Universidad de las Ciencias Informáticas

Facultad 9

Tema: "Aplicación de técnicas comunicativas para los Subsistemas de transmisión de la Plataforma de Transmisión Abierta para Radio y Televisión."

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autora:

Karelia Puertas Mejías

Tutor:

Ing. José Andrés Hernández Bustio

Cotutor:

Ing. Jean Michael Suárez Pérez

Ciudad de la Habana, Julio de 2010

¨Año 52 de la Revolución¨

Lo que sabemos es una gota de agua; lo que ignoramos es el océano.

Isaac Newton (1642-1727) Matemático y físico británico.

Vale más saber alguna cosa de todo, que saberlo todo de una sola cosa.

Blaise Pascal (1623-1662) Científico, filósofo y escritor francés.

En los momentos de crisis, sólo la imaginación es más importante que el conocimiento.

Albert Einstein (1879-1955) Científico nacido en Alemania, nacionalizado estadounidense.

Dedicatoria

A mis padres.

A mi familia.

A Fidel y la Revolución.

Agradecimientos

A mi mamá por siempre preocuparse por mi futuro.

A mi papá por los consejos basados en sus experiencias.

A mi familia, mi hermano, mi tía, mi prima, mi abuela, mi abuelo.

A Yilena y Nedita que aunque estemos algo distante siempre mantenerme al tanto de sus problemas y también ayudarme con los míos.

A Dianin por siempre mantenerme al tanto de sus locuras y no reprochar las mías.

A la PMM (Liumi, Ari, Diosbel, Osmel, Wilfre), por darme muchos de los mejores momentos que pasé en la UCI, nunca los olvidaré.

Al Yano por siempre hacerme reír en los momentos difíciles, por nunca cansarse de mis preguntas y siempre estar dispuesto a ayudarme.

A todos los del antiguo 9107, por haberme dado la oportunidad de pertenecer al mejor grupo que ha existido en la UCI.

A todas las chicas con las que compartí apartamento, a Zule que es una de las personas más buena que he conocido, a Wiki que es una de las que vengo cargando desde 1er año, Lily la flaquita más sexy de la UCI, Yanly, Lourdes, Yake, Yullita, Suyen, Idalmis, Yayo, Margelys, Liudmila, Dayi y Elizabeth.

A Fabian y Cheyner mi gran amigo desde 1er año siempre dándome consejos, al Miche que aunque ya no se encuentre en la escuela y ya sea un muchacho trabajador siempre encuentra

Agradecimientos

un huequito para vernos, al Pache que por ser el mayor de todos siempre trata de guiarnos lo mejor posible, al Yuma, Yosiel, Falerinti, Olivito, Gustavito, Pepito, Ale, Ottico, Joaquín, Osly los chicos con los que compartí muy buenos momentos.

A Alfro que no por último menos importante, por ayudarme siempre, y por estar presente tanto en las buenas como en las malas.

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo al Departamento de Señales Digitales de la Facultad 9 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Karelia Puertas Mejías José A. Hernández Bustio Jean M. Suarez Pérez

_____ _____ _____

Tutor:

Ing. José Andrés Hernández Bustio

Graduado de Ciencias Informáticas

e-mail: jahernandez@uci.cu

Cargo: Segundo jefe del departamento de Señales Digitales del centro de desarrollo GEYSED.

Profesor Instructor con dos años de experiencia docente en la Universidad de las Ciencias Informáticas.

Co-Tutor:

Jean Michael Suárez Pérez

Graduado de Ciencias Informáticas

e-mail: jmsuarez@uci.cu

Profesor Instructor con 1 año de experiencia docente en la Universidad de las Ciencias Informáticas.

Resumen

En el presente documento se refleja la investigación realizada a partir de la necesidad de conocer sobre las diferentes formas de comunicación existentes en el mundo informático, con el fin de lograr comunicar dos aplicaciones de escritorio en lenguaje Java, donde se planteó la necesidad de utilizar como sistema de comunicación un *middleware*. Orientado al proyecto Plataforma de Transmisión Abierta para Radio y Televisión del departamento de Señales Digitales de la Facultad 9. Se plantea luego de dar a conocer los conceptos asociados a la investigación, una comparación entre dos *middlewares* mundialmente reconocidos con el objetivo de su futura utilización. Se llega a la conclusión del uso de ICE como *middleware* para la comunicación entre los Subsistemas Administración de la Transmisión y Transmisión. Se hace referencia al modelo, lenguaje y herramientas utilizadas en el transcurso de la investigación. Para validar la propuesta de tecnología de comunicación se identificó un caso de estudio del proyecto Plataforma de Transmisión Abierta para Radio y Televisión por el cual se explican los pasos básicos que se deben llevar para llevar a cabo la comunicación entre dos aplicaciones Java con el uso del *middleware* ICE.

PALABRAS CLAVES:

Java, *middleware,* Plataforma de Transmisión Abierta para Radio y Televisión, Subsistema Administración de la Transmisión, Subsistema Transmisión.

ÍNDICE DE FIGURAS

Figura 1: Distribución física de PTARTV. 11
Figura 2: Arquitectura de CORBA. (18) 25
Figura 3: Modelo cliente/servidor 33
Figura 4: Logo de Ubuntu 35
Figura 5: Logo de MVJ 35
Figura 6: NetBeans IDE 36
Figura 7: Código Slice 38
Figura 8: Clases Generadas (36) 39
Figura 9: Código de la clase Reprod.java 39
Figura 10: Código de la clase Main.java 40
Figura 11: Código de la clase Main.java 41
Figura 12: Clases en el Subsistema Transmisión 49
Figura 13: Clases en el Subsistema Administración de la Transmisión 50
Figura 14: Paquetes de la biblioteca zeroc 51
Figura 15: Corriendo la aplicación del Subsistema Transmisión 51
Figura 16: Interfaz de la aplicación del Subsistema Administración de la Transmisión 52

ÍNDICE DE TABLAS

Tabla 3: Tabla de tipos de datos Slice y Java 37
Tabla 4: Slice keywords 37

Índice

INTRODUCCIÓN 1
CAPÍTULO 1 5
1.1 Introducción 5
1.2 Conceptos Asociados 5
Sistemas distribuidos. 5
Middleware 6
Streaming 6
Transmisión de radio por medio de la Web 7
Transmisión de televisión por medio de la Web. 7
Comunicación entre procesos (IPC) 8
Plataforma de Transmisión Abierta para Radio y Televisión. 8
1.3 Descripción General del objeto de estudio. 12
1.3.1 Técnicas comunicativas entre aplicaciones informáticas existentes: 12
1.4 Situación Problemática 16
1.5 Conclusiones del capítulo 17
CAPÍTULO 2 18
2.1 Introducción 18
2.2 Antecedentes 18
2.3 Características de las posibles soluciones 18
2.3.1 CORBA: 19
2.3.2 ICE: 26
2.4 Comparando las posibles técnicas a utilizar. 28
2.4.1 Algunas diferencias que existen entre ICE y CORBA en cuanto a: 29

2.5 Elección de la técnica a utilizar 30
2.6 Conclusiones Parciales 31
CAPÍTULO 3 32
3.1 Introducción 32
3.2 Modelo, lenguaje y herramientas 32
3.2.1 Modelo cliente-servidor 32
3.2.2 Lenguaje de programación 33
3.2.3 Sistema operativo 34
3.2.4 Software de desarrollo 35
3.3 Desarrollo de la aplicación ICE 36
3.4 Conclusiones Parciales 41
CONCLUSIONES GENERALES 42
Recomendaciones 43
Referencias Bibliográficas 44
Anexos 47
Anexo #1 47
Anexo #2 48
Anexo #3 49
Glosario de términos 53

INTRODUCCIÓN

En los últimos años la informática se ha convertido en una ciencia conocida por casi todo el mundo a partir del gran desarrollo que tuvieron las computadoras personales en los años 70 y la llegada de Internet al mundo de las comunicaciones que hoy en día son parte indispensable de la humanidad. La informática ha creado un nuevo mundo en la manera de trabajar y de estudiar de las personas, ya que tienen la posibilidad de hacerlo desde sus hogares.

En Cuba desde hace unos años se ha planteado la necesidad de informatizar el país. Para esto se comenzó la formación de ingenieros informáticos calificados para dar cumplimiento a esta tarea, es una labor en la que se han visto involucrados. Se tiene como objetivo además estar acordes al desarrollo mundial informático y a la era actual, denominada, era de la información.

Desde los inicios de la Universidad de las Ciencias Informáticas (UCI) se ha tenido gran desarrollo en las áreas de multimedia[1]. Entre los recursos más conocidos se encuentra el sitio Web de Televisión Universitaria (Inter-Nos) el cual se ha ido mejorando continuamente para poder cumplir con la gran demanda que tiene diariamente este sitio. La UCI también cuenta con una infraestructura televisiva que permite transmitir 24 canales televisivos y uno para la emisora radial de la institución.

En la actualidad con el crecimiento de la población universitaria y al mismo tiempo el incremento del uso de las redes de computadoras como medio para el envío y recepción de contenido multimedia, se hace necesario un sistema que gestione y ofrezca integradamente los servicios de audio y video con nuevas técnicas de interactividad. De esta forma surge la necesidad de distribuir contenidos audiovisuales por los sistemas convencionales de transmisión de señales, incluyendo las redes de computadoras, haciendo uso de tecnologías de desarrollo Web[2] y tecnología streaming.

En el 2008 a partir de la creación del Polo de Video y Sonido Digital (actualmente Departamento de Señales Digitales del centro de desarrollo GEYSED) de la facultad 9. Se crearon los proyectos productivos

[1] **Multimedia** se le conoce a la combinación de sonido, gráficos, animación y video que se usan para la transmisión de una información por medio de un ordenador.

[2] **Desarrollo Web** se le conoce a la tecnología de software que unifica una base de datos con el uso de un navegador de Internet con el objetivo de mostrar determinada información o de darle cumplimiento a determinadas tareas.

que tienen como objetivo la creación de productos informáticos para la gestión y difusión de contenido multimedia como PRIMICIA, VIDEOWEB, Captura y Catalogación de Medias y la Plataforma de Transmisión Abierta para Radio y Televisión (PTARTV). Con PTARTV se pretende unificar los servicios de radio y televisión que se ofrecen en nuestra universidad. PTARTV tiene como principal objetivo proveer una solución tecnológica integral que permita automatizar la transmisión, administración y gestión de los canales de radio y televisión.

PTARTV está estructurado actualmente en 8 Subsistemas encargados de las diferentes prestaciones que brindará la plataforma. Con el uso de las nuevas tecnologías de comunicación entre aplicaciones de escritorio se tratará de establecer una comunicación instantánea para una transmisión en vivo. De los 8 Subsistemas se necesita establecer una comunicación entre los Subsistemas Transmisión y Administración de la Transmisión. Por lo que se necesita realizar una investigación destinada a conocer cómo establecer una comunicación entre dos aplicaciones en entornos distribuidos.

Teniendo como base todo lo antes expuesto y a través de su estudio se le dará respuesta al siguiente **Problema científico**:

¿Cómo lograr una comunicación entre los Subsistemas de transmisión en la Plataforma de Transmisión Abierta para Radio y Televisión?

Para dar respuesta a este problema se plantea el **objetivo general**:

Elaborar una propuesta para el uso de una técnica comunicativa entre aplicaciones de escritorio para los Subsistemas de transmisión de la Plataforma de Transmisión Abierta para Radio y Televisión.

Es por este motivo que se ha definido como **objeto de estudio** las técnicas comunicativas entre aplicaciones informáticas y el **campo de acción** lo constituye una interfaz de comunicación entre los Subsistemas de transmisión para la Plataforma de Transmisión Abierta para Radio y Televisión.

Se propone como **Idea a defender**: Con la aplicación de una técnica comunicativa entre aplicaciones de escritorio dentro de la Plataforma de Transmisión Abierta para Radio y Televisión se logrará una mejora de los procesos de transmisión y un aumento de la productividad.

Durante la investigación y con el fin de lograr el objetivo se plantean las siguientes **Tareas de investigación**:

1. Elaborar la fundamentación teórica del tema en interés.
2. Caracterizar las diferentes técnicas de comunicación entre aplicaciones de escritorio.
3. Comparar las técnicas encontradas.
4. Definir la técnica más adecuada para ser aplicada en el proyecto.
5. Demostrar por medio de una aplicación el uso de la técnica escogida.

Para obtener los **Posibles resultados**:

Caracterización de las diferentes técnicas de comunicación entre aplicaciones de escritorio.

Propuesta de una técnica aplicable a la Plataforma de Transmisión Abierta para Radio y Televisión.

En el desarrollo de la presente investigación se tuvo en cuenta los siguientes **métodos científicos**:

Dentro de los **teóricos** se empleó el siguiente:

- ✓ **Analítico-Sintético** para enunciar y describir en qué consisten los conceptos y las características principales de las técnicas de comunicación entre aplicaciones de escritorio.
- ✓ **Histórico-Lógico** se utiliza para conocer la trayectoria histórica de las técnicas de comunicación que se estudiarán.

Dentro de los **empíricos** se empleó el siguiente:

- ✓ **Entrevistas** a los miembros del departamento de Señales Digitales para recopilar la mayor cantidad de información posible acerca del tema de interés y la situación actual del proyecto PTARTV. (Ver Anexo 1)

El presente trabajo está dividido en 3 capítulos:

Capítulo 1: Fundamentación Teórica. En este capítulo se aborda todo lo referido a los conceptos relacionados con la investigación, así como se darán a conocer diferentes técnicas que se utilizan para

lograr comunicar aplicaciones informáticas, además, se hace un mejor enfoque a la situación problemática de la investigación.

Capítulo 2: Descripción de las propuestas. En dicho capítulo se hace una comparación entre los dos *middlewares* posibles a utilizar para luego dar una propuesta.

Capítulo 3: Validación de la propuesta. En el mismo se analizan las herramientas y lenguajes que se utilizan para el desarrollo de la aplicación utilizando el middleware escogido. Se da una breve explicación de cómo realizar la comunicación de dos aplicaciones Java.

CAPÍTULO 1

1.1 Introducción

Actualmente en lo que respecta a la transmisión de radio y televisión por medio de una Web, el mundo se encuentra en gran proceso de desarrollo, esta nueva tendencia ha revolucionado la manera de transmitir información. Hoy en día con solo acceder a Internet se pueden obtener estos servicios, sin límites territoriales de señal o de frecuencia.

