ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

PROYECTO FIN DE CARRERA

DISEÑO E IMPLEMENTACIÓN DE UN SERVICIO DE VIDEO STREAMING USANDO JAVA Y CORBA.

AUTOR: Pedro Zamora Prades
DIRECTORES: Felipe García Sánchez
Antonio Javier García Sánchez

Autor	Pedro Zamora Prades
E-mail	pedrozapra@hotmail.com
Director	Felipe García Sanchez
E-mail	felipe.garcia@upct.es

Titulo del PFC	DISEÑO E IMPLEMENTACIÓN DE UN SERVICIO DE VIDEO STREAMING USANDO JAVA Y CORBA

Resumen

Este proyecto Fin de Carrera describe el diseño y la implementación de un servicio de transmisión de video. Dicho diseño se basa en la especificación A/V Streaming. Dicha especificación define una arquitectura y unos interfaces que permiten transmitir un flujo multimedia que pueda ser utilizado en aplicaciones distribuidas. Para la implementación de la especificación se ha utilizado el lenguaje de programación Java y CORBA.

Titulación	Ingeniería Técnica de Telecomunicaciones, especialidad Telemática
Departamento	Tecnologías de la Información y Comunicaciones
Fecha de Presentación	22/02/08

ÍNDICE

CAPÍTULO 1
INTRODUCCIÓN

Este Proyecto desarrolla e implementa una aplicación basada en la especificación del servicio A/V Streaming mediante la utilización de Java y CORBA. Estos conceptos se desarrollarán en profundidad los aspectos de este servicio.

El servicio de A/V Streaming es una tecnología que permite la transmisión y la recepción de video y audio a través de internet, o cualquier red de telecomunicaciones. En este proyecto sólo se centra en la transmisión y recepción de vídeo, dejando la transmisión y recepción de audio para trabajos posteriores

La transmisión de video a través de las redes de telecomunicaciones puede efectuarse con calidades de transmisión máximas gracias al aumento del ancho de banda que hemos alcanzado hoy en día. Como ya hemos comentado anteriormente, en este proyecto se ha tratado únicamente el envío y recepción de vídeo sin compresión, mediante los protocolos RTP y UDP. La transmisión de vídeo sin compresión es hoy en día muy utilizado en muchos ámbitos, como por ejemplo, utilizado para hacer diagnósticos en medicina, hacer videoconferencias y también cada vez mas utilizado en el ámbito militar. Todo esto unido al aumento de las posibilidades de comunicación hace que se extienda cada día más el uso de este tipo de servicios a usos más cotidianos.

En la actualidad existen diversas implementaciones del servicio A/V Streaming que se incluyen en distintas distribuciones tales como las de ACE-TAO y Orbit. Sin embargo, entre ellas no se encuentra ninguna que cumpla con dos requisitos fundamentales:

- Que sea programado con el API de Java “nativo” de Sun.
- Que sea de código Java abierto y de libre configuración.

Por ello, la principal constribución de este trabajo es el desarrollo de un servicio de A/V Streaming abierto (código libre), que utilice directamente el API de Java-Sun, y que sea equivalente al de otras distribuciones de CORBA.

1.1.- Objetivos

Los objetivos que se pretenden alcanzar mediante la realización de este proyecto fin de carrera son:

Obj1.- Cumplir los requisitos de la especificación A/V Streaming

Obj2.- Transmitir video en tiempo real

Obj3.- Desarrollar una capa middleware que permita tanto al cliente como al servidor abstraerse de la funcionalidad de la aplicación.

Obj4.- Poder hacer la transmisión de vídeo usando varios protocolos como por ejemplo UDP , RTP etc… y usando varios formatos de transmisión como por ejemplo RGB, H.263, JPEG etc…

1.2.- Requisitos

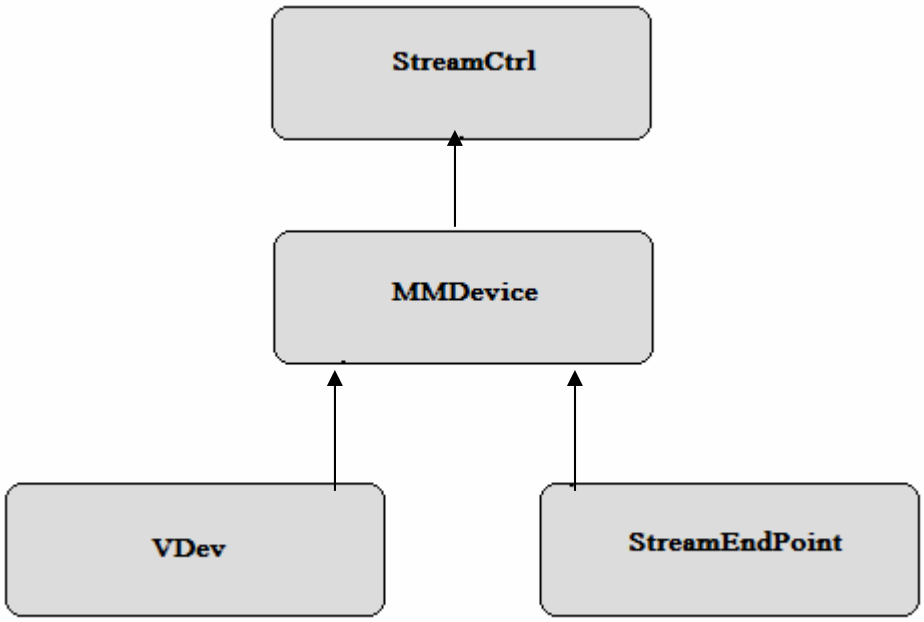
Los requisitos mínimos que deben cumplirse para que este proyecto sea lo más fiel posible a la especificación A/V Streaming son los siguientes:

Req1.- Poder utilizar distintos medios de transmisión por los distintos streams que se puedan crear, como es una red de fibra óptica.

Req2.- Permitir la incorporación de protocolos para la transmisión de los datos multimedia, de forma que con pequeñas modificaciones, se pueden añadir nuevos modos de transmisión y recepción de imágenes a nuestra implementación. Un ejemplo sería la posibilidad de usar RTP de Java con los distintos formatos de compresión de imágenes, o usar un sistema de transmisión sobre UDP enviando imágenes sin ningún tipo de compresión y en paquetes de un determinado tamaño.
En este caso se va a disponer la transmisión de imágenes RGB,JPEG y H.263 sobre RTP y RGB sobre UDP.

Req3.- El usuario cliente deberá poder controlar en todo momento el flujo de datos multimedia. Además deberá ser capaz de conectarse a un servidor, inicializar la visualización de imágenes, pausar , reanudar y parar la transmisión y recepción de vídeo.

Para que todos estos requisitos puedan cumplirse, se ha elaborado este proyecto respetando la especificación A/V Streaming. Dicha especificación define los requisitos anteriormente mencionados y además una arquitectura para poder desarrollar el servicio.



Según la especificación la funcionalidad de cada componente es la siguiente:

- La clase StreamCtrl abstrae la transferencia multimedia entre dos dispositivos virtuales (VDev). Es la encargada de enlazar el MMDevice del cliente y el MMDevice del servidor y utilizar un unico flujo de datos. Este objeto generalmente es creado por el MMDevice del servidor y utilizado por la clase principal del cliente.
- La clase MMDevice representa una factoría de objetos. Es la encargada de crear los objetos VDev y MMdevice. Estas referencias pueden obtenerse tanto de forma local como remota.
- Los StreamEndPoint permiten que el cliente pueda indicarle al servidor quién son los extremos de la conexión.
- Los VDev realizan la configuración de la reproducción multimedia. Mediante los VDev se puede configurar el formato de la transmisión y negociar los parámetros necesarios para la reproducción.

CAPÍTULO 2
TECNOLOGÍA

2.1.- Lenguajes de programación

A lo largo de este apartado se van a explicar los lenguajes de programación empleados en el desarrollo de este proyecto fin de carrera.

2.1.1.- Java

2.1.1.a.- Introducción

El lenguaje de programación empleado para el desarrollo del servicio A/V Streaming es Java, cuyas características se exponen a continuación.

Java es una plataforma de software desarrollada por Sun microsystem, de tal forma que los programas creados en ella puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales.

La principal virtud de Java, es que se trata de un lenguaje de programación orientado a objetos, lo que permite al programador desarrollar aplicaciones de forma más sencilla e intuitiva.

Se puede dividir la plataforma Java en 3 partes diferenciadas:

- El lenguaje de programación.
- La maquina virtual de Java, JRE, que permite la portabilidad en ejecución.
- EL API Java, una biblioteca estándar para el lenguaje.

Originalmente llamado OAK por los ingenieros de Sun Micorsystem, Java fue diseñado para correr en computadoras incrustadas. Sin embargo, en 1995, dada la atención que estaba produciendo la Web, Sun Microsystem la distribuyó para sistemas operativos tales como Microsoft Windows.

El lenguaje mismo se inspira en la sintaxis de C++, pero su funcionamiento es más similar al de Smaltalk que a este. Incorpora sincronización y manejo de tareas en el lenguaje mismo(similar a Ada) e incorpora interfaces como mecanismo alternativo a la herencia múltiple de C++.

A finales del siglo XX, java llegó a ser el lenguaje de mayor acogida para programas de servidor. Utilizando una tecnología llamada JSP (similar a otras tecnologías del lado del servidor como ASP de Microsoft o PHP), se hizo muy fácil para escribir páginas dinámicas para sitios de Internet. Sumando a esto, la tecnología de JavaBeans, al incorporarse con JSP, permitía adaptar al mundo web el patrón MVC(modelo vista controlador) que ya se había aplicado con éxito a interfaces gráficas.

Java llegó a se extremadamente popular cuando Sun Microsystem introdujo la especificación J2EE (Java 2 Enterprise Edition). Este modelo permite, entre otros, una separación entre la presentación de los datos al usuario (JSP o Applets), el modelo de

datos (EJB), y el control (Servlets). Enterprise Java Beans (EJB) es una tecnología de objetos distribuidos que pudo lograr el sueño de muchas empresas como Microsoft e IBM de crear una plataforma de objetos distribuidos como un monitor de transaciones. Con este nuevo estándar, empresas como BEA, IBM, Sun Microsystem, Oracle y otros crearon nuevos “servidores de aplicaciones” que tuvieron gran acogida en el mercado.

Además de programas del servidor, Java permite escribir programas de interfaz gráfica o textual. También se pueden correr programas de manera incorporada o incrustada en los navegadores web de Internet en forma de Java applets, aunque no llegó a popularizarse como se espera en un principio.

Los programas en Java generalmente son compilados a un lenguaje intermedio llamado bytecode, que luego son interpretados por una maquina virtual(JVM). Esta última sirve como una plataforma de abstracción entre la máquina y el lenguaje permitiendo que se pueda “escribir el programa una vez, y correrlo en cualquier lado”. También existen compiladores nativos en Java, tanto software de libre distribución como no libre. El compilador GCC de GNU compila Java a código de maquina con algunas limitaciones al año 2002.

Con la evolución de las diferentes versiones, no sólo se han producido cambios muchos mas importantes en sus bibliotecas asociadas, que han `pasado de unos pocos cientos de en Java 1.0 , a más de tres mil en Java 5.0. En particular, se han añadido APIs completamente nuevas, tales como Swing y Java 2D.

2.1.1.b.- La Máquina virtual de Java.

La máquina virtual de java (en inglés Java Virtual Maquine, JVM) es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial(el bytecode), el cual es generado por el compilador de java.

El código binario de Java no es un lenguaje de alto nivel, sino un verdadero código máquina de bajo nivel, viable incluso como lenguaje de entrada para un microprocesador físico.

La gran ventaja de la maquina virtual java es aportar portabilidad al lenguaje de manera que desde SUN se han creado diferentes máquinas virtuales java para diferentes arquitecturas y así un programa .class escrito en Windows puede ser interpretado en un entorno Linux. Tan solo es necesario disponer de dicha máquina virtual para dichos entornos.

Los intentos de la compañía propietaria de Java y productos derivados, que es Sun Microsystem, de construir microprocesadores que aceptarán el bytecode como su lenguaje de máquina fueron más bien infructuosos.

Actualmente la mayoría de las plataformas de ejecución de Java son máquinas virtuales. La máquina virtual es un programa normal solamente ejecutable en un sistema operativo particular (por ejemplo, Windows) Esta JVM es la que le permite a Java ser multiplataforma debido a que cuando se compila el código java, este queda compilado

en bytecode que es lo que reconoce la JVM instalada en cualquier plataforma Windows, Linux o MacOS.

Existen varias versiones, en orden cronológico, de la máquina virtual de Java. En general la definición del Java bytecode no cambia significamente entre versiones, y si lo hacen , los desarrolladores del lenguaje procuran que exista compatibilidad hacia atrás con los productos anteriores.

2.1.1.c- API JAVA

EL API Java es una interfaz de programación de aplicaciones o API provista por los creadores del lenguaje Java, y que da a los programadores un ambiente de desarrollo completo así como una infraestructura.

Como el lenguaje Java es un lenguaje orientado a objetos, la API de java provee de un conjunto de clases utilitarias para efectuar toda clase de tareas necesarias dentro de un programa.

La API Java está organizada en paquetes, donde cada paquete contiene un conjunto de clases relacionadas semánticamente.

2.1.2.- CORBA

2.1.2.a.-Introducción

CORBA (*Common Object Request Broker Architecture*) , es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.

CORBA fue definido y está controlado por el Object Management Group (OMG) que define las APIs, el protocolo de comunicaciones y los mecanismos necesarios para permitir la interoperabilidad entre diferentes aplicaciones escritas en diferentes lenguajes y ejecutadas en diferentes plataformas, lo que es fundamental en computación distribuida.

En un sentido general CORBA envuelve el código escrito en otro lenguaje en un paquete que contiene información adicional sobre las capacidades del código que contiene, y sobre cómo llamar a sus métodos. Los objetos que resultan pueden entonces ser invocados desde otro programa (u objeto CORBA) desde la red. En este sentido CORBA se puede considerar como un formato de documentación legible por la máquina, similar a un archivo de cabeceras pero con más información.

Por tanto CORBA es una tecnología que oculta la programación a bajo nivel de aplicaciones distribuidas, de tal forma que el programador no se tiene que ocupar de tratar con sockets, flujos de datos, paquetes, sesiones etc. CORBA oculta todos estos detalles de bajo nivel. No obstante CORBA también brinda al programador una tecnología orientada a objetos, las funciones y los datos se agrupan en objetos, estos objetos pueden estar en diferentes máquinas, pero el programador accederá a ellos a través de funciones normales dentro de su programa.

2.1.2.-b.-Funcionamiento

En las aplicaciones CORBA, los métodos y los datos se agrupan formando lo que se denominan interfaces, los interfaces pueden ser interpretados como objetos que agrupan datos y métodos para acceder a estos. Todos estos interfaces se definen usando un lenguaje IDL (Interface Definition Lenguaje), que es precisamente esto, un lenguaje para la definición de interfaces. Este lenguaje es un estándar y lo soportan todas las implementaciones CORBA

Es algo más que una abstracción que oculta la complejidad de red,

- TODAS LAS LLAMADAS PARA EL PROGRAMADOR SON IGUALES
- SE DEFINEN OBJETOS Y MÉTODOS
- LOS OBJETOS SON REMOTOS
- NO PORQUE CORBA OCULTE COSAS DEBEMOS DEJAR DE

  PENSAR EN LA EFICIENCIA DE RED

Hay una multitud de sistemas con un propósito muy similar a CORBA circulando por el mundo, los más usados son: el RPC (Remote Procedure Call) de Sun Microsystems.

2.1.2.c.- Middleware

El middleware es una capa de intermediación que debe proporcionar de forma transparente a los programadores métodos eficaces que permita la construcción de aplicaciones distribuidas.

La base del middleware actual es un sistema de llamada a procedimiento remoto (LPR), que permite con sintaxis similar a la de la función local, invocar funciones que se ejecutan realmente en una máquina servidora.

Para lograr esta transparencia, cada vez que se llama a un procedimiento remoto, el ordenador cliente ejecuta en realidad un “delegado del cliente” o stub, cuya misión es contactar con el servidor, y solicitar a un “delegado en el servidor” o skeleton la ejecución del procedimiento. El delegado en el servidor llama a la implementación del procedimiento en el servidor, recoge los resultados, y los transfiere de regreso al delegado del cliente. Finalmente, el delegado del cliente devuelve el resultado al programa en ejecución. Un sistema de llamada a procedimiento remoto es tanto mejor, cuanto mas se parezca la síntaxis de la llamada local a la de una llamada remota equivalente.

2.1.2.d- Object Request Broker

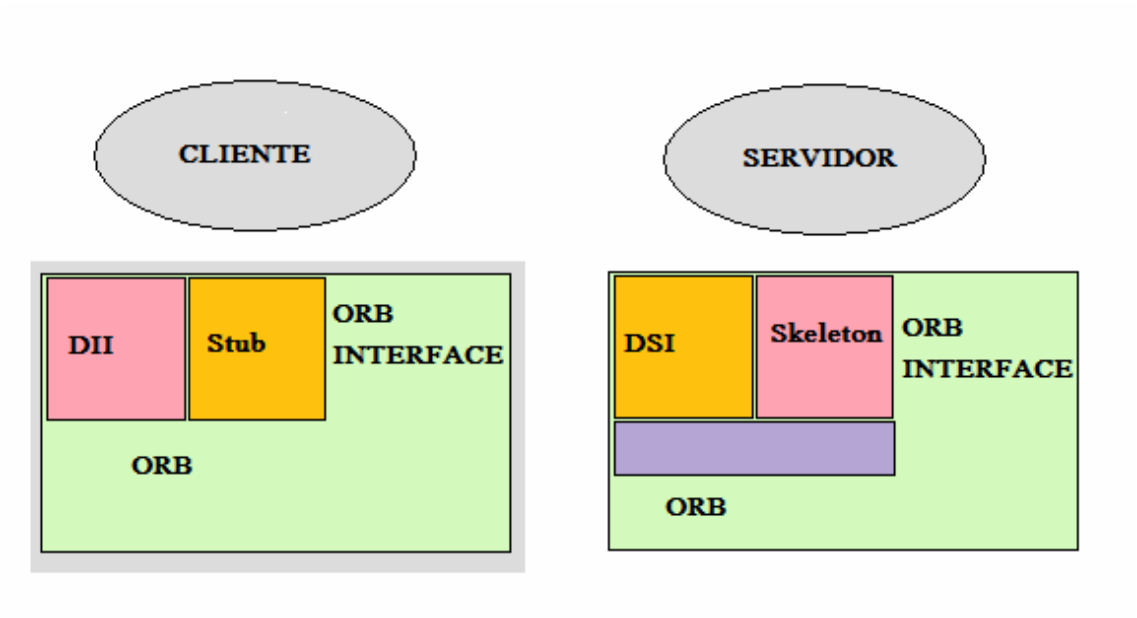
El centro de una red distribuida CORBA es el “Object Request Broker” u ORB. El ORB se encarga de empaquetar y desempaquetar los objetos entre el cliente y el servidor.

Otros servicios como Servicios de Nombres y Servicios de eventos funcionan con el ORB.

La plataforma java 2 incluye un ORB en la distribución llamado el IDL ORB. Este ORB es diferente de otros muchos ORBs porque no incluye el distintivo de "Basic Object Adapter" (BOA)

2.1.2.e – Esquema general de peticiones

Partiendo del siguiente esquema, que representa la arquitectura del middleware de CORBA, los pasos que se siguen al hacer una petición remota son:



1. - El cliente realiza una petición usando stubs estáticos o la interfaz de invocación dinámica DII y esta es dirigida a su ORB.
2. - El ORB del cliente transmite las peticiones al ORB enlazado con el servidor
3. - El ORB del redirige la petición al adaptador de objetos que ha creado el objeto destino.
4. - El adaptador de objetos dirige la petición al servidor que implementa el objeto destino.
5. - El Servidor devuelve su respuesta

2.1.2.f.- IDL

CORBA utiliza un lenguaje de definición de interfaces (IDL) para especificar los interfaces con los servicios que los objetos ofrecerán. CORBA puede especificar a partir de este IDL la interfaz a un lenguaje determinado, describiendo cómo los tipos de datos CORBA deben ser utilizados en las implementaciones del cliente y del servidor. Implementaciones estándar existen para Ada, C, C++, Smalltalk, Java y Python. Hay también implementaciones para Perl y TCL, aunque en nuestro caso la que nos importa es la de Java.

Al compilar una interfaz en IDL se genera código para el cliente y el servidor (el implementador del objeto). El código del cliente sirve para poder realizar las llamadas a métodos remotos. Es el conocido como stub, el cual incluye un proxy (representante) del objeto remoto en el lado del cliente. El código generado para el servidor consiste en unos skeletons (esqueletos) que el desarrollador tiene que rellenar para implementar los métodos del objeto.

CORBA es más que una especificación multiplataforma, también define servicios habitualmente necesarios como seguridad y transacciones. Y así este no es un sistema operativo en si, en realidad es un middleware.

En resumen IDL es un lenguaje de definición de interfaces, el cual proporciona un formato común para representar un objeto que puede ser distribuido a otras aplicaciones. Estas aplicaciones podrían incluso no entender de objetos, pero mientras puedan proporcionar un mapeado entre el formato común IDL y su propia representación de datos, la aplicación podrá compartir datos.

2.1.2.g.- Esquema de Mapeo IDL

Como ya se ha comentado anteriormente muchos lenguajes de programación proporcionan un mapeo entre sus tipos de datos y el formato común denominado IDL, y el lenguaje Java no es una excepción. El lenguaje Java puede enviar objetos definidos por IDL a otras aplicaciones distribuidas CORBA y recibir objetos definidos mediante IDL desde otras aplicaciones distribuidas CORBA.

El mapeo del lenguaje Java a IDL se sitúa en un fichero con extensión .idl. Este fichero es compilado para que pueda ser accedido por los programas CORBA que necesitan enviar y recibir datos.

- **Paquetes e Interfaces** Java : Las sentencias de paquete Java son equivalentes al tipo module de IDL. Este tipo puede ser anidado, lo que resulta en que las clases Java generadas se crean en subdirectorios anidados

- **Métodos** Java : Los métodos Java se mapean a operaciones IDL. Las operaciones IDL son similares a los métodos Java excepto en que no hay concepto de control de acceso. También tenemos que ayudar al compilador IDL especificando qué parámetros son de entrada in, de salida out o de entrada/salida inout definidos de la siguiente forma:

  In – El parámetro se pasa dentro del método pero no se modifica

  Out – El parámetro se podría devolver modificado

  Inout – El parámetro se pasa al método y se podría devolver modificado

- **Arrays** Java: Los arrays Java son mapeados a los tipos array o sequences IDL, usando una definición de tipo

- **Excepciones** Java: Las excepciones Java son mapeadas a excepciones IDL. Las operaciones usan exceptions IDL, especificándolas como del tipo raises.

