UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Título: Módulo de Transmisión de Audio y Vídeo por red para el Proyecto PROLAVI

Trabajo de Diploma para optar por el título de Ingeniero en

Ciencias Informáticas

AUTORES:

Yosuan Santiago Chamizo.

Nestor Manuel Casas Acosta.

TUTORES:

Ing. Sailyn Salas Hechavarria.

Ing. Duany Baró Menéndez.

Cuidad de La Habana, julio del 2010.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Autor: Nestor Manuel Casas Acosta Autor: Yosuan Santiago Chamizo

_____ _____

Tutor: Sailyn Salas Hechavarria Tutor: Duany Baró Menéndez

"Los desafíos pueden ser escalones o muros, es una cuestión de puntos de vista."

DATOS DE CONTACTO

Tutores

Nombre: Sailyn Salas Hechevarria.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Informática.

Categoría Docente: Profesor Instructor.

Correo electrónico: ssalas@uci.cu

Nombre: Duany Baró Menéndez.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Informática.

Categoría Docente: Adjunto a la producción.

Correo electrónico: dbaro@uci.cu

AGRADECIMIENTOS

A toda mi familia, y en especial a mi madre, que ha sido madre y padre a la vez, y a pesar de eso ha sabido educarme correctamente e inculcar en mí cosas que me han servido y me servirán para toda la vida.

A mi tía y a mi abuela que han sido mis segundas madres, a mi abuelo que por cosas de la vida no se encuentra, a mis tíos René, Nacho y José, a mi prima que ha sido como una hermana. Entre todos ellos, que son los que siempre han estado a mi lado, lograron sacar siempre lo mejor de mí y me guiaron por el camino correcto.

A mi tutora, que siempre que me hizo falta estuvo ahí.

Quiero dar dos agradecimientos a Duany, quien en corto plazo de tiempo nos ayudó a salir adelante en la tesis al igual que a Osniel.

A Talancon, Dayrel y Fabian, que también pusieron su grano de arena en mi tesis.

A todos mis amigos y a mi compañero de tesis Nestor con el cual pasé buenos y malos momentos durante el desarrollo de este trabajo, pero que siempre supimos salir adelante.

Yosuan.

A mis padres por darme la vida, por su dedicación y amor incondicional, por ser mis guías y ejemplo, por ser lo más preciado que poseo.

A mis hermanos por ayudarme, reconfortarme y enseñarme a levantarme de los tropiezos.

A Albe por ser no un primo, sino un hermano, por lo sueños que hemos realizado juntos y los que faltan por cumplir.

A tía Olga, por acogerme en su casa y por la constante preocupación por mis estudios y mi bienestar.

A mi abuela Elita y mi tío Pipo, por compartir conmigo lo que han tenido sin reparos.

A mi familia que siempre ha estado presente en cada momento importante de mi vida brindándome su apoyo.

A Sailyn por atendernos siempre, por la ayuda brindada y por ser más que una tutora, una amiga.

A Duany, quien en tan poco tiempo nos acogió como sus tesistas e hizo realidad nuestro deseo de graduarnos.

A Osniel que sin pensarlo dos veces nos apoyó y ayudó incondicionalmente, siempre le estaremos agradecidos.

A Yosuan, por todo el trabajo, las adversidades y los buenos tiempos que compartimos en el desarrollo de esta tesis.

A Lien, Annier, Alfredo, Talancón, Dayrel y Fabián por ayudarnos cuando lo necesitamos.

A todos los amigos de apartamento y de aula.

A todos los que tuvieron que ver de una forma u otra con nuestra tesis.

Nestor.

DEDICATORIA

A mi madre, a mis abuelos, a todos mis tíos y a mi prima, es decir, a todos los que han estado siempre a mi lado.

Yosuan.

A toda mi familia, a los que creyeron y depositaron su esperanza en mí, a ellos, va dedicada esta tesis.

Nestor.

RESUMEN

La educación apoyada en medios digitales plantea ineludibles cambios en los paradigmas educativos, mayormente reflejados en la búsqueda de nuevas tecnologías que proporcionen mejor soporte al proceso de enseñanza-aprendizaje en las distintas áreas y especialidades. Entre estas nuevas tecnologías encontramos el uso del chat, chat de voz y videoconferencias, su utilización con fines educativos constituye un campo abierto a la investigación. En el presente trabajo se aborda el diseño e implementación de un módulo de transmisión de texto, sonido y vídeo por red, basado en una arquitectura cliente–servidor. Para alcanzar esta meta se estudiaron las características de diferentes bibliotecas de red, que en conjunto con las necesidades del proyecto influyeron en la selección de las bibliotecas a utilizar. Además, se hizo un estudio de los principios básicos, conceptos, características y componentes de la programación en redes. El módulo resultante de este trabajo permitirá ser acoplado a las prácticas de laboratorios del Proyecto PROLAVI, constituyendo el soporte para la comunicación entre sus usuarios y la transmisión de datos en general.

PALABRAS CLAVE

Chat, chat de voz, videoconferencia, módulo, arquitectura cliente-servidor, bibliotecas de red, programación en redes, prácticas de laboratorio, PROLAVI.

ABSTRACT

Supported education in digital media, raises the inevitable changes in educational paradigms, mostly reflected in the search of new technologies that provide better support to the teaching-learning process in different areas and specialties. Among these new technologies we can find the use of chat, voice chat and video conferences, its use for educational purposes constitutes an open field for research.

The present work aims to design and implement a module for transmitting data, video and sound, based on client-server architecture. To achieve this goal we study the characteristics of different network libraries, which together with the needs of the project will influence on the selection of the libraries to use for the work. Furthermore, we do a study of the basic principles, concepts, characteristics and components of the network programming.

The module resulting from this work will be coupled to the laboratories practices of the PROLAVI Project, this constitute the support for communication between users and data transmission in general.

KEYWORDS:

Chat, voice, chat, videoconference, module, client- server architecture, network libraries, network programming, laboratories practices, PROLAVI.

TABLA DE CONTENIDO

INTRODUCCIÓN

En la última década el desarrollo de la Realidad Virtual ha sido vertiginoso, nuevas técnicas, nuevas aplicaciones para modelar entornos, nuevas formas de crear ambientes de un realismo increíble, y sobre todo, nuevos usos para estos entornos virtuales. La aplicación de esta disciplina de la informática avanza en distintas esferas, ejemplo: en videojuegos, en el ámbito militar, en el área del entrenamiento profesional (simuladores quirúrgicos, simuladores de vuelo), y en el panorama educativo. La dupla Educación-Tecnología tiene un proceso de retroalimentación en el que ambas partes se benefician y desarrollan. Sin embargo, en Cuba la Realidad Virtual ha sido poco explotada como técnica válida en el proceso educativo. La utilización de escenarios virtuales para el estudio de distintas ramas de la educación puede ser un gran apoyo para el aprendizaje del estudiante, así como una opción viable ante la falta de recursos que impidan hacer de forma tradicional los ejercicios que se están simulando. Estas ideas fueron la base para la creación del proyecto PROLAVI, el cual es parte del polo de Realidad Virtual de la facultad 5 en la UCI, su objetivo principal es crear prácticas de laboratorio virtuales para algunas asignaturas como Física, Biología y Química, y así suplir la necesidad de medios didácticos interactivos en los institutos educativos. En estas prácticas los estudiantes interactuarán con entornos virtuales del mayor realismo posible para realizar distintos ejercicios de laboratorio. En el proyecto PROLAVI no sólo se desea lograr la interacción del estudiante con el entorno virtual, también se hace necesario que este se comunique con otros estudiantes, o con el supervisor de la práctica, y que el trabajo se desarrolle de forma conjunta. Para lograr la comunicación necesaria que permita desarrollar estas prácticas conjuntas se requiere de una interfaz de comunicación por red que brinde las funcionalidades necesarias para ello. Esta interfaz de comunicación entre los usuarios de un entorno virtual, específicamente para los usuarios de las prácticas creadas por el proyecto PROLAVI, no ha sido diseñada ni implementada aún. Debido a esto surge el siguiente problema científico: ¿Cómo lograr la comunicación entre los usuarios de las prácticas de laboratorio virtuales desarrolladas en el proyecto PROLAVI?

Para esto se plantea como objeto de estudio las técnicas de transmisión y recepción de datos en arquitecturas de red y como campo de acción, las técnicas de transmisión y recepción de audio y vídeo por la red.

Este trabajo se propone como objetivo: Desarrollar las funcionalidades de audio y vídeo en el módulo de transmisión de datos del proyecto PROLAVI. En correspondencia con el problema científico planteado se trazaron tareas que contribuyen a darle cumplimiento al objetivo general. A continuación se relacionan las tareas:

- ✓ Estudiar los principios básicos, conceptos, características y componentes de la programación en redes.
- ✓ Analizar las tendencias actuales de las bibliotecas de red existentes para la transmisión de audio y vídeo.
- ✓ Estudiar el módulo desarrollado anteriormente.
- ✓ Desarrollar las funcionalidades relacionadas con el audio y el vídeo.

La idea fundamental a defender de este trabajo es: Mediante la utilización del módulo de transmisión de datos por red se logrará la comunicación entre los usuarios de las prácticas de laboratorio virtuales del proyecto PROLAVI. Durante el desarrollo de la investigación, se utilizó un conjunto de métodos, técnicas y procedimientos para la recopilación, el análisis, el procesamiento y la valoración de la información. Métodos de la Investigación: Los métodos teóricos posibilitaron analizar los resultados obtenidos luego de aplicar los métodos empíricos durante la realización de las tareas investigativas; permitieron establecer conclusiones fiables que posibilitaron dar solución al problema planteado.

Analítico – sintético: Este método se utilizó para realizar una profunda investigación sobre el objeto de estudio y el campo de acción. Se analizaron documentos, teorías y características sobre redes para así extraer de todo lo estudiado los elementos más importantes y que más se relacionan con nuestro trabajo.

Histórico-lógico: Este método se utilizó para profundizar en los antecedentes del desarrollo de software educativos a nivel nacional e internacional y la evolución de la programación en redes.

Por todo lo anterior esperamos obtener como resultado: la actualización del módulo de transmisión de datos por red para incorporar las funcionalidades de audio y vídeo.

El presente Trabajo de Diploma consta de cuatro capítulos:

Capítulo 1. Fundamentación Teórica: En este capítulo se exponen los conceptos fundamentales que sirven de referentes para la programación en redes, se enumeran las características fundamentales de diferentes bibliotecas de red, y se describe un sistema de transmisión de vídeo en tiempo real.

Capítulo 2. Soluciones Técnicas: En este capítulo se especifican los métodos que se utilizarán y las bibliotecas de red que más se adecuan a los requerimientos de nuestro trabajo, que deberán ser tratadas para ofrecer una solución al problema científico de la investigación y cumplimentar el objetivo general.

Capítulo 3. Características del sistema: En este capítulo se profundiza en las características del sistema, se crea el modelo del dominio, se hace el levantamiento de requisitos, se crea el modelo de casos de uso y se hace la descripción de los casos de uso de los subsistemas del módulo de transmisión de datos.

Capítulo 4. Diseño e implementación: En este capítulo se muestra cómo se diseñaron cada uno de los dos subsistemas que constituyen el módulo. Se muestran los diagramas de diseño, de secuencia, y de componentes correspondientes a cada uno de los subsistemas del módulo de transmisión de datos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción.

En el presente capítulo, para una mejor comprensión del problema, se introducen una serie de conceptos básicos para el trabajo con redes, tales como protocolos de red, técnicas de transmisión, y arquitecturas de comunicación. Se hace una reseña de las bibliotecas de red más utilizadas a nivel internacional y nacional, haciendo énfasis en sus características, lo que nos permitirá seleccionar la que se ajuste más a nuestras necesidades de trabajo. Además, se hace un análisis de un sistema de transmisión de vídeo en tiempo real, incluyendo aspectos como la captura de vídeo, codificación de vídeo, transmisión en tiempo real, decodificación de vídeo, y reproducción.

1.2 Laboratorios Virtuales: ventajas y desventajas.

Un laboratorio virtual es un sistema computacional que pretende crear un ambiente aproximado al de un laboratorio tradicional. Los experimentos se realizan paso a paso, siguiendo un procedimiento similar a los que se realizan en las prácticas de laboratorio diseñadas para laboratorios tradicionales: se visualizan instrumentos y fenómenos mediante objetos dinámicos, imágenes o animaciones. Se obtienen resultados numéricos y gráficos, tratándose éstos matemáticamente para la obtención de los objetivos perseguidos en la planificación docente de las asignaturas.

Algunas **ventajas** que presentan los laboratorios virtuales:

- ✓ Son más económicos, siendo una alternativa barata y eficiente donde los estudiantes pueden simular y observar fenómenos como si manipularan objetos reales.
- ✓ Permite realizar un mayor número de experimentos a cada estudiante, ya que no tiene que estar en el laboratorio para efectuarlos, puede hacerlo desde una computadora usando un navegador, haciendo más flexible el horario de prácticas y eliminando los riesgos inherentes a los experimentos.
- ✓ Es una herramienta de auto aprendizaje, donde el alumno altera las variables de entrada, configura nuevos experimentos, aprende el manejo de instrumentos y personaliza el experimento.

- ✓ Los estudiantes pueden repetir las prácticas un número ilimitado de veces porque con ellas no se dañan las herramientas ni se consumen materiales experimentales.

Desventajas que presentan los laboratorios virtuales:

- ✓ El laboratorio virtual no puede sustituir la experiencia práctica, aunque realice una simulación altamente parecida a la realidad se debe considerar una herramienta complementaria para aumentar el rendimiento de los estudiantes.

- ✓ Al interactuar con los laboratorios virtuales se corre el riesgo de que el estudiante se comporte como un mero espectador por lo que es necesario que se realicen las actividades de forma organizada y progresiva para alcanzar los objetivos planteados de la práctica realizada.

- ✓ La experiencia con los laboratorios virtuales es poca, debido a que en la mayoría de los centros se trabaja con instrumentos tradicionales, muchas veces no se encuentra la simulación requerida para determinados experimentos, y aunque los elementos simulados se acerque bastante a la realidad no son objetos reales por lo que hay una pérdida de visión de la realidad.

1.3 Estudio de otros módulos de transmisión de datos aplicados en laboratorios virtuales.

Existe un grupo de laboratorios virtuales llamado el MSDN Labs, dentro de este grupo los dedicados al entorno Visual Studio presentan un módulo de transmisión de datos. Estos laboratorios virtuales consisten en demostraciones *on-line* totalmente interactivas, que permiten ejecutar los ejercicios planteados sin necesidad de poseer las herramientas necesarias, pues se accede a estas en un entorno virtual. La mayoría suelen tener una duración de 30 a 90 minutos y sólo están disponibles en inglés. Además, incluye un pequeño chat para comunicarse con otros usuarios que estén realizando ese mismo laboratorio, aunque presenta inconvenientes ya que solamente envía texto y puede ser ejecutado con Internet Explorer 6.0 u otra versión más reciente. Tanto Visual Studio como Internet Explorer son propietarios. [1]

Existe un laboratorio virtual diseñado por el Msc. Ing. Raúl Fernando Gómez S., docente titular de la materia de Robótica de la Facultad de Informática-Electrónica y Bio-Ingeniería de UNIVALLE, para mejorar la accesibilidad del estudiantado al Laboratorio de Robótica del departamento de Electrónica y Bio-Ingeniería. Este laboratorio emplea tecnologías recientes basadas en el manejo de redes de

computadoras, para controlar la célula de trabajo del robot educacional SCORBOT-ERvPLUS, fabricado por la empresa Eshed Robotec, y como complemento, este sistema es capaz de proveer tanto al operador como a los demás usuarios en red con servicios de chat, clases virtuales, transferencia de archivos, con el propósito optimizar el flujo de información entre los participantes de una sesión remota compartida. La desventaja principal que presenta este módulo de transmisión de datos es que la compañía Eshed Robotec es privada, por lo que el laboratorio virtual también es propietario. [2]

1.4 La voz sobre IP o VoIP.

La Voz sobre IP (VoIP) abre las puertas a la convergencia de las redes de voz y datos en una única red. VoIP supone una reducción de costes en la instalación de cableado, ofreciendo además la flexibilidad de soportar nuevos servicios como la videoconferencia a través de Internet o la conexión con PCs.