En el presente capítulo se plantean los fundamentos teóricos del tema en interés. Se definen una serie de conceptos necesarios para dar a entender el tema de la investigación tales como middleware, sistemas distribuidos, streaming.

1.2 Conceptos Asociados

Sistemas distribuidos.

Los sistemas distribuidos se implementan en diversas plataformas hardware, desde unas pocas estaciones de trabajo conectadas por una red de área local, hasta Internet, una colección de redes de área local y de área extensa interconectadas, que enlazan millones de ordenadores. (1)

Las aplicaciones de los sistemas distribuidos varían desde sistemas bancarios, comunicaciones multimedia y abarcan prácticamente todas las aplicaciones comerciales y técnicas de los ordenadores. Los requisitos de dichas aplicaciones incluyen un alto nivel de fiabilidad, seguridad contra interferencias externas y privacidad de la información que el sistema mantiene. (1)

Un sistema distribuido se define como una colección de computadores autónomos conectados por una red, y con el software distribuido adecuado para que el sistema sea visto por los usuarios como una única entidad capaz de proporcionar facilidades de computación. (1)

Un sistema distribuido se define como una serie de computadoras conectadas por una red que al utilizar un software distribuido los usuarios logran visualizar al sistema como uno solo.

Middleware

En la industria informática, *middleware* (elementos de en medio) es un término general para cualquier programación que sirve para unir o trabajar como puente entre dos o más programas separados, y que por lo general ya existen. Una aplicación común para el *middleware* es permitir que los programas que se han desarrollado puedan acceder a una base de datos. (2)

El software distribuido requerido para facilitar las interacciones cliente-servidor se denomina *middleware*. El acceso transparente a servicios y recursos no locales distribuidos a través de una red se provee a través del *middleware*, que sirve como marco para las comunicaciones entre las porciones cliente y servidor de un sistema. (1)

El middleware se define como un software distribuido constituido por un conjunto de recursos y servicios que permiten a múltiples procesos, corriendo en una o más máquinas, interactuar a través de una red. Es una colección no estructurada de recursos y servicios, que pueden ser utilizados individualmente o conjuntamente por los procesos de la aplicación. Provee una interfaz abstracta que ofrece al programador de aplicaciones una visión uniforme de elementos heterogéneos de bajo nivel. (3)

Middleware es un software que sirve para la conectividad entre dos o más aplicaciones distribuidas y heterogéneas que permite gestionar y coordinar los mecanismos de comunicación.

Streaming

El término inglés "*Streaming*", se refiere a ver u oír un archivo directamente en una página Web sin necesidad de descargarlo antes al ordenador. Esta tecnología aparece por primera vez en abril de 1995 con el lanzamiento de RealAudio 1.0, (formato de audio propietario creado por Real Networks). El

Streaming se compone de códec, Protocolos Ligeros, Precargas, y una red de distribución de contenido. Y se usa para radio y tv por internet, *Interlacing*[1], *Virtual Network Computing*[2]. (4)

Si no se utiliza *streaming*, para mostrar un contenido multimedia en la Red, se tiene que descargar primero el archivo entero hacia el ordenador y más tarde ejecutarlo, para finalmente ver y oír lo que el archivo contenía. Sin embargo, el *streaming* permite que esta tarea se realice de una manera más rápida y que se pueda ver y escuchar su contenido durante la descarga. (5)

El *streaming* es el término que se le otorga a la manera de reproducir un audio o video de una página Web sin tener que descargarlo hacia el disco duro. Con esta tecnología un archivo puede ser descargado y reproducido simultáneamente, por lo que el tiempo de espera es mínimo.

Transmisión de radio por medio de la Web

La transmisión de radio por medio de la Web no es más que un *Stream casting* de audio, en otras palabras es similar a una radio normal solo que su señal es transmitida a través del internet. Propiamente dicho, consiste en la exhibición de contenido auditivo dotado de las características propias del medio radiofónico a través del internet mediante *streaming*. (4)

La transmisión de la radio por una Web trae aparejada la ventaja de que la misma no se encuentra limitada a solamente una transmisión de audio, sino que también puede incluir textos e incluso fotografías, lo cual hace de este un método de transmisión más aceptado por el público.

Transmisión de televisión por medio de la Web.

Los primeras formas de televisión por internet es por el "Video *Streaming*" que es el vídeo seleccionable desde algún lugar de Internet, normalmente un sitio Web (Página de internet), o desde un programa. Actualmente con el aumento de las velocidades de conexión a Internet, el avance de la tecnología, el aumento del número total de internautas en línea, y la disminución en gastos de conexión, se ha hecho

[1] ***Interlacing*** **(Entrelazado)** es un método de codificación que se realiza a las imágenes de un video, utilizado en algunas televisoras.
[2] ***Virtual Network Computing*** es lo que nos permite tomar control de una máquina servidor remotamente desde un ordenador cliente.

cada vez más común encontrar el contenido tradicional de televisión accesible libremente y legalmente sobre Internet. (4)

La transmisión de televisión por medio de una Web es una forma que en la actualidad ha tenido gran aceptación ya que la mayoría de los internautas ya no conocen o utilizan la televisión regular.

Comunicación entre procesos (IPC).

Los procesos pueden estar ejecutándose en una o más computadoras conectadas a una red. Las técnicas de IPC (*Inter-process communication*) están divididas dentro de métodos para: flujo de mensajes, sincronización, memoria compartida y llamadas de procedimientos remotos (RPC). El método de IPC usado puede variar dependiendo del ancho de banda y latencia de la comunicación entre procesos, y del tipo de datos que están siendo comunicados. (6)

En la comunicación orientada a *streams* los modelos RPC y RMI (Invocación remota de métodos) realizan comunicaciones independientes del tiempo. Existen también sistemas donde el tiempo es crucial en la comunicación, o los resultados de salida serán incorrectos; es así el caso de trasmisión de audio, video, sensor de datos, etc. (comunicación continua de datos) donde cortes de comunicación generan retardos no deseados. (6)

IPC provee una forma de comunicación y sincronización para los procesos a través del paso de mensajes, donde los procesos pueden estarse ejecutando en diferentes computadoras conectadas a una red.

Plataforma de Transmisión Abierta para Radio y Televisión.

La plataforma cuenta con 8 Subsistemas: Subsistema Seguridad, Subsistema Transferencia, Subsistema Web, Subsistema Administración de la Transmisión, Subsistema Transmisión, Subsistema de Programación, Subsistema Gestión de Medias, Subsistema Reporte.

- ✓ El Subsistema Seguridad brindará las siguientes prestaciones generales:
    a) Permitir que los usuarios se puedan autenticar al sistema.
    b) Permite según el rol del usuario registrado los privilegios correspondientes.
    c) Permitir la adición, modificación y eliminación de los usuarios de la plataforma.

d) Eliminar un determinado rol para un usuario seleccionado.

- ✓ El Subsistema Transferencia brindará las siguientes prestaciones generales:
  a) Convertir los ficheros a los diferentes formatos de publicación en el entorno Web.
  b) Permitir el chequeo de la integridad de los materiales.
  c) Mover al servidor *streaming* los materiales que se van a publicar o transmitir por la plataforma.
- ✓ El Subsistema Web brindará las siguientes prestaciones generales:
  a) Permitir al usuario la visualización y escucha de los materiales audiovisuales publicados en el sistema.
  b) Visualizar las estadísticas por puntos de los materiales publicados más visitados por los usuarios.
  c) Gestionar las secciones de materiales a demanda u online que se decidan.
  d) Gestionar la publicación materiales en el entorno Web de la plataforma por secciones previamente creadas.
  e) Cargar automáticamente los datos de los materiales, título, año de producción, procedencia, y enlace a material.
  f) Publicar encuestas donde los usuarios seleccionen los programas de la televisión o de la radio de su preferencia quedando delimitada la cantidad de votos por usuario.
  g) Activar o desactivar el estado de los enlaces a los canales de la televisión y de la radio en vivo o a demanda.
  h) Publicación y visualización de informaciones de texto sobre cualquier tema asociado con la plataforma.
  i) Publicación y visualización en el entorno Web de la cartelera por cada canal que transmite la plataforma.
- ✓ El Subsistema Administración de la Transmisión brindará las siguientes prestaciones generales:
  a) Gestionar la programación de un canal de transmisión que incluye la planificación, creación,

modificación y eliminación de la programación de transmisión del canal.

b) Calcular el tiempo que ocupa en la programación un material determinado.

c) Rellenar espacios vacíos en las programaciones de radio y televisión con otros materiales del fondo audiovisual.

d) Modificar en cualquier momento la programación de transmisión de un canal actualizando automáticamente la cartelera canal en cuestión.

e) Permitir monitorear canales.

f) Permitir la interrupción y reanudación de la programación de un canal.

- ✓ El Subsistema Transmisión brindará las siguientes prestaciones generales:
  a) Cargar programación canal de forma automática.
  b) Cargar los ficheros multimedia a transmitir.
  c) Transmitir la señal de audio o video.
  d) Permitir la transmisión del canal interno acorde a la programación generada.
- ✓ El Subsistema Programación brindará las siguientes prestaciones generales:
  a) Gestionar los espacios de radio y televisión.
  b) Gestionar las programaciones de radio y televisión de los canales de la plataforma.
- ✓ El Subsistema Gestión de Medias brindará las siguientes prestaciones generales:
  a) Gestionar las medias en el servidor de la plataforma y chequear la integridad de las mismas.
- ✓ El Subsistema Reporte brindará las siguientes prestaciones generales:
  a) Mostrar los reportes generados por cada Subsistema.

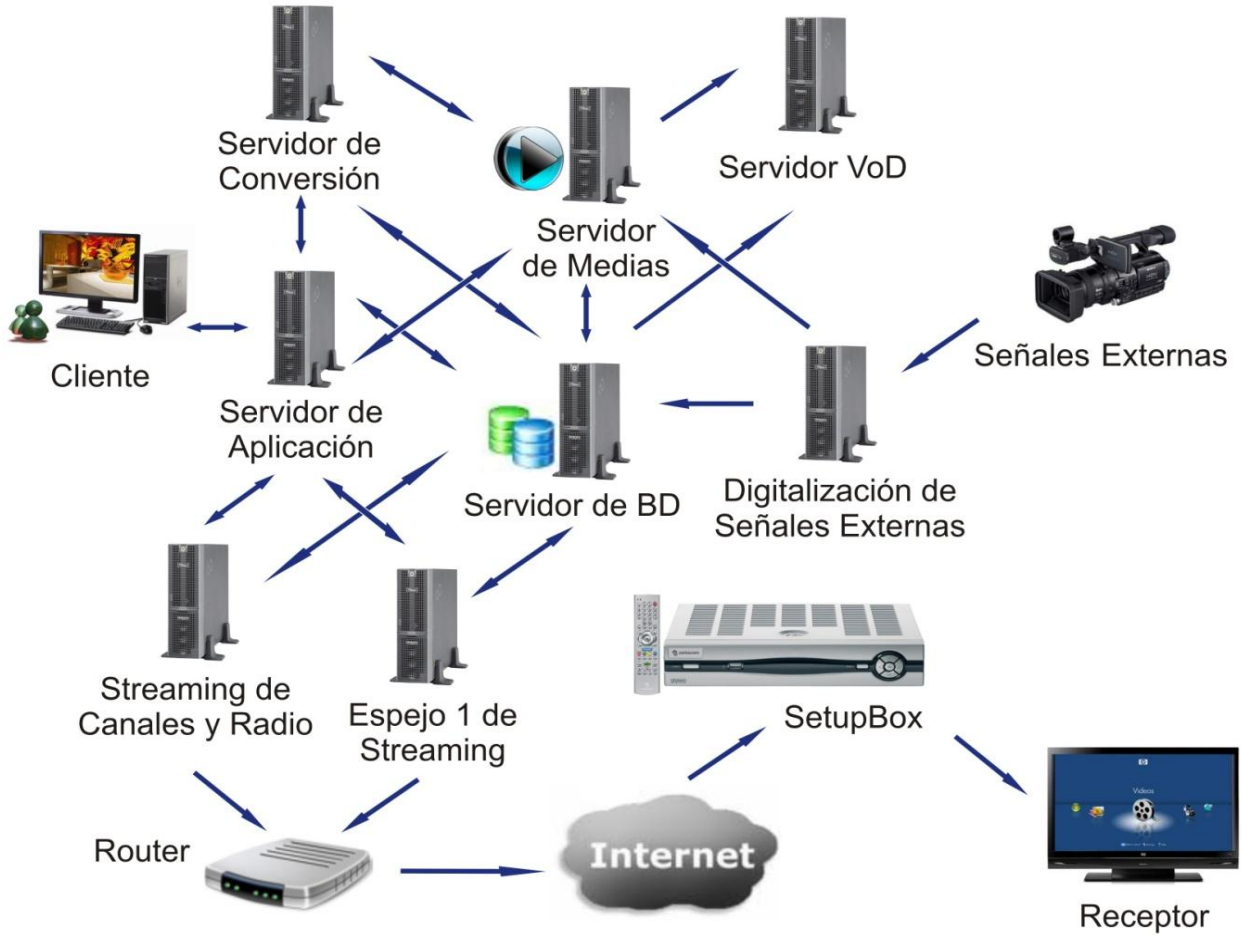

Figura 1: Distribución física de PTARTV.

Con esta figura se puede conocer cómo es el trabajo que hace en sí la PTARTV. Se pudiera comenzar todo este proceso digital desde la cámara con que se filman los videos a transmitir en un futuro, los cuales se pasan directamente hacia una computadora, donde ya se tiene un video digital el cual es un producto que se puede exportar. Se tiene un servidor de medias que es donde se guardarían todos estos videos a transmitir, se tiene el video el cual se registra dentro del sistema de Gestión de Medias el cual luego lo manda al gestor de la transmisión. El Subsistema Programación se guía por el gestor de medias para crear una programación cultural del canal. La aplicación del Subsistema Transmisión recibe esta programación que está asociada a su vez con el servidor de medias. En el servidor de conversión es donde corre el Subsistema Transferencia. En el Servidor de Aplicación es donde corren las otras 5

aplicaciones Web. En el servidor de *Streaming* está trabajando el Subsistema Transmisión. El *Set Top Box* es el encargado de convertir lo que le llega por la red, que sería un fichero o el *Streaming* lo cual lo convirtiéndolo a RCA para poder transmitir dicha señal por un Televisor.