- **Struct** IDL: Un tipo struct IDL puede ser comparado con una clase Java que sólo tiene campos, que es como mapea el compilador IDL

2.1.2.h.- Compilar el Fichero de Mapeos IDL.

Un fichero IDL tiene que ser convertido en clases Java que puedan ser usadas en una red distribuida CORBA. La plataforma Java 2 compila los ficheros .idl usando el programa idltojava. Este programa será reemplazado eventualmente por el comando idlj.

idlj –fall (ARCHIVOS IDL)

Al compilar el archivo .idl generará por cada interfaz idl las siguientes :

- La clase Holder
- La clase Helper
- Operations
- _Stub
- .java
- POA
- POATie

2.1.2.i.- Stubs y skeletons

La compilación de los idl genera un fichero stub para el cliente y un fichero skeleton para el servidor. El stub, y el skeleton se usan para envolver y desenvolver datos entre el cliente y el servidor. El skeleton está implementado mediante el sevidor.

2.1.2.j.- Las clases Helper y Holder

La clase Holder: La idea de estas clases es que puedan resolver el problema de paso de parámetros de salida(out) y entrada salida(in/out). Por ello al compilar un archivo .idl, por cada tipo, se define una clase Holder.

Para acceder al valor lo hacemos a través del campo value.

La clase Helper: Esta clase permite operar con las propiedades de cualquier tipo IDL. Estas propiedades se pueden sacar e introducir mediante el tipo de datos any.

- void insert(org.omg.CORBA.Any any, Tipo t);

- Tipo extract(org.omg.CORBA.Any any);

- Tipo narrow(omg.org.CORBA.Object obj);

- TypeCode type();

- String id();

Esta clase al igual que con la clase Holder es creada por el compilador.

2.2- Especificación A/V Streaming

En este apartado se va a detallar el funcionamiento básico de la especificación asi como su iteración y las premisas que cumplen cada bloque

2.2.1.-Introducción

Streaming es un termino que describe una estrategia sobre demanda para la distribución de contenido multimedia a través de Internet.

La especificación de A/V Streaming, define una arquitectura y unos interfaces que son capaces de obtener un servicio de Streaming que pueda ser utilizado en aplicaciones distribuidas.

2.2.2.- Principales componentes

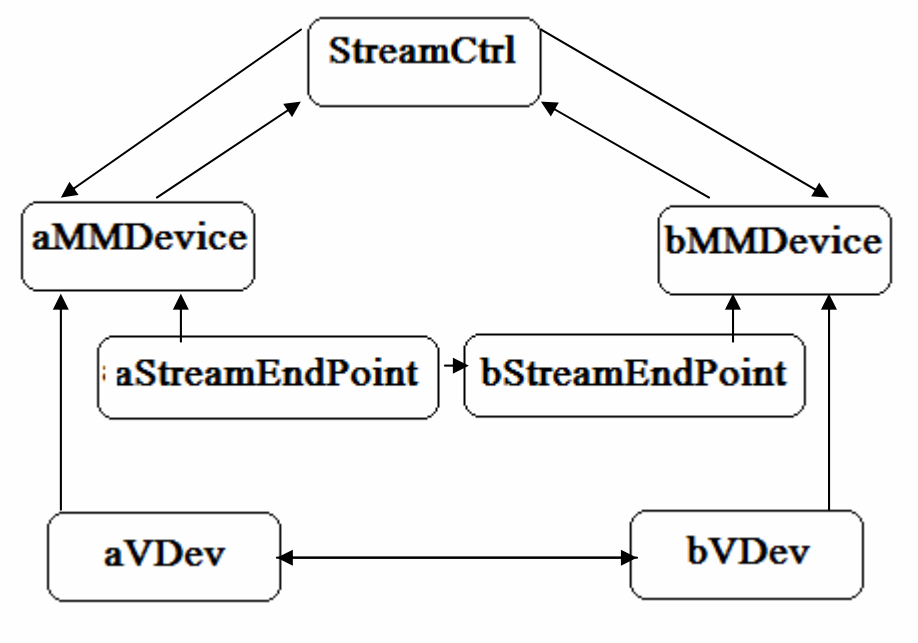
Este apartado se van explicar los principales interfaces que propone la especificación

- Dispositivos virtuales multimedia y dispositivos multimedia representados por VDev y MMDevice respectivamente.
- Streams, representados por el StreamCtrl. Un stream representa una transferencia multimedia continua, generalmente entre dos o mas dispositivos multimedia
- A simple stream between a microphone device.
- Streams endpoints, representados por el interfaz StreamEndPoint.
- Flow Devices - representado por el interfaz FDev

Un stream representa transferencia continua multimedia generalemente entre dos o más dispositivos multimedia

2.2.3.a.- Establecimiento de la conexión

Este proceso se encarga se asociar dos extremos que necesitan comunicarse por medio de streams. A/V Streaming define el mecanismo necesario para conseguir dicha comunicación. Para explicar el funcionamiento, se va a utilizar la siguiente figura:



Supóngase dos extremos “a” y ”b”. En un momenta dado, “a” desea obtener una comunicación multimedia con “b”. A continuación se va a detallar el proceso de establecimiento de la conexión.

a.- StreamCtrl_impl asocia dos MMDevice. La aplicación se encarga de ejecutar en StreamCtrl la operación de asociar los dos MMDevice. El objetivo de esta operación es obtener las referencias a dichos objetos. A partir de estas referencias, se puede acceder a las factorías de objetos de los MMDevice.

b.- Creación de los StreamEndPoint. En esta paso StreamCtrl_impl solicita que aMMDevice y bMMDevice creen los StreamEndPoint y Vdev correspondientes a cada MMDevice. Una vez creados, StreamCtrl_impl obtiene las referencias a dichos objetos a través de los MMDevice.

c.- Configuración de VDev. Los dos extremos se ponen de acuerdo respecto al tipo de transmisión de datos multimedia. Dicha configuración la realizan los VDev obtenidos anteriormente.

d.- Establecimiento de los StreamEndPoint. El extremo “a” se encarga de indicar a los StreamEndPoint obtenidos, cuales son los extremos de la comunicación, es decir, indica por qué puertos y que ips son las que intervienen en la comunicación.

e.- Establecimiento de la reproducción. Una vez configurado todo, los extremos de la conexión tienen todos los parámetros necesarios para comenzar la transmisión de datos multimedia.

2.2.3.b.- Control de la conexión

Una vez configurada la conexión , es cuando entra en juego el control de esta. El control de la conexión consiste en un juego de funciones las cuales se encargan de controlar en todo momento el flujo de datos. Dichas funciones se definen en el interfaz remoto, y el extremo “a” accede a ellas de forma remota, actuando en consecuencia el extremo “b”. Dichas funciones deben ser capaces al menos de iniciar un flujo de datos multimedia, o de pararlo. Es función del programador el diseño de distintas funciones para el control de las comunicaciones.

CAPITULO 3

ARQUITECTURA DE UN SERVICIO DE A/V STREAMING EN JAVA/CORBA

En este capitulo se explica como se ha diseñado el servicio de A/V Streaming utilizando como lenguaje de programación Java y como plataforma middleware CORBA. También se van a explicar las características, asi como las funciones principales de dicho servicio. Y además se comenta el funcionamiento de la aplicación final.

3.1.- Características principales

Este servicio de A/V streaming esta compuesto por dos partes fundamentales: por un lado el cliente y por el otro el servidor. A lo largo de todo este apartado se explicará como se han diseñado.

SERVIDOR:

El servidor se inicia desde la clase ServidorCorba. Es el encargado de coordinar tanto los parámetros de configuración de la reproducción multimedia así como de controlar la transmisión de los datos multimedia hacia el cliente. Tanto el control como la configuración de parámetros, se realiza a través de llamadas Corba. Dichas llamadas se reciben por el puerto 1099. Es el cliente el que realiza dichas llamada y el servidor el que actúa en consecuencia.

CLIENTE:

El cliente se inicia desde la clase CPrincipal y se encarga de indicarle al servidor los parámetros de reproducción seleccionados por el usuario. Es el lado del cliente donde el usuario selecciona todos los parámetros de configuración. También se encarga de controlar la recepción de los datos multimedia y de mostrar las imágenes que recibe.

Una vez definidas las funciones del cliente y del servidor, se indican las posibilidades de reproducción que puede seleccionar el usuario:

El usuario podrá elegir entre varios formatos de reproducción:

1. RGB sobre UDP. Este formato envía video bruto por la red utilizando el protocolo de transporte UDP. Se transmite imágenes de 230400 bytes sin compresión. Dicho protocolo no está orientado a la conexión, por lo que no proporciona ningún tipo de control de errores

ni de flujo, aunque si se utilizan mecanismos de detección de errores. Cuando se detecta un error en un datagrama, en lugar de entregarlo a la aplicación, se descarte. Como el protocolo no está orientado a la conexión cada datagrama UDP existe independientemente del resto de los datagramas UDP. El protocolo UDP es muy sencillo y es muy útil para las aplicaciones que requieren pocos retardos o para ser utilizado en sistemas sencillos que no pueden implementar el protocolo TCP. Las características del protocolo UDP son:

a.- No garantiza la fiabilidad. No podemos asegurar que cada datagrama UDP transmitido llegue a su destino. Es un protocolo del tipo best-effort porque hace lo que puede para transmitir los datagramas hacia la aplicación, pero no puede garantizar que la aplicación los reciban.

b.- No preserva la secuencia de la información que proporciona la aplicación. La información se puede recibir desordenadamente y además la aplicación debe estar preparada por si se perdieran datagramas por el camino, o simplemente, llegan con retardo o desordenados.

En este proyecto se han utilizado los siguientes tamaños de paquete UDP:

- 57600
- 38400
- 28800
- 14400
- 7680
- 5760
- 4608
- 2304
- 1152

2. RGB sobre RTP. En este caso se envían imágenes no comprimidas utilizando el protocolo de transmisión de video RTP (Real Time Protocol). Dicho protocolo se basa sobre UDP, aunque obtiene una eficacia mayor debido a que emplea los siguientes mecanismos:

   a. Identificación del esquema de codificación/compresión. Para reducir los requisitos de video, este se codifica guardando sólo la información correspondiente a los píxeles o líneas de información consecutivas que son diferentes.

   b. Números de secuencia

   c. Marcas temporales

Mediante RTP se obtiene un retardo de paquetización pequeño en comparación con UDP. Al igual que UDP, en RTP tampoco se garantiza la entrega de datos.

3. JPEG sobre RTP. Este formato utiliza compresión de imágenes y utiliza el protocolo RTP para la transmisión.
4. H.263 sobre RTP. Es el formato que utiliza una mayor compresión y sigue utilizando el protocolo RTP para la transmisión.

La tasa de imágenes mostradas por segundo(“ fps ”) varía según el tipo de formato elegido por el usuario.

El servicio diseñado cumple todas las premisas propuestas:

- Gracias al uso de Java, es un sistema capaz de funcionar entre distintas máquinas siempre que estas soporten Java. Además aisla del medio en el que se transmitan estos datos.

- Soporta varios modos de transmisión, en esta caso UDP y RTP, y deja abierta la posibilidad de incorporar más modos.

- Permite un control de la reproducción entre cliente y Servidor

- La implantación se ha diseñado siguiendo una arquitectura punto a punto, dejando abierta la posibilidad de que el desarrollador implemente un sistema punto a multipunto.

De cara a explicar los siguientes apartados, se van a definir los siguientes conceptos:

- Middleware: Capa intermedia que facilita la comunicación distribuida entre los distintos procesos.
- Interfaz: Indica los métodos y funciones de la implementación A/V Streaming que ofrece a las aplicaciones.

3.2.- Componentes generales del sistema de transmisión de vídeo

Este apartado se encarga de explicar los componentes necesarios que cualquier sistema de transmisión de video debe tener para su correcto funcionamiento.

- Servidor CORBA. Es el componente encargado del control de la transmisión de datos. Es capaz de actuar ante las distintas llamadas remotas del cliente. Dichas llamadas son:
    - Play: Esta llamada se encarga de iniciar una conexión entre Cliente y Servidor. Dicha conexión sirve para transmitir un flujo

de datos multimedia. Cuando se realiza esta llamada, el usuario debe indicarle el tipo de conexión a realizar(UDP,RTP,TCP,...) y la dirección IP a la que deben ir dirigidos los datos.

- Stop: Para la transmisión de datos, elimina toda conexión entre cliente y servidor.
- Pause: Para la transmisión de datos, pero mantiene la conexión entre cliente y servidor.
- Resume: Reanuda una transmisión de datos anteriormente pausada

Además se encarga de generar los servidores de datos multimedia configurándolos con las características especificadas por el cliente

- Servidor de datos multimedia. Trasmite las imágenes capturadas por la máquina servidora hacia el cliente. El formato de las imágenes se ha configurado anteriormente por el servidor CORBA.

- Cliente Corba. Se encarga de indicarle al Servidor Corba tanto los parámetros de configuración seleccionados por el usuario, asi como de controlar todo el flujo de datos multimedia a través de las llamadas remotas explicadas anteriormente.

- Cliente de datos multimedia: Tiene la función de recibir y mostrar las imágenes que le envía el Servidor de datos multimedia.

En los próximos apartados se va a explicar como se han programado las distintas clases que forman las interfaces, middleware y aplicación.

3.3.- MIDDLEWARE

Este término hace referencia a un extremo remoto, el cual actúa en respuesta a las llamadas remotas del usuario para poder ofrecerle un determinado servicio, en este caso el servicio de A/V Streaming. Dicho servicio es el de la configuración de los parámetros necesarios para una transmisión multimedia. La configuración de dichos parámetros se realiza de la siguiente forma:

1. Cuando el usuario ha decidido conectarse a una dirección determinada, lo primero que se ha de hacer es obtener las referencias local y remota de los MMDevice (ServidorMMDevice_impl y ClienteMMdevice_impl).

2. A partir de dichas referencias se va a llamar a la función createA y createB, la cual debe crear los Vdev y los StreamEndPoint correspodientes.

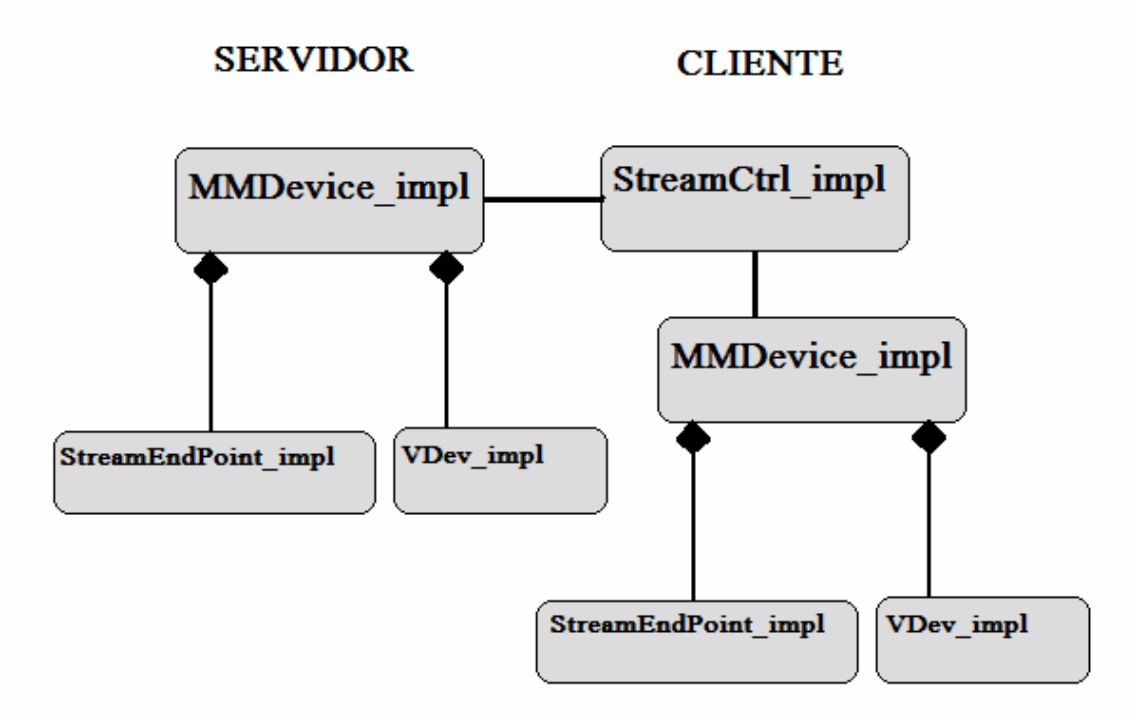
3. El siguiente paso es obtener una referencia de los VDev y StreamEndPoint remotos.

4. Con dichas referencias y con las referencias locales a VDev y StreamEndPoint a las cuales se pueden acceder directamente, se comienza el proceso de configuración de los extremos. En este proceso los VDev los StreamEndPoint deben ponerse de acuerdo en los parámetros de configuración.

5. Lo primero que se debe de hacer es que los VDevs se pongan de acuerdo en el formato de transmisión de datos multimedia.

6. Ahora los StreamEndPoint de ambos extremos deben ponerse de acuerdo. Se indica entonces la dirección y el puerto al que van a ser dirigidos los datos multimedia. Una vez configurados estos parámetros comienza la comunicación multimedia.

Figura:



En los próximos apartados se va a explicar todos los elementos que componen el middleware.

3.4 IMPLEMENTACIÓN DEL SERVICIO

3.4.1.- Paquete AVSTREAMS.idl

El idl que mostramos a continuación es el que propone la especificación para el desarrollo del proyecto. Aunque se han modificado algunas partes respecto al original, su funcionalidad sigue siendo la misma.

```
// AVStreams.idl
// Proyecto fin de carrera
// Diseño e implementacion de un servicio de video
// streaming usando //Java/Corba
// Código de interfaz IDL.
// Pedro Zamora Prades
// Fecha: 01/10/2006

#include "CosPropertyService.idl"

module AVStreams{

////////////////// DECLARACION DE VARIABLES Y
ESTRUCTURAS//////////////////////////

typedef sequence<string> formatSpec;
typedef sequence<string> flowSpec;
typedef sequence<string> protcolSpec;  // UDP, RTP
typedef sequence<octet> key;
```

Los tipos de datos sequence<string> representan arrays de strings en formato Java. Utilizaremos formatSpec para el formato utilizado, flowSpec para aspectos de configuración como IP remota, IP local protocolo utilizado etc…y protocolSpec unicamente para comunicar el protocolo que se utilizará para la transmisión.

```
enum Estado{Enstop, Enstart, dead};
enum TDireccion{Indir, Outdir};

exception notSupported{};
exception PropiedadException{};
exception FPError{string nombreflujo;};
exception OPFailed{string motivo;};
exception OPDenied{string motivo;};
exception noSuchFlow{};

interface Basic_StreamCtrl;
interface Negociador;
interface MMDevice;
interface StreamEndPoint;
interface StreamEndPoint_A;
interface StreamEndPoint_B;
interface VDev;
```

Las interfaces idl representan interfaces (clases) en java.
Todas las interfaces con las que trabajaremos son las anteriormente mencionadas, y cuya funcionalidad se comentó en capítulos anteriores.

```
struct SFPStatus{
   boolean Formateado;
```

```
boolean TiempoTam;
boolean IndicadorFuente;};//fin struct SFPS

boolean PormatoEspecial;boolean SecueNums;
struct flujoStatus{
   string nombre;
   TDireccion direccionalidad;
   Estado est; SFPStatus formato;};
```

Estas estructuras no han sido utilizadas en este servicio, pero quedan disponobles para futuros trabjos.

```
////////////////////////////DEFINICION DE
INTERFACES//////////////////////

interface Basic_StreamCtrl{
   void stop(in flowSpec spec);

   void start(in flowSpec spec);

   void destroy(in flowSpec spec);

   void set_FPStatus(in flowSpec the_spec, in string
   fp_name, in any  fp_settings) raises (noSuchFlow,
   FPError);

   Object get_flow_connection(in string flow_name)
            raises (noSuchFlow, notSupported);

   oid set_flow_connection(in string flow_name,
            in Object flow_connection)
            raises (noSuchFlow, notSupported);


};

////////////////////////////////////////////////////////////
////////////////

interface Negociador{
   boolean negociacion(in Negociador remoto);};

////////////////////////////////////////////////////////////
/////////////////

interface StreamCtrl : Basic_StreamCtrl {

   boolean bind_devs(in MMDevice a_party, in MMDevice
         b_party,
         in flowSpec the_flows)
         raises (OPFailed, noSuchFlow);
```

```
   boolean bind(in StreamEndPoint_A a_party,
         in StreamEndPoint_B b_party,
         in flowSpec the_flows)
         raises (OPFailed, noSuchFlow);

   void unbind_party(in StreamEndPoint the_ep,
         in flowSpec the_spec)
         raises (OPFailed, noSuchFlow);

   void unbind() raises (OPFailed);
};


////////////////////////////////////////////////////////
////////////////////

   interface StreamEndPoint :
   CosPropertyService::PropertySet{

   void stop(in flowSpec the_spec) raises (noSuchFlow);

   void start(in flowSpec the_spec) raises (noSuchFlow);

   void destroy(in flowSpec the_spec) raises (noSuchFlow);

   boolean connect(in StreamEndPoint responder,
         in flowSpec the_spec)
         raises (noSuchFlow, OPFailed);

   boolean request_connection(
         in StreamEndPoint initiator,
         in boolean is_mcast,
         inout flowSpec the_spec)
         raises (OPDenied, noSuchFlow,FPError);

   boolean set_protocol_restriction(in protocolSpec
      the_pspec);

   void disconnect(in flowSpec the_spec)
         raises (noSuchFlow, OPFailed);

   void set_FPStatus(in flowSpec the_spec,
         in string fp_name,
         in any fp_settings)
         raises (noSuchFlow, FPError);

   Object get_fep(in string flow_name)
         raises (notSupported, noSuchFlow);
```

```
   string add_fep(in Object the_fep)
// Can fail for reasons {duplicateFepName, duplicateRef}
         raises (notSupported, OPFailed);

   void remove_fep(in string fep_name)
         raises (notSupported, OPFailed);

   void set_negotiator(in Negociador new_negotiator);

   void set_key(in string flow_name, in key the_key);

   void set_source_id(in long source_id);
};
//////////////////////////////////////////////////////
//////////////////////

interface StreamEndPoint_A : StreamEndPoint{

   boolean multiconnect(inout flowSpec the_spec)
         raises (noSuchFlow,OPFailed);

   boolean connect_leaf(in StreamEndPoint_B the_ep,
         in flowSpec the_flows)
         raises (OPFailed, noSuchFlow, notSupported);

   void disconnect_leaf(in StreamEndPoint_B the_ep,
         in flowSpec theSpec)
         raises(OPFailed, noSuchFlow);
         };

//////////////////////////////////////////////////////
//////////////////////

interface StreamEndPoint_B : StreamEndPoint {

   boolean multiconnect(inout flowSpec the_spec)
         raises (OPFailed, noSuchFlow, FPError);
         };

//////////////////////////////////////////////////////
//////////////////////

interface VDev : CosPropertyService::PropertySet{
   boolean set_peer(in StreamCtrl the_ctrl,
         in VDev the_peer_dev,in flowSpec the_spec)
         raises (noSuchFlow, OPFailed);

   void configure(in CosPropertyService::Property
         the_config_mesg)
         raises(OPFailed);
```

```
   void set_format(in string flowName, in string
         format_name)
         raises (notSupported);

   void set_dev_params(in string flowName,
         in CosPropertyService::Properties new_params)
         raises(OPFailed);    };

////////////////////////////////////////////////////////
//////////////////////

interface MMDevice{
   StreamEndPoint_A create_A(in StreamCtrl the_requester,
         out VDev the_vdev,inout string named_vdev,in
         flowSpec the_spec) raises(OPFailed, OPDenied,
         notSupported, noSuchFlow);

   StreamEndPoint_B create_B(in StreamCtrl the_requester,
         out VDev the_vdev,inout string named_vdev,in
         flowSpec the_spec) raises(OPFailed,OPDenied,
         notSupported, noSuchFlow);

   StreamCtrl bind(in MMDevice peer_device,
         out boolean is_met,in flowSpec the_spec)
         raises (OPFailed, noSuchFlow);

   StreamCtrl bind_mcast(in MMDevice first_peer,   out
         boolean is_met, in flowSpec the_spec)
         raises (OPFailed, noSuchFlow);

   void destroy(in StreamEndPoint the_ep, in string
         vdev_name) raises (notSupported);

   string add_fdev(in Object the_fdev)
         raises(notSupported, OPFailed);

   Object getVdev() raises(notSupported, noSuchFlow);

   Object getStreamEndPoint() raises (notSupported,
         noSuchFlow);
   };
};
```