VoIP consiste en transmitir voz sobre protocolo IP, aunque esto no es de manera sencilla ya que las redes IP fueron diseñadas principalmente para datos, y muchas de las ventajas de las redes IP para los datos resultan ser una desventaja para la voz pues ésta es muy sensible a retardos y problemas de transmisión por muy pequeños que estos sean. [3]

Como se ha podido ver hasta ahora un elemento importante a la hora de la transmisión de voz es el correspondiente a las redes, en este caso las redes informáticas, las cuales podemos definir como un sistema de comunicación que conecta ordenadores y otros equipos informáticos entre sí, con la finalidad de compartir información y recursos.

A través de la compartición de información y recursos en una red, los usuarios de los sistemas informáticos de una organización podrán hacer un mejor uso de los mismos, mejorando de este modo el rendimiento global de la organización. Entre las ventajas que supone el tener instalada una red, pueden citarse las siguientes:

- ✓ Mayor facilidad en la comunicación entre usuarios.
- ✓ Reducción en el presupuesto para *software*.
- ✓ Reducción en el presupuesto para *hardware*.
- ✓ Posibilidad de organizar grupos de trabajo.
- ✓ Mejoras en la administración de los equipos y programas.

- ✓ Mejoras en la integridad de los datos.
- ✓ Mayor seguridad para acceder a la información.

1.5 Protocolos de red

Primeramente decir, que los protocolos de redes, de forma general, se definen como un conjunto de reglas que dos aplicaciones pueden seguir para establecer la comunicación entre ellas. Podemos encontrar protocolos de bajo nivel, estos guían los mensajes desde el origen hasta el destino sin mostrar la dirección en que se realiza la transmisión, estos protocolos se encuentran reunidos en el Protocolo de Internet (Internet Protocol (IP)). El protocolo IP generalmente no se usa en las aplicaciones de red de forma directa, en ellas se utilizan protocolos que estén por encima de IP, los de uso más común son el Protocolo de Datagramas de Usuario (UDP/IP) y el Protocolo de Control de Transferencias (TCP/IP).

UDP/IP: Con este protocolo no se garantiza la llegada de todos los paquetes en que se divide la información, ni se asegura que lleguen en el orden correcto, sin embargo, la transmisión se realiza a una mayor velocidad, y la información se procesa con mayor facilidad pues no le incorpora a los paquetes de datos información adicional.

TCP/IP: Este protocolo crea una conexión punto a punto, la información que va a ser enviada se divide y estructura en paquetes de red a los que se incorporan datos adicionales para el control de errores de transmisión y el orden de llegada. En el momento de extraer la información el receptor ordena los paquetes y realiza un proceso de control de errores donde descarta los datos duplicados, y demanda el reenvío de los datos dañados o perdidos. Es mucho más confiable pero requiere mayor ancho de banda, es más lento y utiliza paquetes más grandes.

- ✓ Las funciones básicas que ha de realizar cualquier protocolo son las siguientes:
- ✓ Establecimiento del enlace (punto de destino y origen).
- ✓ Transmisión de la información y control de flujos.
- ✓ Detección de fallos en la transmisión.
- ✓ Corrección de errores.

1.6 VOIP: protocolos de señalización.

Existen muchos protocolos involucrados en la transmisión de voz sobre IP. Ya de por sí hay protocolos de red involucrados como el propio protocolo IP y otros protocolos de transporte como TCP o UDP. Además, encima de ellos se colocan los protocolos de señalización de voz, que cumplen importantes funcionalidades, como tareas de establecimiento de sesión, control del progreso de la llamada, entre otras. Se encuentran en la capa 5 del modelo OSI, es decir, en la capa de Sesión. [4]

Existen algunos protocolos de señalización, que han sido desarrollados por diferentes fabricantes u organismos como la ITU o el IETF, y que se encuentran soportados por Asterisk. Algunos son:

- ✓ SIP.
- ✓ IAX.
- ✓ H.323.
- ✓ MGCP.
- ✓ SCCP.

Entre estos los más populares en el ámbito de Asterisk son SIP e IAX. En cuanto al Asterisk, se abordará más con el avance de la investigación.

SIP es un protocolo desarrollado por el grupo de trabajo MMUSIC(Multimedia Session Control) del IETF con la intención de ser el estándar para la iniciación, modificación y finalización de sesiones interactivas de usuario donde intervienen elementos multimedia como el vídeo, voz, mensajería instantánea, juegos en línea y realidad virtual. La sintaxis de sus operaciones se asemeja a las de HTTP y SMTP, los protocolos utilizados en los servicios de páginas Web y de distribución de e-mails respectivamente. SIP, como vimos anteriormente, es uno de los protocolos de señalización para voz sobre IP, al igual que H.323 y IAX actualmente IAX2.

En la siguiente imagen se puede ver dónde se encuentra SIP dentro de la arquitectura multimedia de protocolos propuestos por la IETF:

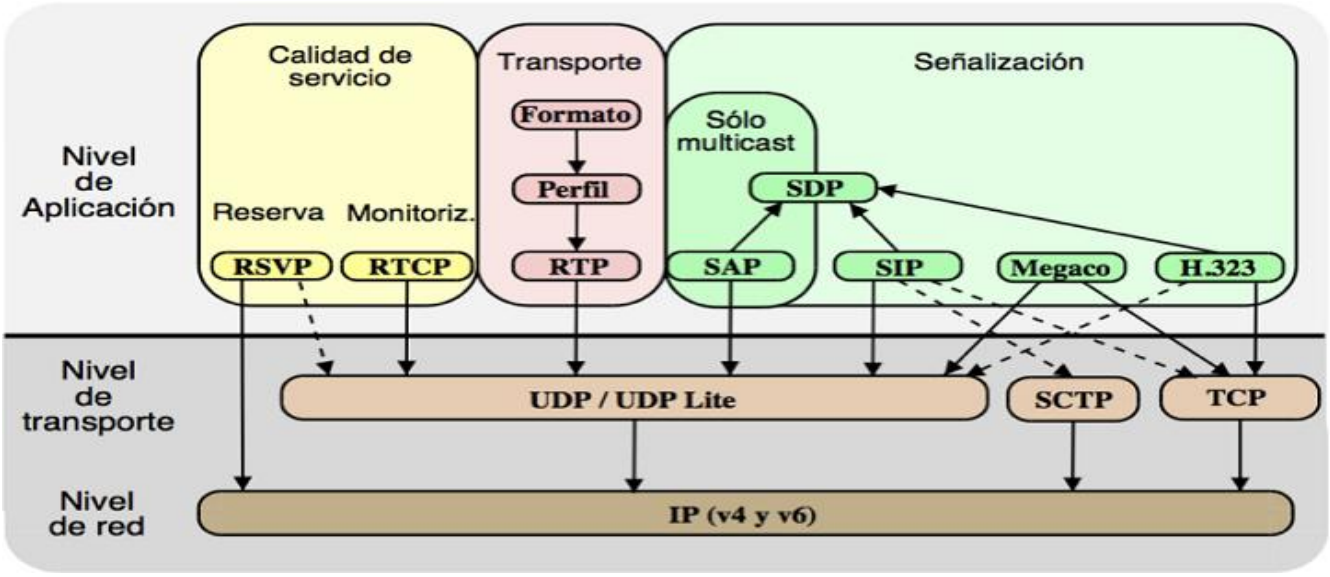


Figura 1 Arquitectura de protocolos propuestos por la IETF.

Desde sus comienzos, SIP ha estado muy relacionado con la transmisión de información multimedia, y se ha enfocado hacia la señalización. SIP será el encargado de gestionar sesiones entre dos extremos SIP y de permitir las negociaciones de flujos multimedia, transportando un protocolo de descripción de sesiones, para lo que se utiliza SDP. La relación que existe entre SIP y SDP es arbitraria. Sin embargo, forman una pareja perfecta para soportar aplicaciones tales como teleconferencias, videoconferencias simples y mejoradas (con posibilidad de incorporar buzones de audio y vídeo) y todo tipo de transmisiones de información en tiempo real que requieran uno ó más flujos de transporte; la forma de asociar dichos flujos usados será mediante una sesión SIP.

SIP fue diseñado por el IETF con el concepto de "caja de herramientas", es decir, el protocolo SIP se vale de las funciones aportadas por otros protocolos, que da por hechas y no vuelve a desarrollar. Debido a este concepto, SIP funciona en colaboración con otros muchos protocolos. Este se concentra en el establecimiento, modificación y terminación de las sesiones, y se complementa entre otros con el SDP (Protocolo de Descripción de la Sesión), que describe el contenido multimedia de la sesión, por ejemplo qué direcciones IP, puertos y códecs se usarán durante la comunicación. También se complementa con el RTP (*Real-time Transport Protocol*) que es el verdadero portador para el contenido de voz y vídeo que intercambian los participantes en una sesión establecida por SIP. [5]

Por último, veremos en la siguiente figura la relación de SIP con protocolos como SDP, RTP y RTCP, así como la forma en que podemos dividir a este protocolo, en un núcleo que se encarga de las transacciones y un controlador de sesiones:

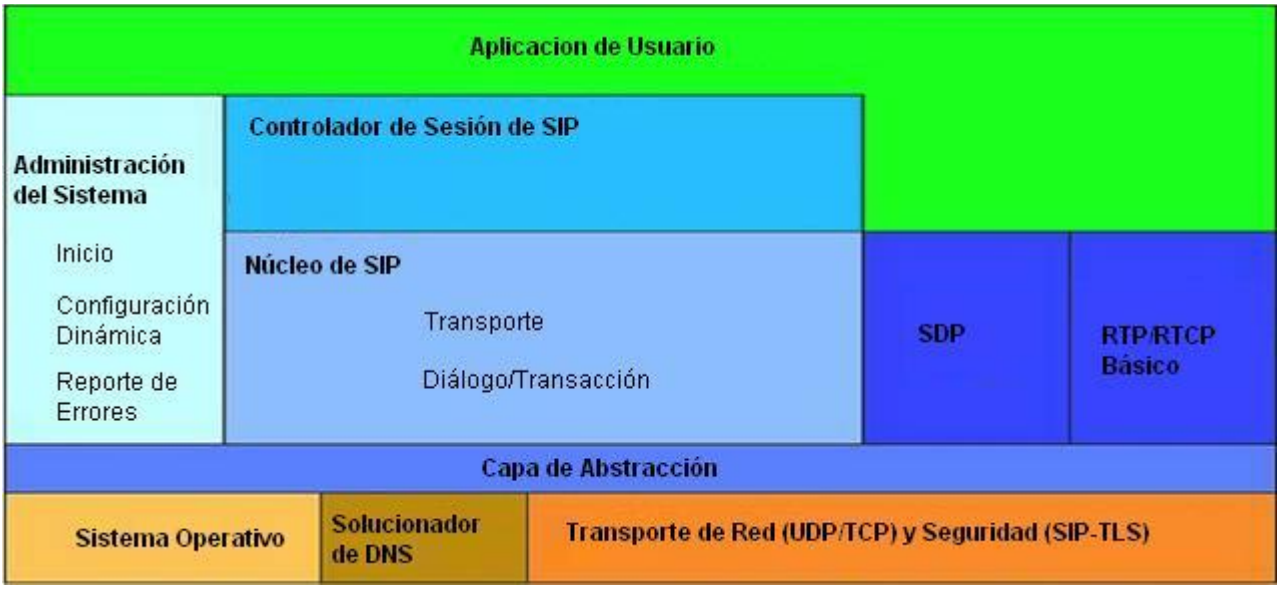


Figura 2 Relación de SIP con protocolos como SDP, RTP y RTCP.

En la siguiente figura podemos observar un hecho curioso y es que pese a que SIP soporta tanto UDP como TCP sólo lo vemos posado sobre UDP. Vale destacar que no se trata de un error sino más bien que en Asterisk la implementación de SIP solamente está disponible para UDP.

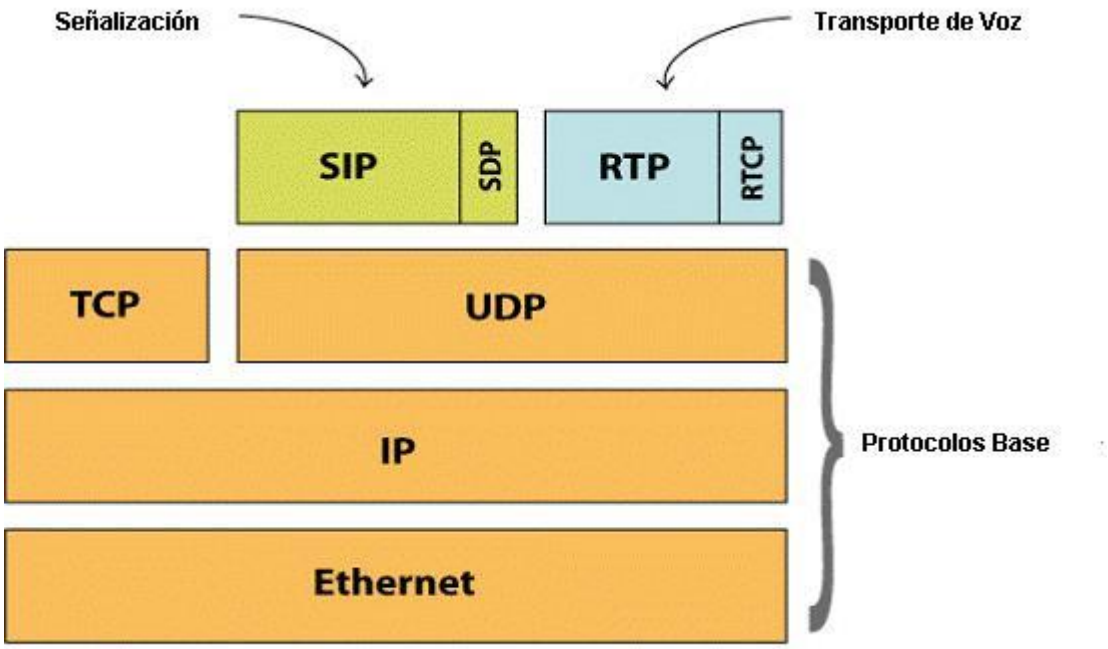


Figura 3 Protocolos involucrados en una llamada SIP. El caso de IAX es muy similar.

Como se ha explicado anteriormente el protocolo SIP depende en buena medida de una serie de protocolos, entre estos es importante abordar el RTP, que es un protocolo a nivel de sesión, utilizado para la transmisión de datos en tiempo real, de uso muy difundido en las aplicaciones de audio y vídeo online, especialmente en las videoconferencias. En un principio se publicó como un protocolo multicast pero después ha sido usado en aplicaciones unicast, su uso más frecuente es en sistemas de control de flujo y streaming, videoconferencias, sistemas para transmitir audio en tiempo real y otras aplicaciones online.

RTP permite la administración de flujos multimedia (voz, vídeo) sobre IP y funciona sobre el protocolo UDP. [6]

Hay que dejar claro que RTP no proporciona QoS (*Quality Of Sevice*) al tráfico transmitido en tiempo real, sino que la función de RTP es dotar a las aplicaciones de una cierta inteligencia para que puedan optimizar la transmisión de contenidos en tiempo real, adaptándola a las características y al estado de la red.

De manera más general, RTP permite:

- ✓ Identificar el tipo de información transmitida.
- ✓ Agregarle marcadores temporales y números de secuencia a la información transmitida.
- ✓ Controlar la llegada de los paquetes a destino.