Aplicaciones Web:

1. Subsistema Gestión de Medias
2. Subsistema Programación
3. Subsistema Web
4. Subsistema Transferencia
5. Subsistema Reporte
6. Subsistema Seguridad

Aplicaciones de escritorio:

1. Subsistema Transmisión
2. Subsistema Administración de la Transmisión

1.3 Descripción General del objeto de estudio.

1.3.1 Técnicas comunicativas entre aplicaciones informáticas existentes:

RPC (*Remote Procedure Call*):

Los mecanismos RPC persiguen que los clientes se abstraigan e invoquen procedimientos remotos (operaciones) para obtener servicios. Así, el procedimiento llamado se ejecuta en otro proceso de otra máquina (servidor). El objetivo de RPC es mantener la semántica de la llamada a procedimiento normal en un entorno de implementación totalmente distinto. La ventaja está en que el desarrollador se preocupa de las interfaces que soporta el servidor. Para especificar dichos interfaces se dispones de un IDL (lenguaje de definición de interfaces). (1)

El *modelo RPC* describe como al cooperar procesos en diferentes nodos de una red pueden comunicarse y coordinar sus actividades. El paradigma de RPC está basado en el concepto de una llamada a un procedimiento en un lenguaje de programación. La semántica de RPC es muy parecida a la semántica de

las tradicionales llamadas a procedimientos. La mayor diferencia es que mientras una llamada a procedimiento normal se realiza entre procedimientos de un proceso simple en el mismo espacio de memoria, una RPC se realiza entre un proceso cliente de un sistema (o máquina) y un proceso servidor de otro sistema, donde ambos, la máquina del cliente y la máquina del servidor están conectadas a una red. (7)

Debido a que RPC proporciona una función orientada a interfaces, a menudo es mucho más fácil de usar que la programación con *sockets*. RPC es también lo suficientemente potente como para ser la base para muchas aplicaciones clientes/servidor. Aunque existen diversas implementaciones incompatibles al protocolo RPC, está fácilmente disponible para muchas plataformas.

***Sockets*:**

Una forma de conseguir que dos programas se transmitan datos, basada en el protocolo TCP/IP, es la programación de *sockets*. Un *socket* no es más que un "canal de comunicación" entre dos programas que corren sobre ordenadores distintos o incluso en el mismo ordenador. (8)

Desde el punto de vista de programación, un *socket* no es más que un "fichero" que se abre de una manera especial. Una vez abierto se pueden escribir y leer datos de él con las habituales funciones de read() y write() del lenguaje C. (8)

Al utilizar los *sockets* es una forma de conseguir que dos programas puedan enviarse mensajes. Dos programas que corren en el mismo o en distintos ordenadores abren una comunicación entre ellos y se envían datos. Simplifica el trabajo de los programadores. En la mayoría de los sistemas operativos se hace uso de los *sockets*.

ICE:

Ice (*Internet communication engine*) es una plataforma *middleware* orientada a objetos. Fundamentalmente esto significa que ICE provee herramientas, interfaz de programación de aplicaciones (API) y soporte de librerías para construir aplicaciones cliente servidor orientadas a objetos. Las aplicaciones ICE son ideales para usar en entornos heterogéneos: cliente y servidor pueden ser escritos en diferentes lenguajes de programación, pueden ejecutarse en diferentes sistemas operativos y en

máquinas con arquitecturas diferentes y pueden comunicarse usando gran variedad de tecnologías de redes. El código fuente de estas aplicaciones es portable sin tener en cuenta el entorno de instalación. (9) ICE es un *middleware* útil para programadores. Una plataforma de comunicación a través de Internet de alto rendimiento, ICE incluye una gran variedad de servicios y *plug-ins*[1]. (10)

ICE fue influenciado por CORBA en su diseño. Es una plataforma ligera desarrollada por la empresa ZeroC[2]. Actualmente este *middleware* es gratuito y es muy utilizado para realizar aplicaciones distribuidas.

ICE es más que una plataforma de RPC. Además de las funciones y alto rendimiento de RPC, ICE proporciona varios servicios. Estos servicios proporcionan funcionalidad que la mayoría de las aplicaciones distribuidas requieren.

CORBA:

CORBA (*Common Object Request Broker Architecture* - arquitectura común de intermediarios en peticiones a objetos), 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 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. (11)

CORBA es un estándar, definido por el OMG[3] (*Object Management Group*), que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos. (12)

CORBA es un *middleware* que hace posible que los objetos interactúen a través de lenguajes de programación, protocolos de comunicación y plataformas heterogéneas.

[1] Un **plug-in** es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API.
[2] **Zeroc** es la empresa que produce el *middleware* ICE.
[3] **OMG** es el consorcio encargado de definir los estándares para la aplicación de tecnologías orientadas a objetos tales como CORBA.

SOA:

SOA (Arquitectura orientada a servicios) supone la utilización de servicios débilmente acoplados y altamente interoperables para dar soporte a los requisitos de los procesos de negocio y los usuarios. En un entorno SOA los recursos de la red se hacen disponibles a través de servicios independientes que pueden ser accedidos a través de métodos estándar y sin necesidad de conocer cómo están implementados internamente. (8)

SOA es una arquitectura de software que permite la creación y/o cambios de las operaciones ejecutadas en una determinada secuencia con el fin de realizar una tarea.

RMI:

La idea que sustenta RMI (Invocación remota de métodos) es un ideal recurrente en tecnologías de Orientación a Objetos: hacer que un objeto invoque a otro objeto remoto, como si fuera un objeto local. Transparencia respecto a la máquina. (13)

RMI es un mecanismo que permite realizar llamadas a métodos de objetos remotos situados en distintas (o la misma) máquinas virtuales de Java, compartiendo así recursos y carga de procesamiento a través de varios sistemas. (14)

Es un mecanismo de expansión de RPC cuyo objetivo es dar soporte a sistemas orientado a objetos. La idea es tener objetos distribuidos. Es usado únicamente para la comunicación entre aplicaciones que usen Java.

DCOM

DCOM (Distributed Component Object Model) en español Modelo de Objetos de Componentes Distribuidos, es una tecnología propietaria de Microsoft para desarrollar componentes distribuidos sobre varios ordenadores y que se comunican entre sí. La base de DCOM es el Microsoft RPC (MSRPC). Si bien se ha trabajado en la implementación de DCOM para su compatibilidad con sistemas UNIX, esta tecnología fue ideada para su uso sobre sistema operativo Microsoft. DCOM soporta varios lenguajes de

programación, como Visual Basic, C y C++. (15) Microsoft y Software AG[1] exportaron DCOM a todos los sistemas operativos del universo conocidos hasta el momento y como es lógico, empezaron con Solaris y Linux. (16)

DCOM es el resultado de la evolución y convergencia de dos tecnologías: la comunicación inter-procesos en los ambientes Windows de Microsoft y los esfuerzos de la OSF (Open Software Foundation) para desarrollar un ambiente de computación distribuido (DCE, Distributed Computing Environment), en particular un protocolo para la invocación de procesos remotos (RPC). (16)

DCOM es un modelo de objetos relativamente sólido que cuenta con un apoyo particularmente bueno en los sistemas operativos de Microsoft, ya que está integrado con Windows 95 y Windows NT.

DCE

DCE (*Distributed Computing Environment*), es una tecnología que es un poco más vieja que CORBA, pero bastante más estable y escalable. No sólo ofrecen los mecanismos básicos de RPC y compiladores stub[2] IDL necesarias para crear aplicaciones distribuidas, pero también ofrece seguridad y autenticación mediante *Kerberos*[3]. (17)

DCE es una tecnología del consorcio OSF[4] (*Open Software Foundation*) que salió al mercado en 1990 por lo que hoy en día existe más bien como una curiosidad histórica.

1.4 Situación Problemática

PTARTV tiene como objetivo proveer de una solución tecnológica integral que permita automatizar la transmisión, administración, gestión de los canales de radio y de televisión. Es un sistema que gestiona, ofrece integradamente los servicios de audio y video con nuevas técnicas de interactividad, haciendo uso además, de tecnologías de desarrollo Web y tecnología *streaming*.

[1] **Software AG:** es una compañía Alemana, encargada de producir software y *middlewares* tales como SOA.
[2] **Stub**: Son librerías que emplean los programadores, las cuales se generan de forma automática.
[3] **Kerberos** es un protocolo de seguridad que usa una criptografía
[4] **OSF** es un consorcio encargado de desarrollar y comercializar software abiertos de cualquier fabricante.

Tal como su nombre lo indica PTARTV es una plataforma de transmisión que debe de proporcionar tanto como el software como el hardware que se necesite para crear una transmisión de radio y televisión. Es una plataforma abierta porque los usuarios que accederán a los servicios de PTARTV no tendrán que pagar. PTARTV presenta las herramientas capaces de difundir informaciones mediante un sitio Web.

Entre el Subsistema Transmisión y Subsistema de Administración de la Transmisión existe como deficiencia que no se encuentran conectados entre ellos por lo tanto se hace necesario realizar una investigación sobre las posibles técnicas aplicables para establecer la conexión. El Subsistema Administración de la Transmisión depende del Subsistema Transmisión que a su vez depende del Subsistema Programación. Lo que hace necesario unificar los Subsistemas Transmisión y el de Administración de la Transmisión a la plataforma como un producto portable que pueda ser personalizado en dependencia de las necesidades del cliente. Con la unión de Administración de la Transmisión se pretende transmitir las medias programadas para cada canal de la plataforma y que el Subsistema Transmisión que es el encargado entre otras cosas de cargar dicha programación automáticamente. De dicha unión depende la correcta transmisión de las medias que se pondrán a disposición de los usuarios de la plataforma.

La carencia de una técnica que gestione la comunicación entre el Subsistema Transmisión y el Subsistema Administración de la Transmisión de PTARTV, técnica que permita además establecer una comunicación instantánea para una transmisión en vivo, constituye un elemento importante a tomar en consideración.

1.5 Conclusiones del capítulo

En este capítulo se ha hecho un recorrido por los principales conceptos y definiciones que servirán de apoyo para el estudio y reflexión de la investigación, los cuales servirán también para la posible toma de decisiones y el correcto desempeño de la investigación. Se ha hecho un breve estudio de las diferentes técnicas de comunicación entre aplicaciones, sirviendo de inicio al estudio que requiere la investigación. Donde se puede decir que los *middlewares* son una pieza muy importante para la creación de aplicaciones distribuidas. Con los *middlewares* se encuentra la forma de establecer conexiones para las grandes plataformas que se están creando actualmente, las cuales se crean en ambientes distribuidos.

CAPÍTULO 2

2.1 Introducción

Como se ha podido apreciar en el capítulo anterior, los *middleware* constituyen una de las partes más importantes en cualquier proyecto de software distribuido, es por eso que mientras más se conozca de ellos y de sus características, mejor será todo el proceso de desarrollo de los mismos, proceso de estudio, evaluación y selección de la tecnología más adecuada para utilizar.

2.2 Antecedentes

La disponibilidad de computadoras personales de altas prestaciones, estaciones de trabajo y ordenadores servidores ha resultado en un mayor desplazamiento hacia los sistemas distribuidos en decaída de los ordenadores centralizados multiusuario. Esta tendencia se ha acelerado por el desarrollo de software para sistemas distribuidos, diseñado para soportar el desarrollo de aplicaciones distribuidas. Este software permite a los ordenadores coordinar sus actividades y compartir los recursos del sistema - hardware, software y datos. (1)

La computación distribuida comenzó alrededor de 1985 que hasta este momento sólo se conocía la computación centralizada, desde entonces la industria de la informática ha estado usando plataformas *middleware* orientadas a objeto, tal como ICE y CORBA. Los *middlewares* eran un paso importante para hacer más fácil para los desarrolladores la creación de aplicaciones distribuidas.

2.3 Características de las posibles soluciones.

Una característica importante de las grandes redes de ordenadores actuales, como Internet, es su heterogeneidad. La heterogeneidad permite utilizar la mejor combinación de hardware y software, aumentando el rendimiento de las aplicaciones sin afectar a su interoperabilidad, consiguiendo un sistema coherente, eficiente y altamente operativo. Pero la práctica demuestra que cumplir los requerimientos de seguridad, eficiencia, flexibilidad y extensibilidad, en sistemas distribuidos heterogéneos es, desafortunadamente, raramente fácil. (18)

Por lo que para crear aplicaciones distribuidas con soporte de red, los programadores recurren a *middlewares* tales como ICE y CORBA, por lo cual se proponen entre las posibles soluciones a aplicar para lograr una comunicación entre los Subsistemas Transmisión en PTARTV.

2.3.1 CORBA:

Fue publicado por primera vez en 1990 por la OMG. La generalización de interfaces de CORBA y la semántica de *middleware* orientado a objetos. Incluye una especificación para el *Object Request Broker* (ORB), una biblioteca de software con objeto CORBA, interfaces normalizados que permite a los clientes y los objetivos que se comunican entre sí a través de una red de una manera bien definida. Además, CORBA aplica automáticamente una gama de servicios útiles a las comunicaciones. Después de que se inicializa el ORB, todos los objetos CORBA pueden ser invocados por aplicaciones como objetos de software locales. (19)

2.3.1.1 Características generales de *CORBA*:

- CORBA es independiente del sistema operativo y soporta varios lenguajes como son Java, C, C++, Ada, Cobol y Visual Basic.

2.3.1.2 Algunos aspectos de los sistemas distribuidos en que se basa CORBA:

- *Interface Definition Lenguaje* (IDL): Es un lenguaje que define una interfaz para los objetos CORBA, con el cual se pueden implementar por separado el cliente y el servidor. La definición de la interfaz IDL es independiente del lenguaje de programación.
- Corba*Services* (Servicios CORBA): Son implementaciones de funciones básicas utilizadas para la construcción de aplicaciones.
- Corba*Facilities* (Facilidades CORBA): Son implementaciones que cubren los servicios de alto nivel como la administración de redes y que facilita la creación de aplicaciones habituales para las empresas.
- Corba*Domains* (Interfaces de dominio CORBA): Son como unos servicios verticales que proveen funcionalidad a usuarios finales en campos de aplicación particulares.
- *General Inter-ORB Protocol* (GIOP). Es un protocolo que define los mensajes y el empaquetado de datos que se transmiten entre objetos. Además, define las implementaciones que permiten la comunicación entre los ORBs.