3.4.1.a.- BASIC_STREAMCTRL_IMPL

```
/* PFC.
   Autor: Pedro Zamora Prades
   Basic_StreamCtrl: esta función define el esqueleto de la
   clase StreamCtrl
   Su misión es la de controlar el flujo de datos.
```

```
    Esta clase es instanciada en el cliente.
    */
package AVStreams;
import CosPropertyService.*;
import java.util.*;

public class Basic_StreamCtrl_impl extends
    Basic_StreamCtrlPOA{

    public Basic_StreamCtrl _this = null;
    public org.omg.CORBA.ORB orb = null;
    public PropertySetDefFactoryImpl property_factory =
    null;
    public PropertySetDefImpl psdi = null;
    public PropertySet property_set = null;

public Basic_StreamCtrl_impl(org.omg.CORBA.ORB orb,
    PropertySetDefFactoryImpl property_factory){
    }

public void stop(String[] the_spec){
    }

public void start(String[] the_spec) {
    }

public void destroy(String[] the_spec){
    }
```

Todos los métodos anteriormente mencionados están debidamente implementados en la clase streamCtrl_Impl.

```
public void set_FPStatus (String[] the_spec, String
    fp_name, org.omg.CORBA.Any fp_settings) throws
    AVStreams.noSuchFlow, AVStreams.FPError{
    }

public org.omg.CORBA.Object get_flow_connection (String
    flow_name) throws AVStreams.noSuchFlow,
    AVStreams.notSupported{
    return null;
    }

public void set_flow_connection (String flow_name,
   org.omg.CORBA.Object flow_connection) throws
   AVStreams.noSuchFlow, AVStreams.notSupported{
    }

   // CosPropertyService
```

```
public void define_property(String property_name,
   org.omg.CORBA.Any property_value) throws
   InvalidPropertyName, ConflictingProperty,
   UnsupportedTypeCode, UnsupportedProperty,
   ReadOnlyProperty{
   this.psdi.define_property(property_name,property_value);
    }

public void define_properties(Property[] nproperties)
   throws MultipleExceptions{
   this.psdi.define_properties(nproperties);
   }

public int get_number_of_properties(){
   return this.psdi.get_number_of_properties();
   }

public void get_all_property_names(int how_many,
   PropertyNamesHolder nproperties,
   PropertyNamesIteratorHolder rest){
   this.psdi.get_all_property_names(how_many, nproperties,
   rest);
   }

public org.omg.CORBA.Any get_property_value(String
   property_name) throws PropertyNotFound,
   InvalidPropertyName{
   return this.psdi.get_property_value(property_name);
   }

public void delete_properties(String[] property_names)
    throws MultipleExceptions{
    this.psdi.delete_properties(property_names);
         }

public boolean delete_all_properties(){
    return this.psdi.delete_all_properties();
         }

public boolean is_property_defined(String property_name)
      throws InvalidPropertyName{
      return
      this.psdi.is_property_defined(property_name);
      }

public void set_this(Basic_StreamCtrl _this){
      this._this = _this;
         }

}
```

3.4.1.b.- STREAMCTRL_IMPL

Esta clase es la encargada de enlazar las clases MMDevice del cliente y del servidor y de esta manera poder utilizar un único flujo de datos. Además hereda de la clase Basic_StreamCtrl_imp

```
/* Esta clase es la encargada de enlazar las clases
MMDevice del cliente y del servidor y de esta manera poder
utilizar un único flujo de datos */


public class StreamCtrl_impl extends StreamCtrlPOA{
  private StreamCtrl _this=null;
  public org.omg.CORBA.ORB orb = null;
  public PropertySetDefFactoryImpl property_factory = null;
  public PropertySetDefImpl psdi = null;
  public PropertySet property_set = null;
  public String[] flows = new String[0];
  public StreamEndPoint_A sepA = null;
  public StreamEndPoint_B sepB = null;
  public VDev vdev_A = null;
  public VDev vdev_B = null;
  public StreamEndPointA_impl sepai=null;
  public MMDevice mmdvs=null;
  public StreamEndPoint_impl sepi=null;

public StreamCtrl_impl(org.omg.CORBA.ORB orb,
  PropertySetDefFactoryImpl property_factory){
  System.out.println("\nStreamCtrl_impl Creado");
    }
```

Constructor de la clase

```
public void set_FPStatus (String[] the_spec, String
      fp_name, org.omg.CORBA.Any fp_settings){}
```

Este método no ha sido implementado en este servicio, pero que a disposición para futuros cambios.

```
public void stop(String[] the_spec){
      String[] stop_flows = null;
      if (the_spec == null || the_spec.length == 0){
      stop_flows = the_spec;
      }

      else{
      stop_flows = the_spec;
      }
```

```
try{
     sepi.stop(stop_flows);
     }catch(Exception e){
    }
  }
```

Este método para la captura de vídeo.

```
   public void start(String[] the_spec) {
     System.out.println("Se va a intentar comenzar la
     captura: basic");

     String[] start_flows = null;
     start_flows = the_spec;

     try{
      if(sepi.direccionlocal.equals(the_spec[2])){
      sepi.direccionremota=the_spec[1];
      sepi.start(the_spec);
      else{
      System.out.println("No Se puede establecer conexión
      remota");
        }
      }catch(Exception e){
    }
  }
```

El método start inicia la captura de imágenes a través de la cámara. La captura de imágenes se inicia tras comprobar que la dirección local coincide con la dirección a la cual se quiere conectar el cliente y que se encuentra en spec[1].

```
public void destroy(String[] the_spec){
     String[] destroy_flows = null;
     if (the_spec == null || the_spec.length == 0){
       destroy_flows = the_spec;
       }
       else{
       destroy_flows = the_spec;
       }
       try{
       sepi.destroy(destroy_flows);
       }catch(Exception e){
      }
   }

public org.omg.CORBA.Object get_flow_connection (String
  flow_name) throws AVStreams.noSuchFlow,
  AVStreams.notSupported{
  return null;
  }
```

```
public void set_flow_connection (String flow_name,
  org.omg.CORBA.Object flow_connection) throws
  AVStreams.noSuchFlow, AVStreams.notSupported{
  }


public boolean bind_devs(MMDevice a_party, MMDevice
  b_party, String[] the_flows) throws OPFailed,
  noSuchFlow{
  //sepA
  System.out.println("Conectando MMDevices en bind_devs");
  VDevHolder vdev_A_holder = new VDevHolder();
  org.omg.CORBA.StringHolder sep_A_named_vdev = new
  org.omg.CORBA.StringHolder("");
  try{
  sepA = a_party.create_A((StreamCtrl)_this,
  vdev_A_holder, sep_A_named_vdev, the_flows);
  System.out.println("MMDeviceA ha creado SEPA");
  vdev_A = vdev_A_holder.value;
  org.omg.CORBA.Any any_strmCtrl = orb.create_any();
  StreamCtrlHelper.insert(any_strmCtrl,(StreamCtrl)_this);
  sepA.define_property("Related_StreamCtrl",any_strmCtrl);
       }
  catch (Exception e){
  throw new OPFailed("");
     }
  //sepB
  VDevHolder vdev_B_holder = new VDevHolder();
  org.omg.CORBA.StringHolder sep_B_named_vdev = new
  org.omg.CORBA.StringHolder("");
  try{
  sepB = b_party.create_B((StreamCtrl)_this,
  vdev_B_holder,
  sep_B_named_vdev, the_flows);
  System.out.println("MMDeviceB ha creado SEPB");
  vdev_B = vdev_B_holder.value;
  org.omg.CORBA.Any any_strmCtrl = orb.create_any();
  StreamCtrlHelper.insert(any_strmCtrl,
  (StreamCtrl)_this);
  sepB.define_property("Related_StreamCtrl",
  any_strmCtrl);
     }
 catch (Exception e){
 throw new OPFailed("");
    }

 boolean b = bind((StreamEndPoint_A)sepA,    return b;
 return true;
   }
```

Con este método se gestiona la conexión de cliente y del servidor. Para ello se manda crear las clases StreamEndPoint mediante los MMDevice correspondientes.

```
  public void unbind_party(StreamEndPoint the_ep, String[]
    the_spec) throws OPFailed,
    noSuchFlow{ throw new OPFailed("");
    }
    public void unbind() throws OPFailed{
    }

  public boolean bind(StreamEndPoint_A a_party,
     StreamEndPoint_B b_party, String[] the_flows)
     throws OPFailed, noSuchFlow{
     if (vdev_A == null || vdev_B == null){
       try{
       org.omg.CORBA.Any any_a =
       sepA.get_property_value("Related_VDev");
       vdev_A = VDevHelper.extract(any_a);

       org.omg.CORBA.Any any_b =
       sepB.get_property_value("Related_VDev");
       vdev_B = VDevHelper.extract(any_b);
            }
       catch (Exception e){
       throw new OPFailed("No se puede obtener la
       propiedades del sep.");
            }
        }

       try{
       boolean config_A =
       vdev_A.set_peer((StreamCtrl)_this, vdev_B,
       the_flows);
            }
       catch (Exception e){
       throw new OPFailed("No se puede establecer el para
       de VDev A.");
       }

      try{
      boolean config_B =
      vdev_B.set_peer((StreamCtrl)_this, vdev_A,
      the_flows);
      }
      catch (Exception e){
      throw new OPFailed("");
       }
      try{
      org.omg.CORBA.Any any_flows_sep_A =
      sepA.get_property_value("Flows");
      flows = flowSpecHelper.extract(any_flows_sep_A);
```

```
      org.omg.CORBA.Any any_flows = orb.create_any();
      any_flows.insert_any(any_flows_sep_A);
      this.define_property("Flows", any_flows);
        }
      catch (Exception e){
      throw new OPFailed("Fallo 3");
        }
      boolean connect = false;
      try {
      org.omg.CORBA.Any any = null;
      String[] the_spec = new String[flows.length];
      for (int i = 0; i < flows.length; i++){
      any = sepA.get_property_value(flows[i] + "_dir");
      String dir = new String(any.extract_string());
      // Prueba
      ((StreamEndPointA_impl)sepA).setDir(dir);

      any = sepA.get_property_value(flows[i] +
      "_currFormat");
      String format = new String(any.extract_string());
      the_spec[i] = new String(flows[i] + "\\" + dir);
      System.out.println("...flow: " + the_spec[i]);
      // Prueba
      ((StreamEndPointA_impl)sepA).setFormato(flows);
      }

      any = sepA.get_property_value("AvailableProtocols");
      String[] sepA_protSpec =
      protocolSpecHelper.extract(any);
      any = sepB.get_property_value("AvailableProtocols");
      String[] sepB_protSpec =
      protocolSpecHelper.extract(any);


((StreamEndPointA_impl)sepA).setTipo(sepA_protSpec);

      sepA.set_protocol_restriction(sepA_protSpec);
      sepB.set_protocol_restriction(sepB_protSpec);

      System.out.println("Sep a conectando a Sep b");
      connect = sepA.connect(sepB, the_spec);
      System.out.println("...stream endpoints connected.");
        }
      catch (Exception e){
      throw new OPFailed("No se puede conectar los seps");
        }

      return true;
    }

  public Request _create_request(Context ctx, String
```

```
    operation, NVList arg_list, NamedValue result) {
     }
    return true;
}
```

3.4.1.c.- STREAMENDPOINT_IMPL

Esta clase es instanciada por MMDevice. Se encarga de almacenar los parámetros IP y puerto con los que se realiza la comunicación multimedia.

```
public class StreamEndPoint_impl extends StreamEndPointPOA{
    protected org.omg.CORBA.ORB orb = null;
    protected PropertySetDefImpl property_set = null;
    protected PropertySetDefFactoryImpl property_factory =
    null;
    protected StreamEndPoint _this = null;
    protected String[] protocol_restriction = null;
    protected VDev_impl related_vdev_impl = null;
    public String direccionlocal;
    public String direccionremota;
    protected String tipo;
    protected String formato;
    protected int puerto;
    public Video video=null;

public StreamEndPoint_impl(org.omg.CORBA.ORB orb,
      PropertySetDefFactoryImpl property_factory){
      this.orb = orb;
      this.property_factory = property_factory;
      this.property_set = property_factory.crear();
      }

public void stop(String[] the_spec) throws noSuchFlow{
    if (the_spec == null || the_spec.length == 0){
    System.out.println("");
    }
    else{
    for (int i = 0; i < the_spec.length; i++){
    video.stop();
       }
      }
    }

public void start(String[] the_spec) throws noSuchFlow{
   System.out.println("Se va a crear una clase Video");
   //video=new Video("RGB","RGB 1152","127.0.0.1",2200);

   if(the_spec[0].equals("RGB/RTP")||the_spec[0].equals("JP
   EG/RTP")||the_spec[0].equals("H.263/RTP")){
   video = new Video("UDP",the_spec[0],the_spec[1],910);
   }
```

```
   else{
   video = new Video ("RGB",the_spec[0],"127.0.0.1",910);
   }

   System.out.println("Se va a comenzar la captura");
   video.play();
   }

public void destroy(String[] the_spec) throws noSuchFlow{
   if (the_spec == null || the_spec.length == 0){
   System.out.println("");
   }
   else{
      for (int i = 0; i < the_spec.length; i++){
      video.pause();
          }
      }
    }

public boolean connect(StreamEndPoint responder, String[]
   the_spec) throws noSuchFlow, OPFailed{
   return true;
   }

public boolean request_connection(StreamEndPoint initiator,
   boolean is_mcast, flowSpecHolder the_spec) throws
   OPDenied,noSuchFlow, FPError{
   if (is_mcast) throw new OPDenied("");
   return true;
   }

public boolean set_protocol_restriction(String[]
   the_pspec){
   String prots[] = {};
   protocol_restriction = prots;
   boolean rtp = false;
   boolean rgb = false;
   for (int i=0; i < the_pspec.length; i++){
      if (the_pspec[i].equals("RTP")){
      rtp = true;
      }
      else{
      if(the_pspec[i].equals("RGB")){
      rgb=true;
      }
      }
    }
   if (rtp) {
   prots[0] = "RTP";
   protocol_restriction = prots;
   }
```

```
   if(rgb){
   prots[0]= "RGB";
   protocol_restriction = prots;
           }
   return true;
     }
```

Este método establece el una restricción en el protocolo de transmisión, obligando a que esta se realice utilizando dicho protocolo

```
public void disconnect(String[] the_spec)
     throws noSuchFlow, OPFailed{
    }

public void set_FPStatus(String[] the_spec, String fp_name,
   org.omg.CORBA.Any fp_settings) throws noSuchFlow,
   FPError {
   throw new FPError("");
   }

public org.omg.CORBA.Object get_fep(String flow_name)
   throws notSupported, noSuchFlow  {
   return null;
   }

public String add_fep(org.omg.CORBA.Object the_fep){
   return null;
   }

public void remove_fep(String fep_name)
   throws notSupported, OPFailed {
   }

public void set_negotiator(Negociador new_negotiator){
   }

public void set_key(String flow_name, byte[] the_key){
   }

public void set_source_id(int source_id) {
   }
```

Los métodos anteriores no han sido implementados en este servicio, pero quedan disponibles para futuras aplicaciones o modificaciones de este servicio.

```
public void setDirlocal(String d){
   direccionlocal=d;
   }
```

Establece la direccion IP local

```
public void setDirRemota(String d){
   direccionremota=d;
   }
```

Establece la dirección IP remota con la que se va iniciar una transmisión multimedia.

```
public void setFormato(String[] f){
   formato=f[0];
   }
```

Establece el formato de transmisión.

```
public void setTipo(String[] t){
   tipo=t[0];
   }

   //CoPropertyService
```

A continución se implementarían los métodos de CosPropertyService
…
…

```
public void set_this(StreamEndPoint _this){
     this._this = _this;
    }

public void set_related_vdev_impl(VDev_impl vdev_impl){
        this.related_vdev_impl = vdev_impl;
    }
```

La implemtación de los métodos de CosPropertyService en esta clase es igual al de la clase StreamCtrl_impl

Encapsula los parámetros de transmisión de un Stream. Los métodos stop, start y destroy paran, reanudan y eliminan la transmisión.
El método set_protocol_restriction almacena el protocolo de restricción para un stream, y puede ser RGB o RTP. Los demas métodos no son implementados en esta clase.

3.4.1.d.- STREAMENDPOINTA_IMPL

```
/*Esta clase añade soporte para multicast. Multicast no es
soportado de momento */

package AVStreams;

import CosPropertyService.*;

   public class StreamEndPointA_impl extends
```

```
       StreamEndPoint_impl implements
       StreamEndPoint_AOperations{

    public StreamEndPointA_impl(org.omg.CORBA.ORB orb,
        PropertySetDefFactoryImpl
        property_factory){
        super(orb, property_factory);
        System.out.println("StreamEndPointA");
        }

    public boolean multiconnect( flowSpecHolder the_spec)
        throws noSuchFlow, OPFailed{
        throw new OPFailed("");
        }

    public boolean connect_leaf(StreamEndPoint_B the_ep,
        String[] the_flows)
        throws OPFailed, noSuchFlow, notSupported{
        throw new OPFailed("");
    }

    public void disconnect_leaf(StreamEndPoint_B the_ep,
        String[] theSpec)
        throws OPFailed, noSuchFlow{
        throw new OPFailed("");
     }

    public boolean connect(StreamEndPoint responder,
        String[] the_spec)
        throws noSuchFlow, OPFailed{
        System.out.println("LLamando operación en
        StreamEndPointA_impl...");

        flowSpecHolder the_spec_holder = new
        flowSpecHolder(the_spec);
        try{
        boolean peticion =
        responder.request_connection(_this, false,
        the_spec_holder);
        return peticion;
        }
        catch (FPError fep){
        System.out.println("FPError.");
        throw new OPFailed("Error de peticion de
        conexión");
        }
        catch (OPDenied opd){
        System.out.println("FPError.");
        throw new OPFailed("fallo");
    }
}
```

```
    public boolean request_connection(StreamEndPoint
      initiator, boolean is_mcast, flowSpecHolder the_spec)
        throws OPDenied, noSuchFlow, FPError{
        throw new OPDenied("Petición de conexión debe ser
        instanciada por el cliente");
     }
}
```

3.4.1.e.- STREAMENDPOINTB_IMPL

```
public class StreamEndPointB_impl extends
StreamEndPoint_impl implements StreamEndPoint_BOperations{

    public StreamEndPointB_impl(org.omg.CORBA.ORB orb,
        PropertySetDefFactoryImpl property_factory){
        super(orb, property_factory);
        System.out.println("StreamEndPointB");
    }

    public boolean multiconnect(flowSpecHolder the_spec)
        throws OPFailed, noSuchFlow, FPError{
        throw new OPFailed("Not supported.");
       }


    public boolean connect(StreamEndPoint responder,
        String[] the_spec)
        throws noSuchFlow, OPFailed{
        throw new OPFailed("Operación no soportada por el
        Cliente");
    }

    public boolean request_connection(StreamEndPoint
      initiator, boolean is_mcast, flowSpecHolder the_spec)
           throws OPDenied, noSuchFlow, FPError{

        if(is_mcast)
        return false;

        System.out.println("Petición de conexión iniciada
        en ClienteStreamEndPoint_impl");
        System.out.println("Petición de conexion
        aceptada");
        return true;
     }
}
```