Normalmente nos referimos a RTP como si se tratase de un único protocolo. No obstante, RTP está formado por dos componentes estrechamente ligados:

- ✓ RTP Protocolo de Transferencia de Datos (RDTP): Este protocolo es el que se encarga del transporte de los datos en tiempo real.
- ✓ RTP Protocolo de Control (RTCP): Que cumple la importante función de encargarse de generar información de control de todos los participantes de una sesión RTP.

RTCP se encuentra relacionado de manera estrecha con el RTP, se definió en la RFC 3550 y provee un control fuera de banda de la información en flujos RTP. Acompaña al protocolo de transporte en la entrada y empaquetamiento de datos multimedia, aunque en sí mismo él no transporta ningún dato. Se usa para emitir de manera periódica paquetes de control que participan en el flujo de sesiones multimedia. Su principal función es proveer una retroalimentación de la calidad del servicio que está brindando el RTP.

Como tal el RTP (Protocolo en tiempo real) y RTCP (Protocolo de control en Real-Time) permiten, respectivamente, transportar y controlar bloques de datos que cuentan con propiedades de tiempo real.[7]

Después de haber abordado, de forma exhaustiva, acerca del protocolo de señalización SIP es momento de hablar de otro de los protocolos de señalización:

El protocolo IAX (*Inter-Asterisk eXchange*) es un protocolo de señalización creado por Mark Spencer, el mismo creador de Asterisk, con el objetivo de solucionar algunos problemas existentes con otros protocolos. Como indica su nombre **fue diseñado como un protocolo de conexiones VoIP entre servidores Asterisk,** aunque hoy en día también sirve para conexiones entre clientes y servidores que soporten el protocolo. El protocolo todavía no es un estándar pero pretende serlo a través de un proceso de estandarización en la IETF.

En esencia IAX presenta tres ventajas muy interesantes sobre otras alternativas como SIP:

- ✓ Consume menos ancho de banda.
- ✓ Soluciona mejor problemas de NAT.
- ✓ Pasa más fácilmente a través de *firewalls*.

La versión actual del protocolo es la versión 2. La versión anterior ha quedado obsoleta por lo que es común ver el nombre IAX2 como sinónimo de IAX. IAX2 es un protocolo diseñado y pensado para su uso en conexiones de VoIP aunque puede soportar otro tipo de conexiones (por ejemplo vídeo).

IAX, a diferencia de SIP (que es un protocolo basado en texto), es un protocolo binario. Esto es una ventaja desde el punto de vista del ancho de banda puesto que en binario se desperdiciarán menos bytes. IAX usa UDP y normalmente utiliza el puerto 4569. Lo interesante de IAX es que por un solo puerto transmite tanto la voz como la señalización y es esto lo que le permite resolver problemas de NAT y pasar a través de firewalls sin mayor inconveniente.

Además de esta característica el protocolo permite la diversificación de varios canales de audio en el mismo flujo de datos. Es decir que en un mismo datagrama se pueden enviar varias sesiones al mismo tiempo, lo que significa una reutilización de datagramas y por consiguiente un ahorro de ancho de banda.

Dentro de los protocolos de señalización uno de los que más ha evolucionado en los últimos tiempos es el H.323, que es una norma creada por la ITU-T (*International Telecommunication Union*) en 1996. Nació para dar soporte a comunicaciones de audio, vídeo y datos a través de la red IP (sin garantizar servicios QoS). También forma parte del estándar de *Audiovisual and Multimedia Systems, the series* H.32X, donde se define la transmisión multimedia a través de diferentes tipos de medios como:

- ✓ H.320: Comunicaciones multimedia sobre RDSI.
- ✓ H.310 y H.321: Comunicaciones multimedia sobre ATM.
- ✓ H.324: Comunicaciones multimedia sobre RTC.

- ✓ H.323: Comunicaciones multimedia sobre IP.

H.323 tiene referencias hacia algunos otros protocolos de ITU-T y de códecs como:

Protocolos:

- ✓ H.225.0 - Protocolo utilizado para describir la señalización de llamada, el medio (audio y vídeo), el empaquetamiento de las tramas, la sincronización de tramas de medio y los formatos de los mensajes de control.
- ✓ H.245 - Protocolo de control para comunicaciones multimedia. Describe los mensajes y procedimientos utilizados para abrir y cerrar canales lógicos para audio, vídeo y datos, capacidad de intercambio, control e indicaciones.
- ✓ H.239 - Describe el uso de la doble trama en videoconferencia, normalmente una para vídeo en tiempo real y la otra para presentación.

Códecs:

- ✓ G.711, G.722, G.723, G.728 y G.729 para audio (voz).

- ✓ H.261, H.263, H.264 para vídeo.

H.323 especifica los protocolos que gestionan la preparación, establecimiento, control de estado, mensajería, códecs de audio/vídeo, transferencia de datos, y fin de llamada. Estos protocolos funcionan sobre un nivel de transporte basado en TCP y UDP y/o (tras la 5ª revisión de H.323) SCTP.

El siguiente gráfico muestra la pila de protocolos H.323:

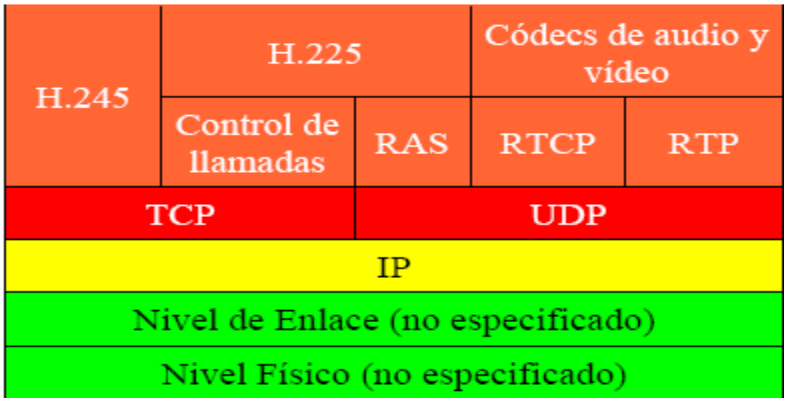


Figura 4 Pila de protocolos H.323.

A modo de conclusión de esta sección les traemos una tabla comparativa entre los protocolos de señalización que desde nuestro punto de vista pudieran ser utilizados:

	H.323	**SIP**
Complejidad Servicios ofrecidos	H.323 es un estándar muy complejo que describe una arquitectura de comunicaciones y servicios completa. Cada revisión del estándar (actualmente la 6ª) añade nuevas funcionalidades que deben implementarse obligatoriamente. Aun así, H.323 mantiene compatibilidad hacia atrás con todas las revisiones anteriores. Los terminales H.323 pueden ofrecer diversos servicios, pero todos son compatibles y pueden ofrecer como	SIP, en cambio, es un protocolo que al ser más abierto y flexible permite una mayor interoperabilidad con otros códecs y protocolos. Por desgracia, la flexibilidad de SIP puede derivar en la incompatibilidad de dispositivos.

	mínimo un servicio básico de llamadas de voz.	
CRÍTICA: **SIP fue creado inicialmente como una alternativa más simple a H.323, pero con el tiempo la especificación de SIP ha incrementado su complejidad hasta el punto de ser parecida a la de H.323: El RFC 3261 (SIP) son 269 páginas, frente a las 317 de H.323rev.6.** **Tanto H.323 como SIP pueden ser opciones demasiado complejas para un simple servicio de telefonía o videotelefonía!**		
Direccionamiento	H.323 soporta múltiples tipos de direcciones tales como: dirección IP:puerto alias H.323 número E164 (teléfono tradicional) URL (etc.)	SIP, por su parte, únicamente aceptad direcciones del tipo URI.
CRÍTICA: **Salvo por el acertado soporte de números E164 por parte de H.323, ambas aproximaciones son inapropiadas. Por un lado, H.323 pretende soportar demasiados tipos de dirección, acumulando una complejidad que podría no verse compensada con utilidad. SIP, por su parte, se basa en un formato de URI (*Unique Resource Identifier*) similar a las direcciones de correo**		

electrónico, que son cómodas de recordar pero inadecuadas para servicios de telefonía.		
ormato de los mensajes	Los mensajes de los protocolos que recoge H.323, como H.225 y H.245, utilizan una codificación binaria, similar a la de los datagramas IP o las tramas Ethernet.	SIP codifica los mensajes en texto plano legible por humanos, como hace HTTP o XML.
CRÍTICA: **En la práctica es mucho más adecuada la codificación binaria, pues el procesado de texto tiene un coste computacional considerable que reduce el rendimiento total. Además, por lo general los mensajes de texto ocupan más espacio y por tanto consumen un mayor ancho de banda.**		
Interconexión con otras redes	H.323 es compatible con las redes H.32x. La especificación además incluye compatibilidad con la red telefónica de circuito conmutado tradicional (PSTN); la traducción de direcciones de red H.323 a números de teléfono E164 es una función obligatoria en los gatekeepers.	SIP no define cómo interoperar con otras redes como la red telefónica tradicional; la funcionalidad queda delegada a los dispositivos implementadores.
CRÍTICA: **La interconexión con la PSTN es de vital importancia para el éxito de la telefonía IP, pues en la vida real no se producen saltos tecnológicos drásticos que dejen atrás los anteriores sistemas. H.323 cuenta con un punto a su favor en este aspecto al estar definida la**		

compatibilidad en el propio estándar.		
Seguridad	Los puntos finales H.323 negocian durante la conexión los puertos que se van a emplear para: - H.245 “Parámetros de llamada” - RTP Audio - RTP Vídeo - RTCP (entre cualquiera de los puertos libres entre el 1024 y el 65535).	SIP especifica los puertos de RTP/RTCP durante el establecimiento de llamada.

CRÍTICA:

Como los puertos RTP/RTCP se escogen dinámicamente, ni H.323 ni SIP funcionan detrás de firewalls, pues éstos no saben qué puertos deben abrirse si no están configurados a priori. Así pues, la única forma de hacer que una conexión funcione es abrir todos los puertos, con el grave riesgo de seguridad que esto conlleva. No obstante, hoy en día existen firewalls “inteligentes” que reconocen los protocolos H.323 y/o SIP y pueden averiguar qué puertos se deben abrir dinámicamente inspeccionando los paquetes en los respectivos canales de control ....aunque esto supone un intento de “parchear” un diseño mal planteado y no es una verdadera solución.

Por contra, protocolos como Inter-Asterisk eXChange 2 (IAX2) solucionan el problema de los firewalls transmitiendo conjuntamente la señalización y los datos mediante UDP. IAX2 utiliza

un único puerto, el 4569.

Tabla 1 Comparación entre los Protocolos de señalización SIP y H.323.

1.7 PBX Asterisk.

Asterisk es un software PBX que usa el concepto de software libre (GPL).Una de las cosas más interesantes de Asterisk es que soporta los servicios de VoIP, así como numerosos protocolos de VoIP como SIP y H.323. El paquete básico de Asterisk incluye muchas características que antes sólo estaban disponibles en caros sistemas propietarios como **creación de extensiones, envío de mensajes de voz a e-mail, llamadas en conferencia, menús de voz interactivos y distribución automática de llamadas**. Este PBX permite conectividad en tiempo real entre las redes PSTN y redes VOIP. [8]

Asterisk usa una CPU de servidor para procesar los canales de voz, en vez de tener un DSP (procesador de señales digitales) dedicado a cada canal.

Entre sus funcionalidades generales se encuentran:

- ✓ Conferencias Múltiples (*MeetMe*) Ilimitadas.
- ✓ Música en espera configurable en diversos formatos.
- ✓ Correo de Voz integrado al Correo Electrónico.
- ✓ Operadoras Automáticas Ilimitadas.

Beneficios que brinda como servidor VoIP:

- ✓ Disminución de costos en llamadas telefónicas entre sucursales.
- ✓ Encriptación en el momento de conexión y durante toda la comunicación.
- ✓ Conexiones remotas a través de autenticación de usuarios.

1.8 Codificación de la voz.

Ya tenemos claro que para transportar la voz se utilizan algunos protocolos como SIP, IAX y otros como RTP o RTCP. Pero la voz es una onda analógica que necesita transformarse a digital en algún formato antes de ser transmitida.

La búsqueda de un formato óptimo generó algunas alternativas de formatos de transmisión llamadas códec. La palabra códec proviene de abreviar las palabras Codificación y Decodficación. Su función principal es la de adaptar la información digital de la voz para obtener algún beneficio. Este beneficio en muchos casos es la compresión de la voz de tal manera quepodamos utilizar menos ancho de banda del necesario. [9]

Algunos códecs, soportados por Asterisk y comúnmente usados en comunicaciones de VoIP, son G.711, G.729, GSM, iLBC, entre otros. A continuación explicaremos algunos de ellos:

- ✓ G.711

Es uno de los códecs más usados de todos los tiempos y proviene de un estándar ITU-T que fue liberado en 1972. Viene en dos sabores llamados u-law y a-law. La primera versión se utiliza en los Estados Unidos y la segunda se utiliza en Europa. Una de sus características es la calidad de voz debido a que casi no la comprime. Utiliza 64kbit/s, es decir un muestreo de 8 bits a 8kHz. Es el códec recomendado para redes LAN, pero hay que pensarlo dos veces antes de utilizarlo en enlaces remotos debido al alto consumo de ancho de banda.

- ✓ G.729

También se trata de una recomendación ITU cuyas implementaciones han sido históricamente licenciadas, o sea, que hay que pagar por ellas. La ventaja en la utilización de G.729 radica principalmente en su alta compresión y por ende bajo consumo de ancho de banda lo que lo hace atractivo para comunicaciones por Internet. Pese a su alta compresión no deteriora la calidad de voz significativamente y por esta razón ha sido ampliamente usado a través de los años por muchos fabricantes de productos de VoIP.

G.729 utiliza 8kbit/s por cada canal. Si comparamos este valor con el de G.711 notaremos que consume 8 veces menos ancho de banda, lo cual a simple vista es un ahorro de recursos significativo.

- ✓ GSM

Muchas personas suelen preguntar si el códec GSM tiene algo que ver con el estándar de comunicaciones celulares y la respuesta es que sí .El estándar que define la tecnología celular GSM (*Global System for Mobile communications*) incluye este códec.

La ventaja de este códec también es su compresión. GSM comprime aproximadamente a 13kbit/s.

1.9 Técnicas de transmisión.

Las técnicas de transmisión para mensajes se pueden dividir en distintos tipos, cada uno de ellos con características distintivas que los diferencian e identifican:

- ✓ **Unicasting (Redes punto a punto):** En esta técnica se establece una conexión entre dos puntos a los cuales se le identifica como nodo receptor y nodo emisor, mediante la misma se puede realizar un control y dar dirección a los paquetes enviados. No es eficiente, ya que en determinadas ocasiones hace un uso excesivo de ancho de banda para enviar un mensaje, por ejemplo ,si ese mensaje es requerido por más de un receptor el emisor lo envía de forma repetitiva malgastando recursos.

- ✓ **Multicasting (multidifusión):** En esta técnica se agrupan los receptores en grupos específicos, a los cuales les interesen mensajes comunes, para hacer el uso más eficiente que se pueda de la red. La conexión se establece entre un emisor y varios receptores agrupados como se dijo anteriormente, de manera que cuando se envía un mensaje determinado este llega a los receptores del grupo interesado y no hay necesidad de repetir o duplicar el mensaje. Se considera una buena técnica para enviar información a grandes números de usuarios o nodos.

- ✓ **Broadcasting (Redes multipunto o redes de difusión):** En esta técnica la comunicación es establecida entre un emisor y todos los nodos restantes de la red, esto trae como consecuencia que cada nodo tenga que realizar el procesamiento de cada mensaje emitido, debido a esta situación en

grandes redes en las cuales se encuentren conectados un gran número de usuarios no se garantiza el broadcasting.

1.10 Arquitecturas de comunicación.

Las arquitecturas de red se pueden organizar de acuerdo con su grado de despliegue y pueden ser utilizadas de acuerdo a las exigencias del cliente o de la tecnología disponible así como la eficiencia que tendrá de acuerdo con las características específicas de la red a la cual va a ser aplicada.