2.3.1.3 Servicios de CORBA

Los Servicios de CORBA son implementaciones de interfaces específicas definidas por el OMG. Estos incluyen cosas como la seguridad, la persistencia, transacciones, consultas, intercambio de objetos, nombres, eventos, control de concurrencia, y las colecciones. Más servicios se están definiendo con regularidad. Relativamente pocos de estos casos aún no aparecen en los productos comerciales, y mucho menos en el *Open Source*. Algunos de los servicios básicos, tales como nombres y acontecimientos, a menudo son incluidos con la aplicación de CORBA base. (17)

Entre los servicios más conocidos se encuentran:

- Nombrado (*Naming*): correspondencia entre nombres convenientes de objetos y referencias a objetos reales. (20)
- Ciclo de Vida (*Object Life Cycle*): creación, borrado, copiado y traslado de objetos. (20)
- Eventos (*Event*): registro para la notificación requerida y esperada de la ocurrencia de eventos. Extensión: Notificación (*Notification*). (20)
- Estado persistente (*Persistent State*), antes Objeto Persistente (*Persistent Object*): existencia a largo plazo de objetos, gestión del almacenamiento de objetos. (20)

Publicados por OMG/Wiley como COSS Volumen I en 1994.

- Relación (*Relationship*): gestión de representación y consistencia de relaciones entre objetos. (20)
- Externalización (*Externalization*): capacidad para almacenar la representación de objetos en medios removibles y permitir más adelante la re-internalización. (20)
- Transacciones (*Transaction*): combina el paradigma de transacciones y el de objetos para tratar los problemas del procesamiento de transacciones comercial. (20)
- Control de Concurrencia (*Concurrency Control*): gestión de la ejecución concurrente en un entorno distribuido. (20)

Publicados como CORBA*Services* en 1995.

- Consulta (*Query*): permite a usuarios y objetos invocar consultas en colecciones de otros objetos (1995). (20)
- Propiedades (*Property*): define operaciones para crear y manipular conjuntos de propiedades (parejas nombre-valor) asociadas a objetos (1995) (20)

- Licencia (*Licencing*): mecanismos para que los productores controlen el uso de su propiedad intelectual (1995). (20)
- Seguridad (*Security*): identificación, autenticación. (1996).
- Tiempo (*Time*): hora y temporizadores (1996)

2.3.1.4 Las facilidades de CORBA se dividen en dos categorías:

- Facilidades horizontales que son las que consisten en interfaces de usuarios, administración de información, administración de sistemas y administración de tareas.
- Facilidades verticales son las especificaciones IDL utilizadas en sectores específicos del mercado, como la salud y el comercio electrónico.

2.3.1.5 Algunos componentes de CORBA

- *Object Request Broker (agente de petición de objetos)* - (u ORB) un ORB es una pieza de "*middleware*", como se le llama, a lo que se sitúa en medio de los clientes y servidores y hace posible la fácil comunicación entre ellos. El ORB es un ente conceptual, que algunas veces toma la forma de una librería compartida y en otras ocasiones toma la forma de un programa externo. El ORB es el responsable de establecer y destruir las sesiones entre clientes y servidores, dirigir, y transportar mensajes entre ellos durante una sesión. (21)
- *Object Adaptor* (adaptador de objetos) - (u OA) un *Object Adaptor* proporciona el canal por el cual un *object server* se comunica con el *Object Request Broker* (ORBit). (21)
- GIOP/IIOP (*General Inter-ORB Protocol - Internet Inter-Orb Protocol*) CORBA toma todos estos lenguajes isms[1] y *handles*[2] orientados a objetos llevándolos entre componentes software orientados a objetos. Si las dos piezas de software están en diferentes lugares, entonces se usará el protocolo GIOP para mover la información entre ellos. IIOP es una especificación de GIOP para el conjunto de protocolos de Internet; teóricamente se podría hacer funcionar GIOP sobre otra cosa distinta al IP, y en este caso se le llamaría de otra manera distinta. (21)

[1] **ISMS** quiere decir Sistema de Información de Gestión de la Seguridad.
[2] **Handles** es un puntero que es usado cuando una aplicación hace referencia a bloques de memoria u objetos gestionados por otros sistemas, como una base de datos o un sistema operativo.

2.3.1.6 El ORB de CORBA

Misión del ORB

Una parte fundamental de la arquitectura CORBA es el ORB, componente software cuyo fin es facilitar la comunicación entre objetos. El ORB se encarga de enviar las peticiones a los objetos y retornar las respuestas a los clientes que invocan las peticiones. (18) Se le conoce como el núcleo de las comunicaciones entre las aplicaciones CORBA ó como el mensajero de las solicitudes entre los objetos y los clientes en una red remota. El ORB está configurado para acceder a los servicios de CORBA.

La principal característica del ORB es la transparencia, cómo facilita la comunicación cliente/servidor. Generalmente, el ORB oculta lo siguiente:

- Ubicación de los objetos. El cliente no sabe dónde se encuentra el objeto destino. Puede residir en un proceso diferente en otra máquina a través de la red, o dentro del mismo proceso. (18)
- Implementación de los objetos. El cliente no sabe cómo esta implementado el objeto remoto, en qué lenguaje de programación o de *scripts* está escrito, o el sistema operativo y el hardware, sobre el que se ejecuta. (18)
- Estado de ejecución del objeto. Cuando el cliente lanza una petición sobre un objeto remoto, no necesita saber si el objeto está en ese momento en ejecución, y listo para aceptar peticiones. El ORB de forma transparente inicializa el objeto en caso de ser necesario, antes de enviarle la petición. (18)
- Mecanismos de comunicación de los objetos. El cliente no sabe qué mecanismos de comunicación (por ejemplo, TCP/IP (Protocolo de Control de Transmisiones / Protocolo de Internet), memoria compartida, llamada a método local) utiliza el ORB para enviar la petición al objeto y retorna la respuesta al cliente. (18)

Con las características del ORB se logra que los programadores de las aplicaciones sólo deban preocuparse por las cuestiones propias del dominio de sus aplicaciones, ya que éste le oculta la dificultad existente en las comunicaciones de redes.

2.3.1.7 Paquetes de CORBA

- AdaBroker es un conjunto de herramientas y librerías para desarrollar aplicaciones en lenguaje de programación Ada. Proporciona un analizador de IDL, generador de código Ada. AdaBroker es liberado bajo la GPL (*General Public License*). (17)
- ORBit es el ORB para el proyecto GNOME para lenguajes de programación C y Ada que ofrece los servicios de *Naming* y *Events*.
- OmniORB tiene un puerto explícito para Linux. Características de C++ IDL, utiliza IIOP (*Internet Inter-Orb Protocol*) y la IIOP ha sido probado con otros ORB. OmniORB está licenciado bajo GPL. (17)
- MICO: Como resultado de su origen académico en la Universidad de Frankfurt, en 1996, la aplicación MICO tiene un diseño claro y modular, incluso para implementaciones internas que permite asegurar el fácil uso de su extensibilidad. Sólo se basa en C++, la API estándar de Unix y bibliotecas no-propietario. (19) Compatible con los SO Linux, Solaris y Windows NT.
- Orbix es un ORB desarrollado para crear aplicaciones en lenguajes C++, Java y Cobol. Compatible con los SO Linux, Solaris y Windows NT.
- VisiBroker es un ORB desarrollado para crear aplicaciones en lenguajes C++, Java. Compatible con los SO Linux, Solaris y Windows NT.
- OpenORB es un ORB desarrollado para crear aplicaciones en Java. Compatible con los SO Linux, Solaris y Windows NT.
- ORBacus (OmniBroker) es una implementación de CORBA de las más flexibles y mejor implementadas. Además, aunque no es gratis, su código está disponible para ser usado en universidades sin cargo. (22) Es un ORB utilizado para desarrollar aplicaciones en lenguajes de programación C++ y Java. Compatible con los SO Linux, Solaris y Windows NT. Ofrece los servicios de *Naming, Events, Time, Property y Security.*
- OAK (Java) incluye extensiones para tolerancia a fallos.
- COOL ORB (*Contrat Net Protocol Agent Talk*), es un lenguaje para la coordinación, que modela la conversación entre los agentes mediante una máquina de estados finitos. Cada estado representa el punto de la conversación en el que se encuentren. El paso de un estado a otro viene determinado por la llegada del mensaje correcto. De este modo se puede establecer la conducta de los agentes frente a la llegada de los mensajes. (23)

- ILU (*Inter-Language Unification*) Incluye compiladores IDL para C + +, ANSI C, Python, Java, *Common* Lisp y Modula-3. ILU también incluye un equipo autónomo de implementación de la ONC (*Open Network Computing*) RPC, lo que hace posible el uso de los servicios existentes de RPC como objetos ILU. ILU también incluye un equipo autónomo de aplicación de HTTP (*Hyper Text Transfer Protocol*), lo que permite crear aplicaciones CORBA basados en servidores Web. (17)
- Arachne surge de un esfuerzo para desarrollar los componentes reutilizables para el ámbito médico. Arachne se ejecuta en una variedad de plataformas, incluyendo 95/NT, Mac, SunOS, HP / UX, pero Linux es la plataforma primaria de desarrollo. (17)
- TAO está ampliamente desplegada en sistemas de misión crítica, pero pueden fácilmente ser aplicado a los escenarios de uso general para los usuarios de desarrollo en C++. (24) Publicado bajo licencia GNU es compatible con los SO Linux, Solaris y Windows NT.
- Electra Este ORB sobre Isis define un mecanismo básico de transferencia de estado, en la que el BOA (*Basic Object Adapter*) con el que trabaja (el POA, *Portable Object Adapter*, es una estandarización bastante posterior a Electra). (25)
- COPE es un ORB escrito completamente en Perl. No proporciona un IR (*Interface Repository*), pero puede utilizar las herramientas de terceros para ese fin (como Orbacus (OmniBroker)). (17)
- JacORB es un CORBA ORB escrito completamente en Java. Dispone de un compilador de Java IDL, soporta hilos, IIOP, IR. Se conoce que funciona bien en Linux y es más compatible que algunos productos reconocidos de CORBA. Libre, bajo licencia GPL.
- Jorba es una implementación de CORBA escrito completamente en Java. Incluye el ORB, y un compilador de IDL. Se suministra bajo licencia GPL.
- CORBAPlus de ExpertSoft ofrece una aplicación CORBAv2 escrito en Java. Incluye soporte para un compilador de Java IDL, IIOP. Los términos de licencia no están del todo claras, pero parece ser libremente disponible para cualquier uso. No está claro si se ha probado en Linux. (17)
- JavaIDL es una instantánea de compilador de Java SunSoft IDL. Incluye un núcleo de ORB genéricos. (17)
- Fnorb es un ORB experimental escrito en Python. No proporciona un IR (*Interface Repository*), pero puede utilizar las herramientas de terceros para ese fin (como Orbacus (OmniBroker)). (17)

- ROBIN (RPC y *Object Broker Interface*) implementa un subconjunto de la especificación CORBAv2.1. Apoya un IDL, no es compatible con IIOP. Se centra en la adquisición de datos y sistemas de control, incluyendo los ordenadores integrados. (17)
- DOME de Object Oriented Technologies ofrece *callbacks*[1], operación asincrónica, la intermediación de localización, así como monitores de red y herramientas. Puntos fuertes: se ejecuta en una gran variedad de plataformas, y es compatible con muchos protocolos de red diferentes. Ofrece una versión libre para Linux. (17)
- Bionic Buffalo: Este es un producto comercial, una versión libre de la ORB. (17)

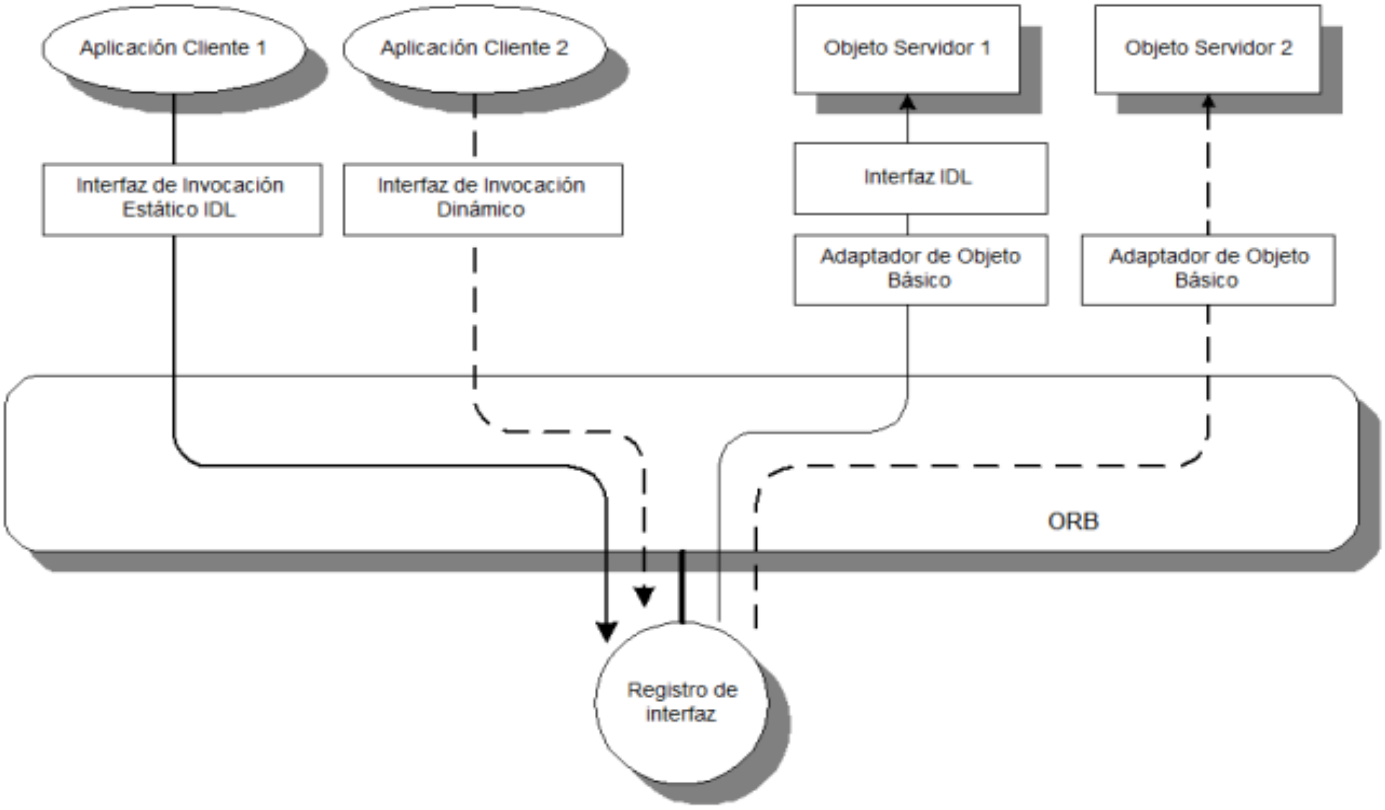

Figura 2: Arquitectura de CORBA. ***(18)***

[1] En programación, un **retrollamado** (en inglés: **callback**) es un código ejecutable que recibe como parámetro la dirección o puntero de otra función, cuando el callback es llamado este recurre al puntero de la función y la ejecuta.