3.4.1.f.-VDEV_IMPL

```
public class VDev_impl implements VDevOperations{
    protected org.omg.CORBA.ORB orb = null;
    protected PropertySetDefFactory propiedades = null;
    protected PropertySet property_set = null;
    protected StreamCtrl sc = null;
    protected VDev otroVdev = null;

    public VDev_impl(org.omg.CORBA.ORB orb,
    PropertySetDefFactory property_factory){
        this.orb = orb;
        propiedades = property_factory;
        property_set =
     (PropertySet)property_factory.create_propertysetdef();
    }

    public boolean set_peer(StreamCtrl the_ctrl, VDev
       the_peer_dev, String[] the_spec) throws noSuchFlow,
       OPFailed{
       if (sc == null){
       sc = the_ctrl;
       otroVdev = the_peer_dev;
       }
       else{
       if (!sc._is_equivalent(the_ctrl)){ throw new
       OPFailed();}
       else{
       sc = the_ctrl;
       otroVdev = the_peer_dev;
       }
     }

       String[] vdev_flows = null;
       try{
       org.omg.CORBA.Any any_flows =
       this.get_property_value("Flows");
       vdev_flows = flowSpecHelper.extract(any_flows);
       } catch (Exception e){
         throw new OPFailed();
       }

       String[] flows = null;
       if (the_spec != null && the_spec.length > 0){
         for (int i=0; i<the_spec.length; i++){
           boolean encontrado = false;
           int j=0;
           while (!encontrado && j<vdev_flows.length){
             if (the_spec[i].equals(vdev_flows[j]))
             encontrado = true;
```

```
            else
            j++;
            }
            if (!encontrado){throw new noSuchFlow();
            }
          }
         flows = the_spec;
        }
        else{
          flows = vdev_flows;
          }
        for (int i=0; i<flows.length; i++){
            System.out.println("Estableciendo par para el
            fujo: " + flows[i]);
            try{
            org.omg.CORBA.Any any_available_formats =
     this.get_property_value(flows[i]+"_availableFormats");
            String[] available_formats =
           formatSpecHelper.extract(any_available_formats);
            org.omg.CORBA.Any any_peer_available_formats =
the_peer_dev.get_property_value(flows[i]+"_availableFormats
");
            String[] peer_available_formats =
      formatSpecHelper.extract(any_peer_available_formats);
            org.omg.CORBA.Any a_format_defecto =
this.get_property_value(flows[i]+"_format_defecto");
             String format_defecto =
             a_format_defecto.extract_string();
             org.omg.CORBA.Any any_peer_current_format =
   the_peer_dev.get_property_value(flows[i]+"_currFormat");
              String peer_current_format =
              any_peer_current_format.extract_string();

              boolean common_format = false;

          if (!format_defecto.equals(peer_current_format)){
          int j=0;
           while (!common_format&&
           j<peer_available_formats.length){
            if
        (format_defecto.equals(peer_available_formats[j])){
        the_peer_dev.set_format(flows[i], format_defecto);
            common_format = true;
             }
            else
            j++;
            }
            j=0;
         while (!common_format &&
            j<available_formats.length){
            int k=0;
```

```
            while (!common_format &&
            k<peer_available_formats.length){
            if (available_formats[j].equals(
            peer_available_formats[k])){
            this.set_format(flows[i],
            available_formats[j]);
            the_peer_dev.set_format(flows[i],
            available_formats[j]);
            common_format = true;
            }
            else
            j++;
            }

            j++;
            }
           }
          else{
          common_format = true;
          }
          if (!common_format) throw new OPFailed();
          }
          catch (Exception e){
          throw new OPFailed();
          }
      }
    return true;
    }

  public void configure(CosPropertyService.Property
  the_config_mesg)throws OPFailed{
     }

  public void set_format(String flowName, String
  format_name)
      throws notSupported {
      try {
          org.omg.CORBA.Any any = orb.create_any();
          // Crea objeto Corba
          any.insert_string(format_name);
          //  inserta el String
          org.omg.CORBA.Any any_sep =
       this.get_property_value("Related_StreamEndPoint");
          // Nombre
          StreamEndPoint sep =
          StreamEndPointHelper.extract(any_sep); // valor
          this.define_property(flowName +
          "_format_defecto", any);
          // Define propiedad para VDev
          sep.define_property(flowName +
          "_format_defecto", any);
```

```
            // define propiedad para StreamEndPoint
        }
        catch (Exception e){}
    }

       public void set_dev_params(String flowName,
        CosPropertyService.Property[] protocolo)
        throws OPFailed {
        try{
            org.omg.CORBA.Any any = orb.create_any();
            CosPropertyService.PropertiesHelper.insert(any,
            protocolo);
            org.omg.CORBA.Any any_sep =
         this.get_property_value("Related_StreamEndPoint");
            StreamEndPoint sep =
            StreamEndPointHelper.extract(any_sep);
            this.define_property(flowName + "_protocol",
            any);
            sep.define_property(flowName + "_protocol",




   // CosPropertyService

A continuación iría la implementación de todos los métodos
de CosPropertyService
...
...
...
```

3.4.1.g.- CLIENTEVDEV_IMPL

```
public class ClienteVDev_impl extends VDev_impl{

    private String[] the_spec = null;

    public ClienteVDev_impl(org.omg.CORBA.ORB orb,
    PropertySetDefFactoryImpl property_factory, String[]
    the_spec){
    super(orb, property_factory);
    this.the_spec = the_spec;
    }

// Este método establece las propiedades para ClienteVDev

    protected boolean set_properties(){
        try{
```

```
            //  Dirección Flujo
            org.omg.CORBA.Any direcc_flujo =
            orb.create_any();
            direcc_flujo.insert_string("in");
            this.define_property("direcc_flujo",
            direcc_flujo);

            //  Formato por defecto
            org.omg.CORBA.Any format_defecto =
            orb.create_any();
            format_defecto.insert_string("RTP");
            this.define_property("format_defecto",
            format_defecto);
        }

        catch (Exception e){
            return false;
          }
        return true;
      }
   }
```

3.4.1.h.- SERVIDORVDEV_IMPL

```
public class ServidorVDev_impl extends VDev_impl{
    private String[] the_spec = null;
    private ServidorCorba.Servidor application = null;

    public ServidorVDev_impl(org.omg.CORBA.ORB orb,
    PropertySetDefFactoryImpl property_factory,
    ServidorCorba.Servidor application, String[] the_spec){
        super(orb, property_factory);
        this.application = application;
        this.the_spec = the_spec;
    }

    public void set_format(String flowName, String
      format_name) throws notSupported{
          try{
           super.set_format(flowName, format_name);
           return;
           }
          catch (Exception e){
           System.out.println("***fallo al establecer
           formato.");
           throw new notSupported(format_name);
           }
            }

    public void set_dev_params(String flowName,
```

```
      CosPropertyService.Property[] new_params)
      throws OPFailed{
              try{
       super.set_dev_params(flowName, new_params);
       return;
           }
       catch (Exception e){
       System.out.println("***Setting device parameters
       fails.");
       throw new OPFailed(flowName);
              }
         }

    public boolean set_peer(StreamCtrl the_ctrl, VDev
      the_peer_dev,  String[] the_spec)
      throws noSuchFlow, OPFailed{

      return super.set_peer(the_ctrl,the_peer_dev,
      the_spec);
      }

    protected boolean set_properties(){
        try{
          org.omg.CORBA.Any any_flows = orb.create_any();
          String[] flows = new String [1];
          flows[0] = "video";
          flowSpecHelper.insert(any_flows, flows);
          this.define_property("Flows", any_flows);
          org.omg.CORBA.Any any_video1_dir =
          orb.create_any();
          any_video1_dir.insert_string("out");
          this.define_property("video_dir",any_video1_dir);
           }

        catch (Exception e){
          System.out.println("***Setting properties on VDev
          fails.");
          e.printStackTrace();
          return false;
        }
        return true;
    }

    private synchronized boolean waitForState(Processor p,
      int state){
      p.addControllerListener(new StateListener());

       if (state == Processor.Configured){
       p.configure();
       }
       else
```

```
        if (state == Processor.Realized){
        p.realize();
        }
       while (p.getState() < state && !failed){
       synchronized (getStateLock()){
        try{
        getStateLock().wait();
        }
        catch (InterruptedException ie){
        return false;
            }
         }
       }
       if (failed)
       return false;
       else
       return true;
    }

  class StateListener implements ControllerListener{
      public void controllerUpdate(ControllerEvent ce){
         if (ce instanceof ControllerClosedEvent){
            // setFailed();
         System.out.println("***ControllerClosedEvent: " +
         ce);
         }
          if (ce instanceof ControllerEvent){
            synchronized (getStateLock()){
            getStateLock().notifyAll();
               }
             }
           }
         }
       }
```

3.4.1.i.-.- MMDEVICE_IMPL

Esta clase se trata de una factoria de objetos. Su papel principal es la de recolectar la información necesaria para crear un objeto deseado y luego llamar al constructor de la

Paquetes necesarios:

```
public class MMDevice_impl implements MMDeviceOperations{
    protected MMDevice _this = null;
    protected VDev vdev = null;
    protected StreamEndPoint sep;
    protected org.omg.CORBA.ORB orb = null;
    protected PropertySetDefFactoryImpl property_factory =
    null;
    protected PropertySet property_set = null;
```

```
    protected PropertySetDefFactory prueba = null;
    protected PropertySetDefImpl psdi = null;
    private ServidorVDev_impl vdev_impl;
    private StreamEndPoint_A sep_A;
    private StreamEndPointA_impl sep_A_impl;

    public MMDevice_impl(org.omg.CORBA.ORB orb,
        PropertySetDefFactoryImpl property_factory){
        this.orb = orb;

        this.property_factory = property_factory;
        this.property_set =
(PropertySet)this.property_factory.create_propertysetdef();
        psdi = this.property_factory.crear();
        System.out.println("Inicializando
        MMDevice_impl...");
        }

Se procede a la exposición de las funciones que componen la
clase.
Esta clase crea un StreamEndPointA y un VdevA.

    public StreamEndPoint_A create_A(StreamCtrl
           the_requester, VDevHolder the_vdev,
           org.omg.CORBA.StringHolder named_vdev,
           String[] the_spec) throws OPFailed, OPDenied,
           notSupported, noSuchFlow{

        System.out.println("Quien tiene que hacer esti
        es ServidorMMDevice_impl");


         vdev_impl = new ServidorVDev_impl(orb,
         property_factory,application, the_spec);
         VDevPOATie vdev_tie = new VDevPOATie(vdev_impl);
         vdev = vdev_tie._this(orb);

         sep_A_impl = new StreamEndPointA_impl(orb,
         property_factory);
         StreamEndPoint_APOATie sep_A_tie = new
         StreamEndPoint_APOATie(sep_A_impl);
         sep_A = sep_A_tie._this(orb);

         sep_A_impl.set_this(sep_A);
         sep_A_impl.set_related_vdev_impl(vdev_impl);

        try{
            // para el vdev define una propiedad referente
               al sep y otra al mmdevice

            //sep
```

```
            org.omg.CORBA.Any any_sep = orb.create_any();
            StreamEndPointHelper.insert(any_sep, sep_A);
            vdev.define_property("Related_StreamEndPoint",
            any_sep);

            //mmdev
            org.omg.CORBA.Any any_mmdev = orb.create_any();
            MMDeviceHelper.insert(any_mmdev, _this);
            vdev.define_property("Related_MMDevice",
            any_mmdev);
        }
        catch (Exception e){
            throw new OPFailed("fallo");
        }
        if (!vdev_impl.set_properties()){
            throw new OPFailed("fallo");
        }

        try{
           //para el sep.

            org.omg.CORBA.Any any_vdev = orb.create_any();
            VDevHelper.insert(any_vdev, vdev);
            sep_A.define_property("Related_VDev",
            any_vdev);

            org.omg.CORBA.Any any_flows = orb.create_any();
            String[] flows = new String[1];
            flows[0] = "video";

            flowSpecHelper.insert(any_flows, flows);
            sep_A.define_property("Flows", any_flows);

            org.omg.CORBA.Any any_video_dir =
            orb.create_any();
            any_video_dir.insert_string("out");
            sep_A.define_property("video_dir",
            any_video_dir);

            org.omg.CORBA.Any any_video_currFormat =
            orb.create_any();
            any_video_currFormat.insert_string("");
            sep_A.define_property("video_currFormat",
            any_video_currFormat);

            org.omg.CORBA.Any any_video1_address =
            orb.create_any();

            int port_video = port_base;
            any_video1_address.insert_string("RTP=" +
            network_address + ":" + port_video);
```

```
            sep_A.define_property("video1_address",
            any_video1_address);

            org.omg.CORBA.Any any_AvailableProtocols =
            orb.create_any();
            String[] prots = {"RTP"};

protocolSpecHelper.insert(any_AvailableProtocols, prots);
            sep_A.define_property("AvailableProtocols",
            any_AvailableProtocols);

            De momento no restricción de protocolo
            org.omg.CORBA.Any any_ProtocolRestriction =
            orb.create_any();
            String[] empty = new String[0];

protocolSpecHelper.insert(any_ProtocolRestriction, empty);
            sep_A.define_property("ProtocolRestriction",
            any_ProtocolRestriction);
            System.out.println("Todas las propiedades
            establecidas en servidor");
        }

        catch (Exception e){
            throw new OPFailed();
        }

        the_vdev = new VDevHolder(vdev);
        System.out.println("Retorno de sepa");
        return sep_A;
    }

    public StreamEndPoint_B create_B(StreamCtrl
       the_requester,VDevHolder the_vdev,
       org.omg.CORBA.StringHolder named_vdev, String[]
       the_spec) throws OPFailed, OPDenied, notSupported,
       noSuchFlow{
       System.out.println("Quien tiene que hacer esto es el
       ClienteMMDevie_impl");
       return null;
    }

   public StreamCtrl bind(MMDevice peer_device,
      org.omg.CORBA.BooleanHolder is_met, String[]
      the_spec) throws OPFailed, noSuchFlow {
      StreamCtrl streamCtrl = create_StreamCtrl();
      System.out.println("StreamCtrl creado Mediante los
      MMDevices...");
      return streamCtrl;
      }
```

```
    public StreamCtrl bind_mcast(MMDevice first_peer,
       org.omg.CORBA.BooleanHolder is_met, String[]
       the_spec)
       throws OPFailed, noSuchFlow {
       throw new OPFailed("");
      }

    public void destroy(StreamEndPoint the_ep, String
        vdev_name)
        throws notSupported{
        }

    public String add_fdev(org.omg.CORBA.Object the_fdev)
        throws notSupported, OPFailed {
        throw new OPFailed("");
        }

    public org.omg.CORBA.Object getVdev() throws
        notSupported, noSuchFlow{
        return vdev;
        }

    public org.omg.CORBA.Object getStreamEndPoint() throws
       notSupported, noSuchFlow{
      return sep;
      }

       // A continuación se implementaría todos los metodos
          de CosPropertySevice

    public void set_this(MMDevice _this){
        this._this = _this;
    }

    private StreamCtrl create_StreamCtrl(){
         StreamCtrl_impl streamCtrl_impl = new
         StreamCtrl_impl(orb, property_factory);
         StreamCtrlPOATie streamCtrlPOA = new
         StreamCtrlPOATie(streamCtrl_impl);
         StreamCtrl streamCtrl = streamCtrlPOA._this(orb);
         System.out.println("Creando StremCtrl en
         métodos....");
         streamCtrl_impl.set_this(streamCtrl);
         return streamCtrl;
        }
}
```

3.4.1.j.- CLIENTEMMDEVICE_IMPL

Esta clase basicamente implementa el metodo createB cuya implementación y funcionamiento se desccribe posteriormente.

```
package AVStreams;
import javax.media.*;
import CosPropertyService.*;
import org.omg.CORBA.*;
import ClienteCorba.*;

public class ClienteMMDevice_impl extends MMDevice_impl{
    private String network_address = null;
    private VDev vdev = null;
    private int port_base;
    private ClienteVDev_impl vdev_impl = null;
    private StreamEndPoint_B sep_B = null;
    private StreamEndPointB_impl sep_B_impl = null;
    private ClienteCorba.CPrincipal application = null;

    public ClienteMMDevice_impl(org.omg.CORBA.ORB orb,
           PropertySetDefFactoryImpl property_factory,
           ClienteCorba.CPrincipal application, String
           network_address, int port_base){

        super(orb, property_factory);
        this.application = application;
        this.network_address = network_address;
        this.port_base = port_base;
    }

    public StreamEndPoint_A create_A(StreamCtrl
           the_requester, VDevHolder the_vdev,
           org.omg.CORBA.BooleanHolder met_qos,
           org.omg.CORBA.StringHolder named_vdev,
           String[] the_spec) throws OPFailed, OPDenied,
           notSupported, noSuchFlow{
        throw new notSupported("Este dispositivo solo
        soporta SEP_B");
    }

    public StreamEndPoint_B create_B(StreamCtrl
           the_requester, VDevHolder the_vdev,
           org.omg.CORBA.BooleanHolder met_qos,
           org.omg.CORBA.StringHolder named_vdev, String[]
           the_spec)
        throws OPFailed, OPDenied, notSupported,
        noSuchFlow{

         vdev_impl = new ClienteVDev_impl(orb,
         property_factory, the_spec);
```

```
VDevPOATie vdev_tie = new VDevPOATie(vdev_impl);
vdev = vdev_tie._this(orb);
sep_B_impl = new StreamEndPointB_impl(orb,
property_factory);
StreamEndPoint_BPOATie sep_B_tie = new
StreamEndPoint_BPOATie(sep_B_impl);
sep_B = sep_B_tie._this(orb);

sep_B_impl.set_this(sep_B);
sep_B_impl.set_related_vdev_impl(vdev_impl);

          //Para VDev
try{
    // para el vdev define una propiedad referente
       al sep y otra al mmdevice

    //sep
    org.omg.CORBA.Any any_sep = orb.create_any();
    StreamEndPointHelper.insert(any_sep, sep_B);
    vdev.define_property("Related_StreamEndPoint",
    any_sep);

    //mmdev
    org.omg.CORBA.Any any_mmdev = orb.create_any();
    MMDeviceHelper.insert(any_mmdev, _this);
    vdev.define_property("Related_MMDevice",
    any_mmdev);
}
catch (Exception e){
    throw new OPFailed("No se pueden establecer las
    propiedades para VDev");
}
if (!vdev_impl.set_properties()){
    throw new OPFailed("No se pueden establecer las
    propiedades para VDev");
}


     /* Para sepB */

try{

    // sepB -> se le asocia el VDev correspodiente
    org.omg.CORBA.Any any_vdev = orb.create_any();
    VDevHelper.insert(any_vdev, vdev);
    sep_B.define_property("Related_VDev",
    any_vdev);

    // sepB -> almacena el tipo de flujo soportado
    (video)
    org.omg.CORBA.Any any_flows = orb.create_any();
```

```
            String[] flows = new String[1];
            flows[0] = "video";
            flowSpecHelper.insert(any_flows, flows);
            sep_B.define_property("Flows", any_flows);

            // sepB -> Almacena la dirección del flujo
            org.omg.CORBA.Any any_video_dir =
            orb.create_any();
            any_video_dir.insert_string("in");
            sep_B.define_property("direcc_flujo",
            any_video_dir);
```

Se crea un tipo de datos any, al que insertamos un string cuyo valor es el de la dirección del flujo de datos en el cliente.En este caso el flujo de datos en el cliente será entrante.

```
            // sepB -> Alamacena el formato común
            org.omg.CORBA.Any any_video_currFormat =
            orb.create_any();
            any_video_currFormat.insert_string("RTP");
            sep_B.define_property("video_currFormat",
            any_video_currFormat);
```

Se crea un tipo de datos any, al que insertamos un string cuyo valor es el que comparten tanto cliente como servidor.
En este caso y para una mayor simplificación hemos puesto “RTP”. Difinimos un nombre para esta propiedad y se la asocimos a la clase sep_B.

```
            // sepB -> almacena la dirección y puerto.
            org.omg.CORBA.Any any_video1_address =
            orb.create_any();
            any_video1_address.insert_string("RTP=" +
            network_address + ":" + port_base);
            sep_B.define_property("video1_address",
            any_video1_address);

            // sepB -> Protocolos disponibles
            org.omg.CORBA.Any any_AvailableProtocols =
            orb.create_any();
            String[] prots = {"RTP"};

protocolSpecHelper.insert(any_AvailableProtocols, prots);
            sep_B.define_property("AvailableProtocols",
            any_AvailableProtocols);

            // sepB -> Protocolo de restriccion
            org.omg.CORBA.Any any_ProtocolRestriction =
            orb.create_any();
```

```
            String[] restriccion = new String[0];

protocolSpecHelper.insert(any_ProtocolRestriction,
            restriccion);
            sep_B.define_property("ProtocolRestriction",
            any_ProtocolRestriction);
        }

        catch (Exception e){
            throw new OPFailed();
        }

        the_vdev = new VDevHolder(vdev);
        System.out.println("...create B operation
        returns.");
        return sep_B;
    }

    public boolean set_properties(String f){
        // tipo de flujo
        org.omg.CORBA.Any any_flows = orb.create_any();
        String[] flow_names = {"video"};
        flowSpecHelper.insert(any_flows, flow_names);

        // direccion del flujo (entrada)
        org.omg.CORBA.Any any_v_out = orb.create_any();
        any_v_out.insert_string("in");

        // formatos disponibles.
        org.omg.CORBA.Any any_v_formats = orb.create_any();
        any_v_formats.insert_string(f);

        try{
            define_property("Flows", any_flows);
            define_property("direcc_flujo", any_v_out);
            define_property("video_availableFormats",
            any_v_formats);
            return true;
        }
        catch (Exception e){
            return false;
        }
    }
}
```

3.4.2.-Paquete ServidorCorba

A continuación explicaremos las clases implementadas que se ejecutaran en el servidor.

3.4.2.a.-SERVIDOR

```
public class Servidor{
        private org.omg.CORBA.ORB orb = null;
        private org.omg.PortableServer.POA rootPOA = null;
        private org.omg.PortableServer.POAManager
        poaManager = null;
        private NamingContext nc = null;
        private PropertySetDefFactory property_factory =
        null;
        private MMDevice mmdev = null;
        private ServidorMMDevice_impl smmdev=null;
        private String a;
        private String hostCliente;
        private String c;
        private PropertySetDefFactoryImpl psdfi;

   public Servidor(){
          hostCliente="localhost";
```

Constructor de la clase servidor.

```
   public static void main(String args[]){
          Servidor s= new Servidor();
          s.start(args);
         }
```

Este código es el primero que se ejecuta cuando se arranca el servidor, tras crear un objeto de la clase servidor, se llama a la clase start, que implementamos y describimos a continuación.

```
   public void start(String[] args){
           if (args.length > 1)
               hostCliente= args[0];

        int status = 0;
        try{
            orb = org.omg.CORBA.ORB.init(args,null );
            status = run( orb, args );
        }
        catch( Exception ex ){
            ex.printStackTrace();
            status = 1;
        }
        System.out.println("POR FIN");
        java.lang.Object sync=new java.lang.Object();
        System.out.println("Esperando clientes...");
        synchronized(sync){
```

```
            try{
            sync.wait();
            System.out.println("Esto no deberia hacerlo");
            }catch(Exception e){}
      }

        if ( orb != null ){
            System.out.println(orb);
             try{
                orb.destroy();
             }
             catch( Exception ex ){
                ex.printStackTrace();
                status = 1;
        }

        System.exit( status );
        }
    }

        // get root POA

        try{
         rootPOA = org.omg.PortableServer.POAHelper.narrow(
         orb.resolve_initial_references( "RootPOA" ) );
         poaManager = rootPOA.the_POAManager();
        }
        catch( org.omg.CORBA.ORBPackage.InvalidName ex ){
            System.out.println("***Can't resolve
            `rootPOA'");
            return 1;
        }

        // get naming service

        org.omg.CORBA.Object nameObj = null;
        try{
            nameObj =
            orb.resolve_initial_references("NameService");
            System.out.println(nameObj);
        }
        catch( org.omg.CORBA.ORBPackage.InvalidName ex ){
            System.out.println("***Can't resolve
            `NameService'.");
            return 1;
        }
        nc = NamingContextHelper.narrow(nameObj);

        // get property service
        org.omg.CORBA.Object propObj = null;
        psdfi=new PropertySetDefFactoryImpl(orb);
```

```
        String a = crearServidorMMDevice();
```

```
        try{
        poaManager.activate();
        }
        catch(
   org.omg.PortableServer.POAManagerPackage.AdapterInactive
   ex ){
        return 1;
        }
        System.out.println("Esperando a clientes.....");
        orb.run();
        return 0;
       }

    public String crearServidorMMDevice(){
        InetAddress host = null;
        try{

           System.out.println("");
            if (hostCliente != null)
                host = InetAddress.getByName(hostCliente);
            else
                host = InetAddress.getLocalHost();
        }
        catch (UnknownHostException uhe){
            return "Fallo al crear servidor MMDevice";
        }
           ServidorMMDevice_impl mmdev_impl = new
           ServidorMMDevice_impl(orb, psdfi, this,
           host.getHostAddress(), 26000);
           MMDevicePOATie mmdev_poatie = new
           MMDevicePOATie(mmdev_impl);
           mmdev = mmdev_poatie._this(orb);
           mmdev_impl.set_this(mmdev);
           mmdev_impl.set_properties();
      System.out.println("ServidorMMDevice_impl
creado....");

            try{
            NameComponent comp[] = new NameComponent[1];
            comp[0] = new NameComponent("MMDevice","");
            nc.rebind(comp,mmdev);
            System.out.println("MMDevice en Servidor de
            Nombres ");
        }
       catch(Exception e){
           System.out.println("Algo falla");
    }
   return "va bien";
```

El método crear ServidorMMDevice crea un objeto MMDevice_impl y lo añade al servidor de nombres para que el cliente pueda obtener una referencia de esta clase.