- ✓ **Arquitectura punto a punto (peer to peer):** En las redes con esta arquitectura todos los nodos están conectados a los restantes nodos que integran la red, no existe intermediario entre ellos y cada uno puede emitir mensajes a los restantes, esto afecta de manera positiva a la red con respecto al problema de la latencia. Este tipo de red no tiene jerarquía entre los nodos por lo cual no resulta escalable. Muy útil cuando se aplica a un número no muy grande de computadoras conectadas y en redes locales LAN.

- ✓ **Arquitectura cliente servidor:** En esta arquitectura se escoge un nodo específico el cual hará la función de servidor, el resto de los nodos en la red son receptores, el nodo servidor controla y administra la comunicación en la red. Normalmente, el servidor es una máquina bastante potente que actúa de depósito de datos y funciona como un sistema gestor de base de datos (SGBD). Por otro lado, los clientes suelen ser estaciones de trabajo que solicitan varios servicios al servidor. La comunicación directa entre los nodos receptores no existe, al servidor controlar todos los envíos realizados hay un retardo en el flujo de paquetes, sin embargo, hay un mejor uso de recursos porque el mismo mensaje no tiene que ser enviado a todos los nodos. Ambas partes deben estar conectadas entre sí mediante una red. Este tipo de arquitectura es la más utilizada en la actualidad, debido a que es la más avanzada y la que mejor ha evolucionado en estos últimos años.

- ✓ **Arquitectura red de servidores:** Está constituida por un grupo de nodos servidores conectados punto a punto, a cada uno de estos servidores hay un grupo de nodos clientes conectados, es decir, un conjunto de subredes que están conectadas a través de los nodos servidores. Esta arquitectura brinda una gran escalabilidad y disminuye los problemas de capacidad que brindaba un solo servidor, sin

embargo, cuando llega a gran escala surgen problemas para el control y manipulación de datos por la red.

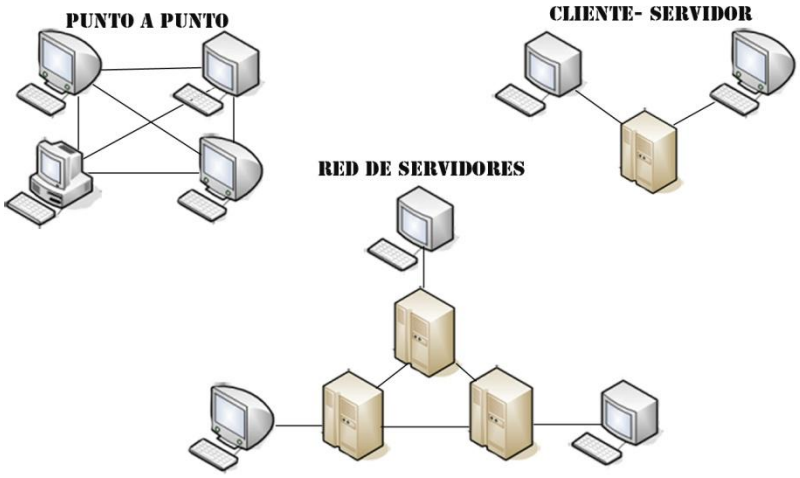


Figura 5 Arquitecturas de comunicación.

1.11 Bibliotecas de red.

En la actualidad el desarrollo en las redes es enorme, el auge de Internet impone un desarrollo acelerado para el control y envío de la gran cantidad de información que viaja por la red. Los juegos *online* y multijugador exigen constantemente nuevas técnicas para manipular los datos y establecer comunicación con el servidor evitando los problemas de latencia. Surgen nuevas opciones de negocio y comunicación, se generaliza el uso del vídeo-conferencia y la transmisión de vídeo y audio en tiempo real. Los simuladores avanzan hacia simulaciones colectivas y de forma conjunta necesitando no sólo la conexión entre los elementos simulados sino también opciones válidas de comunicación para los participantes de la simulación. Esto ha traído como consecuencia el desarrollo de un enorme número de bibliotecas que se dedican al control y manipulación de datos para el envío de estos por la red. Muchos desarrolladores han creado sus propias bibliotecas debido a que cada aplicación puede requerir opciones específicas de ella y que no hayan sido tratadas en ninguna biblioteca existente. En la actualidad hay un gran número de estas, cada una posee características distintivas y fueron programadas para su aplicación en distintas áreas.

1.11.1 HawkNL.

HawkNL es una API de red orientada a juegos libres, de código abierto liberada bajo la licencia GNU LGPL (*GNU Library General Public License*), se considera una API de bajo nivel ya que trabaja bastante cercano a los sockets, tiene funciones sobre Berkeley/Unix Sockets y WinSock. HawkNL también brinda otras ventajas incluyendo soporte para distintos sistemas operativos, grupos de sockets, un temporizador de alta exactitud, estadísticas de los sockets, macros para leer y enviar datos en paquetes con conversión "endian" y soporte para múltiples transportes de red. Ha sido probada en Windows 9X/ME/NT/2000/XP/CE, Linux, IRIX, AIX, BSDs, Mac OS. Esta biblioteca trabaja básicamente con los sockets brindando funciones para la conexión, la lectura y escritura, los cierres de conexión y todo lo referente con el envío de estos datos, haciendo más fácil el trabajo. Es una de las bibliotecas más usadas internacionalmente como base para las conexiones en juegos *online*. Posee comunicación de voz como *software* propietario. [10]

1.11.2 Raknet.

Raknet está programada en C++, constituye un engine (motor) para la gestión y trabajo con redes en juegos, diseñada para tener un alto desempeño, fácil de integrar y constituir una solución completa para juegos y otras aplicaciones. Posee un sistema de replicación de objetos que automáticamente crea, destruye, serializa y transmite los objetos de juego. Tiene una robusta capa de comunicación con un control de flujo automático, ordenamiento de mensajes por múltiples canales, reagrupamiento del mensaje, fraccionamiento y el posterior re ensamblaje del mensaje. Llamadas remotas a procedimientos propios de C y C++, con una lista automática de parámetros serializados. También posee comunicación de voz, por lo que incluye bindings de audio para Port Audio, FMOD y Direct Sound. [11]

1.11.3 Biblioteca STKNET.

STKNET es una biblioteca libre desarrollada en la Universidad de las Ciencias Informáticas por el ingeniero Yessy Cedeño. STKNET brinda funcionalidades para envío y procesamiento de datos, actualmente está en desarrollo y se propone ser la base de los módulos de red en los proyectos del polo de Realidad Virtual. Su diseño orientado a objeto permite a los programadores no tener que interactuar con las funciones de bajo nivel o trabajar directamente con los sockets de la HawkNL, que es la biblioteca que sirve de base a STKNET y soporta todas las rutinas de conexión y lectura/escritura de datos. También

cuenta con un buffer en cuya estructura se determina como serán interpretados los datos, especificando claramente en las funciones de envío y lectura de datos como será la sintaxis del buffer, se pueden enviar en este una gran cantidad de datos, convirtiendo el uso de este recurso en un elemento flexible y multiuso, dando al programador la oportunidad de escribir sus propias funciones, obteniendo módulos personalizados en cada proyecto que use la biblioteca. En STKNET encontramos varias clases que brindan funciones de apoyo y que sirven de estructura base a la hora de conformar los paquetes o de efectuar el envío o recepción de datos. Sin embargo, el núcleo funcional lo constituyen las clases STKPeer, STKServer y STKClient. STKPeer es una clase abstracta de la cual heredan STKServer y STKClient y brinda las funciones básicas de la biblioteca. STKServer y STKClient también son clases abstractas pero ya se enfocan en la función específica que van a desarrollar como cliente o servidor, al trabajar con la biblioteca STKNET hay que heredar de estas dos clases para crear clases cliente y servidor propias, donde se implementan los eventos según las necesidades del proyecto para el que se use, personalizando de este modo el transporte de datos.

1.11.4 DirectPlay.

DirectPlay es una biblioteca de la API de Microsoft DirectX destinada a la comunicación de red prevista para el desarrollo del juego en computadoras. Entre las principales interfaces con las que trabaja DirectPlay encontramos:

IDirectPlayXServer: En esta interfaz se manejan los datos y se hacen los trámites necesarios para acceder al servidor, se maneja todo lo referente a la conexión y el establecimiento de sesiones con el servidor.

IDirectPlayXClient: Esta es la interfaz donde se maneja y se da acceso a la PC cliente, se manejan las solicitudes y el envío de datos de las terminales que están conectadas al servidor.

IDirectPlayXPeer: Interfaz para manipular el envío de datos entre máquinas que estén conectas con la arquitectura de red punto a punto (peer to peer), se controla el envío de datos entre computadoras “peers” o sea de igual categoría sin jerarquía entre ellas.

Esta biblioteca trabaja sobre un sistema UDP, de esta forma, su aprovechan las características de este protocolo que permite la rapidez en la comunicación entre computadoras, también facilita la comunicación

con otros dispositivos de Windows asociados a DirectX. Esta biblioteca es propietaria y no se puede usar sobre sistemas UNIX.

1.11.5 DataReel.

DataReel es una plataforma desarrollada en C++ que constituye una herramienta de desarrollo usada para crear bases de datos multi-hilos y aplicaciones de comunicación. C++ es un lenguaje de programación que produce una rápida ejecución en programas compilados y ofrece poderosas capacidades para la programación. A diferencia de JAVA y PERL el lenguaje C++ no contiene él mismo interfaces de programación base de datos, comunicaciones ni programación multi-hilo. Usando DataReel se puede potenciar el lenguaje de programación C++ utilizando interfaces de alto nivel para la programación de bases de datos, comunicaciones y programación multi-hilo. El paquete de desarrollo de DataReel fue producido por trabajo independiente y bajo contrato y fue liberado al publico bajo una licencia de no exclusividad. El trabajo de creación comenzó de manera independiente en 1997 y fue aumentado del 1999 al 2004 por código bajo contrato para servir de soporte a varias aplicaciones. Son muchos los desarrolladores a través del mundo que han aportado código para hacer de DataReel una biblioteca robusta. En 2005 el código de DataReel se puso bajo el escrutinio para producir un código fuente estable y seguro que permitiera su uso en complejos sistemas comerciales.

Entre las ventajas de DataReel tenemos que simplifica la complejidad temporal consumida al trabajar con bases de datos, sockets y programación multi-hilo brindando una interfaz de programación semejante a la de JAVA para la programación de estos. Es flexible, modular y portable, permite operar entre distintas plataformas y constituye un acercamiento a la programación para redes y bases de datos. [12]

DataReel está disponible para desarrolladores comerciales, individuales y académicos, el código base de esta biblioteca es distribuido un formato de código abierto. El código abierto también promueve la selección del lenguaje C++ para la creación de cualquier aplicación. Al mantener el código abierto esta biblioteca mejora con el aporte de desarrolladores de todo el mundo.

1.11.6 Libtcp++.

Libtcp++ es una biblioteca de clases de C++ que facilita la creación de clientes y servers TCP/IP. Esta biblioteca contiene tres clases, TcpClient, TcpServer y TcpRuletSet. TcpServer ha construido un método

para la detección de las capacidades de logueo de una computadora “peer”, y un control de acceso basado en IP para regular la funcionalidad del servidor. La clase de TcpClient brinde en la función connect () un parámetro timeout que resulta útil para los escaneos de puertos de *host*, y para otras situaciones en las que estés tratando de conectarte a *host* apagados o a puertos protegidos por reglas de *firewall*. La biblioteca ha sido probada en Linux y BSD, en otros sistemas aún no ha sido probada, con el ajuste de algunas líneas debe ser posible correrla en distintos compiladores y sistemas operativos. Brinda el código libre y se pueden realizar las modificaciones necesarias que estime el usuario. [13]

1.11.7 Zoidcom.

Zoidcom es una biblioteca de red de alto nivel, esta biblioteca basada en el protocolo UDP provee ventajas para la replicación de objetos de juego y la sincronización de sus estados sobre la conexión de red de una manera altamente eficiente referente al ancho de banda. Esto se logra al multiplexar y demultiplexar la información de los objetos desde y hasta secuencias de bits, lo que hace posible evitar el envío de datos redundantes. Los booleanos necesitan un solo bit, los datos enteros y flotantes se separan en cuantos bits sean necesarios. Zoidcom es libre para el uso no comercial. Tiene versiones para Windows, Linux y Mac OS. [14]

Entre las ventajas de Zoidcom tenemos:

- ✓ **Fácil de usar y flexible:** Es una API simple y directa hecha en C++, posee una extensa documentación, hay una gran cantidad de programas ejemplos disponibles, no se necesitan pasos de construcción adicionales, administración de memoria personalizada.

- ✓ **Conectividad:** Rápida, basada en el protocolo de red UDP usando secuencias de bits para el envío aunque se tiene muy poco control de los datos enviados. El sistema completo puede operar sobre un solo puerto UDP, de forma automática crea los paquetes de la talla requerida. Tiene una dinámica distribución y limitación del ancho de banda y envía paquetes UDP a destinos arbitrarios.

1.11.8 PJSIP.

PJSIP es un conjunto de bibliotecas y utilidades VoIP multiplataforma para la comunicación basada en el protocolo SIP. PJSIP es extremadamente portable ya que incluye soporte para Windows, Windows Mobile,

Linux, Unix, MacOS, RTEMS, Symbian OS y está programada en C. Presenta un alto rendimiento, lo que significa menos demanda de potencia de CPU y más transacciones SIP, además, es ideal para sistemas embebidos donde el espacio es costoso y también para aplicaciones más generales. Presenta una comunidad amplia y documentación útil y precisa, haciendo sencillo a los desarrolladores usar este conjunto de bibliotecas. [15]

Sus principales características son:

- ✓ Múltiples líneas/identidades.
- ✓ Múltiples llamadas.
- ✓ Retención y transferencia de llamadas.
- ✓ Mensajería instantánea.
- ✓ Conferencias.
- ✓ Auto-responder.
- ✓ Grabación.