La principal tarea de los adaptadores de objetos es la de hacer de interfaz entre la implementación de un objeto y su ORB.

2.3.2 ICE:

Las aplicaciones ICE son ideales para usar en entornos heterogéneos: cliente y servidor, pueden ser escritos en diferentes lenguajes de programación, pueden ejecutarse en diferentes sistemas operativos y en máquinas con arquitecturas diferentes y pueden comunicarse usando gran variedad de tecnologías de redes. El código fuente de estas aplicaciones es portable sin tener en cuenta el entorno de instalación. (9)

ICE proporciona herramientas, APIs, y apoyo de bibliotecas para construir aplicaciones orientado a objeto cliente–servidor. Las aplicaciones de ICE son adecuadas para uso en entornos heterogéneos: donde cliente y servidor pueden ser escritos en lenguajes de programación diferentes, pueda correr sobre sistemas operativos diferentes, y pueda comunicarse usando una variedad de tecnologías de red. El código fuente para estas aplicaciones es portable a pesar del entorno de despliegue. (26)

El núcleo de ICE ofrece al cliente una sofisticada plataforma cliente-servidor, para el desarrollo de aplicaciones distribuidas. Sin embargo, las aplicaciones realistas por lo general requieren más que sólo una capacidad de interacción remota, también se necesita una forma de iniciar los servidores de la demanda, distribución de *proxies* a los clientes, distribuir eventos asincrónicos, configurar su aplicación, distribuir los parches para una aplicación, y así sucesivamente. (26)

2.3.2.1 Características generales de ICE:

- Es compatible con C++, Java, .NET, Python, PHP, Ruby, y Objetive-C en la mayoría de los principales sistemas operativos como Linux, Solaris, Windows y Mac OS X. (27)
- ICE es software libre, disponible con código fuente completo, y liberado bajo los términos de la GNU *General Public License*[1] (GPL). (27)
- Puede ser utilizado para aplicaciones de Internet sin la necesidad de utilizar el protocolo HTTP y es capaz de atravesar los *firewalls*[2] a diferencia de la mayoría de los *middlewares*.

[1] ***General Public License*** es una licencia creada para proteger de la apropiación de los software libres que estén cubierto por esta.
[2] ***Firewalls*** (cortafuegos) es el software que es utilizado en las redes de computadoras para no permitir el acceso no autorizado a las mismas, en el que se cuentan los usuarios no autorizados o programas dañinos.

- Entre muchas otras características, ICE permite la comunicación utilizando un protocolo altamente eficiente (incluyendo compresión y soporte para los protocolos TCP (Protocolo de Control de Transmisión) y UDP[1]).

2.3.2.2 ICE está formado por los siguientes paquetes:

Usando los servicios ICE se pueden reducir considerablemente el esfuerzo de desarrollo, al tomar la ventaja de una ejecución de fortaleza industrial. (27)

- IceGrid: Es un servicio usado para facilitar la creación de aplicaciones a gran escala. Este servicio habilita a los clientes para hallar los servidores, actuando como intermediario ayuda a mejorar la ejecución y fiabilidad de las aplicaciones.
- IceStorm: es un servicio de distribución de eventos sofisticado. Si un servidor se cae IceStorm desvía automáticamente la comunicación hacia otro servidor que aún esté corriendo, y cuando el antiguo servidor se levanta IceStorm sin intervención manual reasume de forma automática la entrega de eventos hacia el servidor que se levanta.
- Freeze: Con unas pocas líneas de código, una aplicación puede incorporar un vector altamente escalable que gestiona eficientemente objetos persistentes. (10)
- FreezeScript: es necesario para cambiar tipos de datos persistentes, especialmente en proyectos de software grandes. Para minimizar el impacto de estos cambios, FreezeScript proporciona herramientas de inspección y migración para bases de datos *Freeze*. La herramienta soporta scripts basados en XML ya que es potente y fácil de usar. (10)
- Glaciar: Algunos retos de los sistemas middleware son la seguridad y los *firewalls*. Glaciar es una solución firewall de Ice, que simplifica el despliegue de aplicaciones seguras. Autentica y filtra las solicitudes de los clientes y permite callbacks al cliente de una manera segura. Utilizado en combinación con IceSSL, proporciona una solución segura que no permite intrusos y es fácil de configurar. (10)
- IcePath: es un servicio de parcheado para distribuciones software. Mantener el software actualizado es una tarea tediosa. IcePath automatiza las actualizaciones tanto de archivos

[1] ***User Datagram Protocol*** (**UDP**) es un protocolo del nivel de transporte no orientado a conexión. Permite el envío de mensajes para aplicaciones informáticas sin que exista una conexión previa.

individuales y de jerarquías de directorios. Solo los archivos modificados son descargados en las máquinas cliente, utilizando eficientemente algoritmos de compresión. (10)

- Slice: El lenguaje de especificación para ICE. Establece el contacto entre servidores y clientes y además es utilizado para describir datos persistentes. (10)
- Compiladores Slice: Las especificaciones en Slice se pueden compilar en varios lenguajes de programación: C++, Java, Python, PHP, C# y Visual Basic. Los servidores y clientes ICE trabajan juntos, a pesar del lenguaje de programación. (10)
- ICE: El núcleo de librerías ICE, entre otras características este paquete gestiona todas las tareas de comunicación utilizando un protocolo de altamente eficiente, proporciona soporte de hilos para servidores multihilo[1] y funcionalidades adicionales para soportar escalabilidad extrema. (10)
- IceUtil: una colección utilidades como manejo de Unicode y programación con hilos (C++ solo). (10)
- IceBox: Un servidor de aplicación específico para aplicaciones Ice. IceBox puede ejecutar y administrar servicios ICE que son cargados dinámicamente como una DLL[2], librería compartida o clase Java. (10)
- IceSSL: Un plug-in de transporte SSL (*Security Socket Layer*[3]) dinámico para el núcleo Ice. Proporciona autenticación, encriptación e integridad de mensaje, utilizando un protocolo SSL estándar. (10)

2.4 Comparando las posibles técnicas a utilizar.

ICE fue influenciado en su mayor parte por CORBA por lo que los hace parecidos pero también gracias a esto ICE logra ser más simple, con mayor facilidad de uso, y por último pero no menos importante posee una amplia gama de paquetes especificados en el epígrafe anterior.

CORBA por su lado es una plataforma de mucha experiencia en lo que respecta a la creación de aplicaciones distribuidas, la cual salió al mercado desde Octubre de 1991 y se ha ido perfeccionando a través de los años.

[1] **Servidor multihilo** permite atender alrededor de 15 consultas por segundos.
[2] Una **biblioteca de enlace dinámico** tiene varias funciones como la de almacenar funciones o clases, o para guardar cualquier tipo de recursos como texto imágenes y sonido.
[3] ***Security Socket Layer*** es un sistema de seguridad de comunicación.

2.4.1 Algunas diferencias que existen entre ICE y CORBA en cuanto a:

Lenguaje

ICE por su lado soporta C++, Java, Python, Ruby, PHP y .NET. No se tiene conocimiento de ninguna empresa CORBA que ofrezca tantas opciones. De hecho, la mayoría de los fabricantes sólo ofrecen CORBA C++ y Java. Si se desea utilizar otro lenguaje, tiene que cambiar a diferentes proveedores (diferentes ORBs), o considerar el uso de las implementaciones CORBA sin apoyo experimental.

Control de versiones

ICE tiene a Freeze, el servicio de persistencia de ICE, que permite una fácil migración de bases de datos cuando la descripción de Slice persistentes a cambios de datos. CORBA no tiene ningún mecanismo para apoyar el control de versiones.

Seguridad

Con ICE las comunicaciones entre cliente y servidor se pueden asegurar mediante encriptación SSL, de forma transparente para la aplicación. Además por medio de Glacier2 se puede combinar esta tecnología con los *firewalls.*

CORBA no ofrece soluciones prácticas que permitan coexistir al *middleware con firewalls.* Aunque tiene la seguridad de que todas las comunicaciones viajan encriptados.

Actualización de software

ICE cuenta con el paquete IcePatch que le permite mantener el software del cliente actualizado. CORBA no ofrece ningún mecanismo para mantener las actualizaciones del software.

Complejidad

ICE es una plataforma simple. Los servicios o paquetes (Ver epígrafe 2.3.2.2) que ofrece Ice permiten obtener todo lo necesario para el manejo de sistemas distribuidos de la forma más simple posible. En cambio CORBA es grande y complejo. Lo que se debe en parte, a su estandarización ya que debe acomodarse a las condiciones de todas las partes interesadas del OMG, lo que conlleva implementaciones grandes.

Escalabilidad

A ICE se le conoce como un *middleware* eficiente y escalable pero por su parte CORBA fue diseñado para adaptarse a nuevos entornos como Internet, por lo que se conoce que es una tecnología muy escalable.

2.5 Elección de la técnica a utilizar

Teniendo en cuenta todo lo anteriormente planteado en lo que respecta a cada *middleware* comparado, al haberse analizado las características de los mismos, se llega al punto de hacer la propuesta de la técnica que se va utilizar para comunicar los Subsistemas Administración de la Transmisión y Transmisión.

Para decidir cuál va ser la técnica a utilizar, se debe de tener en cuenta algunos aspectos que deben cumplirse. Por lo que se debe de tomar en cuenta que el *middleware* que se escogerá debe de ser implementado por herramientas libres. El *middleware* también debe de cumplir la característica de que sea compatible con lenguaje que se utiliza en PTARTV para desarrollar la plataforma, se ha podido demostrar que tanto ICE como CORBA soportan Java. Ambos son compatibles al sistema operativo Linux (Ubuntu Linux) que es el sistema operativo (SO) que está utilizando el proyecto PTARTV, ya que es un SO gratuito y de código abierto.

Por todo lo anteriormente planteado se puede reconocer que ICE es un *middleware* mejor estructurado y mucho más fácil de utilizar que CORBA, ya que como se ha planteado ICE es una plataforma simple. Aunque CORBA como se conoce es un *middleware* de más experiencia, ICE lo supera en los diversos aspectos por los que fueron comparados en el epígrafe anterior, donde se debe de tomar en cuenta la facilidad que presentan con el Slice independiente del lenguaje de programación que se utilice. Al ser ICE un tecnología actual, se le conoce por ser una plataforma sencilla a la hora de aprender para los programadores y por lo tanto también lo es a la hora de ser utilizada como puente de comunicación entre aplicaciones.

Por lo que se elige el *middleware* ICE para ser utilizado en la plataforma PTARTV, que es una plataforma de mucha experiencia para el desarrollo de aplicaciones distribuidas tal como es el caso de lo que se necesita en la PTARTV para la comunicación de los Subsistemas. (Ver Anexo # 2)

2.6 Conclusiones Parciales

En este capítulo se ha dado a conocer las características de los *middleware* posibles a utilizar para poder lograr la unión de los Subsistemas en cuestión. Se hizo una comparación de las técnicas encontradas para así lograr definir cuál de ellas sería la técnica que se aplicará en el proyecto PTARTV.

Con el trabajo investigativo desarrollado hasta el momento, se ha logrado como resultado una comparación de los aspectos principales entre ICE y CORBA, donde se llegó a la conclusión de que ICE es el *middleware* más correcto de utilizar, con el objetivo de lograr una comunicación entre dos aplicaciones Java.

CAPÍTULO 3

3.1 Introducción

En este capítulo se estarán viendo las tecnologías que se utilizarán en la creación de una aplicación utilizando el *middleware* ICE, para la unión de dos aplicaciones Java. Con el objetivo de plantear en este capítulo un ejemplo del uso de este *middleware*, para que sea utilizado a la hora de unificar los Subsistemas Transmisión y Administración de la Transmisión.

3.2 Modelo, lenguaje y herramientas

Debido al proceso de evolución del mundo de las tecnologías, con el paso de los años las aplicaciones son cada vez más grandes, por lo que se necesitan tecnologías capaces de poder procesar todos los datos de una forma más ágil y eficiente.

3.2.1 Modelo cliente-servidor

La arquitectura cliente-servidor, llamada modelo cliente-servidor o servidor-cliente es una forma de dividir y especializar programas y equipos de cómputo a fin de que la tarea que cada uno de ellos realice, se efectúe con la mayor eficiencia y simplificación posible. (28)

El modelo cliente-servidor es muy conocido por su uso para las comunicaciones de las aplicaciones Web. En este caso se ve su uso con dos aplicaciones de escritorio Java que realizan su comunicación por dicho modelo utilizando el *middleware* seleccionado.

La aplicación cliente (Subsistema Administración de la Transmisión) es el que comienza con la solicitud de un servicio que es respondido en la aplicación servidora (Subsistema Transmisión). Se recomienda el uso del modelo cliente/servidor para entornos que requieran un alto grado de fiabilidad como es el caso de la plataforma PTARTV.