2.3.4.b.-SERVIDORRGB

```
public class ServidorRGB implements Runnable{
   private int tamaño;
   private FrameGrabber fg;
   // argado de la captura de datos
   private InetAddress d_ip;
   private DatagramSocket serverSocket;
   private byte[] alin;
   private int i;
   private DatagramPacket sync;
   private Thread t;
   private Buffer B;
   private Object O;
   private String IP;
   // Cliente
   private int par;
   private int puerto;

      public ServidorRGB(int t, String ip, int p){
         tamaño=t;
         IP=ip;
         puerto=p;
         par=2;
         d_ip=null;
         serverSocket=null;
         fg=null;
         alin=new byte[6];
         alin[0]=1;
         alin[1]=0;
         alin[2]=1;
         alin[3]=0;
         alin[4]=1;
         alin[5]=0;
         System.out.println("Dentro del ServidorRGB");

                        try{
            fg=new FrameGrabber();

System.out.println("FrameGrabber iniicializado");
            d_ip=InetAddress.getByName("localhost");
            serverSocket=new DatagramSocket();
            serverSocket.setSendBufferSize(10*(230400+8));
            }catch(Exception e){
         }
         i=230400/tamaño;
         play();
```

```
            }
        public void play(){
            t = new Thread(this);
            t.start();
            }
        public void stop(){
            t.stop();
            serverSocket.close();
            fg.stop();
            }
        public void pause(){
            t.stop();
            }
        public void resume(){
            t=new Thread(this);
            t.start();
            }

        public void run(){

                        while(true){
                        B=null;
            try{
                            System.out.println("Se va a iniciar
getBuffer En SRGB");
            B=fg.getBuffer();
            }catch(Exception e){
            System.out.println(e.getMessage());
                            }

            O=B.getData();
            byte datos[]=(byte[])O;
            byte aux[]=null;

            if(par%2==0){
               par=1;
               }
            else{
               par=par+1;
               }
            try{
               sync=new DatagramPacket(alin,6,d_ip,2020);
               serverSocket.send(sync);
               }catch(Exception e){
               System.out.println(e.getMessage()+"Error al
               enviar sincronismo");
            }

            for(int a =0;a<i;a++){
               try{
               aux=new byte[tamaño];
```

```
            System.arraycopy(datos,tamaño*a,aux,0,tamaño);
            //DatagramPacket sendPacket=new
            DatagramPacket(aux,tamaño,d_ip,2020);
            serverSocket.send(sync);

                                }catch(Exception e){
               System.out.println("Error");
               System.exit(0);
               }

            try{
               t.sleep(10/i);
               }catch(Exception e){
               System.out.println("ERROR");
            }
                        }
      }
   }
}
```

3.4.2.c.-SERVIDORRTP

```
public class ServidorRTP{ //extends Thread{
   private MediaLocator locator;
   // dispositivo de captura
   private String d_ip;
   // destino -> Cliente
   private String puerto;
   private String formato;
   private Integer stateLock=new Integer(0);
   private Processor processor=null;
   private DataSink rtptransmisor=null;
   //lectura de datos
   private DataSource dataOutput=null;
   //como acceder a los datos

   public ServidorRTP(String format, String ip, String
      port){
      formato="JPEG/RTP";//=format;
      locator=new MediaLocator("vfw://0");
      d_ip=ip;
      puerto="2020";//port;
      System.out.println("Dentro del servidor RTP");
      start();
        }
   public synchronized String start(){
```

```
\\ssmjv01dat02\
      String result;

      result=createProcessor();

      if(result!=null)
      return result;

      result=createTransmitter();
      if(result!=null){
         processor.close();
         processor=null;
         return result;
         }

         //processor.configure();
                        processor.start();
         return null;
      }

      private String createProcessor(){
                     System.out.println("Creando
                     Processor");
                     if(locator==null)
         return "Locator nulo";
         DataSource ds;
         DataSource clone;

         try{
                        ds =
         Manager.createDataSource(locator);
         }catch(Exception e){
              return "No se puede crear DataSource";
              }

         try{
         processor = Manager.createProcessor(ds);
         }catch(Exception e){
         return "No puede crear processor";
                        }

         System.out.println("Encima de wait");
         boolean result = waitForState(processor,
         Processor.Configured);
                      if(result==false){
                         return "h";}

                       System.out.println("fuera de wait");
                       TrackControl[] tracks=
                       processor.getTrackControls();
```

```
          if(tracks == null || tracks.length<1)
          return "i";

          boolean programado= false;

          for(int i=0;i<tracks.length;i++){
             Format format=tracks[i].getFormat();
             if(tracks[i].isEnabled()&& format instanceof
             VideoFormat && !programado){
             Dimension size
             =((VideoFormat)format).getSize();
             float frameRate =
             ((VideoFormat)format).getFrameRate();
             int w = (size.width % 8 == 0 ? size.width:
                                  (int)(size.width/8)*8);
             int h = (size.height % 8 == 0 ? size.height:
                                  (int)(size.height/8)*8);
             VideoFormat jpegFormat=null;


if(formato.equals("RGB/RTP")){
             System.out.println("Dentro de VideoFormatRGB");
                                  jpegFormat=new
VideoFormat(VideoFormat.RGB, new Dimension(w,h),
Format.NOT_SPECIFIED, Format.byteArray, frameRate);
                }
else if (formato.equals("JPEG/RTP")){
                jpegFormat= new
VideoFormat(VideoFormat.JPEG_RTP,new Dimension(w,h),
Format.NOT_SPECIFIED, format.byteArray, frameRate);
                  }
                                  else
if(formato.equals("H.263/RTP")){
                jpegFormat = new
VideoFormat(VideoFormat.H263_RTP, new Dimension(w,h),
Format.NOT_SPECIFIED, Format.byteArray, frameRate);
                  }
                tracks[i].setFormat(jpegFormat);
                System.err.println("Video transmitiendo
                como: "+jpegFormat);
                programado = true;
                } else
                tracks[i].setEnabled(false);
             }

             if(!programado)
             return "No puede encontrar video track";

             ContentDescriptor cd = new
             ContentDescriptor(ContentDescriptor.RAW_RTP);
             processor.setContentDescriptor(cd);
```

```
            result= waitForState(processor,
            Controller.Realized);


                                if(result == false)
              //  return "No puede realizar el proccesor";

            setJPEGQuality(processor, 0.5f);

            dataOutput= processor.getDataOutput();
            return null;
   }// fin

//Crea una sesión RTP para transmitir imagenes

   private String createTransmitter(){
            System.out.println("Creando Transmisor");
      String rtpURL = "rtp://"+d_ip+":"+puerto+"/video";
      MediaLocator outputLocator = new
MediaLocator(rtpURL);

      try{
      rtptransmisor= Manager.createDataSink(dataOutput,
outputLocator);
      rtptransmisor.open();
      rtptransmisor.start();
      dataOutput.start();
      }catch(Exception e){}
      return null;
                }

        public void setJPEGQuality(Player p,float val){
        Control es[]=p.getControls();
        QualityControl qe=null;
        VideoFormat jpegFmt = new
VideoFormat(VideoFormat.JPEG);
        for(int i=0;i<es.length;i++){
            if(es[i]instanceof Owned){
            Object owner=((Owned)es[i]).getOwner();
                if(owner instanceof Codec){
                Format
fmts[]=((Codec)owner).getSupportedOutputFormats(null);
                for(int j = 0;j<fmts.length;j++){
                if(fmts[j].matches(jpegFmt)){
                qe=(QualityControl)es[i];
                qe.setQuality(val);
                break;
                }
                }
                }
            if(qe!=null)
```

```
                break;
            }
        }
    }

//Para la transmisión y cierra la sesion RTP

   public void stop(){
      synchronized(this){
         if(processor !=null){
            processor.stop();
            processor.close();
            processor=null;
            rtptransmisor.close();
            rtptransmisor=null;
            }
         }
      }
//Pausa la transmisión del flujo de datos

   public void pause(){
      processor.stop();
      }
// Reanuda la transmisión
   private boolean failed=false;

        public void resume(){
      processor.start();
      }
        public Integer getStateLock(){
        return  stateLock;
        }

        public void setFailed(){
        failed=true;
        }



     private synchronized boolean waitForState(Processor
p,int state){
        p.addControllerListener(new StateListener());
        failed=false;
        if(state==Processor.Configured){
        p.configure();
        }else if (state==Processor.Realized)
        p.realize();

        while(p.getState()<state&&!failed){
        synchronized(getStateLock()){
        try{
```

```
        getStateLock().wait();
        }catch(Exception e){
        return false;
        }
        }
        }
        if(failed)
            return false;
        else
            return true;
     }

class StateListener implements ControllerListener{
        public void controllerUpdate(ControllerEvent ce){
                if (ce instanceof ControllerClosedEvent){
                setFailed();

System.out.println("***ControllerClosedEvent: " + ce);
            }
            if (ce instanceof ControllerEvent){
                synchronized (getStateLock()){
                    getStateLock().notifyAll();
                }
            }
             }
          }
        }
```

3.4.2.d.-FRAMEGRABBER

La clase FrameGrabber es la clase encargada de la captura de datos. Al igual que en la clase ServidorRTP existe un método createProcessor encargado de tal captura a traves del MediaLocator.

Para el caso de una cámara web el medialocator viene definido por el dispostivo vfw://0 que extrae datos de la capturadora de vídeo de la máquina servidor

```
public class FrameGrabber implements ControllerListener{
   private final static String DEFAULT_DEV_NAME="vfw://0";
   private Object stateLock=new Object();
   private PushBufferStream camStream;
   private BufferToImage converter;
   private RGBFormat RGBF=null;
   private Processor deviceProc;
   private DataSource formattedSource;
   private PushBufferDataSource source;

      public FrameGrabber(){
         String cameraDevice="vfw://0";
         Dimension viewSize=null;
```

```
          int viewDepth=0;
          int xRes = 320;
          int yRes = 240;
          viewSize = new
          Dimension(xRes,yRes);
          viewDepth=8;
          RGBFormat userFormat=null;
          userFormat=new
RGBFormat(viewSize,Format.NOT_SPECIFIED,Format.byteArray,25
          ,viewDepth,10,10,10);

          MediaLocator loc = new
          MediaLocator("vfw://0");

           if(loc==null)
           System.out.println("No hay camara");
           formattedSource=null;


             try{
             formattedSource=Manager.createDataSource(loc);
             }catch(Exception e){}

             if(!(formattedSource instanceof CaptureDevice))
             System.out.println("Tampoco hay camara");
             FormatControl[] fmtControls =

       ((CaptureDevice)formattedSource).getFormatControls();

             Format setFormat = null;

             for(int i = 0;i<fmtControls.length;i++){
               if(fmtControls[i]==null)
continue;

if((setFormat=fmtControls[i].setFormat(userFormat))!=null)
               break;
                  }
               if(setFormat==null)
               System.out.println("Error en FG");

               try{
               formattedSource.connect();
               }catch(Exception e){}

               deviceProc=null;

               try{

deviceProc=Manager.createProcessor(formattedSource);
               }catch(Exception e){}
```

```
            deviceProc.addControllerListener(this);
            deviceProc.realize();
            System.out.println("Realizando Captura");


while(deviceProc.getState()!=Controller.Realized){
            synchronized(stateLock){
            try{
            stateLock.wait();
            }catch(Exception e){
            }
           }
          }
       deviceProc.start();
       source=null;
       try{

source=(PushBufferDataSource)deviceProc.getDataOutput();
              }catch(Exception e){}

         PushBufferStream[] streams=source.getStreams();
         camStream=null;

         for(int i =0;i<streams.length;i++){
         if(streams[i].getFormat() instanceof RGBFormat){
         camStream=streams[i];
         RGBFormat rgbf=(RGBFormat)streams[i].getFormat();
         RGBF=rgbf;
         converter = new BufferToImage(rgbf);
         break;
         }
        }

         System.out.println("Captura Iniciada");
         }

        public Image getImage(){
            Buffer b = new Buffer();
            try{
            camStream.read(b);
            }catch(Exception e){}
            Image i = converter.createImage(b);
            return i ;
            }

         public Buffer getBuffer(){
            Buffer b = new Buffer();
            try{
            System.out.println("Capturando Imagen...");
            camStream.read(b);
```

```
            }catch(Exception e){}
            return b;
            }

         public RGBFormat getFormat(){
            return RGBF;
            }

         public void controllerUpdate(ControllerEvent ce){
            if(ce instanceof RealizeCompleteEvent){
            System.out.println("Realizando Trasnmision");
            synchronized(stateLock){
            stateLock.notifyAll();
              }
             }
           }

         public void stop(){
            deviceProc.stop();
            formattedSource.disconnect();
            source.disconnect();
            }

         }
```

3.4.2.e.-VÍDEO

Esta clase es la encargada de enlazar el streamEndPoint con el ServidorRGB y ServidorRTP según corresponda el formato con el que se quiere transmitir.

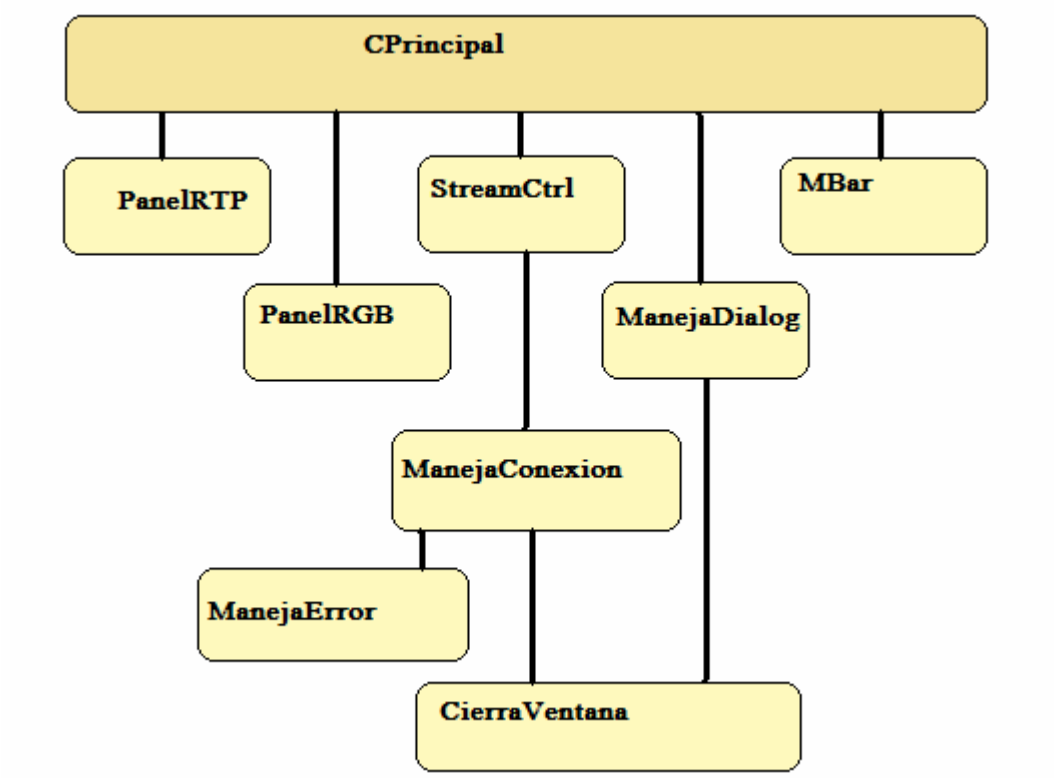
```
public class Video{
   private String tipo;
   private String formato;
   private String IP;
   private NamingContext nc = null;
   private ServidorRTP srtp;
   private ServidorRGB srgb;
   private int puerto;

   public Video(String t,String f,String direcc,int port){
      super();
      tipo=t;
      formato=f;
      IP=direcc;
      puerto=port;
      System.out.println("Creando clase Video");
      }

   public void play(){
```

```
      if(tipo.equals("RGB")){
         if(formato.equals("RGB 57600")){
            srgb=new ServidorRGB(57600,IP,puerto);
            }
         if(formato.equals("RGB 28800")){
            srgb=new ServidorRGB(28800,IP,puerto);
            }
         if(formato.equals("RGB 14400")){
            srgb=new ServidorRGB(14400,IP,puerto);
            }
         if(formato.equals("RGB 7680")){
            srgb=new ServidorRGB(7200,IP,puerto);
            }
         if(formato.equals("RGB 5760")){
            srgb=new ServidorRGB(3600,IP,puerto);
            }
         if(formato.equals("RGB 4608")){
            srgb=new ServidorRGB(3600,IP,puerto);
            }
         if(formato.equals("RGB 2304")){
            srgb=new ServidorRGB(2304,IP,puerto);
            }
         if(formato.equals("RGB 1152")){
            System.out.println("Va a entrar aqui 1152");
            srgb=new ServidorRGB(1152,IP,puerto);
            }
                }
         else{
            System.out.println("Llamando a Servidor RTP");
            srtp=new ServidorRTP("UDP",IP,"puerto");
            srgb=null;
      }
   }catch(Exception e){}
}

   public void stop(){
      if (srtp==null){
         srgb.stop();
         srgb=null;
         }
         else{
            srtp.stop();
            srtp=null;
            }
         System.runFinalization();
      }

   public void pause(){
      if(srtp==null){
         srgb.pause();
```

```
            }
         else{
            srtp.pause();
            }
         }

      public void resume(){
         if(srtp==null){
           srgb.resume();
            }
            else{
               srtp.resume();
               }
         }
}
```

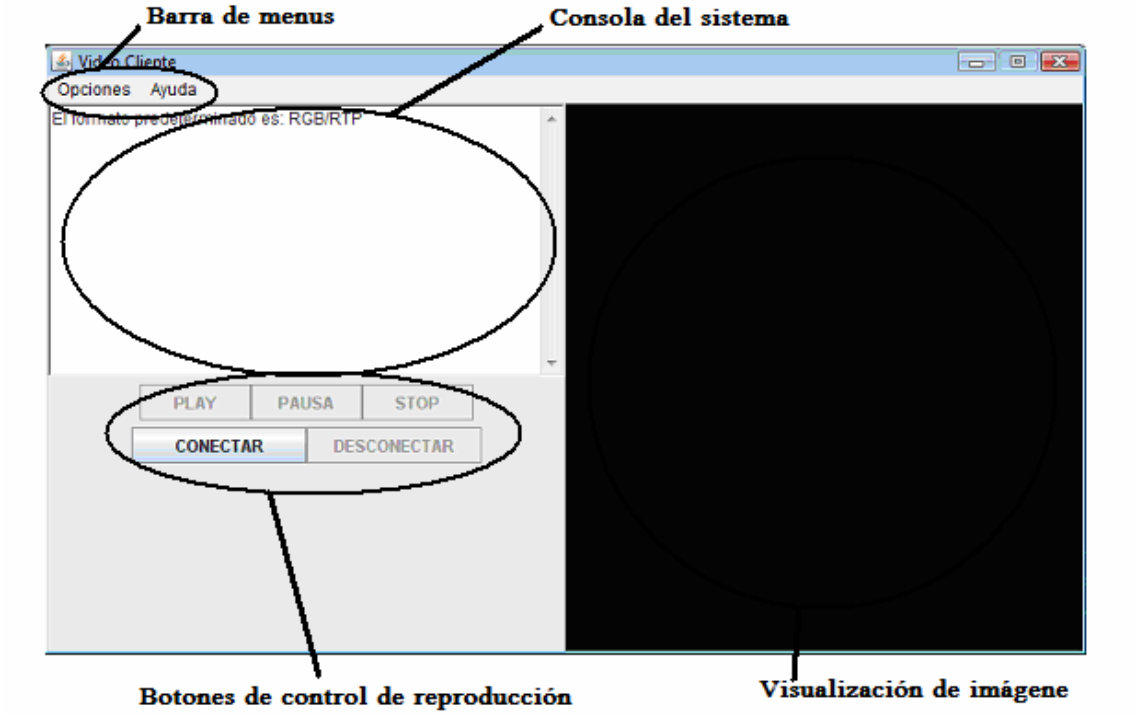
CLIENTE

A lo largo de este apartado se va a explicar todas las clases de la aplicación que se encuentran en el lado del cliente. En la siguiente figura se muestra el esquema de clases de Aplicación en el lado del cliente:



3.4.3.a- CPRINCIPAL

CPrincipal es la clase principal del entorno gráfico del cliente. El entorno gráfico que utiliza el cliente es el siguiente:



A continuación se va a explicar la implementación de todas las clases que forman el cliente, para ello comentaremos las partes que forman el interfaz gráfico del cliente.

- Frame principal : Se trata del contenedor principal donde se situan todos los elementos que conforman el interfaz gráfico
- Barra de menús : Esta barra de menús ppermite al cliente elegir entre los distintos protocolos de transmisión.
- Botones de control de reproducción: Estos botones permiten controlar el flujo de datos. Permiten tanto conectarse a un servidor como comenzar,pausar y parar una reproducción.
- Zona de visualización de imágenes : En esta zona se muestran las imágenes recibidas.
- Consola del sistema: Informa al usuario de todos los eventos ocurridos.