Investigando de manera más exhaustiva sobre PJSIP percibimos que la virtud que tenía era que no solamente es una API para el manejo del protocolo de sesiones SIP, sino que consistía en una completa arquitectura de componentes para las transmisiones multimedia, tal y como se puede ver en la en la siguiente figura:

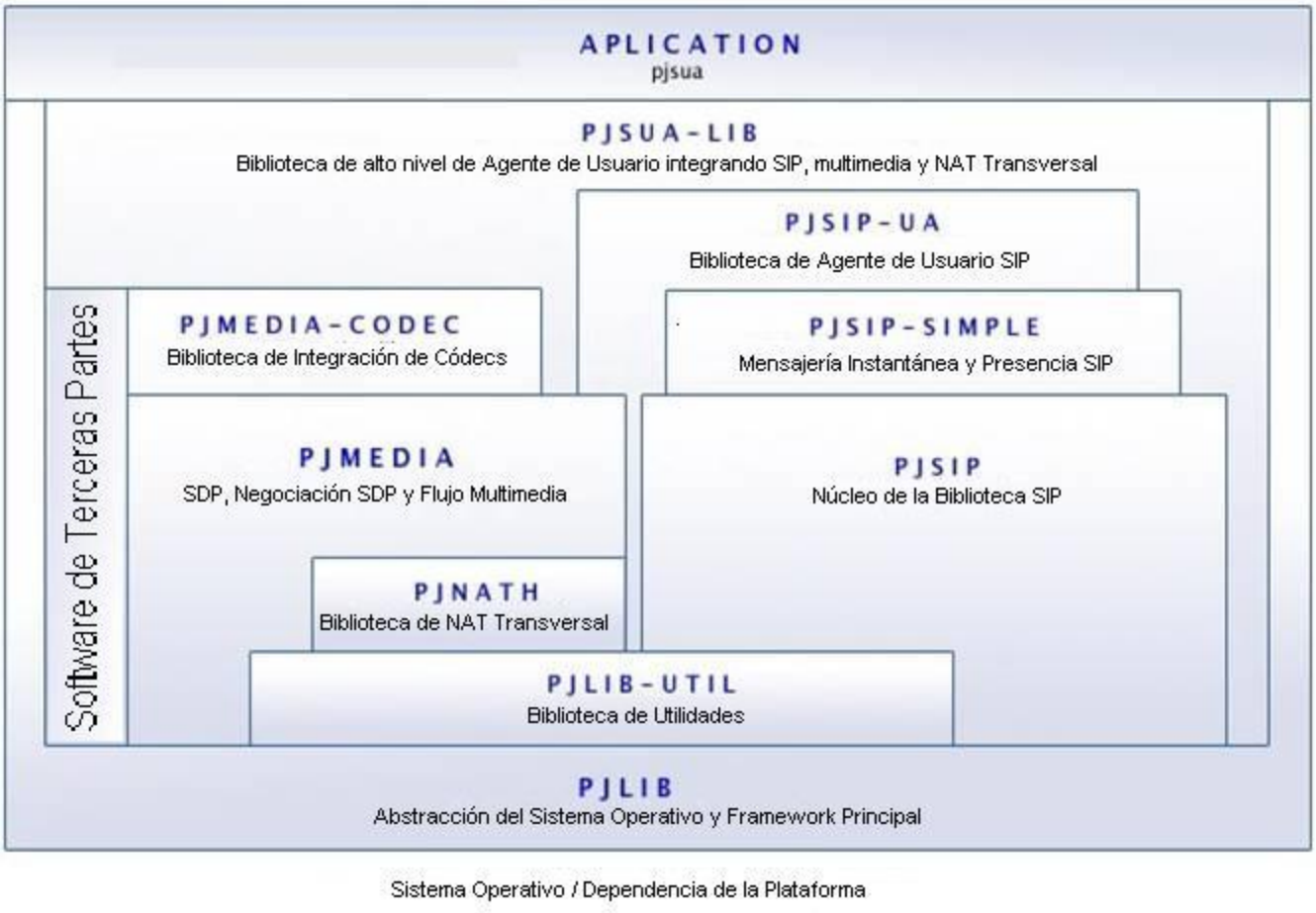


Figura 6 Arquitectura de componentes de PJSIP.

1.11.9 PJMEDIA.

PJMEDIA es una biblioteca complementaria para PJSIP, que se encarga de gestionar todos los aspectos referentes a los flujos de audio y vídeo y su transmisión en tiempo real, dando soporte para ello mediante la implementación de RTP/RTSP. La unión de estas bibliotecas permite desarrollar un agente basado en SIP con todas las funcionalidades necesarias en una comunicación VoIP. Al igual que PJSIP es multiplataforma y soporta el codificado y decodificado de señales, presenta funcionalidades para evitar la pérdida de paquetes y permite grabación y reproducción de archivos de audio. También permite transacciones bidireccionales y cancelación de eco. [16]

1.11.10 PJSUA.

Es una biblioteca de alto nivel que se encarga de abstraer al desarrollador de la complejidad de las bibliotecas PJSIP y PJMEDIA al unir las funcionalidades de ambas y permitir el desarrollo de un Agente de Usuario Simple (SUA por sus siglas en inglés). [17]

1.12 Bibliotecas de audio.

1.12.1 Open Audio Library (OpenAL).

Uno de los aspectos más importantes para lograr la calidad tanto en un juego como en un simulador es el sonido, de un buen efecto de sonido depende en gran parte el realismo que se le puede imprimir al entorno. También es necesaria una biblioteca que sea capaz de transmitir, guardar y controlar los tipos de datos relacionados con el sonido, tanto del entorno como de medios de comunicación que se puedan establecer entre los usuarios.

Cuando se habla de OpenAL no se puede dejar de mencionar su característica distintiva: esta API es capaz de proporcionar sonido tridimensional (sonido 3D). Qué quiere esto decir, pues el sonido se oirá de formas distintas de acuerdo con disímiles parámetros y situaciones. Dependerá de la posición de la fuente con respecto al oyente, con su movimiento, con la velocidad entre ambos, y el tipo de entorno, entre muchos otros parámetros.

A la hora de trabajar con OpenAL hay pasos que son esenciales, debemos establecer la fuente del sonido. Para esto hay que indicar la posición y velocidad de la fuente, al llenar estos vectores se le establece la ubicación a la fuente, hay que crear el buffer y la fuente e indicar la posición, velocidad y la orientación del oyente.

En OpenAL también se inicializa el buffer de sonido y la fuente de este, a la hora de cargar los sonidos hay distintas variables que son imprescindibles: tamaño, frecuencia, el formato y los datos en específico. La forma más común es utilizar la misma biblioteca como dispositivo de salida, sin embargo, hay otros dispositivos de salida con los cuales la calidad y elegancia del sonido aumentan, tal es el caso de DirectSound3D, SDL, ALSA y MMSytem. [18]

La función con la que se carga el sonido en OpenAL de forma general no es la misma para Windows y Linux, esto se debe a que la biblioteca “alut” no se incluye dentro de los archivos oficiales de OpenAL, constituye una biblioteca auxiliar. Se pueden encontrar el caso de bibliotecas auxiliares que utilicen el mismo código en ambas plataformas, ejemplo la “libSDL”, con el uso de esta se logra un código más multiplataforma.

Esta biblioteca constituye una buena opción para el trabajo con sonidos, tanto para el manejo de datos de audio para su posterior envío o almacenamiento, así como para aplicarle sonido 3D a los entornos creados en simuladores y juegos.

1.13 Sistema de transmisión de vídeo en tiempo real.

En la actualidad el desarrollo de Internet y de las tecnologías multimedia ha traído como consecuencia que la transmisión de vídeo, audio y animaciones multimedia en tiempo real se ha convertido en centro de debate y desarrollo constante de tecnologías y protocolos que trabajan con este tipo de transmisión. A esta tecnología se le conoce como tecnología de transmisión de flujo multimedia, su función principal es mandar en tiempo real datos de vídeos, sonido y animaciones multimedia a los usuarios desde servidores, de esta formas los usuarios no tienen que esperar a que se descarguen todos los datos, sino sólo unos segundos y comienzan a visualizar en tiempo real el flujo multimedia.

Un sistema de transmisión de vídeo en tiempo real incluye la captura del vídeo (en caso del uso de webcams, en otros casos se ubica las secuencias de vídeo digital a transmitir), la codificación del vídeo en caso de ser necesario, la transmisión por la red en tiempo real y finalmente la decodificación y reproducción del vídeo.

1.13.1 Captura del vídeo.

Para realizar la captura de vídeo desde un webcam es necesario instalar los drivers necesarios según el sistema operativo que se use par el correcto desempeño de la herramienta, una vez instalados se prepara una aplicación para la captura de la secuencia de vídeo. Estos pasos proveen un vídeo analógico o ya directamente un vídeo digitalizado que se utilizara para la transmisión.

1.13.2 Codificación del vídeo.

En el momento de transmitir la información esta deber comprimida y codificada para generar una secuencia de vídeo orientada a la comunicación por la red y que presenta las características requeridas por los estándares de transmisión y de la red. Existen distintos APIs que brindan estas funcionalidades, por ejemplo la biblioteca libavcodec, esta contiene todos los codificadores y decodificadores de audio y vídeo FFmpeg, y vale aclarar que es una herramienta libre.

Un estándar que se debe tener en cuenta para llevar a cabo este trabajo es el MPEG-4, que requiere una baja tasa de transmisión y usa eficientemente el ancho de banda, comprime los datos a través de la reconstrucción de frames y obtiene una mejor calidad de imagen usando menor cantidad de información, provee una gran tasa de compresión y al mismo tiempo hay una menor pérdida de datos, manteniendo de esta forma la calidad del vídeo.

1.13.3 Transmisión en tiempo real.

En este paso se envían la secuencia de bits codificados a través de un transmisor de tiempo real por la red. Esta transmisión es regida por el protocolo de transporte en tiempo real (RTP), que constituye el estándar a utilizar y la tecnología clave en este tipo de transmisión. Existen disímiles bibliotecas de red que utilizan RTP para la transmisión de datos, estas bibliotecas manejan internamente el protocolo de control de RTP (RTCP), y brindan las funcionalidades necesarias para llevar a cabo la transmisión de datos en tiempo real, la mayoría de estas bibliotecas. [19]

1.13.4 Decodificación del vídeo y reproducción.

Finalmente, se decodifica la secuencia de vídeo, se reconstruye la señal y se reproduce en las herramientas de salida del receptor. Para llevar a cabo este proceso es muy útil el uso de las APIs de Simple DirectMedia Layer (SDL).

SDL - Simple DirectMedia Layer:

SDL es una biblioteca multimedia con licencia GNU LGPL (GNU Library General Public License) que permite que los programas usen audio, el teclado, mouse, *joystick*, 3D hardware (vía OpenGL) y un *framebuffer* de vídeo 2D. Esta biblioteca es utilizada por MPEG software reproducción, en emuladores y

en varios juegos. Soporta una gran cantidad de sistemas operativos entre los que se incluye Linux, Windows, BeOS, MacOS Classic, Mac OS X, FreeBSD, OpenBSD, Solaris, IRIX y QNX. Se puede acceder a la biblioteca a través de C, C++, Ada, Eiffel, Java, Lua, ML, PHP, Python y otros. [20]

1.14 Conclusiones del capítulo.

Durante este capítulo, donde está sentada la base del estudio del tema en que se desenvolverá nuestro trabajo, realizamos una profunda investigación sobre una serie de conceptos básicos como la VoIP, los protocolos de red y de señalización, las técnicas de transmisión de datos por la red y las arquitecturas de comunicación, que pensamos son fundamentales para el entendimiento del trabajo a realizar. Además, se estudiaron características distintivas de varias bibliotecas de red, con el fin de escoger la que más se ajusta a las necesidades de nuestro trabajo. También se profundizó en el tema de transmisión de vídeo por la red en tiempo real, teniendo en cuenta aspectos como la captura, decodificación, transmisión por la red, codificación y reproducción del vídeo. También se profundizó en los códecs que permitirán la codificación de la voz para su posterior transmisión.

CAPÍTULO 2: SOLUCIONES TÉCNICAS.

2.1 Introducción.

En el capítulo anterior estudiamos las bibliotecas de red más utilizadas para el envío de datos, sus características particulares, así como las técnicas y sistemas utilizados en la transmisión de información desde archivos y en sesiones en tiempo real. En este capítulo se proponen las soluciones técnicas para la creación del módulo de transmisión de audio y vídeo, los métodos específicos que se utilizarán y las bibliotecas más adecuadas según los requerimientos del producto.

2.2 Bibliotecas de red, protocolo de señalización y códecs a utilizar.

Teniendo en cuenta lo estudiado anteriormente utilizaremos el protocolo de señalización SIP para el establecimiento, mantenimiento y terminación de sesiones interactivas entre usuarios; estas sesiones pueden tratarse de conferencias multimedia, chat, sesiones de voz o distribución de contenidos multimedia; esta decisión fue tomada debido a que SIP es un protocolo pensado desde un primer momento para Internet, con arquitectura cliente/servidor y mensajes tipo texto similares a los mensajes HTTP. Además, SIP constituye una de los protocolos de señalización que soporta PBX Asterisk, el cual nos servirá de intermediario ente los usuarios, es decir, lo utilizaremos como servidor, ya que soporta los servicios de transmisión de voz sobre IP (VoIP). Para la transmisión de audio y vídeo en tiempo real usaremos las bibliotecas de redes PJSIP, PJMEDIA y PJSUA, pues permiten comunicación basada en el protocolo SIP. Como códec de audio se propone el uso de GSM y para la codificación de vídeo el H .263+.

2.3 Lenguajes.

2.3.1 De Modelado: UML.

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar.

Un modelo UML nos indica que es lo que supuestamente hará el sistema, pero no como lo hará. Mediante las combinaciones de sus elementos gráficos UML nos permite conformar una serie de diagramas de manera estándar con los que podemos generar un anteproyecto fácil de entender para cualquier persona involucrada en el proceso de desarrollo.

Con la notación UML podemos compartir y comunicar el conocimiento de una arquitectura gracias a la combinación simultánea de cinco perspectivas:

1. **Definir:** Fijar, determinar, decidir, explicar un concepto a través de sus atributos distintivos. Señalar sus límites y dar una idea exacta de lo que es esencial y de lo que es circunstancial.
2. **Organizar:** Establecer unos recursos, disponer un orden de responsabilidades y formalizar unas reglas de relación y actuación; todo ello orientado a conseguir un propósito.
3. **Visualizar:** Representar mediante imágenes y/o símbolos el contenido y la organización de los conceptos que configuran un sistema. Hacer visible su naturaleza y su complejidad.
4. **Actuar:** Pensar y tomar decisiones de manera ágil y sistemática, siguiendo un método; éste a su vez, define el modo de actuar sobre la base de la relación de un conjunto de actores, actividades, entregables y certificaciones posibles en un escenario concreto.
5. **Certificar:** Comprobar de manera fehaciente que un entregable es completo, coherente y usable para el propósito que ha sido creado.

El resultado, es una mayor comprensión y claridad sobre la naturaleza de los objetos, eventos y hechos que tienen consecuencias dentro de un dominio.

2.3.2 De Programación: C++.

Primero que todo C++ es el lenguaje de programación de la Facultad 5, a lo largo de la carrera hemos aprendido que con C++ se pueden hacer todo tipo de programas, desde los más simples hasta los más complejos. C++ es un lenguaje de programación orientado a objetos lo cual es esencial para incrementar la productividad, calidad y reutilización del software. Ofrece gran riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Además, otras de las grandes ventajas de C++, es que es un lenguaje multi-nivel, es decir, puedes usarlo tanto para programar directamente el hardware, como para crear aplicaciones tipo Windows, definidas todas por poseer una misma interfaz. Además, es muy importante

que es portable, es decir, que todo programa escrito en C++ puede ser compilado en cualquier Sistema Operativo.

2.4 Metodología: RUP.

RUP (Racional Unified Process). Proceso Unificado Racional es una metodología cuyo fin es entregar un producto de software donde se estructuran todos los procesos y se mide la eficiencia de la organización. Utiliza el lenguaje unificado de modelado UML para conformar los esquemas y una serie de metodologías adaptables al contexto y necesidades de cualquier sistema de software.

RUP constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Con el garantizamos una forma disciplinada de asignar tareas y responsabilidades además de verificar la calidad del software.

Características de RUP:

- ✓ **Iterativo e Incremental:** En el Proceso Unificado Los proyectos se entregan en etapas iteradas. En cada iteración se analiza la opinión, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados. Esta serie de iteraciones trae como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo.

- ✓ **Dirigido por casos de uso:** En el Proceso Unificado los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. Cada iteración toma un conjunto de casos de uso o escenarios y desarrolla todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, etc.

- ✓ **Centrado en la arquitectura:** El Proceso Unificado asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema.

2.5 Herramientas a utilizar.

2.5.1 Qt Designer.

Para la construcción de la interfaz gráfica se utilizará Qt Designer una herramienta para el diseño y construcción de Interfaces Gráficas de Usuario (GUIs), gratuita para aplicaciones de código abierto. Este software está provisto de herramientas que facilitan la creación de formas, botones, ventanas de diálogo muy elegantes, además permite su previsualización, lo que ayuda en gran medida a crear mejores diseños. Qt Designer se puede descargar de Internet para casi todas las plataformas, ya sean sistemas Linux, MacOS, o Windows, por lo que la aplicación resultante puede ser compilada y utilizada en cualquier sistema operativo sin necesidad de cambiar el código. Qt hace uso de una extensa biblioteca de clases y herramientas, las cuales se encuentran bien documentadas en la ayuda del software.

