117.140.96 and ip.src == 163.117.140.35 and eth.src == 00:0d:9d:59:18:24(WANem PC)"* y pinchamos Apply. Ahora si nos vamos a Capture>Interfaces… y pinchamos en "Start" sobre el interfaz deseado.

- **Servidor de Vídeo (S.V.)**

Una vez realizados los anteriores pasos en el cliente, al S.V. solo le queda lanzar los diferentes flujos de vídeo. Cómo se explicó en el capítulo 5 de Validación de la Propuesta, al igual que en el cliente se ha creado un fichero ejecutable .bat para lanzar estos flujos simultáneamente.

```
c:

cd "C:\Archivos de programa\VideoLAN\VLC"

start vlc "C:\Documents and Settings\video\video1.mpg" --
sout="#std{access=rtp,mux=ts,dst=163.117.140.96:1234}"

start vlc "C:\Documents and Settings\video\video2.mpg" --
sout="#std{access=rtp,mux=ts,dst=163.117.140.96:1236}"

start vlc "C:\Documents and Settings\video\video3.mpg" --
sout="#std{access=rtp,mux=ts,dst=163.117.140.96:1238}"

start vlc "C:\Documents and Settings\video\video4.mpg" --
sout="#std{access=rtp,mux=ts,dst=163.117.140.96:1240}"
```

B.3. Procesado de Datos

La fase de procesado solo se realiza en el cliente partiendo de las tramas capturadas con Wireshark. Los pasos que hay que dar con Wireshark son:

1. Parar la captura una vez haya finalizado el streaming de vídeo pinchando en Capture>Stop.
2. Como las tramas ya han sido filtradas previamente ahora hay que decodificarlas para que Wireshark las identifique como tramas RTP. Esto lo tendremos que hacer para los diferentes flujos de vídeo.
   Por ejemplo, seleccionamos una trama con puerto UDP destino 1234, pinchamos con el botón derecho sobre ella y marcamos “Decode As”. En la nueva ventana que aparece marcamos en la lista de la izquierda “UDP destination (1234) port(s) as”, de esta forma generalizamos esta regla para próximas pruebas ya que el puerto origen varia. En la lista de la derecha seleccionamos RTP y pinchamos “OK”. Este paso lo deberemos realizar con el resto de los flujos.
3. Si pinchamos en Statistics>RTP>Show All Streams, tendremos un resumen (figura 58) sobre la información de cada stream: número de paquetes, porcentaje de pérdidas, máxima delta, máximo jitter y jitter medio.

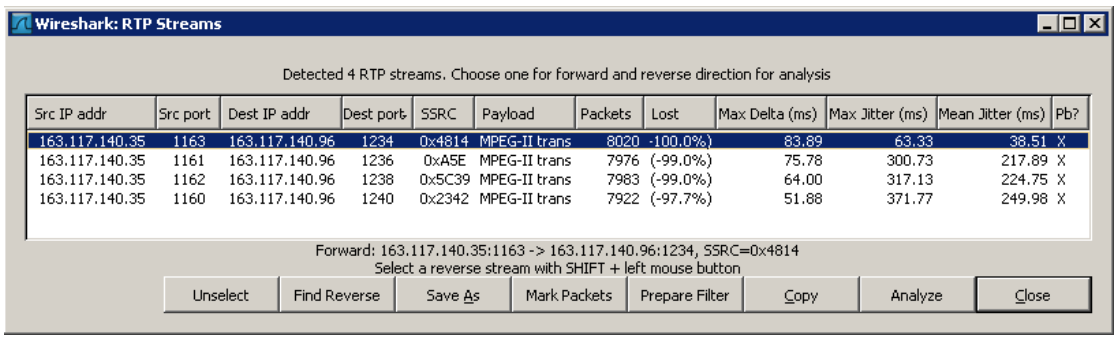

Figura 58: Pantalla de Show All Streams

4. Finalmente tenemos que almacenar los datos de las tramas RTP en un fichero que posteriormente se procesará en Matlab. Seleccionamos una trama de un determinado flujo (ej. Puerto destino UDP 1236) y pinchamos en Statistics>RTP>Stream Analysis. Nos aparecerá la pantalla de la figura 59 donde pincharemos en "Save as CSV…" para guardar la tabla de datos en un fichero con formato CSV. Buscamos la ruta destino del fichero, escribimos su nombre (ej. 1236.txt) y pinchamos en "OK". Este paso se repite con cada stream hasta tener cuatro ficheros de texto: 1234.txt, 1236.txt, 1238.txt y 1240.txt.

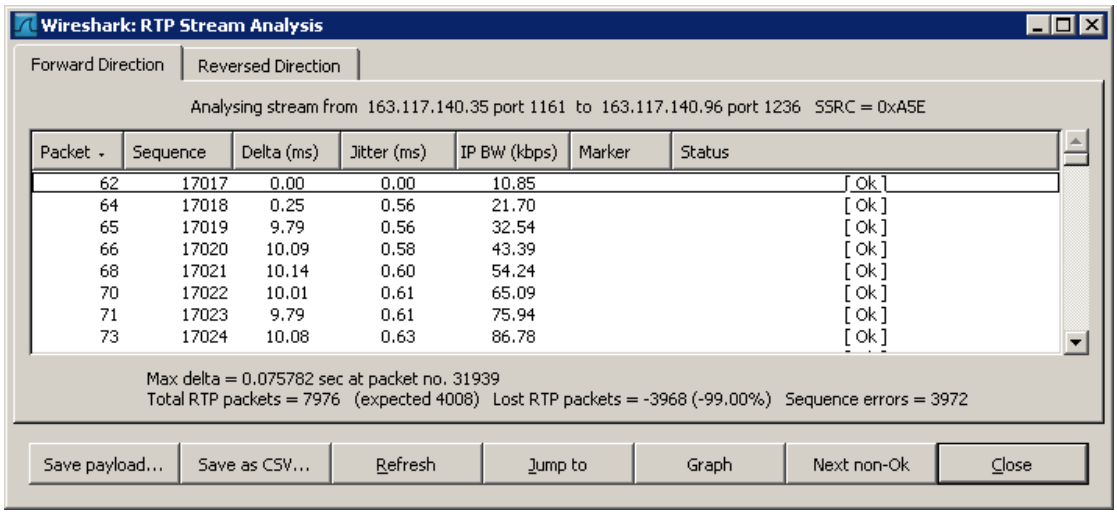

Figura 59: Pantalla de Stream Analysis

Partiendo de los ficheros .txt podemos utilizar el Excel para extraer los datos con formato CSV, para almacenarlos posteriormente en una variable de Matlab. Esta variable es una matriz de 6 columnas con la información sobre él nº de paquete, el nº de secuencia, la delta, el jitter, la tasa acumulada y el tiempo de llegada de cada paquete. Tendremos una variable por cada stream de vídeo recibido.

Una vez tenemos esta variable almacenada en Matlab, hay que ejecutar el fichero .m "simulate.m" que permite extraer la información sobre el número de paquetes buenos y malos de cada stream, y el número de streams que se tiene a lo largo de la transmisión.

En el CD ajunto a este proyecto se incluye el fichero "simulate.m" y "processStream.m" dentro de la carpeta Matlab.