Con todo esto podemos concluir que la clase CPrincipal consiste en un Frame que además de interactuar con el cliente permite que este controle la comunicación multimedia.

```
public class CPrincipal extends JFrame{
     private org.omg.CORBA.ORB orb = null;
     private org.omg.PortableServer.POA rootPOA = null;
     private org.omg.PortableServer.POAManager
```

```
    poaManager = null;
    private NamingContext nc = null;
    private PropertySetDefFactory property_factory =
    null;
    public String ipremota;
    public String iplocal;
    private String tipo;     // Puede ser RGB o RTP
    private MBar mb;         // Barra de menú
    private TextArea ta;     // Area de Texto
    private MMDevice mmdev_c = null;
    private ClienteMMDevice_impl mmdev_impl=null;
    private MMDevice mmdev_s = null;
    private StreamEndPointA_impl sepa = null;
    public StreamEndPointB_impl sepb=null;
    private StreamCtrl_impl streamc = null;
    private AVStreams.StreamCtrl sct = null;
    private Panel pv;
    private String consola;
    // Lo que pondremos en el area de texto
    private PanelRGB prgb=null;    // Panel para RGB
    private PanelRGB prgba=null;
    private PanelRTP prtp=null;
    private String []formato = new String[1];
    private ClienteCorba.StreamCtrl sc;
    private JPanel contenedor;  // Contenedor Principal
    private boolean reproduciendo;
    // Para comprobar si está reproduciendo
     Panel Virtual donde se verá el video
    private PropertySetDefFactoryImpl a = null;
    private StreamCtrl_impl impl;
    private String flu[];

public CPrincipal(String[] host){
     start(host);
      }

public void Inicio(String host){
     setTitle("Video Cliente");
     mb=new MBar(this);
     setMenuBar(mb);
     resize(750,440);
     setResizable(false);
     setLayout(new GridLayout(1,2));
     // 1 Fila y 2 Columnas
     tipo="RGB/RTP";
     consola="El formato predeterminado es: "+tipo;
     ta=new
   TextArea(consola,10,40,ta.SCROLLBARS_VERTICAL_ONLY);
     ta.setCursor((new Cursor(Cursor.TEXT_CURSOR)));
     ta.setEditable(true);      // No permite escribir
     ta.setBackground(Color.white);
```

```
        reproduciendo= false;
        pv= new Panel();
        pv.setBackground(Color.black);
        contenedor=new JPanel();
        contenedor.setLayout(new GridLayout(2,1));
        sc= new StreamCtrl(this, host);
        sc.setSize(new Dimension(320,100));

        WindowListener windowListener = new
        WindowAdapter(){
        public void windowClosing(WindowEvent e){
           hide();
           dispose();
           System.exit(0);
           }
        };
        addWindowListener(windowListener);

        setBackground(Color.black);
        reproduciendo=false;
        contenedor.add(ta);
        contenedor.add(sc);
        add(contenedor);
        add(pv);
        show();
        }//fin constructor


    /* Método para capturar las acciones de la barra de
    menu */

public boolean handleEvent(Event e){

         if(e.target instanceof CheckboxMenuItem){
             String a = tipo;
             tipo = mb.desactivar(tipo);
    if(!tipo.equals(a)){
    setConsola(getConsola()+"\nSe ha seleccionado el
    formato "+tipo);
    actualizaConsola();
     }
 }
     else if (e.target instanceof MenuItem){
             if(e.arg.equals("Salir")){
             hide();
             dispose();
             System.exit(0);
              }

             else if(e.arg.equals("Ayuda")){
             Runtime rt;
```

```
            ManejaDialog md=new
            ManejaDialog(this,"Ayuda","Pedro Zamora
            Prades");
            setConsola(getConsola()+"\nAyuda ...");
            actualizaConsola();
                                }
            else if(e.arg.equals("Acerca de ...")){
            ManejaDialog md=new ManejaDialog(this,"Acerca
            de ...","Acerca del prototipo de Video 1.0");
            setConsola(getConsola()+"\nSe ha consultado
            Acerca de ...");
            actualizaConsola();
          }
         }
      return super.handleEvent(e);
      }//fin metodo

  public void setConsola(String c){consola=c;}
  public String getConsola(){return consola;}
  public String getTipo(){return tipo;}
  public void setOption(boolean b){mb.setOption(b);}
  public void actualizaConsola(){
  ta.replaceText(consola,0,consola.length());}

  public void setPanel(String ip, int port){
      System.runFinalization();
      System.gc();

      setConsola(getConsola()+"\nCargando Panel...");
      actualizaConsola();
      setConsola(getConsola()+"\nConectando con la IP
      "+ip);
      actualizaConsola();
```

Se comprueba que formato de trasmision ha elegido el cliente para la transmision y recepcion de imágenes. En caso de tratarse del formato RGB sobre UDP , crea una clase PanlRGB indicandole el tamaño del paquete y el puerto por el que se quieren recibir los datos.

```
      if(tipo=="RGB 57600"){
          prgb=new PanelRGB(57600,port);
          prtp=null;
          }

      else if (tipo=="RGB 38400"){
          prgb=new PanelRGB(8400,port);
          prtp=null;
          }

      else if (tipo=="RGB 28800"){
          prgb=new PanelRGB(28800,port);
```

```
            prtp=null;
            }

        else if (tipo=="RGB 14400"){
            prgb=new PanelRGB(14400,port);
            prtp=null;
            }

         ...
         ...
         ...

        else{
           try{
            System.out.println("Entrando en panel RTP");
            prtp=new PanelRTP(iplocal,port);
            prgb=null;
            }catch (Exception e){
            sc.setCancel();
                   }
             }

            setMenuBar(mb);
            setLayout(new GridLayout(1,2));

            if(prgb==null){
            System.out.println("PRTP");
            remove(pv);
            add(prtp);
            prtp.play();
            contenedor.remove(sc);
            contenedor.add(sc);
            }

            else{
            System.out.println("PRGB");
            remove(pv);
            add(prgb);
            prgb.play();
            contenedor.remove(sc);
            contenedor.add(sc);
            }

            show();
            setConsola(getConsola()+"\nReproduciendo");
            actualizaConsola();
            }



      public void setStop(){
```

```
            if(prgb==null){
            remove(prtp);
            prtp.stop();
            prtp=null;
            }
            else{
            remove(prgb);
            prgb.stop();
            prgb=null;
            }

            add(contenedor);
            add(pv);

            show();
            setConsola(getConsola()+"\n Parando.....");
            actualizaConsola();
            stop();
            System.runFinalization();
            }

        public void setPause(){
            if(prgb==null){
            prtp.pause();
            }
            else{
            prgb.pause();
            }
            pause();
            }

        public void setResume(){
            if(prgb==null){
            prtp.resume();
            }
            else{
            prgb.resume();
               }
            }

    public static void main(String []args){
            CPrincipal cp= new CPrincipal(args);
            }

    public void start(String[] args){

        int status = 0;


        try{
```

```
        // Inicializa un ORB y retorna una referencia
        pseudo-objeto a este ORB.

        orb = org.omg.CORBA.ORB.init( args, null );
        status = run( orb, args );
        }
        catch( Exception ex ){
        ex.printStackTrace();
        status = 1;
        }

      if ( orb != null ){
         try{
         orb.destroy();
           }

      catch( Exception ex ){
         ex.printStackTrace();
         status = 1;
         }

      System.exit( status );
        }
    }


  int run(org.omg.CORBA.ORB orb, String[] args){

      // get root POA

      try{
      rootPOA = org.omg.PortableServer.POAHelper.narrow(
      orb.resolve_initial_references( "RootPOA" ) );
      poaManager = rootPOA.the_POAManager();
      }
      catch( org.omg.CORBA.ORBPackage.InvalidName ex ){
      System.out.println("***Can't resolve `rootPOA'");
      return 1;
      }

      // get naming service

      org.omg.CORBA.Object nameObj = null;
      try{
      nameObj =
      orb.resolve_initial_references("NameService");
      }

      catch( org.omg.CORBA.ORBPackage.InvalidName ex ){
          System.out.println("***Can't resolve
          `NameService'.");
```

```
            return 1;
            }

        if (nameObj == null){
            System.out.println("***`NameService' is a nil
            object reference.");
            return 1;
        }
        nc = NamingContextHelper.narrow(nameObj);
        if (nc == null){
            System.out.println("`NameService' is not " +
            "a NamingContext object reference\n");
            return 1;
        }

        // get property service

        org.omg.CORBA.Object propObj = null;

        a = new PropertySetDefFactoryImpl(orb);
        try{
          crearClienteMMDevice();
          poaManager.activate();
          }
        catch(
org.omg.PortableServer.POAManagerPackage.AdapterInactive ex
){
        return 1;

        }
        Inicio("127.0.0.1");
        orb.run();
        return 0;
        }

        public String crearClienteMMDevice(){
        InetAddress host = null;

        try{
           host = InetAddress.getLocalHost();
           iplocal=host.getHostAddress();
           }
        catch (UnknownHostException uhe){
           return "Fallo al crear ClienteMMDevice";
        }

        mmdev_impl = new ClienteMMDevice_impl(orb,
        a, this, host.getHostAddress(), 4321);

        MMDevicePOATie mmdev_poatie = new
        MMDevicePOATie(mmdev_impl);
```

```
        mmdev_c = mmdev_poatie._this(orb);
        mmdev_impl.set_this(mmdev_c);

((ClienteMMDevice_impl)mmdev_impl).set_properties(tipo);


        try
        {
        NameComponent[] name2 = new NameComponent[1];
        name2[0] = new NameComponent();
        name2[0].id = "ClienteMMDevice_impl";
        name2[0].kind = "";
        nc.rebind( name2, mmdev_c );
        return "Creando ClienteMMDevice correctamente";
        }
        catch(Exception e)
        {
            return "fallo 2";
        }
    }
```

Se ha añadido el objeto al servidor de nombres.

```
   public String obtenerServidorMMDevice(String host, int
        puerto){
        try
        {
          NameComponent[] name = new NameComponent[1];
          name[0] = new NameComponent("StreamCtrl","");

          org.omg.CORBA.Object obj = nc.resolve( name );
          sct = StreamCtrlHelper.narrow( obj );

          NameComponent[] name1 = new NameComponent[1];
          name1[0] = new NameComponent("MMDevice","");

          org.omg.CORBA.Object obj1 = nc.resolve( name1 );
          mmdev_s=MMDeviceHelper.narrow( obj1);

          return "Resolviendo StreamCtrl  y MMDevice
          correctamente";

        }
        catch(Exception e)
        {
        return "fallo 3";
              }
        }

    public String bindMMDevices(){
        if (mmdev_s == null || !(mmdev_s instanceof
```

```
      MMDevice))
      System.out.println("fallo del mmdev_s");
      if (mmdev_c == null || !(mmdev_c instanceof
      MMDevice))
      System.out.println("fallo del mmdev_c");
      try{
      org.omg.CORBA.BooleanHolder b = new
      org.omg.CORBA.BooleanHolder();
      String flows[]=new String[1];
      flows[0]="video";

      if(sct==null){
      System.out.println("Sct = null");
      }

      boolean a = sct.bind_devs(mmdev_s,mmdev_c,flows);

      return "Enlace de MMDevices correcto";
      }
      catch (Exception e)
      {
      return "Fallo al conectar ambos MMDevices";
          }
        }

 public void pause(){
      sct.destroy(null);
    }

 public void stop(){
    sct.stop(flu);
         }

 public void start(){
     flu = new String[3];
     flu[0]=tipo;
     flu[1]=iplocal;
     flu[2]=ipremota;
     setPanel(ipremota,2020);
     sct.start(flu);
     }
     else{
     setConsola(getConsola()+"\nNo se puede conectar con
     la ip:"+ipremota+"\nPare y luego desconecte para
     volver a intentarlo");
     actualizaConsola();
    }
}
```

El metodo start() crea un array con la siguiente información:

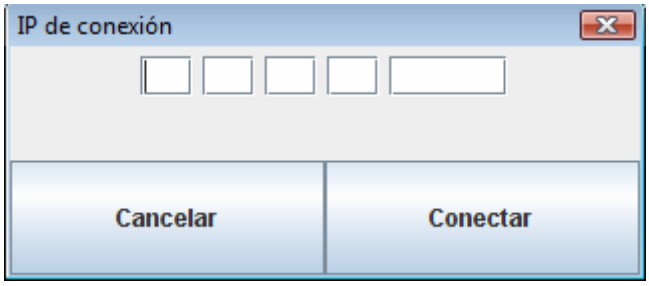
- Formato de transmisión
- Direccion IP local
- Direccion IP remota

Un vez creado se pasa este array como argumento para llamar al metodo start de la StremCtrl.

```
   public void setformato(String t){
      tipo=t;
      }
   public void setIP(String i){
      ipremota=i;
   }

   public void setTipo(String t){
      tipo=t;
   }
    }// fin class
```

3.4.3,b.- MANEJACONEXIÓN

Esta clase se encarga de recoger la dirección IP a la que el servidor se quiere conectar y el puerto por donde el cliente quiere recibir los datos.




```
public class ManejaConexion extends JDialog implements
KeyListener, ActionListener{
   private JButton b1;
   private CPrincipal vd;
   private JTextField IP[];
   private StreamCtrl sc;
   private String ip;
   private JTextField port;
   private int puerto;
   private JButton cancelar;
```

Constructor

```
   public ManejaConexion(CPrincipal v, StreamCtrl mr){
      super(v,"IP de conexión");
      setResizable(false);
      vd=v;
      sc=mr;
      setLayout(new GridLayout(2,1));
      IP=new JTextField[4];
      JPanel p1= new JPanel();
```

Se crean 4 contenedores de texto donde el cliente intriducirá la direccion IP.

```
      for(int i=0;i<4;i++){
         IP[i]=new JTextField(2);
         IP[i].setEditable(true);
         p1.add(IP[i]);
         IP[i].addKeyListener(this);
         }
```

Se crea el contenedor de texto donde el cliente introducirá el número de puerto por el que desea recibir los datos.

```
         port=new JTextField(5);
         port.setEditable(true);
         p1.add(port);
         port.addKeyListener(this);

         add(p1);
         JPanel pb1=new JPanel();
         pb1.setLayout(new GridLayout(1,2));

         cancelar=new JButton("Cancelar");
         cancelar.addActionListener(this);
         pb1.add(cancelar);
         b1=new JButton("Conectar");
         b1.addActionListener(this);
         pb1.add(b1);
         add(pb1);
         resize(320,140);
         show();
      }
```

En esta clase se atienden los siguientes evenetos:

- Eventos provocados al pulsar el boton conectar cancelar.
  - Si se pulsa el boton conectar se comprueba la IP introducida es valida conforme el formato de Ips, y en caso de ser conforme comprueba la validez del puerto.

```
      public void keyPressed(KeyEvent e){}
      public void keyTyped(KeyEvent e){}
```

```
      public void actionPerformed(ActionEvent e){
         if(e.getSource()==cancelar){
            hide();
            dispose();
            sc.setCancel();
            }
         if(e.getSource()==b1){
            try{

if(((IP[0].getText()).length()>3)||((IP[1].getText()).lengt
h()>3)||((IP[2].getText()).length()>3)||((IP[3].getText()).
length()>3)||((IP[0].getText()).length()==0)||((IP[1].getTe
xt()).length()==0)||((IP[2].getText()).length()==0)||((IP[3
].getText()).length()==0)||(Integer.parseInt(IP[0].getText(
))>254)||(Integer.parseInt(IP[1].getText())>254)||(Integer.
parseInt(IP[2].getText())>254)||(Integer.parseInt(IP[3].get
Text())>254))
                                                {
            System.out.println("Aqui no entra");
            hide();
            ManejaError me = new
            ManejaError(vd,this,"Parametros incorrectos", "Se
            debe pasar un @ IP correcta");
                  }// fin if

      else{

       System.out.println("ENTRA?");
       if(((port.getText()).length()<=5)
       &&((port.getText()).length()>0)&&
       (Integer.parseInt(port.getText())>1024)
       &&(Integer.parseInt(port.getText())<=65500)
       &&(Integer.parseInt(port.getText())!=1099)){
       System.out.println("y aqui??");


ip=IP[0].getText()+"."+IP[1].getText()+"."+IP[2].getText()+
"."+IP[3].getText();
          System.out.println("Lo último");
          puerto=Integer.parseInt(port.getText());
          String t=vd.obtenerServidorMMDevice(ip,puerto);
          String k=vd.bindMMDevices();
          sc.habilitar();

                }

         else{
            hide();
            ManejaError me= new ManejaError(vd,this,
            "Parametros incorrectos", "Erro aqui");
```

```
                    }// fin else
                  }// fin else

            System.out.println("HOLA CARACOLA");
                                    hide();
              }// fin try
               catch(Exception ee){
               hide();
               ManejaError me = new
               ManejaError(vd,this,"JEJJEE","No sé que
               pasa!!!!");
                }
               }
              }
```

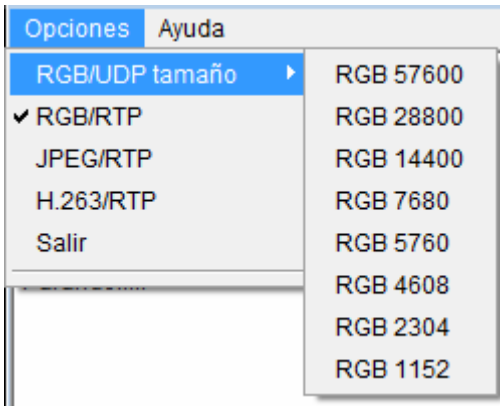
- Eventos provocados al pulsar botones del teclado

```
   public void keyReleased(KeyEvent e){
      if(e.getKeyCode()==KeyEvent.VK_ESCAPE){
         hide();
         dispose();
         sc.setCancel();
            }
      else if
          (e.getKeyCode()==KeyEvent.VK_ENTER){
          try{

if(((IP[0].getText()).length()>3)||((IP[1].getText())...
...
...
)){
            hide();
            ManejaError me = new
ManejaError(vd,this,"Parametros incorrectos", "Se debe
pasar un @ IP correcta");
                }// fin if

         else{
            if(((port.getText()).length()<=5)
            &&((port.getText()).length()>0)&&
            (Integer.parseInt(port.getText())>1024)
            &&(Integer.parseInt(port.getText())<=65500)
            &&(Integer.parseInt(port.getText())!=1099)){

ip=IP[0].getText()+"."+IP[1].getText()+"."+IP[2].getText()+
                     "."+IP[4].getText();

            puerto=Integer.parseInt(port.getText());
            String t=vd.obtenerServidorMMDevice(ip,puerto);
            String u=vd.bindMMDevices();
```

```
                    }
        else{
           hide();
           ManejaError me= new
           ManejaError(vd,this, "Parametros
           incorrectos", "Error aqui tb");

                    }// fin else
                 }// fin else
              }// fin try
        catch(Exception ee){
           hide();
           ManejaError me = new ManejaError(vd,this,"Que
           passa??","");
              }
           }
        }
     }
```

3.4.3.c.- MBAR

Esta clase se implementa la barra de menu que utiliza el reproductor multimedia




```
Constructor de la clase Mbar.

public class MBar extends MenuBar{
   Menu m;
   Menu RGB;
   Menu ayuda;
   CPrincipal cp;
```

```
    public MBar(CPrincipal cliente){
       cp=cliente;
       m=new Menu("Opciones");
       RGB= new Menu("RGB/UDP tamaño");
       RGB.add(new CheckboxMenuItem("RGB 57600",false));
       RGB.add(new CheckboxMenuItem("RGB 28800",false));
       RGB.add(new CheckboxMenuItem("RGB 14400",false));
       RGB.add(new CheckboxMenuItem("RGB 7680",false));
       RGB.add(new CheckboxMenuItem("RGB 5760",false));
       RGB.add(new CheckboxMenuItem("RGB 4608",false));
       RGB.add(new CheckboxMenuItem("RGB 2304",false));
       RGB.add(new CheckboxMenuItem("RGB 1152",false));

       m.add(RGB);
       m.add(new CheckboxMenuItem("RGB/RTP",true));
       m.add(new CheckboxMenuItem("JPEG/RTP", false));
       m.add(new CheckboxMenuItem("H.263/RTP",false));
       m.add(new MenuItem("Salir"));
       m.insertSeparator(4);
       add(m);

       ayuda=new Menu("Ayuda");
       ayuda.add(new MenuItem("Ayuda"));
       ayuda.add(new MenuItem("Acerca de ..."));
       ayuda.insertSeparator(1);
       add(ayuda);
       }

     public String desactivar(String t){
        if(t=="RGB/RTP"){
           CheckboxMenuItem
           mi=(CheckboxMenuItem)m.getItem(1);
           mi.setState(false);
           }
        else if(t=="JPEG/RTP"){
           CheckboxMenuItem
           mi=(CheckboxMenuItem)m.getItem(2);
           mi.setState(false);
           }
       else if(t=="H.263/RTP"){
           CheckboxMenuItem
           mi=(CheckboxMenuItem)m.getItem(3);
           mi.setState(false);
           }
       else if(t=="RGB 57600"){
           CheckboxMenuItem
           mi=(CheckboxMenuItem)RGB.getItem(0);
           mi.setState(false);
           }
       else if(t=="RGB 28800"){
           CheckboxMenuItem
```

```
          mi=(CheckboxMenuItem)RGB.getItem(1);
          mi.setState(false);
          }
       ...
       ...
       ...

          }
       else if(t=="RGB 1152"){
          CheckboxMenuItem
          mi=(CheckboxMenuItem)RGB.getItem(7);
          mi.setState(false);
          }
          cp.setformato(t);
          return activo(t);

         }

    public String activo(String t){
       if(((CheckboxMenuItem)m.getItem(1)).getState())
       return "RGB/RTP";
       else if(((CheckboxMenuItem)m.getItem(2)).getState())
       return "JPEG/RTP";
       else if(((CheckboxMenuItem)m.getItem(3)).getState())
       return "H.263/RTP";

       else if
(((CheckboxMenuItem)RGB.getItem(0)).getState()) return "RGB
       57600";
       else if
(((CheckboxMenuItem)RGB.getItem(1)).getState()) return "RGB
       28800";
...
...
...


       else if (t=="RGB/RTP"){
         CheckboxMenuItem
         mi=(CheckboxMenuItem)m.getItem(1);
         mi.setState(true);
          }
       else if (t=="JPEG/RTP"){
          CheckboxMenuItem mi =
          (CheckboxMenuItem)m.getItem(2);
          mi.setState(true);
          }
       else if (t=="H.2/RTP"){
          CheckboxMenuItem mi =
          (CheckboxMenuItem)m.getItem(3);
          mi.setState(true);
```

```
            }

        return t;
        }

        public void setOption(boolean b){
           m.setEnabled(b);
           }

    }//fin class
```

3.4.3.d.- STREAMCTRL

Esta clase consiste en un Jpanel y es el encargado del control de los botones Play, Pause, Stop, Conectar y Desconectar.


```
public class StreamCtrl extends JPanel implements
ActionListener{
   private JButton play;
   private JButton stop;
   private org.omg.CORBA.ORB orb = null;
   private PropertySetDefFactory property_factory = null;
   private JButton pause;
   private JButton conectar;
   private JButton desconectar;
   private CPrincipal vd;
   private String ip;
   private ManejaConexion mc;
   private boolean pausado;
   private JPanel aux;
   private JPanel aux1;
   private StreamCtrl_impl sci;
   private JPanel aux2;
   private ServidorMMDevice_impl mmd_s;
   private StreamEndPointA_impl sep_s;
   private ServidorVDev_impl vd_s;
   private StreamEndPointB_impl sep_c;
   private ClienteVDev_impl vd_c;
   private String host;
   private boolean can;
   private JPanel p1;
   private JPanel p2;
```

```
    GridLayout g2;

    public StreamCtrl(CPrincipal a, String h){
       vd=a;
       host=h;
       p1=new JPanel();
       p2=new JPanel();
       g2=new GridLayout();
       g2.setColumns(3);
       g2.setRows(1);

       p1.setLayout(g2);
       p2.setLayout(g2);
       play=new JButton(" PLAY ");
       stop=new JButton(" STOP ");
       pause=new JButton(" PAUSA ");
       aux=new JPanel();
       aux1=new JPanel();
       aux2=new JPanel();
       aux.setSize(10,30);
       conectar = new JButton(" CONECTAR ");
       desconectar = new JButton(" DESCONECTAR ");
       pause.addActionListener(this);
       stop.addActionListener(this);
       play.addActionListener(this);
       conectar.addActionListener(this);
       pause.setCursor(new Cursor(Cursor.HAND_CURSOR));
       stop.setCursor(new Cursor(Cursor.HAND_CURSOR));
       play.setCursor(new Cursor(Cursor.HAND_CURSOR));
       pause.setEnabled(false);
       play.setEnabled(false);
       stop.setEnabled(false);
       desconectar.setEnabled(false);
       play.setToolTipText("Para abrir archivo");
       stop.setToolTipText("Para abrir archivo");
       pause.setToolTipText("Para abrir archivo");
       p1.add(play);
       p1.add(pause);
       p1.add(stop);

       p2.add(conectar);
       p2.add(desconectar);
       add(p1);
       add(p2);

       can=false;

    }
         public void habilitar(){
         play.setEnabled(true);
         System.out.println("Habilitado boton play");
```

```
}
```

Al pulsar los botones se generan eventos. Estos eventos son escuchados por el sistema y tratados por el método public void actionPerormed(ActionEvent e).