2.5.2 Visual Paradigm.

Visual Paradigm for UML (VP-UML) es una poderosa herramienta de modelado visual, está diseñada para apoyar a arquitectos, desarrolladores, diseñadores, analistas de procesos de negocio y modeladores de datos. Este software de modelado UML constituye una gran ayuda en la rápida construcción de aplicaciones de calidad a un menor costo. Además de su Interfaz amigable y fácil de usar (VP-UML EE) nos brinda motores de generación de código para diez lenguajes de programación diferentes entre los cuales se encuentra el C++ que es el lenguaje en el que se programará. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Algunas de sus principales características:

- ✓ Ingeniería Inversa, Código a modelo, código a diagrama.
- ✓ Generación de Código: Modelo a código, diagrama a código.
- ✓ Generación de Informes para elaboración de documentación.
- ✓ Distribución automática de diagramas, reorganización de las figuras y conectores de los diagramas UML.
- ✓ Carácter multiplataforma ya que se puede usar tanto en sistemas Linux, Unix o Windows.

2.6 Conclusiones del Capítulo.

En este capítulo sentamos las bases técnicas sobre las cuales se va a implementar el módulo, que dará solución al objetivo propuesto. Se escogieron las bibliotecas de red necesarias, así como las metodologías y herramientas que se van a utilizar. Es importante destacar que todo el software que se utilizará es software libre, de código abierto y multiplataforma, por lo que el módulo resultante no necesitará ninguna licencia para su uso y podrá ser portado fácilmente a otros sistemas operativos.

CAPÍTULO 3: CARACTERISTICAS DEL SISTEMA.

3.1 Introducción.

En este capítulo se comienza a tener una visión más clara del módulo que será implementado, las necesidades del cliente que este satisface, sus características y funcionalidades. Se determinan las reglas del negocio y se hace el levantamiento de requisitos funcionales (capacidades o funciones que el sistema debe cumplir) y los no funcionales (propiedades o cualidades que el producto final debe presentar). También se presenta el modelo del dominio para representar los conceptos principales que rigen y se manejan en el sistema.

3.2 Reglas del negocio.

- ✓ Los datos de audio transmitidos serán codificados con el códec GSM.
- ✓ Las direcciones de destino para los envíos de audio y vídeo serán instancias de RTPIPv4Address.
- ✓ Los vídeos transmitidos serán codificados con H .263+.

3.3 Modelo del dominio.

A continuación se presentan los principales conceptos para el sistema que se va a construir. Este modelo nos sirve como base para diseñar el módulo. Se utilizarán técnicas para modelar con similitud a las de la programación orientada a objetos, creando de esta forma un modelo con objetos del dominio que tengan un gran parecido a los objetos reales que se van a construir e identifiquen claramente el problema a resolver.

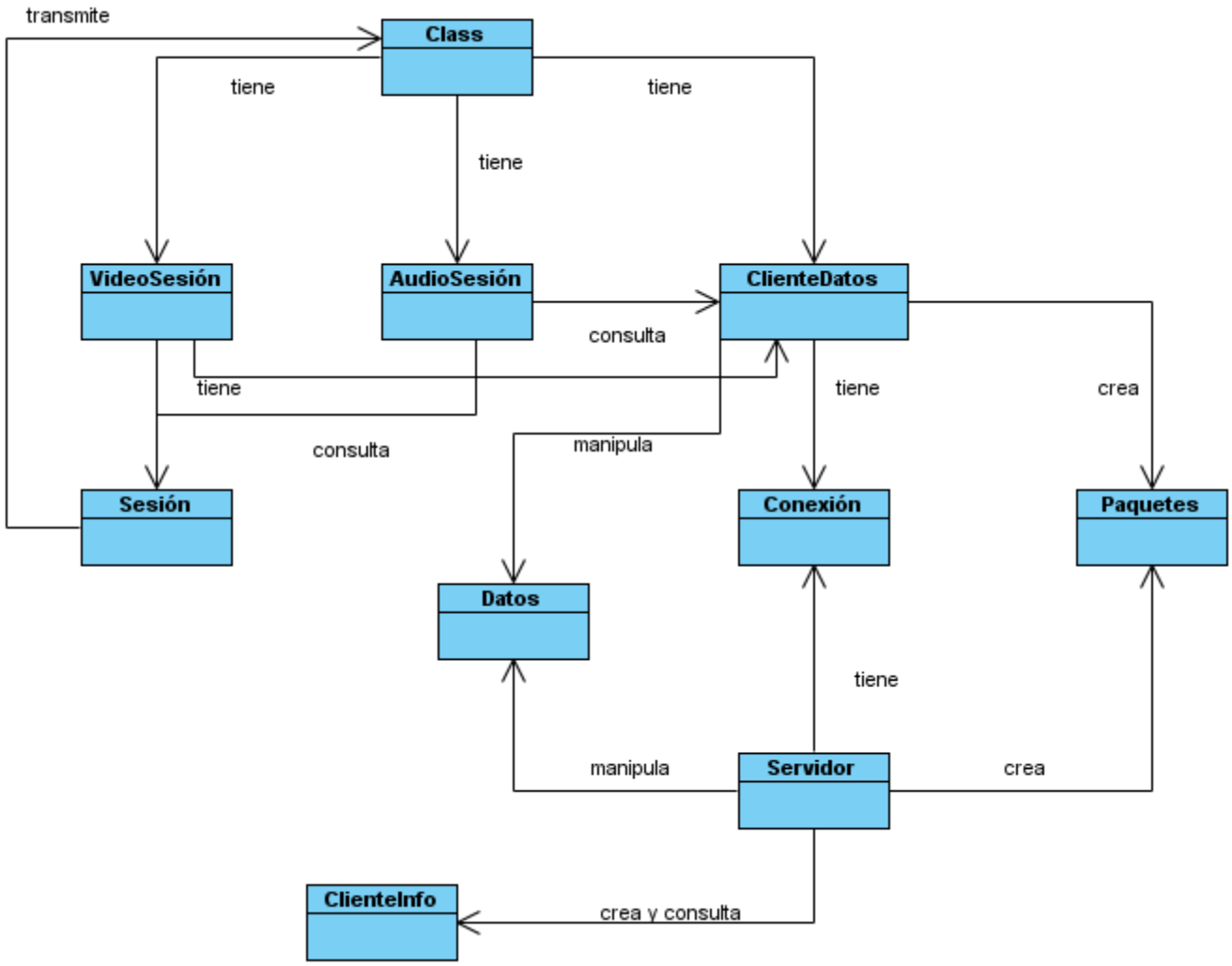


Figura 7 Modelo de Dominio.

En este modelo presentamos las relaciones entre los conceptos manejados para reflejar la funcionalidad del sistema. Aquí el “Cliente” se va a servir de las funciones que se encuentran en “ClienteDatos”, “AudioSesión” y “VideoSesión” que es donde se desarrolla la lógica de la transmisión de los datos según el tipo que el Cliente desee transmitir. En “ClienteDatos” se manipulan los Datos y es el encargado de estructurar los Paquetes y a través de Conexión establece la comunicación con Server. En “AudioSesión” y “VideoSesión” se establece, como tal, la comunicación con los otros clientes mediante Sesión que es el método que utilizan las bibliotecas usadas para la transmisión de vídeo y sonido en tiempo real. En Sesión se crea una conexión directa entre los clientes para el envío de sonido y vídeo en tiempo real.

3.3.1 Glosario de términos del Modelo del dominio.

- ✓ **Cliente:** Es quien se va a servir de los servicios de transmisión del módulo de red, pasándole a este los datos que quieren que sean transportados.
- ✓ **ClienteInfo:** Representa la estructura en la que el servidor almacena los datos de los clientes conectados en ese momento.
- ✓ **ClienteDatos:** En él se extraen los datos recibidos de los paquetes y se manipulan los datos a enviar para conformar los paquetes, por medio de Conexión efectúa la lógica de conexión, transmisión y recepción de paquetes. También a través de este se obtiene la información necesaria para la transmisión de sonido y vídeo.
- ✓ **Conexión:** Medio de comunicación entre ClienteDatos y Servidor, a través de este se envían y reciben los datos.
- ✓ **Datos:** Representan la información que se quiere transmitir, los datos que conformarán los paquetes de comunicación entre el servidor y el cliente.
- ✓ **Paquetes:** *Buffer* dentro del cual se transportan los datos, la sintaxis de construcción y lectura del *buffer* determina el uso y funcionalidad de los datos que son transportados.
- ✓ **Servidor:** Asegura la conexión entre los clientes, permanece escuchando en espera de nuevas conexiones y de paquetes entrantes y los redirecciona hacia los destinatarios. Atiende los llamados especiales de los clientes cuando estos piden datos que se necesitan para la transmisión interna de vídeo o sonido y le brinda la información necesaria para inicializar las sesiones de transmisión.
- ✓ **Sesión:** Conexión que se estable entre los clientes a través de la cual se hace el envío de audio o vídeo en tiempo real.
- ✓ **VideoSesión:** Se encarga de la transmisión de vídeo en tiempo real, se inicializan los parámetros y componentes necesarios para iniciar una Sesión de transmisión de vídeo.

3.4 Captura de requisitos.

Los requisitos son las condiciones o capacidades que deben ser alcanzadas o poseídas por un sistema para satisfacer las necesidades del cliente. Se pueden clasificar como funcionales y no funcionales. Los funcionales representan las capacidades y condiciones que deben ser brindadas por el módulo, y los no funcionales son las propiedades y cualidades del producto. En este trabajo se han definido claramente dos estructuras funcionales, que se podrían clasificar como subsistemas dentro del módulo (subsistema cliente y subsistema servidor), cada uno posee sus propios requisitos, debido a esto y para una mejor comprensión especificaremos a que cual de los dos pertenece cada requisito.

3.4.1 Requisitos funcionales.

Requisitos funcionales del subsistema Cliente:

- ✓ RF1C - Conectarse al servidor cuya dirección IP y puerto se especifiquen.
- ✓ RF1.1C – Informar si se pudo o no realizar la conexión al servidor.
- ✓ RF2C – Enviar datos de tipo audio a los otros clientes.
- ✓ RF3C – Enviar datos de tipo vídeo a los otros clientes.
- ✓ RF4C – Recibir audio.
- ✓ RF5C – Recibir vídeo.
- ✓ RF6C – Terminar la conexión con el servidor.
- ✓ RF6.1C - Avisar al servidor cuando abandone la conexión.

Requisitos funcionales del Servidor:

- ✓ RF1S – Estar en línea (*online*), abrir correctamente un puerto y permanecer esperando la conexión de los clientes.
- ✓ RF2S – Aceptar la conexión de un nuevo cliente.

- ✓ RF2S.1 – Verificar la información enviada por el cliente, comprobar si puede ser aceptado o no este nuevo cliente.
- ✓ RF2S.2 – Guardar la información de los clientes aceptados.
- ✓ RF3S – Recibir y redireccionar los paquetes que envíen los clientes.
- ✓ RF4S – Notificar a los clientes cuando el servidor deje de prestar servicios.
- ✓ RF5S – Mostrar todas las acciones realizadas por los clientes a través del servidor (server log).
- ✓ RF6S – Desconectar.

3.4.2 Requisitos no funcionales.

Software:

- ✓ Para la codificación y decodificación de vídeo deben estar disponibles: Sistema Windows: AVCodec y DirectShow, familia UNIX: Video4Linux.
- ✓ Para la codificación y decodificación de sonido debe estar disponible el códec GSM para el sistema operativo que se esté utilizando.
- ✓ Para la transmisión de audio y vídeo entre los usuarios es necesario que se posea el PBX Asterisk, ya que es este el que se va encargar de realizar las funciones del servidor.

Hardware:

- ✓ Tarjeta de Red.
- ✓ Canal físico de comunicación.

Para transmitir y recibir vídeo:

- ✓ Webcam.

- ✓ Tarjeta de vídeo.

Para transmitir y recibir sonido:

- ✓ Micrófono.
- ✓ Tarjeta de sonido.
- ✓ Audífonos.

Diseño e implementación:

- ✓ Lenguaje de programación C++.
- ✓ Se utilizará la herramienta de desarrollo integrado QT Creator para programar.
- ✓ Se utilizará la aplicación QT Designer para diseñar la interfaz gráfica del módulo.
- ✓ Se utilizará la biblioteca PJSUA para la transmisión de audio y vídeo por la red en tiempo real.
- ✓ Se utilizará la biblioteca de clases QT para construir la aplicación visual del servidor.

3.5 Modelo de casos de uso del sistema.

En este trabajo se han definido dos subsistemas funcionales (cliente y servidor), los cuales al interactuar brindan todas las funcionalidades del módulo. En cada uno de estos subsistemas se definen Actores del Sistema y Casos de Uso del Sistema.

3.5.1 Definición de actores del Sistema.

- ✓ **SUBSISTEMA CLIENTE**

Actores	**Justificación**
Cliente	Usuario que interactúa con las funcionalidades que brinda el módulo.

Servidor	Aplicación que envía mensajes especiales para iniciar funcionalidades en el cliente.

Tabla 2 Actores del sistema (clientes).

- ✓ **SERVIDOR**

Actores	**Justificación**
Administrador	Usuario que se encarga de levantar y retirar los servicios del servidor.
Servidor	Aplicación que envía mensajes al servidor para obtener los servicios de transmisión de datos.

Tabla 3 Actores del sistema (servidor).

3.5.2 Casos de uso del sistema.

- ✓ **SUBSISTEMA CLIENTE**

CU1	**Conectar Cliente**
Actor	Cliente
Descripción	Se inicializa la parte del módulo que tiene que ver con el cliente y este intenta conectarse con el servidor. En caso de no realizarse la conexión se le informa.

Referencia	RF1C, RF1.1C

Tabla 4 CU1 Conectar Cliente.

CU2	**Enviar Audio**
Actor	Cliente
Descripción	Se envía el audio capturado a través del micrófono a los demás clientes conectados al servidor.
Referencia	RF2C

Tabla 5 CU2 Enviar Audio.

CU3	**Enviar Vídeo**
Actor	Cliente
Descripción	Se envía el vídeo capturado con *webcam* a los demás clientes conectados al servidor.
Referencia	RF3C

Tabla 6 CU3 Enviar Vídeo.

CU4	Recibir Audio
Actor	Servidor
Descripción	Inicializa los componentes necesarios para reproducir audio.
Referencia	RF4C

Tabla 7 CU4 Recibir Audio.

CU5	Recibir Vídeo
Actor	Servidor
Descripción	Inicializa los componentes necesarios para reproducir vídeo.
Referencia	RF5C

Tabla 8 CU5 Recibir Vídeo.

CU6	Desconectar
Actor	Cliente
Descripción	Cierra la conexión con el servidor.

Referencia	RF6C, RF6.1C

Tabla 9 CU9 Desconectar.

- ✓ **SERVIDOR**

CU1	**Inicializar el Servidor**
Actor	Administrador
Descripción	Abre un puerto para la conexión y comienza a mostrar todas las acciones realizadas por los clientes que se conectan a través de él.
Referencia	RF1S, RF7S

Tabla 10 CU1 Inicializar el Servidor.

CU2	**Procesamiento de paquetes**
Actor	Servidor
Descripción	Identifica el paquete enviado por el cliente y de acuerdo con este realiza las acciones definidas para responder a la petición.
Referencia	RF2S, RF2S.1, RF2S.2, RF2S.3, RF3S, RF4S, RF4S.1, RF6S

Tabla 11 CU2 Procesamiento de Paquetes.

CU3	Desconectar
Actor	Administrador
Descripción	Termina los servicios del servidor, informa a los clientes que se ha cerrado.
Referencia	RF5S

Tabla 12 CU3 Desconectar.