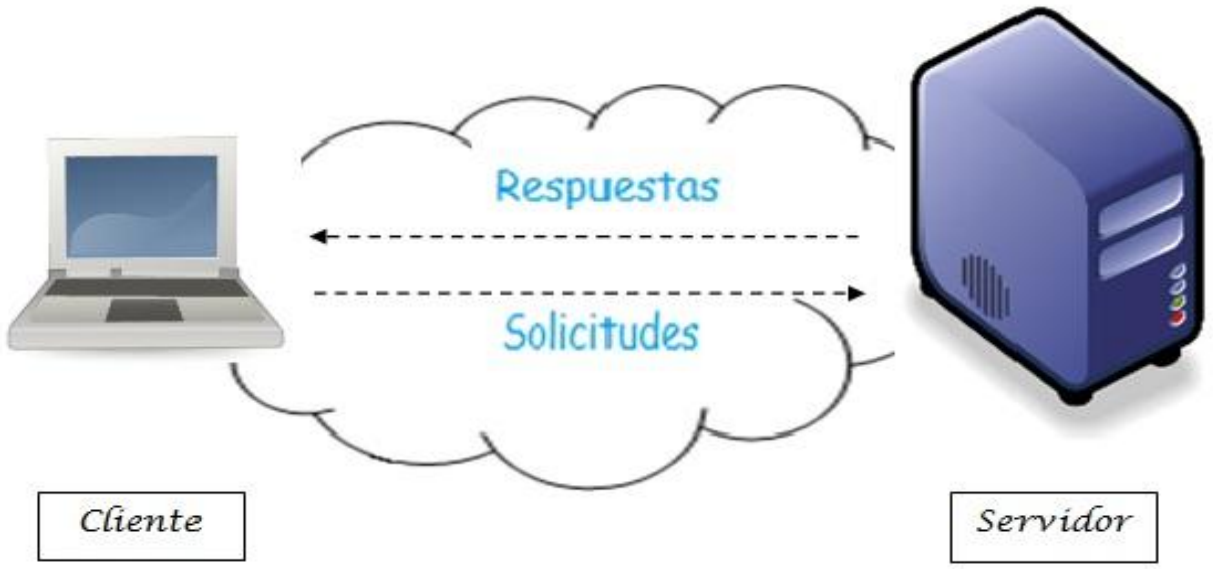

Figura 3: Modelo cliente/servidor

3.2.2 Lenguaje de programación

Los lenguajes de programación facilitan la tarea de programación, ya que disponen de formas adecuadas que permiten ser leídas y escritas por personas, a su vez resultan independientes del modelo de computador a utilizar. Los lenguajes de programación representan en forma simbólica y en manera de un texto los códigos que podrán ser leídos por una persona. (29)

Hoy en día los lenguajes de programación se dividen según donde se implementan guiándose por el modelo cliente/servidor, lo que este caso las dos aplicaciones cliente/servidor son realizadas en Java.

Java

Lenguaje de programación orientado a objetos con el que se puede realizar cualquier tipo de programa, desarrollado por *Sun Microsystems*[1], se basó en los lenguajes C++ y C lo cual hace a Java un lenguaje muy parecido a estos a la hora de programar. Es compatible con los Sistemas Operativos Windows y Linux. Las aplicaciones Java se ejecutan en una máquina virtual, por lo tanto son multiplataforma.

[1] ***Sun Microsystems*** empresa fabricante de semiconductores y software, recientemente adquirida por Oracle *Corporation*.

La programación en Java, permite el desarrollo de aplicaciones bajo el esquema de Cliente Servidor, como de aplicaciones distribuidas, lo que lo hace capaz de conectar dos o más computadoras u ordenadores, ejecutando tareas simultáneamente, y de esta forma logra distribuir el trabajo a realizar. (30)

Sun describe a Java como "simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico". (31)

3.2.3 Sistema operativo

El sistema operativo es el programa (o software) más importante de un ordenador. Para que funcionen los otros programas, cada ordenador de uso general debe tener un sistema operativo. Los sistemas operativos realizan tareas básicas, tales como reconocimiento de la conexión del teclado, enviar la información a la pantalla, no perder de vista archivos y directorios en el disco, y controlar los dispositivos periféricos tales como impresoras, escáner. (32)

En sistemas grandes, el sistema operativo tiene incluso mayor responsabilidad y poder, es como un policía de tráfico, se asegura de que los programas y usuarios que están funcionando al mismo tiempo no interfieran entre ellos. El sistema operativo también es responsable de la seguridad, asegurándose de que los usuarios no autorizados no tengan acceso al sistema. (32)

Los sistemas operativos proporcionan una plataforma de software encima de la cual otros programas, llamados aplicaciones, puedan funcionar. Las aplicaciones se programan para que funcionen encima de un sistema operativo particular, por tanto, la elección del sistema operativo determina en gran medida las aplicaciones que puedes utilizar. (32)

Entre las políticas que sigue la universidad se encuentra el de utilizar software libre para el desarrollo de los diferentes proyectos productivos por lo que PTARTV hace uso del SO Ubuntu Linux.

Ubuntu Linux

Ubuntu es un sistema operativo completamente de código abierto construido sobre la base del *kernel Linux* .La comunidad Ubuntu está basada en las ideas establecidas en la filosofía Ubuntu: que el *software* debe ser accesible totalmente libre de cualquier cargo, que las herramientas de *software* deben ser utilizadas por las personas en su idioma local y sin tener en cuenta cualquier tipo de discapacidad, que las

personas pueden tener la libertad de personalizar y modificar su *software* en la forma que les resulte más provechoso. (33)

Figura 4: Logo de Ubuntu

3.2.4 Software de desarrollo

Maquina Virtual de Java (MVJ) - JDK 1.6.0_04

La Máquina Virtual Java es el núcleo del lenguaje de programación Java. De hecho, es imposible ejecutar un programa Java sin ejecutar alguna implantación de la MVJ. En la MVJ se encuentra el motor que en realidad ejecuta el programa Java y es la clave de muchas de las características principales de Java, como la portabilidad, la eficiencia y la seguridad. Sun utiliza el término "Máquina Virtual Java", para referirse a la especificación abstracta de una máquina de software para ejecutar programas Java. (34)

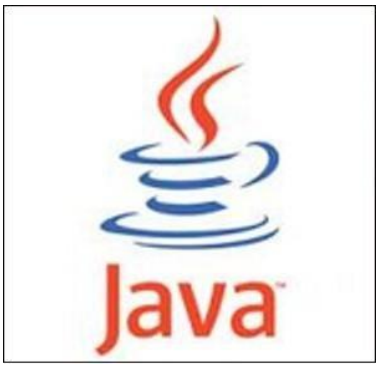

Figura 5: Logo de MVJ

JDK (*Java Development Kit*) es el software que presenta la empresa Sun Microsystems, el cual provee las herramientas y documentación necesarias para el desarrollo de aplicaciones Java. JDK brinda el soporte para cargar aplicaciones en la máquina virtual de Java donde se puede compilar y correr las aplicaciones que se crean, donde se prueba si las mismas cumplen con los requisitos para las que fueron creadas.

Netbeans

NetBeans IDE es un entorno de desarrollo - una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. (35)

Figura 6: NetBeans IDE

3.3 Desarrollo de la aplicación ICE

La aplicación se basará en una de las necesidades de comunicación que presentan los dos Subsistemas. En este caso se desea satisfacer la necesidad de que, al estar encendido el servidor ubicado en la aplicación del Subsistema Transmisión, desde el Subsistema Administración de la Transmisión al darle

click al botón de reproducir media (Play) o Parar media, la aplicación servidora debe de recibir la orden que se le está indicando y ejecutarla. (Ver anexo #3)

Generando el Slice

Slice	Java
bool	boolean
byte	byte
short	short
int	int
long	long
float	float
double	double
string	String

Tabla 1: Tabla de tipos de datos Slice y Java

Slice soporta los tipos: enumerado, estructura, secuencias, diccionarios, constantes, trabajos con excepciones, facilidades para crear interfaces.

bool	enum	implements	module	struct	byte	exception
int	Object	throws	class	extends	interface	out
true	const	false	local	sequence	dictionary	float
void	LocalObject	short	idempotent	double	long	string

Tabla 2: Slice keywords

```
module PTARTV
{
      interface ControlReproductor
      {
            int Stado(int accionX);
      };
};
```

Figura 7: Código Slice

Para lograr la comunicación se especificó en el Slice un método Stado donde se le pasa un parámetro accionX, que es por donde desde la aplicación del Subsistema Administración de la Transmisión se le especifica si debe de realizarse o no dicho método.

Al ejecutar **slice2java PTARTV.ice** dentro de la carpeta donde se encuentra ubicado el archivo Slice, se generan las clases java necesarias a utilizar con el *middleware* ICE en las aplicaciones cliente y servidora.

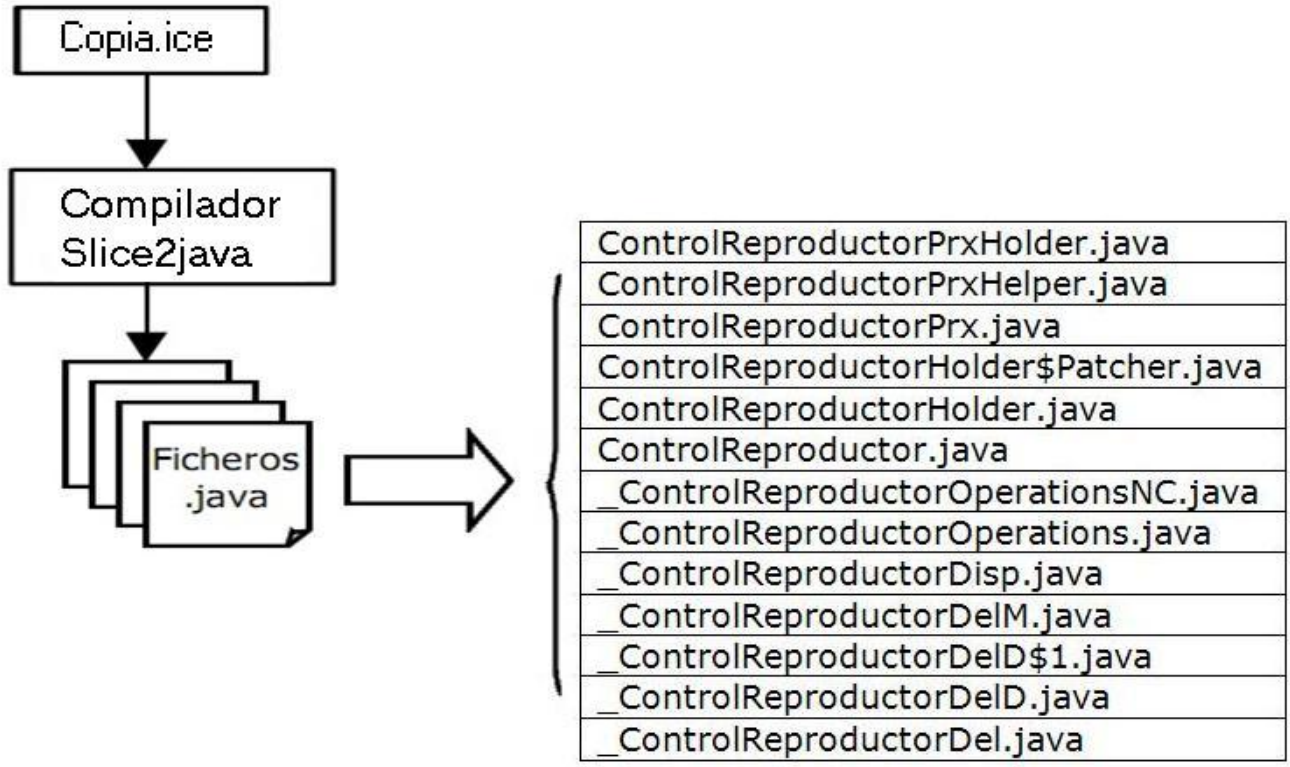

Figura 8: Clases Generadas (36)

Generando el servidor

Al crearse la aplicación que tomará la función de servidora, en la clase Main.java que se genera de la misma, se inicializa el servidor de ICE, se le da un nombre al servidor que se está publicando en la red (ServidorMedias). En esta clase se realizan las siguientes operaciones básicas de cualquier aplicación ICE que se va a implementar:

```
public int Stado(int accionX, Current __current) {
   if(accionX == 1)
   {
      ReproducirMedia.Reproducir_Media("/home/ptartv/Escritorio/a.mp3");
   }
   return 0;
}
```

Figura 9: Código de la clase Reprod.java

Esta clase hereda de la clase _ControlReproductorDisp.java generada por el compilador Slice.

```
1  int status = 0;
2  Ice.Communicator iceO = null;
3  int port=10005;
4  iceO = Ice.Util.initialize(args);
5  Ice.ObjectAdapter adapter = iceO.createObjectAdapterWithEndpoints("ServerMedias",
   "tcp -p "+String.valueOf(port));
6  Ice.Object object = new Reprod();
7  adapter.add( object, ic.stringToIdentity("ServerMedias"));
8  adapter.activate();
9  System.out.println("El servidor de medias esta a la escucha del puerto:
   "+String.valueOf(port));
10 iceO.waitForShutdown();
```

Figura 10: Código de la clase Main.java

En esta clase es donde se inicializa ICE, se crea el objeto adapter para escuchar las peticiones que llegan por parte del cliente, hasta este punto el servidor está inicializado y se publica en la red con el nombre de ServerMedias. Luego se crea un objeto (object) de la clase Reprod() y se pone el servidor a la escucha de las peticiones del cliente por el puerto especificado. Por último se suspenden las entradas de las llamadas hasta que la ejecución del servidor termine en la línea 10.

Generando el cliente

Al crearse la aplicación java donde se implementarán los métodos correspondientes a las funciones de la aplicación cliente. También se genera en la misma una clase Main.java en la cual se realizan las siguientes operaciones básicas para cualquier implementación en la que se utilice ICE:

```
int status = 0;
Ice.Communicator iceO = null;
iceO = Ice.Util.initialize(iCliente.Aux);
Ice.ObjectPrx base = iceO.stringToProxy("ServerMedias :tcp -p 10005 -h 10.7.2.7");
ControlReproductorPrx Repro = ControlReproductorPrxHelper.checkedCast(base);
if (Repro == null)
Repro.Stado(1);
```

Figura 11: Código de la clase Main.java

En esta clase se inicializa ICE tal y como se realiza en la aplicación servidora, es donde se especifica a que servidor es al que se va a conectar la aplicación cliente. Aquí es donde se ejecutan las llamadas de los métodos que se realizan en el servidor.