- Cuando se pulsar el boton play se abre un cuadro de dialogo (ManejaConexion) donde se puede introducir la dirección remota a la que se quiere conectar y el puerto por el que se quieren recibir datos.

```
    public void actionPerformed(ActionEvent e){
       if(e.getSource()==conectar){

vd.setConsola(vd.getConsola()+"\nReproduciendo");
          vd.actualizaConsola();
          play.setEnabled(false);
          mc=new ManejaConexion(vd, this);


          vd.setOption(false);
          }
       else if(e.getSource()==play){

          stop.setEnabled(true);
                 pause.setEnabled(true);
                 conectar.setEnabled(false);
                 System.out.println("El cliente se va a
                 hacer cargo de ir a por el cliente");
          vd.start();

          }

       else if(e.getSource()==stop){
             if(pausado==false){
                pause.setLabel("Pause");
                pausado=true;
                }
                mc.hide();
                mc.dispose();
                vd.setConsola(vd.getConsola()+"\nParando
                Reproducción");
                vd.actualizaConsola();
                setStop();
          }
       else if(e.getSource()==pause){
          if(pausado==true){
             pause.setLabel("Reanudar");
             pausado=false;
```

```
vd.setConsola(vd.getConsola()+"\nReproduciend");
            }
         else{
            pause.setLabel("Pausa");
            pausado=true;
            vd.setConsola(vd.getConsola()+"\nReproducion
            reanudada");
            vd.actualizaConsola();
            setResume();
            }
         }
      }

     public void setPause(){
            vd.setPause();
            try{
               vd.pause();
               vd.pause();
            }catch(Exception e){
            }

      }

      public void setResume(){
         try{
            v.resume();
            }catch(Exception e){
         }
         vd.setResume();
         }

      public void setStop(){
         play.setEnabled(true);
         stop.setEnabled(false);
         pause.setEnabled(false);
         conectar.setEnabled(false);
         desconectar.setEnabled(true);
         vd.setOption(true);
         try{
         vd.stop();
         v=null;
            }catch(Exception e){
         }
         }

      public void setCancel(){
         play.setEnabled(true);
         stop.setEnabled(false);
         pause.setEnabled(false);
         vd.setOption(true);
```

```
            vd.setConsola(vd.getConsola()+"\nSin
            conexión");
            vd.actualizaConsola();
            }
    }
```

Cuando el usuario una vez que pulsa play, cancela el cuadro de diálogo mostrado, el sistema debe saber que se ha cancelado la reproducción. El botón play debe volver a ser visible, y los demás botones inaccesibles. Se debe activar el menú de opciones, e informar de la situación a traveé de la consola del sistema.

3.4.3.e.- PANELRTP

Esta clase es la encargada de mostrar las imágenes recibidas en una conexión RTP. Se ejecuta desde CPrincipal.
Su única funcionalidad es gestionar y visualizar las imágenes a través de RTP. El mayor peso de esta clase recae sobre otra clase:

Javax.media.bean.playerbean.MediaPlayer


```
public class PanelRTP extends JPanel{
```

Variables de instancias necesarias:

```
    private String iplocal;
    private int puertolocal;
    private javax.media.bean.playerbean.MediaPlayer
    mediaPlayer1;
```

El constructor de la clase necesita como argumentos la dirección IP y el numero de puerto

```
    public PanelRTP(String ip,int port){
       puertolocal=port;
       iplocal=ip;
       mediaPlayer1= new
       javax.media.bean.playerbean.MediaPlayer();
       mediaPlayer1.setMediaLocation(new
       java.lang.String("rtp://127.0.0.1:2020/video"));
       System.out.println("En rtp panel tras conexion");
                 mediaPlayer1.setControlPanelVisible(true);
      }
```

Las siguientes funciones permiten el control de la visualización de las imágenes.

```
       public void play(){
          System.out.println("Play en PanelRtp");
          setLayout(new GridLayout(1,1));
          add(mediaPlayer1);
          mediaPlayer1.setBounds(0,120,324,240);
          mediaPlayer1.start();
          }
```

La función play inicia la visualización de las imágenes.

```
       public void stop(){
          mediaPlayer1.stop();
          mediaPlayer1.close();
          removeAll();
          }
```

La función play permite parar la recepción de datos multimedia

```
       public void pause(){
          mediaPlayer1.stop();
          }
```

La función pause detiene la recepción de datos multimedia, manteniendo la conexión con el servidor.

```
       public void resume(){
          mediaPlayer1.start();
          }
    }
```

Por último la función resume reanuda la visualización de datos multimedia que anteriormente ha sido parada

3.4.3.f.- PANELRGB

Clase que se encarga de mostrar las imágenes recibidas en una conexión UDP


```
public class PanelRGB extends Panel implements Runnable{
   private int trama;
   private RGBFormat rgbf;
   private Image img;
   private DatagramSocket clientSocket;
   private DatagramPacket receivePacket;
   private DatagramPacket sync;
   private byte[]array;
   private byte[]array2;
   private byte[]sincro;
   private byte[]aux;
   private byte[]alin;
   private byte[]dat;
   private int i;
   private Thread t;
   private BufferToImage converter;
   private int puerto;
```

Constructor de la clase.

```
public PanelRGB(int t, int puerto){
      array=new byte[23040];
      array2=new byte[230400];

      trama=t;
      rgbf=null;
      img=null;
      clientSocket=null;
```

```
      System.out.println("Estamos en panelRGB");

      try{
      float f = 15;
      rgbf= new RGBFormat(new
Dimension(320,240),230400,Format.byteArray,f,24,3,2,1);
      clientSocket=new DatagramSocket(2020);
      clientSocket.setReceiveBufferSize((230400+6)*10);
      System.out.println("RGB con Sockets");
      }catch(Exception e){
      }
      i=230400/trama;
      aux=null;
      sincro=null;
      alin=new byte[6];
      dat=new byte[trama];
      receivePacket=new DatagramPacket(dat,trama);
      sync=new DatagramPacket(alin,6);
      converter=new BufferToImage(rgbf);
      System.out.println("REHOLA");
   }
```

Métodos utilizados en esta clase.

```
public void setData(byte array[],byte datos[],int
        pos){
        System.arraycopy(datos,0,array,pos,datos.length);
             }
```

La utilización de este método permite ensamblar todas la imágenes fragmentadas en una variable.

```
   public void play(){
         System.out.println("panelRGB PLAY");
         t = new Thread(this);
         System.out.println("Se va a realizar t.start()");
         t.start();
         }
```

Inicializa un Thread para la recepcion de y posterior visualización de imágenes.

```
   public void pause(){
         t.stop();
         }
```

El Thread pausa la recepción y visualización de imagenes

```
   public void resume(){
```

```
        t=new Thread(this);
        t.start();
        }
```

El Thread reanuda la recepción y visualización de imágenes

```
    public void stop(){
        t.stop();
        clientSocket.close();
        }
```

El Thread para la recepción y visualización de imágenes.

A continuación se va a exponer el funcionamiento del Thread de recepción y visualización de imágenes. Dicho Thread se llama con el método run(). Este método consiste en un bucle infinito que constantemente esta recibiendo imágenes

```
        while(true){
        try{
        sincro=null;
```

A continuacion se reciben los bytes del sincronismo y se comprueban si estos bytes corresponden con la secuencia “101010”.

```
        clientSocket.receive(sync);
        sincro=sync.getData();
        System.out.println("Panel RGB Recibiendo Datos en
        RUN");
        } catch(Exception
        e){System.out.println(e.getMessage()+"Error de
        sincronismo 2");
        }

if(sincro[0]==1&&sincro[1]==0&&sincro[2]==1&&sincro[3]==0&&
sincro[4]==1&&sincro[5]==0){
```

Cuando se reciben los bytes de sincronismo se pasan a recibir imágenes que se reciben en varios paquetes UDP.

```
        for(int a=0;a<i;a++){
          try{
          aux=null;
          System.out.println("Esperando paquetes");
          clientSocket.receive(receivePacket);
          aux=receivePacket.getData();
          setData(array,aux,trama*a);

          System.out.println("Recibiendo paquetes");
          }catch(Exception e){
          }
```

A continuación se convierten las imagenes en el formato adecuado.

```
            Object O=(Object)array;
            Buffer b =new Buffer();
            b.setData(O);
            b.setFormat(rgbf);
            img =converter.createImage(b);

            while(PRGBA.getState()){}

            System.out.println("Se va a hacer el
            repaint()");
            repaint();
```

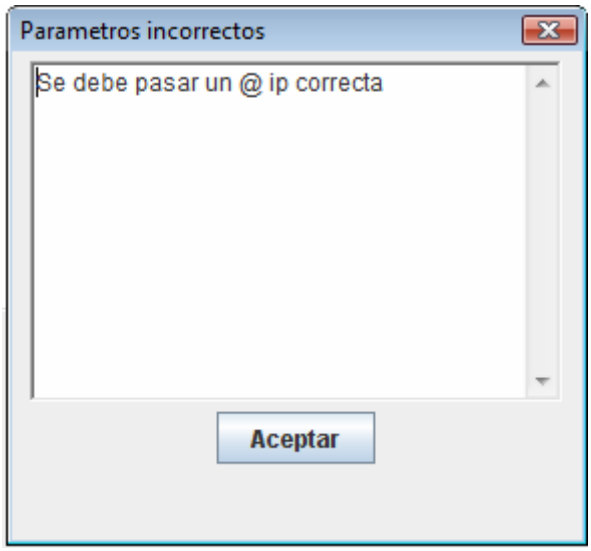
Por ultimo se visualizan las imágenes recibidas.

```
   public void paint(Graphics g){
            for(int i =0;i<10000;i++){
            System.out.println("Visualizando video");}
```

```
g.drawImage(img,0,235,315,0,0,0,img.getWidth(this),img.getHei
ght(this),this);
          }
           }
```

3.4.3.g.- MANEJAERROR

Esta clase solo puede ser invocada por la clase maneja conexión. Se trata de un cuadro de dialogo, el cual aparece cuando se produce un error al introducir una dirección IP errónea.




```
public class ManejaError extends JDialog implements
  ActionListener{
```

Las variables de instancia de esta clase son:

```
private JButton botoncomo;
   private TextArea textcomo;
   private JPanel panelcomo1;
   private JDialog jd;
```

El siguiente constructor inicializa la clase y solicita a la clase maneja conexión el texto de la consola del sistema.

```
public ManejaError(Frame parent, JDialog jd, String
      title, String messageString){
      super(parent, title,false);
      this.jd=jd;
      this.jd.hide();
      panelcomo1=new JPanel();
      textcomo=new TextArea(messageString,10,35,
      textcomo.SCROLLBARS_VERTICAL_O
      NLY)
```

```
textcomo.setEditable(false);
      textcomo.setBackground(Color.white);
      panelcomo1.add(textcomo);
      botoncomo=new JButton("Aceptar");
      botoncomo.setCursor((new
      Cursor(Cursor.HAND_CURSOR)));
      botoncomo.addActionListener(this);
      panelcomo1.add(botoncomo);
      setContentPane(panelcomo1);
      setResizable(false);
      resize(290,270);
      addWindowListener(new CierraVentana());
      show();
```

Añadimos un evento al boton aceptar y tras pulsarlo este cierra la ventana

```
public void actionPerformed(ActionEvent ae){
      setVisible(false);
      this.jd.show();
      }
     }
```

CAPITULO 4
ACERCA DEL SERVICIO A/V STREAMING Y FUNCIONAMIENTO

4.1- PREGUNTAS FRECUENTES

4.1.1.- ¿Qué es el programa Servicio de Video Streaming 1.0?

Servicio de Video Streaming 1.0 es un programa que permite realizar transferencia de video con otros usuario a través de internet.

4.1.2.- ¿Qué requisitos se debe cumplir para que funcione?

Para que funcione el programa es necesario:

- Disponer de una capturadora de video en el ordenador que transmita las imágenes
- Tener instalado Windows
- Tener instalada la maquina virtial de Java 1.4 o superior.

4.2.- Instalación

Copiar la carpeta del Servicio de AVStreams dentro de la carpeta donde está instalado Java. De esta forma se puede evitar tener que modificar los permisos de ejecución de java, ya que la carpeta de instalación de java posee todos los permisos.
Dichos permisos están almacenados en el fichero java.policy y java.security de la carpeta de instalación de Java. Modificando dichos permisos un usuario avanzado puede instalar el programa en cualquier otra carpeta.

4.3.- Ejecución

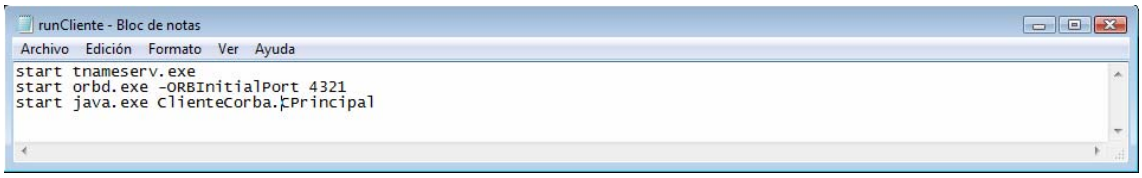
La ejecución de Cliente y del sevidor se realiza a través de dos ejecutables .bat

Para arrancar el cliente, se ejecuta el archivo runCliente.bat. Este archivo arranca desde la carpeta de java el programa Corbaregistry, el cual permite el uso de llamadas remotas a través de CORBA, y arranaca la clase Servidor.class de nuestro programa servidor

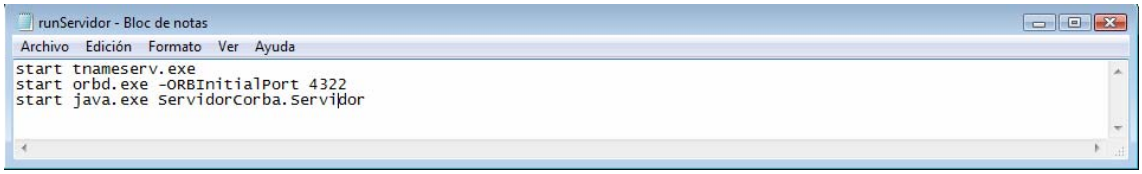
4.4.-Ejecutables .bat

En este apartado se va a mostrar como se han crado los scripts que ejecutan los programas cliente y Servidor. Estos ejecutables están diseñados para la versión 1.4 de la java Virtual Machine. Dichos ejecutables pueden ser modificados por un editor de texto cualquiera como por ejemplo el Bloc de notas de windows.

runCliente.bat



runServidor.bat



4.5.- Funcionamiento

Una vez ejecutado el servidor de vídeo a través de runServidor.bat, el usuario no tiene que preocuparse de nada más que de conectar la cámara, puesto que el servidor se encarga de administrar todas las conexiones automáticamente.

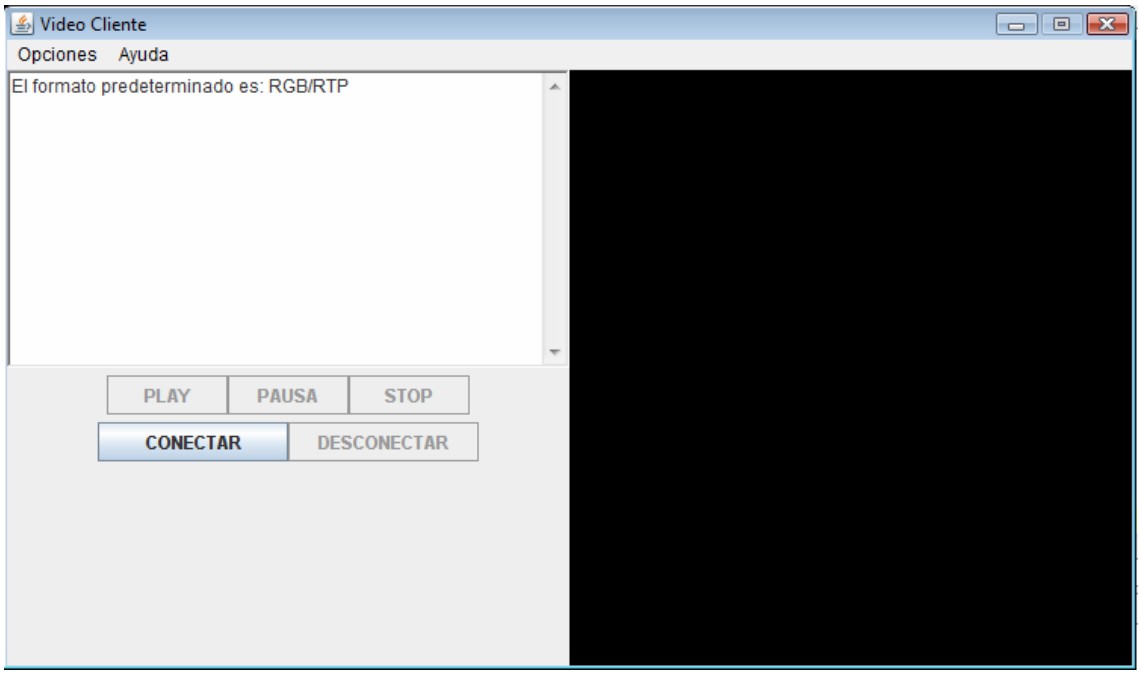
A continuación vamos a explicar la puesta en marcha de la aplicación, además iremos explicando todo el proceso de llamada a clases y funciones que se ejecutan en el Cliente y en el Servidor cuando se produce una conexión.

Lo primero que se debe de hacer es ejecutar la clase Servidor en el Servidor y la clase CPrincipal en el cliente.

Cuando se ejecuta Servidor, se crea un ServidorMMDevice y dicha clase se asocia a la conexión CORBA, para que posteriormente desde el servidor de nombres, el cliente pueda obtener una referencia de la clase y poder operar con ella como si se encontrará realmente en el cliente.

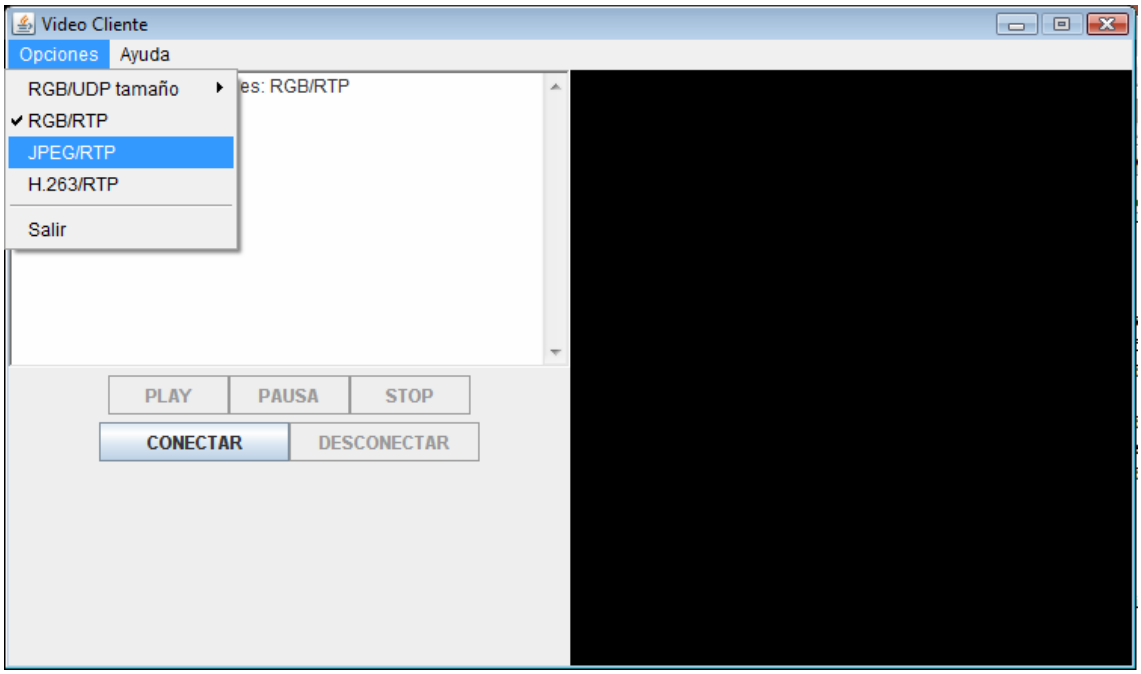
Dichas clases se quedan a la escucha de peticiones del cliente por el puesrto 900.

Una vez ejecutado el cliente, el usuario se encuentra con el siguiente entorno gráfico. Dicho entorno gráfico está contenido en la clase CPrincipal.