3.5.3 Diagrama de casos de uso del sistema.

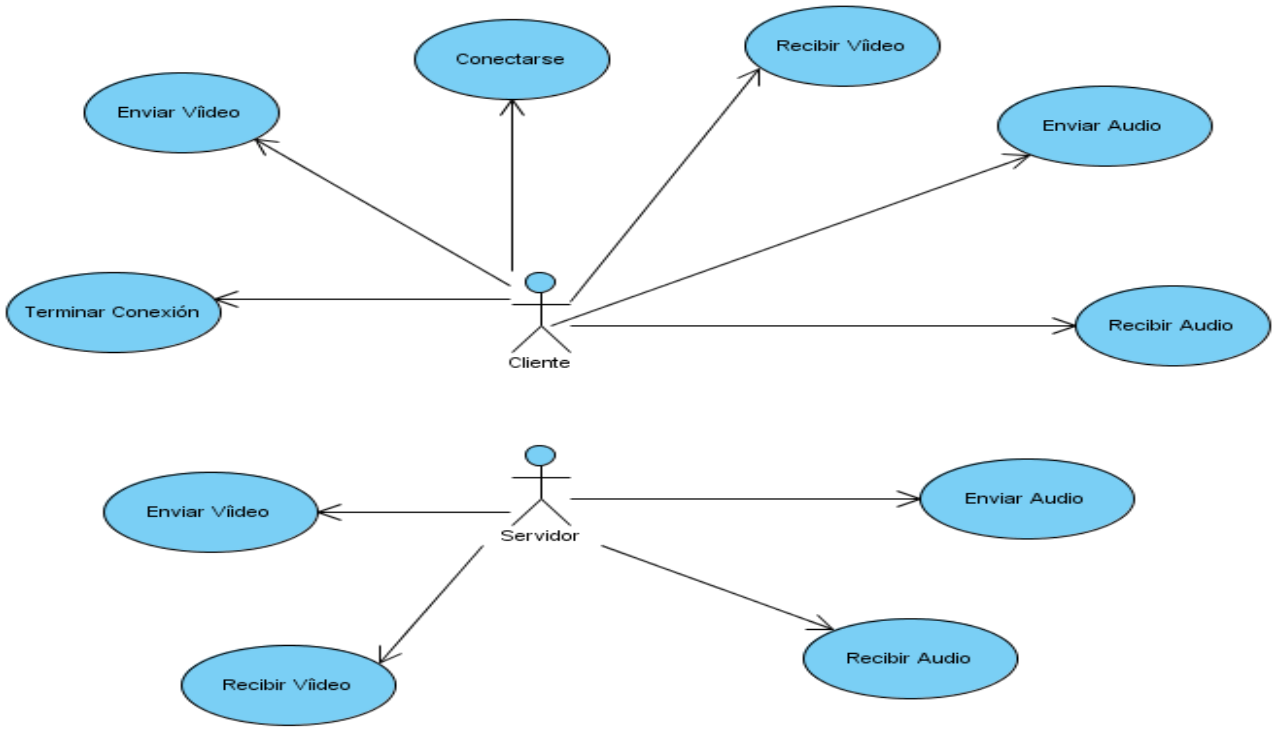


Figura 8 Diagrama de casos de uso del sistema (subsistema cliente).

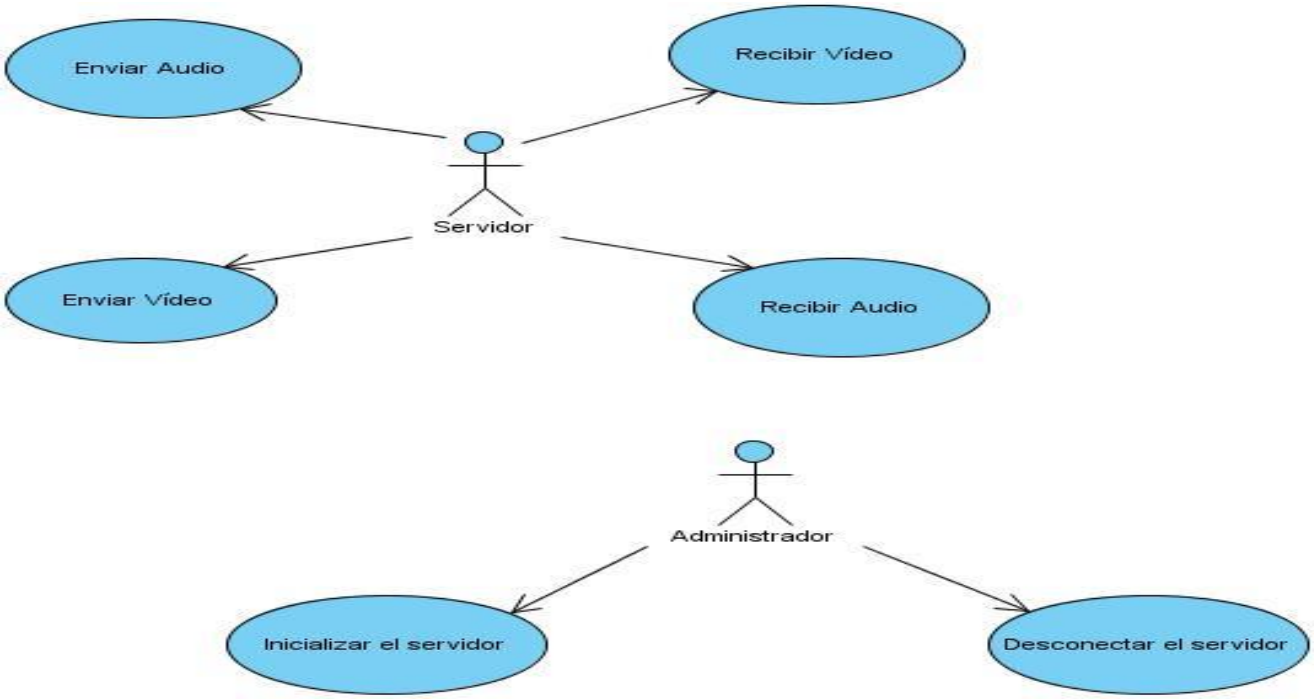


Figura 9 Diagrama de casos de uso del sistema servidor.

3.6 Expansión de casos de uso.

- ✓ **SUBSISTEMA CLIENTE**

Caso de Uso	
CU1	**Conectar Cliente.**
Propósito	Conectar la Aplicación al servidor.
Actores	Cliente

Resumen	El caso de uso se inicia cuando el Cliente desea conectarse al servidor usando la función de conexión que brinda el módulo.
Referencias	RF1C, RF1.1C
Precondiciones	**-**

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El cliente indica la dirección del Servidor, el nombre de usuario, la contraseña y pide conectarse.	1.1- Abre el puerto de conexión, se pone en línea y guarda el nombre de usuario y la contraseña.
	1.2- Se conecta al servidor.
	1.3- Informa de que la conexión se estableció.

Flujo Alterno 1	
Acción del Actor	**Respuesta del Sistema**
	1.1- No se puede abrir el puerto para la conexión.

	1.2- Informa un error al abrir el cliente.
Flujo Alterno 2	
Acción del Actor	**Respuesta del Sistema**
	1.2 No se puede establecer conexión con el servidor.
	1.3 Informa de que no se pudo conectar al servidor indicado.
Pos condiciones	Cambia el estado ha conectado y se establece la conexión con el servidor.
Prioridad	Crítico.

Tabla 13 Descripción del CU1 Conectar Cliente.

Caso de Uso	
CU2	**Enviar Audio.**
Propósito	Transmitir el sonido capturado a través del micrófono al resto de los clientes.

Actores	Cliente
Resumen	El caso de uso se inicia cuando el Cliente llama a la función de transmitir Audio del módulo.
Referencias	RF2C
Precondiciones	Estar conectado al Servidor.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Cliente llama a la función de enviar Audio.	1.1 Crea la URL del protocolo SIP con los datos entrados por el usuario.
	1.2 Añade la dirección del usuario de destino para la transmisión de Audio.
	1.3 Inicializa los dispositivos de entrada para capturar sonido a través del micrófono.
	1.4 Envía el paquete al Servidor, inicializando la llamada.

Pos condiciones	Se comienza a transmitir audio en tiempo real a los clientes.
Prioridad	Crítico.

Tabla 14 Descripción del CU2 Enviar Audio.

Caso de Uso		
CU3	**Enviar Vídeo.**	
Propósito	Transmitir el vídeo capturado a través de una webcam a otro cliente.	
Actores	Cliente	
Resumen	El caso de uso se inicia cuando el Cliente llama la función de transmitir Vídeo del módulo.	
Referencias	RF3C	
Precondiciones	**Estar conectado al Servidor.**	
Flujo Normal de Eventos		
Acción del Actor		**Respuesta del Sistema**

1. El Cliente llama a la función de enviar Vídeo.	1.1 Crea la URL del protocolo SIP con los datos entrados por el usuario.
	1.2 Añade la dirección del usuario de destino para la transmisión de Vídeo.
	1.3 Inicializa los dispositivos de entrada para capturar sonido y el vídeo a través del micrófono y la webcam.
	1.4 Envía el paquete al Servidor, inicializando la transmisión.
Pos condiciones	Se comienza a transmitir vídeo en tiempo real a los clientes.
Prioridad	Crítico.

Tabla 15 Descripción del CU3 Enviar Vídeo.

Caso de Uso	
CU4	**Recibir Audio.**
Propósito	Recibir el audio enviado por otro cliente.
Actores	Servidor.

Resumen	El caso de uso se inicia cuando el servidor re direcciona el audio hacia el cliente destino.
Referencias	RF4C
Precondiciones	Estar conectado al Servidor.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Servidor envía el audio recibido hacia el cliente destino.	1.1- Identifica el paquete.
	1.2- Inicializa los componentes para recibir la transmisión de audio.
Pos condiciones	Se recibe el audio que es transmitido en tiempo real por otro cliente.
Prioridad	Crítico.

Tabla 16 Descripción del CU4 Recibir Audio.

Caso de Uso

CU5	**Recibir Vídeo.**
Propósito	Recibir el audio y el vídeo enviado por otro cliente.
Actores	Servidor.
Resumen	El caso de uso se inicia cuando el servidor re direcciona el audio y el vídeo hacia el cliente destino.
Referencias	RF5C
Precondiciones	Estar conectado al Servidor.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Servidor envía el audio y el vídeo recibido hacia el cliente destino.	1.1- Identifica el paquete.
	1.2- Inicializa los componentes para recibir la transmisión de vídeo.
Pos condiciones	Se recibe el audio y el vídeo que es transmitido en tiempo real por otro cliente.

Prioridad	Crítico.

Tabla 17 Descripción del CU5 Recibir Vídeo.

Caso de Uso	
CU6	**Desconectar.**
Propósito	Cerrar la conexión con el servidor.
Actores	Cliente
Resumen	El caso de uso se inicia cuando el Cliente quiere terminar la conexión y llama a la función desconectar.
Referencias	RF6C, RF6.1C
Precondiciones	Estar conectado al Servidor.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Cliente llama a la función desconectar Cliente.	1.1 Manda un paquete al servidor para avisar de que va a cerrar la conexión.

	1.2 Envía el paquete al servidor y cierra el puerto de la conexión.
Pos condiciones	Se cierra la conexión con el Servidor.
Prioridad	Crítico.

Tabla 18 Descripción del CU6 Desconectar.

- ✓ **SERVIDOR**

Caso de Uso	
CU1	**Inicializar Servidor.**
Propósito	Abrir correctamente el puerto para brindar servicio a los clientes, ponerse en línea (*online*).
Actores	Administrador.
Resumen	El caso uso se inicia cuando el Administrador selecciona la opción de poner en línea el Servidor.
Referencias	RF1S,

Precondiciones	**-**
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador selecciona la opción de poner en línea el Servidor.	1.1- Abre el puerto usado para la conexión Correctamente.
	1.2- Cambia su estado a *online* (en línea).
	1.1- Comienza a mostrar las acciones de los clientes conectados.
Flujo Alterno 1	
	1.1 No se puede abrir el puerto para la conexión.
	1.2 Informa error al iniciar el servidor.
Pos condiciones	Se inician los servicios del Servidor.
Prioridad	Crítico.

Tabla 19 Descripción del CU1 Inicializar Servidor.

Caso de Uso	
CU2	**Desconectar.**
Propósito	Terminar los servicios del servidor.
Actores	Administrador.
Resumen	El caso uso se inicia cuando el Administrador selecciona la opción Cerrar del Servidor.
Referencias	RF6S
Precondiciones	Estar *online* (en línea).

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador selecciona la opción cerrar del Servidor.	1.1 Se informa a los clientes que el servidor va a finalizar los servicios.
	1.2- Cambia su estado a *offline* (fuera de línea) y se cierra el servidor.

Pos condiciones	Se finalizan los servidos del Servidor.
Prioridad	Crítico.

Tabla 20 Descripción del CU2 Desconectar.

3.7 Conclusiones del Capítulo.

En este capítulo se definieron las funcionalidades del sistema a partir de la recogida de los requisitos funcionales, a los que el módulo debe responder para dar total cumplimiento a los objetivos propuestos. Se dividió la estructura del módulo en dos subsistemas funcionales, con el objetivo de encapsular las responsabilidades del cliente y el servidor, y de esta forma independizar su desarrollo.

CAPÍTULO 4: DISEÑO E IMPLEMENTACIÓN.

4.1 Introducción.

En el capítulo anterior se hizo una captura de los requisitos funcionales y no funcionales para el módulo a desarrollar, también se ha tenido un acercamiento a las características principales del sistema y el agrupamiento de sus funcionalidades en dos subsistemas. En este capítulo se traducirán esos requisitos del módulo en especificaciones que hagan una descripción de cómo se pueden implementar cada una de estas funcionalidades, estas especificaciones constituyen el diseño del sistema. Luego de tener este diseño se hará el paso de las clases del diseño a componentes físicos, al utilizar el lenguaje C++ estos componentes físicos serán ficheros .h y .cpp. También se elabora el diagrama de despliegue para el módulo, presentando la ubicación física de los componentes del sistema.

4.2 Diagrama de paquetes del diseño.

A continuación se muestra el diagrama de paquetes donde se ve la relación entre las bibliotecas que usamos para el desarrollo de nuestro módulo y la dependencia entre ellas, así como la relación de todas con el paquete módulo, donde están representadas la clases que van a ser implementadas para la transmisión de audio y vídeo.

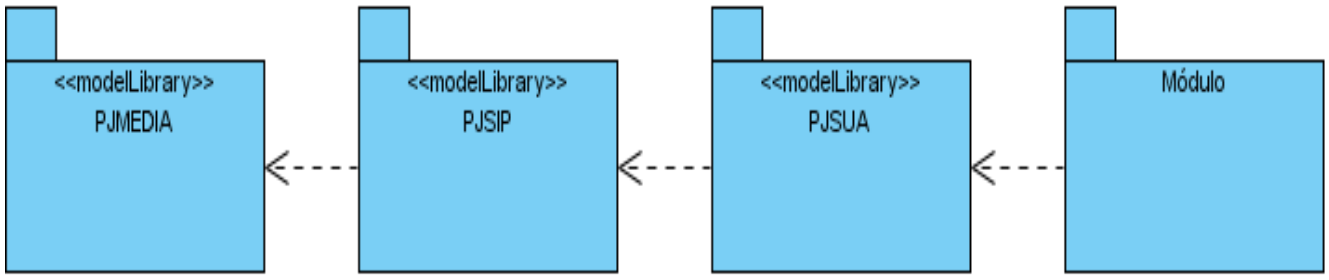


Figura 10 Diagrama de paquetes del diseño.