3.4 Conclusiones Parciales

En este capítulo se ha tomado en consideración la utilización de tecnologías o software libres a la hora de implementar las dos aplicaciones Java. Se ha realizado un análisis del *middleware*, lenguajes de programación, metodologías de desarrollo y herramientas que serán usados en el desarrollo de la propuesta de desarrollo del *middleware*. Se ha logrado mostrar por pasos la construcción de las aplicaciones que servirán de ejemplo para la propuesta del *middleware* que se utilizará en la PTARTV, propuesta hecha con el objetivo de lograr la comunicación de los subsistemas en cuestión.

CONCLUSIONES GENERALES

Durante el desarrollo del trabajo de diploma y el tiempo dedicado a la investigación e implementación de las aplicaciones con el fin de demostrar el uso del *middleware* propuesto se logró:

- ✓ Realizar un análisis sobre las técnicas comunicativas entre aplicaciones informáticas existentes.
- ✓ Demostrar las grandes posibilidades que brinda los *middlewares* a la hora de comunicar dos aplicaciones informáticas.
- ✓ Validar la propuesta del *middleware* ICE, basado en una de las necesidades surgidas en el proyecto PTARTV.
- ✓ Desarrollar y documentar una aplicación con el uso de ICE.

Por todo lo anterior se concluye que los objetivos propuestos para el presente trabajo investigativo, han sido cumplidos satisfactoriamente.

Recomendaciones

Teniendo en cuenta que la propuesta realizada se basa en las necesidades presentadas en el proyecto PTARTV se plantean las siguientes recomendaciones.

- ✓ Aplicar la propuesta a la plataforma PTARTV.
- ✓ Dar continuidad al estudio e implementación práctica del uso de ICE.
- ✓ Mantener el *middleware* pendiente a las actualizaciones.
- ✓ Utilizar la propuesta en otros proyectos productivos del Departamento de Señales Digitales.

Referencias Bibliográficas

1. **Rojo, J. Oscar.** augcy. *augcy.* [En línea] 2003. [Citado el: 26 de 11 de 2009.] http://www.augcyl.org/?q=glol-intro-sistemas-distribuidos.

2. **T, José Camilo Daccach.** PROX.com. *prox.com.* [En línea] 1999/2009. [Citado el: 22 de 11 de 2009.] http://www.proz.com/kudoz/english_to_spanish/it_information_technology/1232972-base_platform_middleware.html.

3. **Blatecky, Alan R. y Sun, Xian-He.** Middleware: the key to next generation computing (Preface). *Journal of Parallel and Distributed Computing.* 2004, 64.

4. El Nuevo Empresario. *el nuevo empresario.* [En línea] 2 de mayo de 2008. [Citado el: 24 de Noviembre de 2009.] http://www.elnuevoempresario.com/noticia_1278_radio-y-tv-por-internet.php.

5. **Alvarez, Miguel Angel.** desarrolloweb.com. *desarrolloweb.com.* [En línea] 1 de julio de 2009. [Citado el: 11 de Diciembre de 2009.] http://www.desarrolloweb.com/articulos/482.php.

6. **Stallings, William.** *Sistemas operativos : aspectos internos y principios de diseño.* s.l. : Pearson Alhambra, 07/2007.

7. **David Cano, Robert Salla, Cèsar Fdez.** [En línea] [Citado el: 17 de Marzo de 2010.] http://fermat.eup.udl.es/~cesar/sh/rpc/modelo.html.

8. **García, Fco. Manuel Márquez.** *Unix, Programación avanzada.*

9. **Muñoz, Jesús M. Conesa.** *Desarrollo de un sistema para toma de decisiones en situaciones de intrusión en instalaciones e infraestructuras.* Madrid : s.n., 2009.

10. ditec.um.es. *ditec.* [En línea] Universidad de Murcia. [Citado el: 18 de Diciembre de 2009.] http://ditec.um.es/ssdd/trabajos/0506/Memoria-icegrid.doc.

11. **Castillo, Alvaro del.** programatium.com. *programatium.com.* [En línea] 1999. [Citado el: 17 de Marzo de 2010.] http://www.calcifer.org/documentos/librognome/corba.html.

12. **Pinedo, Miguel Angel Martínez.** [En línea] Junio de 2009. [Citado el: 12 de Diciembre de 2009.] https://arco.esi.uclm.es/publicdav/pfcs/miguel.martinez.pinedo.pdf.

13. **Lago, Ramiro.** proactividad.com. [En línea] Septiembre de 2008. [Citado el: 11 de 12 de 2009.] http://www.proactiva-calidad.com/java/rmi/introduccion.html.

14. **Montenegro, Enrique Medina.** adictosaltrabajo.com. *adictosaltrabajo.com.* [En línea] 05 de Septiembre de 2003. [Citado el: 15 de Marzo de 2010.]

15. **Sanchez, Francisco García.** tesisenxarxa.net. [En línea] 2007. [Citado el: 11 de Diciembre de 2009.] http://www.tesisenxarxa.net/TDX/TDR_UM/TESIS/AVAILABLE/TDR-0121108-122701//GarciaSanchezFrancisco.pdf.

16. **Hurd, Jim.** clarín.com. *clarin.com.* [En línea] [Citado el: 21 de Marzo de 2010.] http://www.clarin.com/suplementos/informatica/1999/01/13/t-01001d.htm.

17. **Vepstas, Linas.** linas.org. *linas.org.* [En línea] Octubre de 2007. [Citado el: 17 de marzo de 2010.] http://linas.org/linux/corba.html.

18. **Tejedor, Ramón Jesús Millán.** ramonmillan.com. *ramonmillan.com.* [En línea] [Citado el: 18 de marzo de 2010.] http://www.ramonmillan.com/tutoriales/corba.php.

19. mico is corba. [En línea] [Citado el: 17 de Marzo de 2010.] http://www.mico.org/.

20. **Ríos, Juan José Gil.** it.uc3m.es. *it.uc3m.es.* [En línea] [Citado el: 2010 de Marzo de 17.] http://www.it.uc3m.es/mcfp/docencia/si/material/9_services.pdf.

21. **Zoll, Todd Graham Lewis y David "Gleef".** www.linuxlots.com. *linuxlots.com.* [En línea] [Citado el: 18 de Marzo de 2010.] http://www.linuxlots.com/~barreiro/spanish/gnome-es/faq/corba.html#AEN680.

22. **Ruiz, Diego Sevilla.** ditec.um.es. [En línea] [Citado el: 12 de Marzo de 2010.] http://ditec.um.es/ssdd/orbacus.pdf.

23. **Barrera, Crisman Martínez.** ucentral.edu.co. [En línea] [Citado el: 12 de Marzo de 2010.] http://www.ucentral.edu.co/NOMADAS/nunme-ante/11-15/Pdfs%20Nomadas%2015/19c-Agentes.pdf.

24. PrismTech. [En línea] [Citado el: 2010 de Marzo de 21.] www.prismtech.com/section-item.asp?id=634.

25. **Cabañas, Luis Miguel Peña.** [En línea] [Citado el: 2010 de Marzo de 12.] http://grasia.fdi.ucm.es/sensei/docs/sensei.pdf.

26. **Michi Henning y Mark Spruiell.** *Distributed Programming with ICE.* [pdf] s.l. : Zeroc, 2008.

27. zeroc.com. *zeroc.com.* [En línea] Zeroc. [Citado el: 16 de Marzo de 2010.] http://www.zeroc.com/ice.html.

28. **Villamor Lugo, Pickin,Gil Ríos.** www.it.uc3m.es. *www.it.uc3m.es.* [En línea] [Citado el: 20 de Abril de 2010.] http://www.it.uc3m.es/mcfp/docencia/si/material/1_cli-ser_mcfp.pdf.

29. lenguajes de programacion. *lenguajes de programacion.* [En línea] [Citado el: 3 de Mayo de 2010.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

30. Lenguajes de Programacion. [En línea] [Citado el: 4 de Mayo de 2010.] http://www.lenguajes-de-programacion.com/programacion-java.shtml.

31. **Alvarez, Miguel Angel.** desarrolloweb.com. *desarrolloweb.com.* [En línea] 18 de Julio de 2001. [Citado el: 3 de Mayo de 2010.] http://www.desarrolloweb.com/articulos/497.php.

32. **Alojamiento, Posicionamiento y Dominios en Alicante.** [En línea] [Citado el: 22 de Abril de 2010.] http://www.masadelante.com/faq-sistema-operativo.htm.

33. [En línea] [Citado el: 15 de Mayo de 2010.] http://www.ubuntu.com/.

34. **Carballeira, Dr. Félix García.** *Revista Digital Universitaria.* [En línea] 1 de Octubre de 2000. [Citado el: 4 de Mayo de 2010.] http://www.revista.unam.mx/vol.1/num2/art4/.

35. www.netbeans.org. [En línea] [Citado el: 3 de Mayo de 2010.] http://webcache.googleusercontent.com/search?q=cache:lpTiSm7--wUJ:netbeans.org/index_es.html+Netbeans&cd=2&hl=es&ct=clnk&gl=cu.

36. Departamento Tecnologías de la Información y las Comunicaciones . *www.tic.udc.es.* [En línea] [Citado el: 10 de Mayo de 2010.] http://www.tic.udc.es/~fbellas/teaching/adoo-2000-2001/Tema2.pdf.

Anexos

Anexo #1

1. ¿Con cuántos Subsistemas cuenta actualmente la plataforma PTARTV?
2. ¿Qué SO se utiliza actualmente en la realización del proyecto PTARTV?
3. ¿Qué lenguajes de programación se utilizan para el desarrollo de la plataforma?
4. ¿En qué consisten los Subsistemas Transmisión y Administración de la Transmisión?
5. ¿Por qué se necesita realizar la unión de los Subsistemas Transmisión y Administración de la Transmisión?
6. ¿Cuál sería el Subsistema que cumpliría con las labores de servidor a la hora de implementar la unión de los Subsistemas Transmisión y Administración de la Transmisión?
7. ¿Cuál sería el Subsistema que cumpliría con las labores de cliente a la hora de implementar la unión de los Subsistemas Transmisión y Administración de la Transmisión?

Anexo #2

Durante el desarrollo de la investigación para la selección del middleware, se identificó que ICE al ser más actual y por haberse enfocado en CORBA a la hora de desarrollarse, era una mejor opción a la hora de desarrollar la comunicación entre dos aplicaciones Java. Por lo que primeramente se elige dicha tecnología para ser usada en el desarrollo de la comunicación entre los subsistemas Administración de la Transmisión y Transmisión de PTARTV.

ICE una tecnología actual y por esto no es muy conocida, es decir, no se encontraba mucha información referente a sus aplicaciones o uso en el mercado mundial, de hecho sólo se encontraban informaciones importantes en el sitio oficial de la empresa ZeroC, el cual en ocasiones no se puede acceder a él pues aparece dentro de los sitios prohibidos para nuestro país. Por dichas razones se hace un cambio en la dirección que estaba tomando la investigación y se decide luego utilizar el *middleware* CORBA, el cual sí cuenta con abundante documentación, aspecto necesario para conocer cómo desarrollar el uso de este *middleware* en PTARTV. Luego de comenzar a utilizar dicho *middleware* en un caso estudio escogido de unas de las necesidades encontradas en el proyecto PTARTV, se comenzó a encontrar dificultades en su desarrollo en el SO Ubuntu Linux, principalmente a la hora de desarrollar (implementar) la comunicación entre las aplicaciones, más tarde se encontraron algunas de las herramientas necesarias para desarrollar el *middleware* ICE en los instaladores de paquetes del (Synaptic) SO Ubuntu en su versión 9.04 y se hace de nuevo un cambio necesario destinado a proponer como *middleware* definitivo ICE.

Con el uso del *middleware* ICE y luego de haber utilizado CORBA se notó la diferencia en lo que respecta a facilidad de uso entre los dos *middlewares*, ratificando lo expuesto en el epígrafe 2.4 de que ICE es una tecnología simple de utilizar.

Anexo #3

Para validar la propuesta de tecnología de comunicación se identificó un caso de estudio del proyecto PTARTV, donde es necesario desde el Subsistema Administración de la Transmisión invocar a los métodos (play, pause) alojados en el Subsistema Transmisión. Para demostrar la comunicación entre dichas aplicaciones se realizó un fichero **.Ice** donde se describe toda la interfaz de la comunicación y luego se re implementa en la aplicación servidora (Transmisión) los métodos de la interfaz. Para poder ejecutar funciones desde aplicaciones remotas y publicar servicios remotamente visibles.