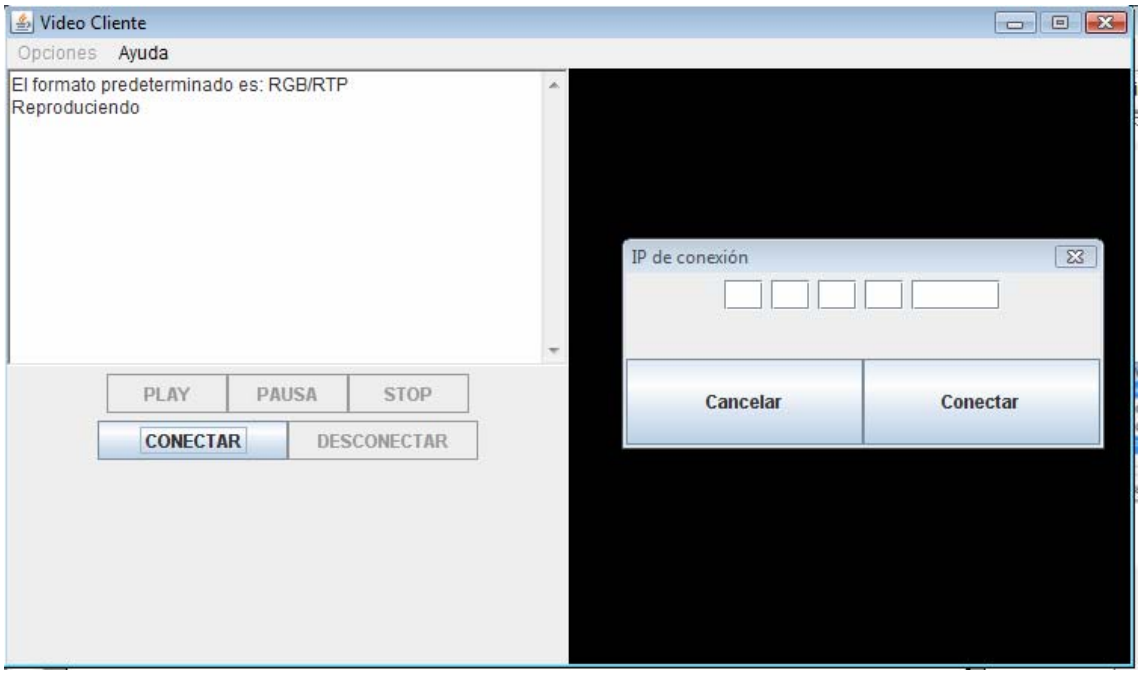


Cuando el usuario quiere conectarse con el extremo remoto, debe realizar los siguientes pasos.

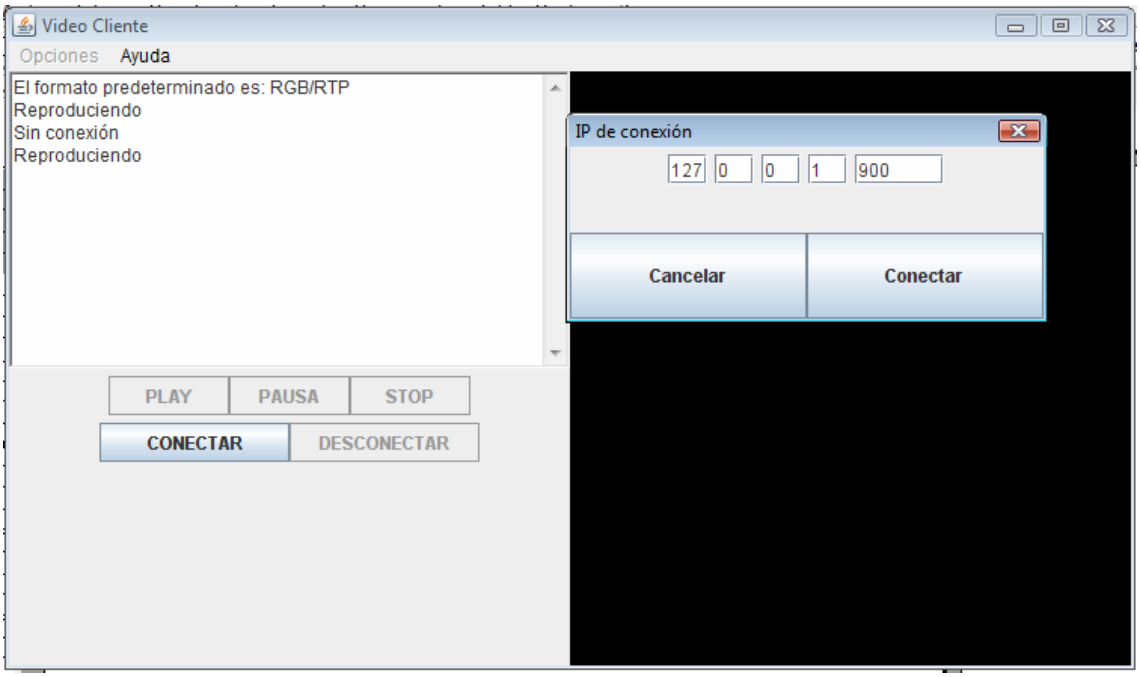
1. Seleccionar el formato de transmisión desde el menú de opciones. Dicho menú pertenece a la clase Mbar. Para evitar que haya más de un formato seleccionado, la clase MBar tiene las funciones activar() y desactivar(). Dichas funciones son llamadas desde CPrincipal que es quien gestiona los menus. CPrincipal se encarga emtonces de que cuando se seleccione un menú se desactive la opcion seleccionada anteriormente.



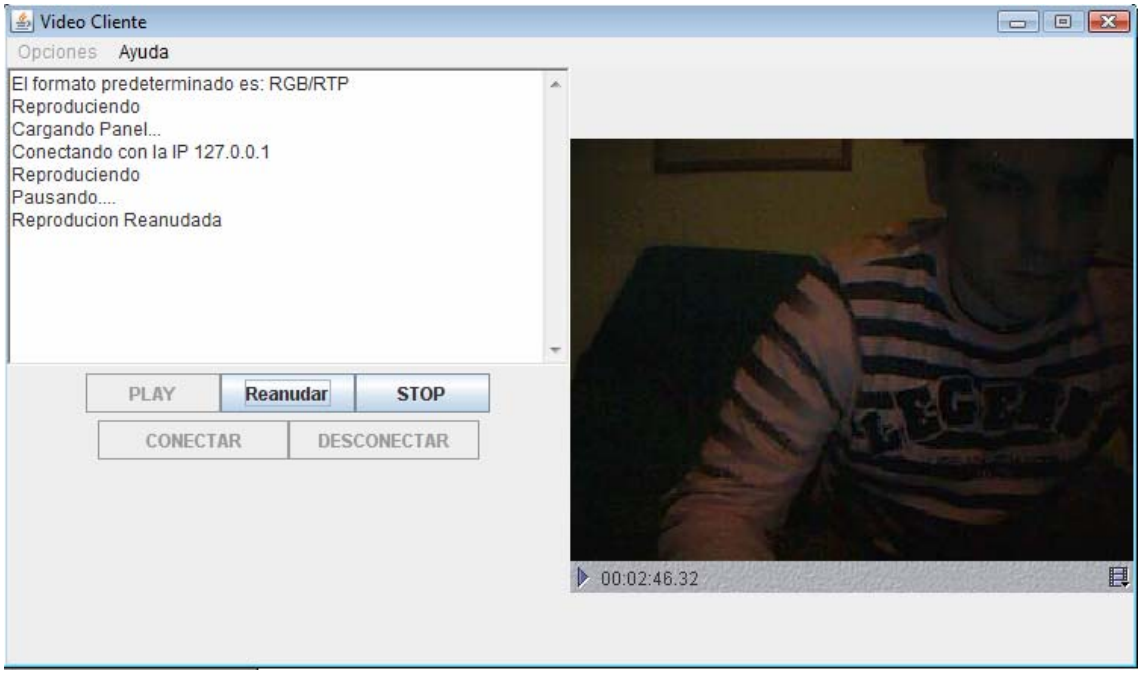
2. Una vez seleccionado el formato deseado, el usuario debe pulsar sobre el botón de CONECTAR. Dicho botón pertenece a la clase StreamCtrl. Una vez seleccionado el boton CONECTAR, StreamCtrl ejecuta el constructor de la clase ManejaConexion y entonces aparece un cuadro de dialogo para introducir la dirección IP del extremo remoto y el puerto por el que se quieren recibir datos. En caso de que el cliente escriba una dirección IP no válida o un numero de puerto inferior a 1025 aparecerá un cuadro de dialogo informando del tipo de error que se ha producido.



3. Una vez introducida la dirección IP del extremo remoto y el puerto por el que se van a recibir los datos, se pulsa sobre conectar o sobre la tecla de inicio, cuando esto ocurre, la clase manejaConexion atiende el evento y pasa a comprobar si los datos introducidos son correctos. En caso de no ser correctos aparece un mensaje de error utilizando la clase manejaError. En caso contrario, manejaConexion le pasa a CPrincipal la dirección IP y el puerto , para que esta puede obtener las referencias de MMDevice remoto. A partir de aquí se inicia el proceso de establecimiento de la conexión con el servidor. En todo momento el usuario el usuario puede cerrar el cuadro de dialogo de manejaConexion. Para gestionar el evento de cerrar el cuadro de dialogo de manejaError, se utiliza la clase CierraVentana.



4. A continuación se muestra una captura de una reproducción en formato RGB sobre RTP.



.

CAPITULO 5
CONCLUSIONES

En este proyecto final de carrera se ha desarrollado un servicio de *A/V Streaming* utilizando el lenguaje de programación Java y el *middleware* CORBA. Debido a la versatilidad de estos lenguajes de programación es posible desarrollar diferentes variantes del servicio desarrollado.

El lenguaje Java es seleccionado por la expansión de su uso en diversos sistemas de telecomunicación. Debido a que Java es un sistema heterogéneo, puede ser usado en distintas plataformas sin que el código tenga que ser compilado para cada plataforma. Esta propiedad de Java permite la creación de muchas y variadas aplicaciones.

La capa de intermediación *middleware*, como es CORBA, permite la interoperabilidad de aplicaciones en distintas arquitecturas y con distintos lenguajes de programación, lo que posibilita crear servicios distribuidos en un número muy amplio de entornos.

Se puede sumar a este razonamiento que cada año surge la necesidad de integrar nuevos servicios a la sociedad. Un ejemplo de estos servicios, son las aplicaciones de transmisión de video. En este caso tanto la integración de los distintos protocolos de transmisión de imágenes como la incorporación de nuevos mecanismos de negociación en las comunicaciones, son un ejemplo de ello. Además, ofrecen una nueva utilidad capaz de controlar la calidad de servicio desde la capa *middleware*.

En lo que se refiere a este proyecto fin de carrera, y en concreto a la especificación de *A/V Streaming* proporciona una interfaz que permite al desarrollador crear aplicaciones multimedia. Además permite al programador de aplicaciones Java realizar implementaciones de manera sencilla y con un coste de tiempo mínimo.

A partir de este servicio y como consecuencia de las funcionalidades de Java y CORBA se sugieren los siguientes trabajos como variantes del desarrollado en este proyecto:

- Implementación de un servicio parecido al del proyecto actual, pero con sistemas de negociación diferentes.

- Ampliación del servicio para incorporar nuevos protocolos que en el futuro sean empleados por CORBA y nuevos servicios de transmisión.

- Implementación de servidores capaces de transmitir video a varios usuarios a la vez, teniendo cada usuario unas necesidades particulares.

- Desarrollo de aplicaciones específicas como un sistema capaz de asignar un servidor libre, de entre una lista de servidores, a un usuario.

- Como aplicación del anterior el usuario podría decidir a que servidor conectarse.

Bibliografía

- The Design and Perfomance of a CORBA Audio/Video Streaming Service
- PFC: Desarrollo e implementación de un servicio de A/V Streaming usando Java/RMI.
- Programing with Java Corba
- The Java Class Libraries, Second Edition, Volume 2 - Chan, Lee
- http://www.programacion.com
- http://ava.sun.com
- http://www.java.com
- http://wikipedi.org
- http://www.webdelprogramador.com
- http://javalobby.com
- http://www.java.com/en/everywhere/javavideo.jsp

ANEXO I
MEDIDAS DE RETARDO

Durante este capitulo explicaremos como hemos obtenido las estadísticas de retardo entre cliente y servidor.

Definiremos retardo entre cliente y servidor como el tiempo que transcurre entre que el servidor comienza a enviar una imagen y el cliente la recibe. A partir de este retardo y para un servicio dado podemos calcular dos parámetros estadísticos más, como son el Jitter y retardo medio:

- Retardo medio: Es una media de todos los retardos obtenidos.

- Jitter: Es la variación máxima de retardo respecto del retardo medio.

En nuestro proyecto sólo vamos a obtener estas medidas para el caso de transmisión en UDP. Las tasas de reproducción utilizadas para la transmisión con 15FPS y 5FPS

Esta toma de estadísticos está basada en la herramienta de Benchmarking.
La herramienta de Benchmarking funciona a partir de marcas temporales. Estas marcas temporales no son más que las que resultan de obtener el valor del reloj del sistema en un momento determinado y tomado como un valor temporal en milisegundos.

Entonces esta herramienta funciona de la siguiente manera:

a.- Cuando el servidor comienza a transmitir la primera imagen se obtiene una marca temporal, esta marca es un tiempo de referencia en el Servidor y por tanto se tomo como tiempo cero del servidor. Este tiempo se toma como el tiempo que hay que restarle a las siguientes marcas temporales en el servidor para obtener el tiempo en el que se envia el paquete.

b.- Cuando el cliente .recibe el primer paquete de sincronismo realiza una marca temporal, la cual llamamos tiempo de referencia en el Cliente. Dicho tiempo se toma como tiempo cero del cliente , y al igual que en el servidor, es el tiempo que hay que retarle a las siguientes marcas temporal en el cliente para obtener el tiempo en el que se recibe el paquete.

c.- Cuando el servidor termina de transmitir una imagen, realiza una marca de tiempo la cual asocia a un número de secuencia de imagen

d.- Cuando el cliente recibe un imagen completa, hace una marca de referencia y la almacena junto al número de secuencia que le indique el paquete de sincronismo.

e.- Una vez se ha terminado el proceso de captura de video, se puede calcular el retardo extremo a extremo entre imágenes como se explica a continuación:

Retardo(n) = C(n)-S(n)

C(n)= Cmt(n)-TRefC
S(n)= Smt(n)-TRefS

Siendo :

Retardo(n) : Retardo extremo a extremo de la imagen n.
C(n) : Tiempo de llegada del paquete n al Cliente
S(n) : Tiempo en que el Servidor envía el paquete n.
Cmt(n) : Marca temporal del paquete n en el Cliente.
Smt(n) : Marca temporal del paquete n en el Servidor.
TRefC : Tiempo de referencia en el Cliente.
TRefS : Tiempo de referencia en el Servidor.

Una vez obtenidos los retardos extremo a extremo de varias imágenes, se pasa a hacer una media aritmética de todos los retardos, así como el cálculo del Jitter.

A1. IMPLEMENTACIÓN

SERVIDORRGB

```
package ServidorCorba;

import javax.media.*;
import java.net.*;
import javax.media.rtp.*;
import javax.media.protocol.*;
import javax.media.Manager.*;
import javax.media.format.*;
import java.util.*;
import java.io.*;
import java.awt.*;

public class ServidorRGB implements Runnable{
   private int tamaño;
   private FrameGrabber fg;// encargado de la captura de
   datos
   private InetAddress d_ip;
   private DatagramSocket serverSocket;
   private byte[] alin;
   private int i;
   private FileOutputStream FOS;
   private DatagramPacket sync;
   private Thread t;
   private Buffer B;
   private Object O;
   private String IP; // Cliente
   private int par;
   private int puerto;
```

```
        // RETARDO

        private boolean cero;
        private String dat;
        private long tini;
        private long tfin;


    public ServidorRGB(int t, String ip, int p){
       tamaño=t;
       IP=ip;
       puerto=p;
       par=2;
       d_ip=null;
       serverSocket=null;
       fg=null;

        // PARA RETARDO

                alin = new byte[8];
                alin[0]=1;
                alin[1]=0;
                alin[2]=1;
                alin[3]=0;
                alin[4]=1;
                alin[5]=0;
                alin[6]=-128;
                alin[7]=-128;


              System.out.println("Dentro del ServidorRGB");

       try{
       fg=new FrameGrabber();
       System.out.println("FrameGrabber iniicializado");
       d_ip=InetAddress.getByName(IP);
       serverSocket=new DatagramSocket();
       serverSocket.setSendBufferSize(10*(230400+8));
       FOS=new
    FileOutputStream("logServidor/ServidorUDP"+tamaño+".log"
);
       }catch(Exception e){}
       i=230400/tamaño;
                 play();
       }
    public void play(){
       t = new Thread(this);
       t.start();
       }
    public void stop(){
       t.stop();
```

```
   serverSocket.close();
   fg.stop();
   }
public void pause(){
   t.stop();
   }
public void resume(){
   t=new Thread(this);
   t.start();
   }

public void run(){
      while(true){
      B=null;
      try{
      System.out.println("Se va a iniciar getBuffer En
      SRGB");
      B=fg.getBuffer();
      }catch(Exception e){}

      O=B.getData();
      byte datos[]=(byte[])O;
      byte aux[]=null;

      // PARA RETARDO

     if (par%1==0){
     tini=System.nanoTime();
     par=1;
        if (alin[7]==127){
           if (alin[6]==127){
           alin[6]=-127;
           alin[7]=-127;
           }
           else{
           alin[6]=(byte)((int)alin[6]+1);
           alin[7]=-127;
           }
         }
           else{
           alin[7]=(byte)((int)alin[7]+1);
           }



       try{
          sync=new DatagramPacket(alin,8,d_ip,2020);
```

```
                serverSocket.send(sync);
                }catch(Exception e){
System.out.println(e.getMessage()+"Error al enviar
sincronismo");
                    }

      for(int a=0;a<i;a++){
        try{
        aux=new byte[tamaño];
        System.arraycopy(datos,tamaño*a,aux,0,tamaño);

        DatagramPacket sendPacket=new
        DatagramPacket(aux,tamaño,d_ip,2020);
        serverSocket.send(sendPacket);
        System.out.println("Enviando paquete....." + a);
        }catch(Exception e){
        System.out.println(e.getMessage());
        System.exit(0);
         }
        }

        if(cero){
        cero=false;
        tini=System.nanoTime();
        }
        tfin=System.nanoTime();

        dat=(((int)alin[7]+128)+(((int)alin[6]+128)*255))+"
        "+((tfin-tini)/1000000)+"\n";
        try{
        FOS.write(dat.getBytes());
        }catch(Exception e){
        System.out.println("Error al escribir log");
                }
              }
            else{
            par=par+1;}
        }
   }
}
```

PANELRGB

```
package ClienteCorba;

import AVStreams.*;
```

```
import javax.media.format.*;
import java.awt.*;
import javax.media.*;
import java.net.*;
import javax.media.util.*;
import javax.swing.*;
import java.io.*;
import java.util.*;

public class PanelRGB extends Panel implements Runnable{
   private int trama;
   private RGBFormat rgbf;
   private Image img;
   private DatagramSocket clientSocket;
   private DatagramPacket receivePacket;
   private DatagramPacket sync;
   private byte[]array;
   private byte[]array2;
   private byte[]sincro;
   private byte[]aux;
   private byte[]alin;
   private byte[]dat;
   private String data;
   private int i;
   private Thread t;
   private BufferToImage converter;
   private int puerto;
   private long tin;
   private long tfi;
   private DataOutputStream dos;
   private boolean cero;



   public PanelRGB(int t, int puerto){
      array=new byte[230400];
      array2=new byte[230400];

      trama=t;
      rgbf=null;
      img=null;
      clientSocket=null;

      cero=true;

      ImageIcon ii=new ImageIcon("azul.gif");
      img= ii.getImage();
      System.out.println("Estamos en panelRGB");

      try{
      float f = 15;
```

```
      rgbf= new RGBFormat(new

Dimension(320,240),230400,Format.byteArray,f,24,3,2,1);
      clientSocket=new DatagramSocket(2020);
      clientSocket.setReceiveBufferSize((230400+6)*10);
      dos = new DataOutputStream(new
   FileOutputStream("logCliente/ClienteUDP"+trama+".log"));
      System.out.println("RGB con Sockets");
      }catch(Exception e){}

      i=230400/trama;
      aux=null;
      sincro=null;

      alin = new byte[8];
      dat=new byte[trama];
      receivePacket=new DatagramPacket(dat,trama);
      sync=new DatagramPacket(alin,8);
      converter=new BufferToImage(rgbf);
                System.out.println("REHOLA");
                }

public void setData(byte array[],byte datos[],int pos){
      System.arraycopy(datos,0,array,pos,datos.length);
      }

   public void play(){
       System.out.println("panelRGB PLAY");
       t = new Thread(this);
       System.out.println("Se va a realizar t.start()");
       t.start();
       }

   public void pause(){
      t.stop();
      }

   public void resume(){
      t=new Thread(this);
      t.start();
      }

   public void stop(){
      t.stop();
      clientSocket.close();
      }

   public void run(){
        while(true){
           try{
      sincro=null;
```

```
        clientSocket.receive(sync);
         sincro=sync.getData();

          // PARA RETARDO

         if (cero){
         cero=false;
         tin=System.nanoTime();
         }

    System.out.println("Panel RGB Recibiendo Datos en RUN");
        }catch(Exception
    e){System.out.println(e.getMessage()+"Error de
    sincronismo
    2");
    }

if(sincro[0]==1&&sincro[1]==0&&sincro[2]==1&&sincro[3]==0&&
sincro[4]==1&&sincro[5]==0){
    for(int a=0;a<i;a++){
         try{
         aux=null;
         System.out.println("Esperando paquetes");
         clientSocket.receive(receivePacket);
         aux=receivePacket.getData();
         setData(array,aux,trama*a);
         System.out.println("Recibiendo paquetes");
         }catch(Exception e){
        }

        Object O=(Object)array;
        Buffer b =new Buffer();
        b.setData(O);
        b.setFormat(rgbf);
        img = converter.createImage(b);

        System.out.println("Se va a hacer el repaint()");
        repaint();
        tfi=System.nanoTime();

       data=(((int)sincro[7]+128)+(((int)sincro[6]+128
       )*255))+" "+((tfi-tin)/1000000)+"\n";
       try{
           dos.writeBytes(data);
           }catch(Exception e){
           System.out.println(e.getMessage()+" error al
           almacenar estadísticas");

                      }
                    }
                  }
```

```
                }
            }

    public void paint(Graphics g){

g.drawImage(img,0,235,315,0,0,0,img.getWidth(this),img.getH
eight(this),this);
        }
```

DATOS OBTENIDOS VIDEO RGB

PAQUETE 57600

TX	3	0	4	11	3	11	4	9	3	11	12	3	15	13	12	13	16	13	3
RX	12	38	54	36	66	81	9	78	4	20	40	43	51	76	13	26	44	69	4

Media: 38,409 ms
Jitter: 5,42

ANEXO II
MATERIAL UTILIZADO

SOFTWARE EMPLEADO PARA LA REALIZACIÓN DEL PROYECTO:

- **WINDOWS VISTA**

- **j2sdk 1.4.2_15**

- **Netbeans 5 .0**

- **JMF**

HARDWARE EMPLEADO PARA LA REALIZACIÓN DEL PROYECTO:

PORTÁTIL:

WEBCAM.

ANEXO III
ENTORNO DE TRABAJO

El entorno de trabajo en el cual se probado y validado este proyecto fin de carrera consta de 2 equipos conectados a un switch. Uno de los equipos hace de router para obtener servicios de Internet.

1 SWITCH
2 ORDENADORES