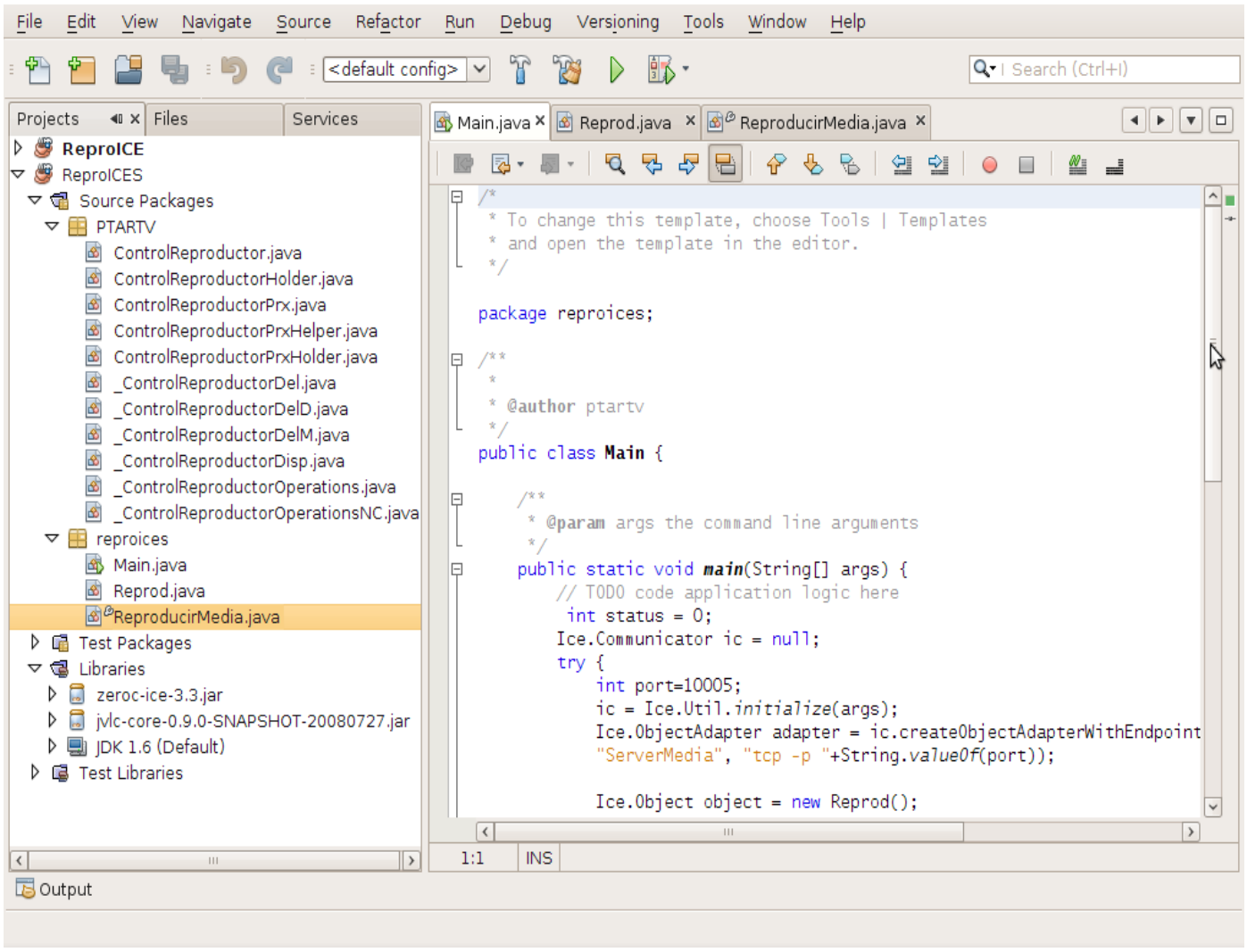

Figura 12: Clases en el Subsistema Transmisión

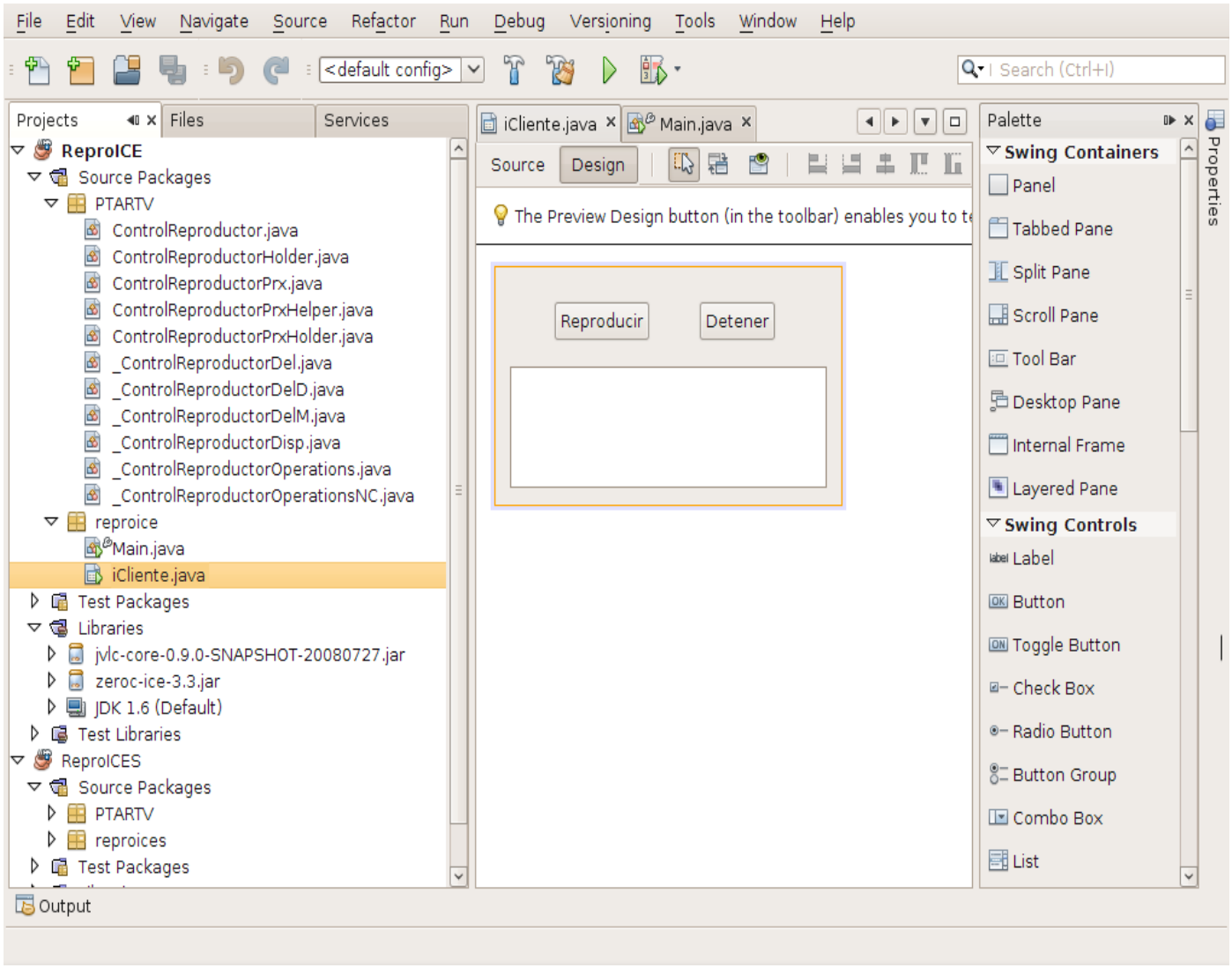

Figura 13: Clases en el Subsistema Administración de la Transmisión

En dichas aplicaciones se hace uso de la biblioteca **zeroc-ice-3.3** necesaria para realizar la implementación del reproductor con el uso del *middleware* ICE que logra la comunicar dichos subsistemas.

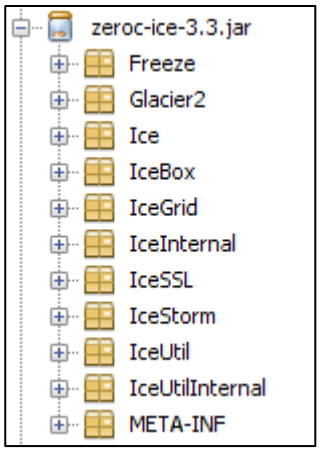

Figura 14: Paquetes de la biblioteca zeroc

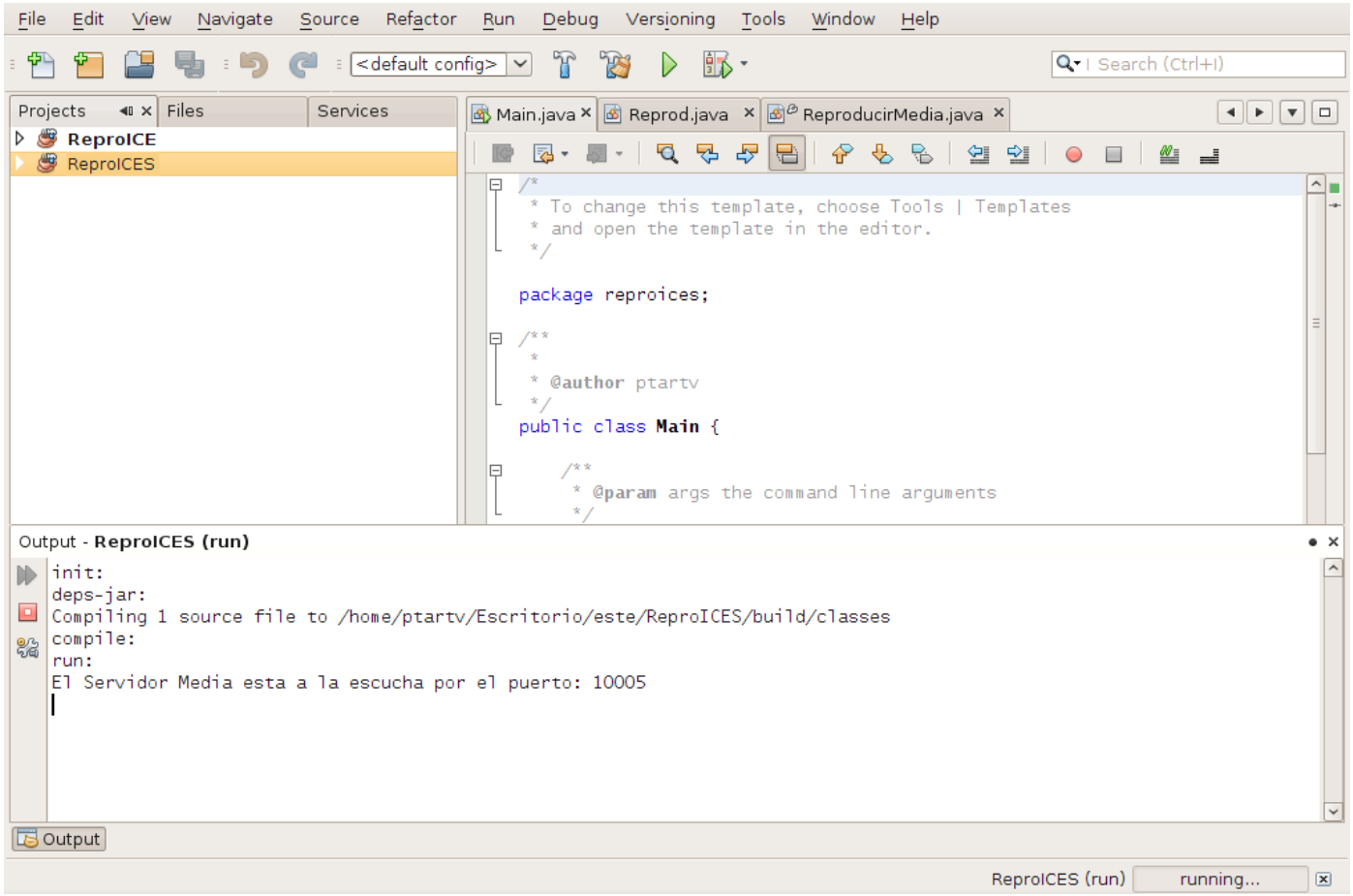

Figura 15: Corriendo la aplicación del Subsistema Transmisión

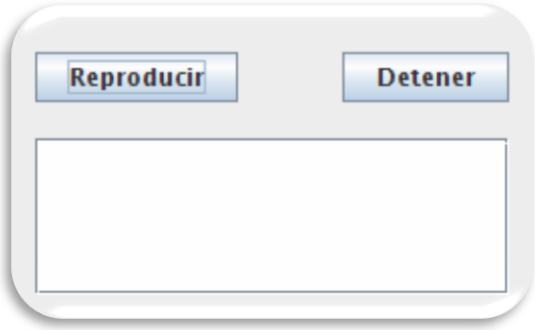

Figura 16: Interfaz de la aplicación del Subsistema Administración de la Transmisión

Al darle clic al botón Reproducir se comienza a reproducir el archivo especificado en la aplicación del Subsistema Transmisión el cual ejecuta el método play que se encuentra implementado en dicho subsistema.

Glosario de términos

API (*Application Programming Interface* – Interfaz de programación de aplicaciones): Conjunto de rutinas y definiciones de funciones que abstraen los detalles de la implementación y hace más fácil el desarrollo de aplicaciones.

Biblioteca de enlace dinámico (DLL - *Dinamic-link Library*) tiene varias funciones como la de almacenar funciones o clases, o para guardar cualquier tipo de recursos como texto imágenes y sonido.

Broadcast es un modo de transmisión de información o paquetes.

Desarrollo Web se le conoce a la tecnología de software que unifica una base de datos con el uso de un navegador de Internet con el objetivo de mostrar determinada información o de darle cumplimiento a determinadas tareas.

Entrelazado es un método de codificación que se realiza a las imágenes de un video, utilizado en algunas televisoras.

Firewalls (cortafuegos) es el software que es utilizado en las redes de computadoras para no permitir el acceso no autorizado a las mismas, en el que se cuentan los usuarios no autorizados o programas dañinos.

General Public License (GPL - Licencia Pública General de GNU) es una licencia creada para proteger de la apropiación de los software libres que estén cubierto por esta.

Handles es un puntero que es usado cuando una aplicación hace referencia a bloques de memoria u objetos gestionados por otros sistemas, como una base de datos o un sistema operativo.

Interoperabilidad es la capacidad que tiene un programa u ordenador de acceder a múltiples sistemas e intercambiar datos.

ISMS quiere decir Sistema de Información de Gestión de la Seguridad.

Kerberos es un protocolo de seguridad que usa una criptografía de clave simétrica para evitarse el envío de contraseñas a través de una red insegura. Al usar este protocolo se evita que usuarios no autorizados traten de robar las contraseñas en la red.

Multimedia se le conoce a la combinación de sonido, gráficos, animación y video que se usan para la transmisión de una información por medio de un ordenador.

OMG (*Object Management Group* - Grupo de Gestión de Objetos) es el consorcio encargado de definir los estándares para la aplicación de tecnologías orientadas a objetos tales como CORBA.

OSF (*Open Software Foundation*) es un consorcio encargado de desarrollar y comercializar software abiertos de cualquier fabricante.

Plug-in es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API.

Retrollamado (en inglés: ***callback***) En programación, es un código ejecutable que recibe como parámetro la dirección o puntero de otra función, cuando el *callback* es llamado este recurre al puntero de la función y la ejecuta.

Security Socket Layer (SSL) es un sistema de seguridad de comunicación.

Servidor multihilo es un servidor que permite atender alrededor de 15 consultas por segundos.

Software AG: es una compañía Alemana, encargada de producir software y *middlewares* tales como SOA.

Stub: Son librerías que emplean los programadores, las cuales se generan de forma automática.

Sun Microsystems es una empresa fabricante de semiconductores y software, recientemente adquirida por Oracle *Corporation*. Desarrolla tecnologías innovadoras para la red.

User Datagram Protocol (UDP - Protocolo de datagrama de usuario) es un protocolo del nivel de transporte no orientado a conexión. Permite el envío de mensajes para aplicaciones informáticas sin que exista una conexión previa.

Virtual Network Computing (VNC - Computación Virtual en Red) es lo que nos permite tomar control de una máquina servidor remotamente desde un ordenador cliente.

Zeroc es la empresa que produce el *middleware* ICE.