4.3 Diagrama de Clases del Diseño.

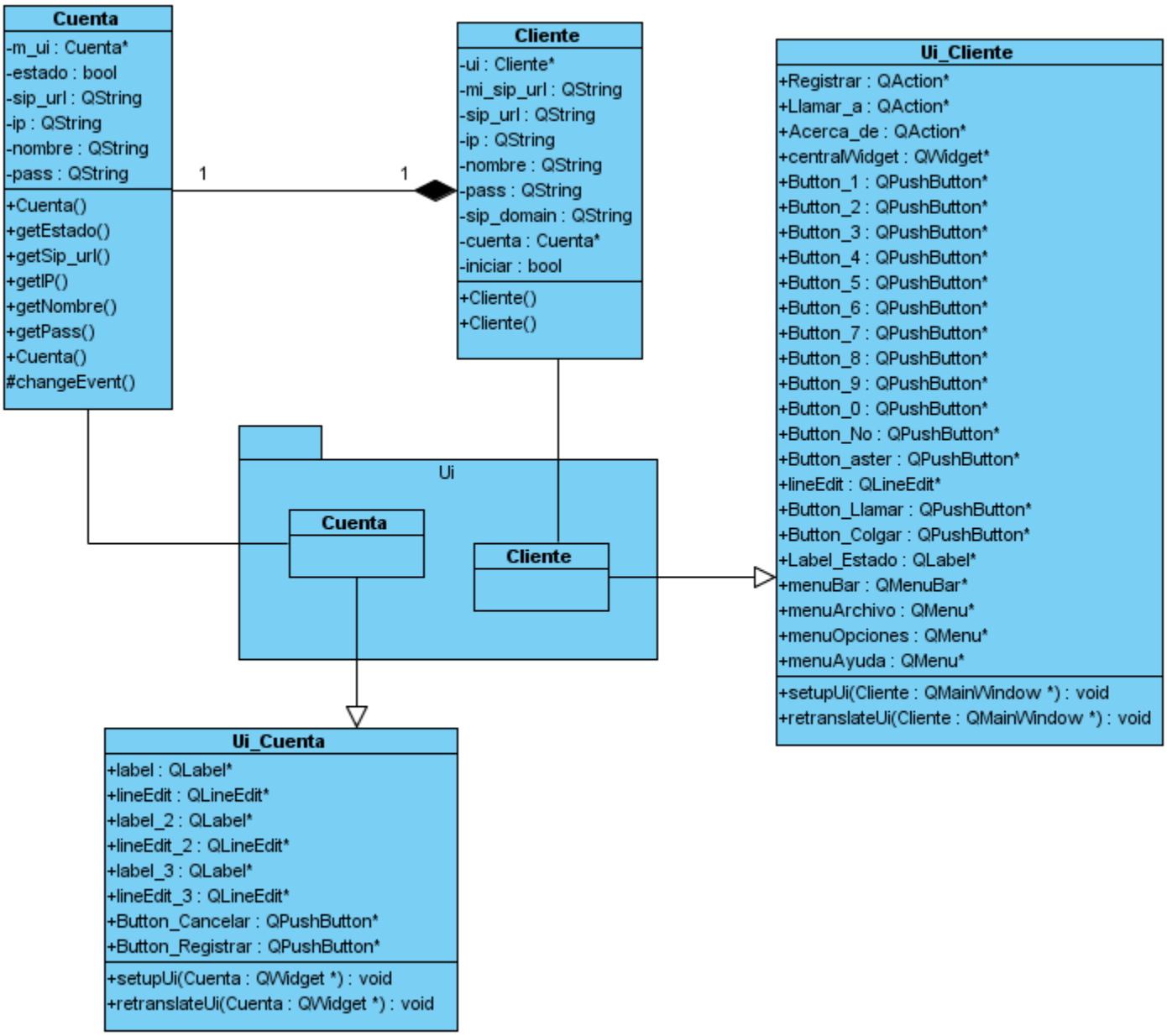


Figura 11 Diagrama de Clase para el trabajo con el audio.

4.4 Descripción de las Clases del diseño.

Nombre: Audio	
Tipo de clase: Controladora	
Atributo	**Tipo**
SIP_DOMAIN	String
SIP_USER	String
SIP_PASSWORD	String
Para cada responsabilidad:	
Nombre:	static void on_incoming_call(pjsua_acc_id acc_id,pjsua_call_id call_id, pjsip_rx_data *rdata)
Descripción:	Este método se ejecuta cuando se recibe una llamada, envía señales de respuesta indicando que se ha recibido la llamada.
Nombre:	static void on_call_state(pjsua_call_id call_id, pjsip_event *e)
Descripción:	Este método se ejecuta cuando cambia el estado de una llamada.

Nombre:	static void on_call_media_state(pjsua_call_id call_id)
Descripción:	Este método se ejecuta cuando cambia el estado de la transferencia de la llamada.
Nombre:	static void error_exit(const char *title, pj_status_t status)
Descripción:	Se muestra un error y se sale de la aplicación.
Nombre:	int main(int argc, char *argv[1])
Descripción:	En este método se capturan los datos entrados por el usuario, se crean las conexiones UDP y son establecidas las mismas, se registra al usuario en el servidor Asterisk y se tratan las excepciones.

Tabla 21 Descripción de la Clase Cliente.

4.5 Diagramas de Secuencia.

- ✓ **SUBSISTEMA CLIENTE.**

Los diagramas de secuencia permiten tener una visión más clara de las funcionalidades del sistema y de las secuencias de acciones realizadas en cada caso de uso. Los casos de uso del Cliente tienen acciones en común, por tanto, se ha decidido mostrar en este documento dos de ellos que representen la funcionalidad del sistema y la secuencia de pasos seguidos para cumplir los pedidos de la Aplicación Usuario. El resto de los diagramas de secuencia se encuentran en el anexo “Diagramas de Secuencia”, para ser consultados en caso de interés.

Caso de Uso “Conectar Cliente”

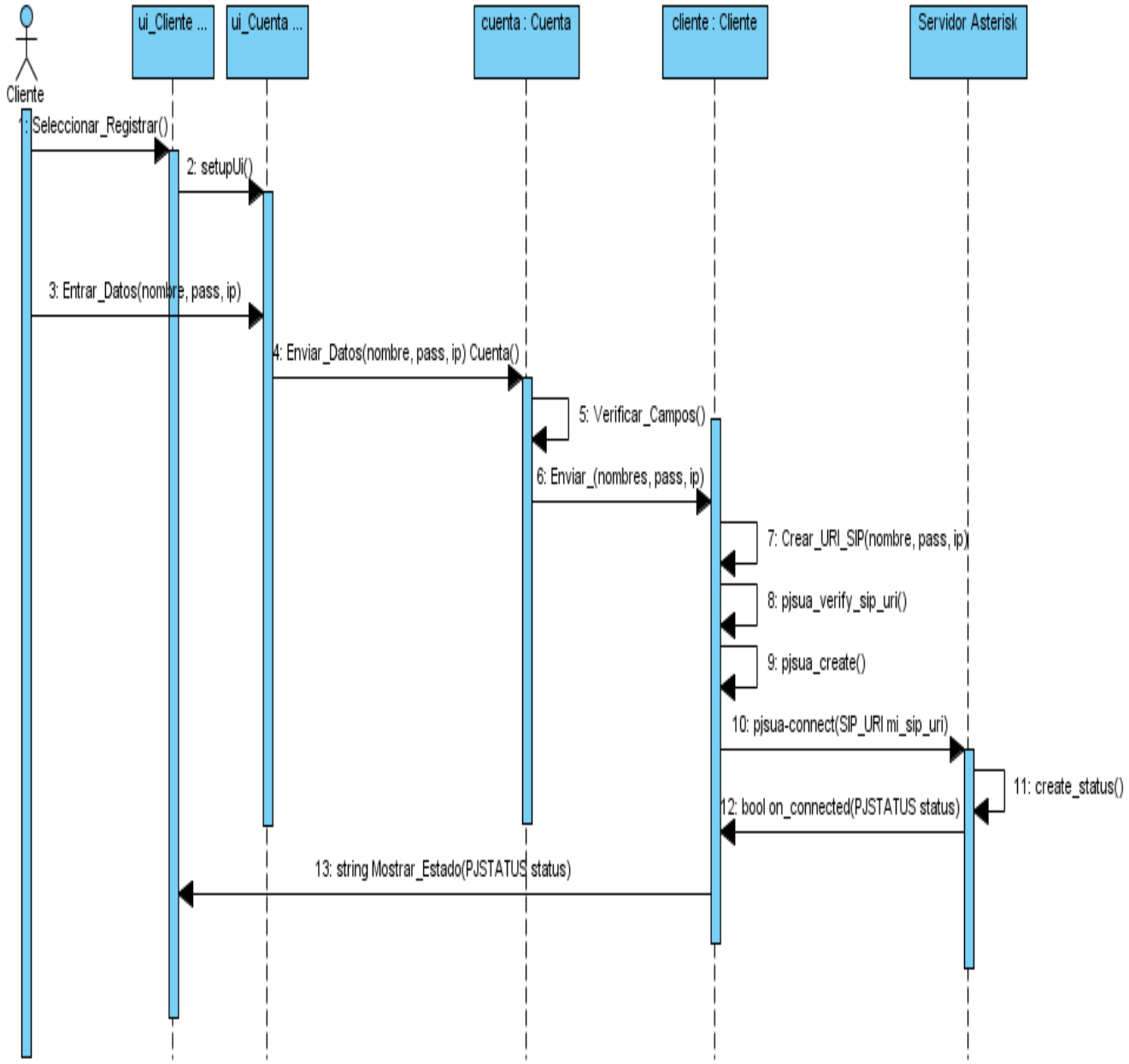


Figura 12 Diagrama de Secuencia, Caso de Uso Conectar Cliente.

Caso de Uso “Enviar Audio”

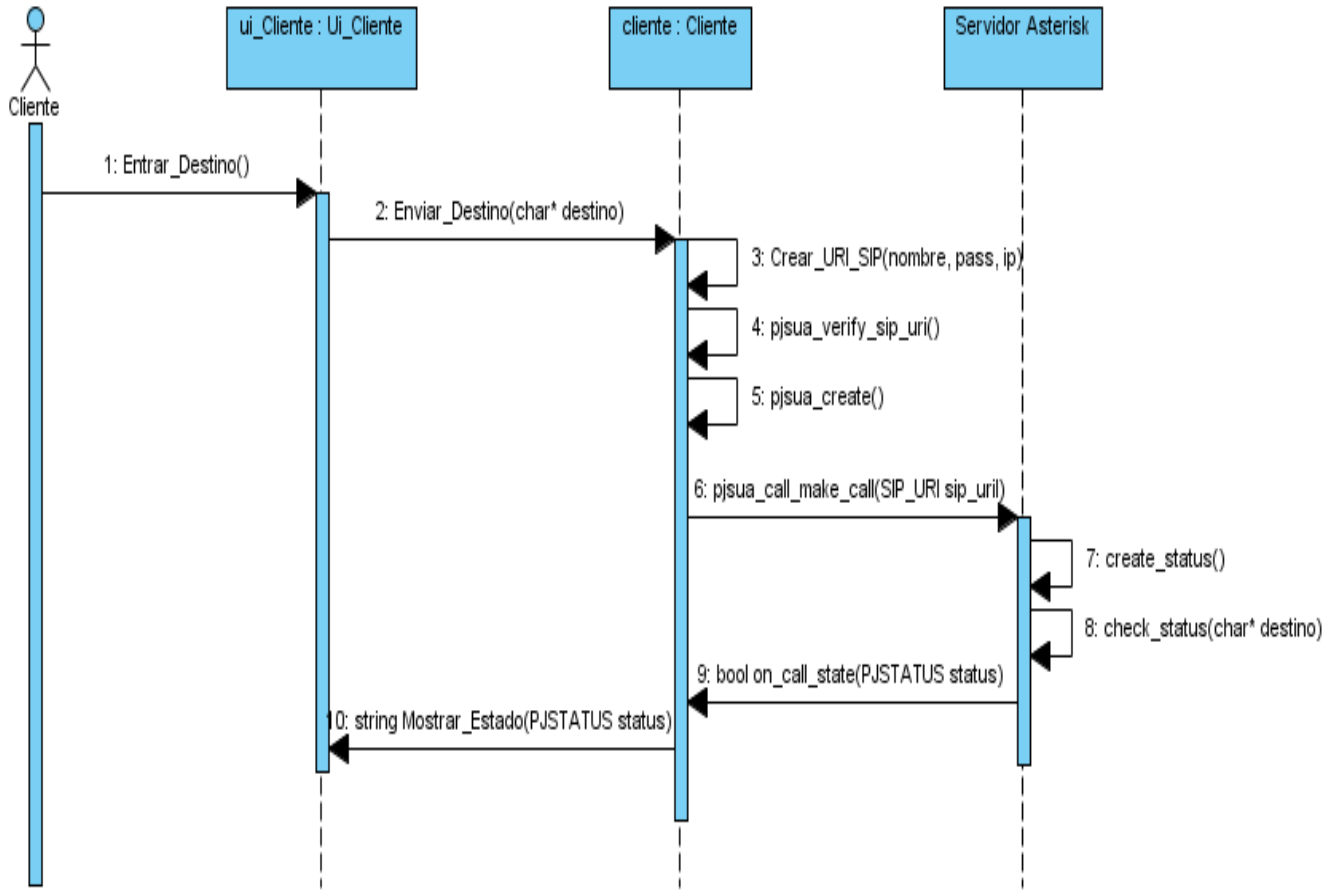


Figura 13 Diagrama de Secuencia, Caso de Uso Enviar Audio.

- ✓ **SUBSISTEMA SERVIDOR.**

Caso de Uso “Desconectar”

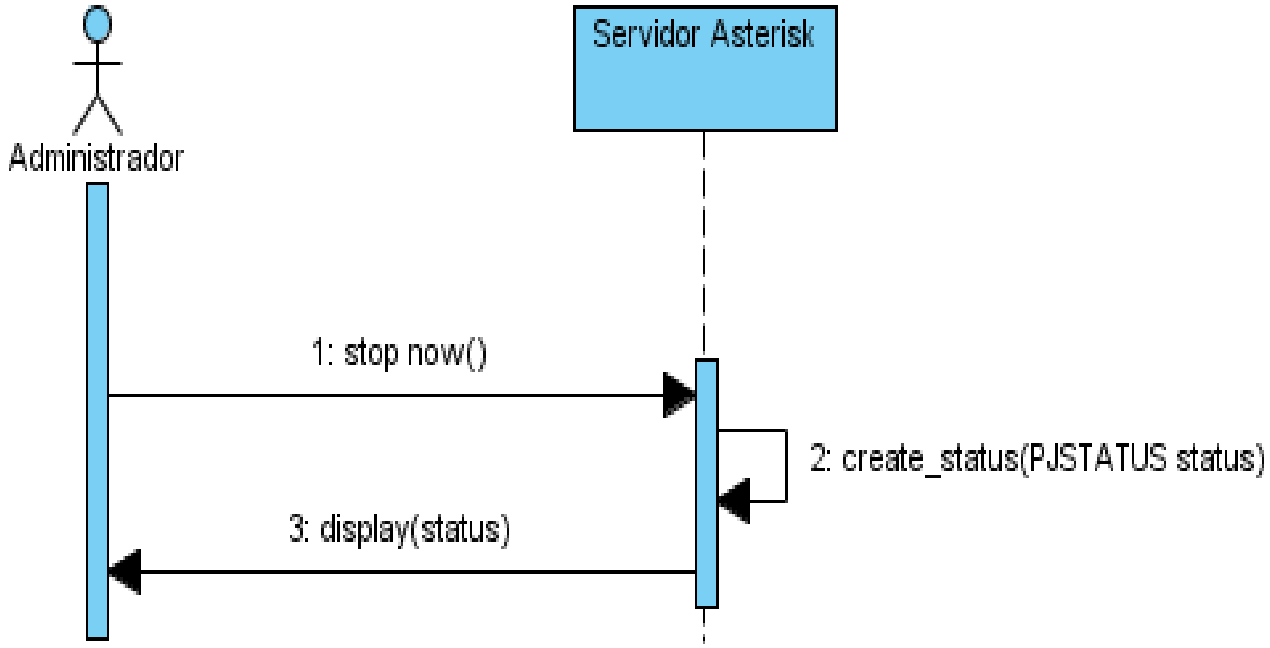


Figura 14 Diagrama de Secuencia, Caso de Uso Desconectar.

4.6 Diagrama de componentes.

A continuación, mostramos los diagramas de componentes de los paquetes de nuestros subsistemas Cliente y Servidor, con el objetivo de tener una idea de los componentes que los conforman y las dependencias entre estos. Los diagramas mostrados a continuación incluyen las bibliotecas y los archivos de nuestras clases.

- ✓ Diagrama de componentes subsistema Cliente

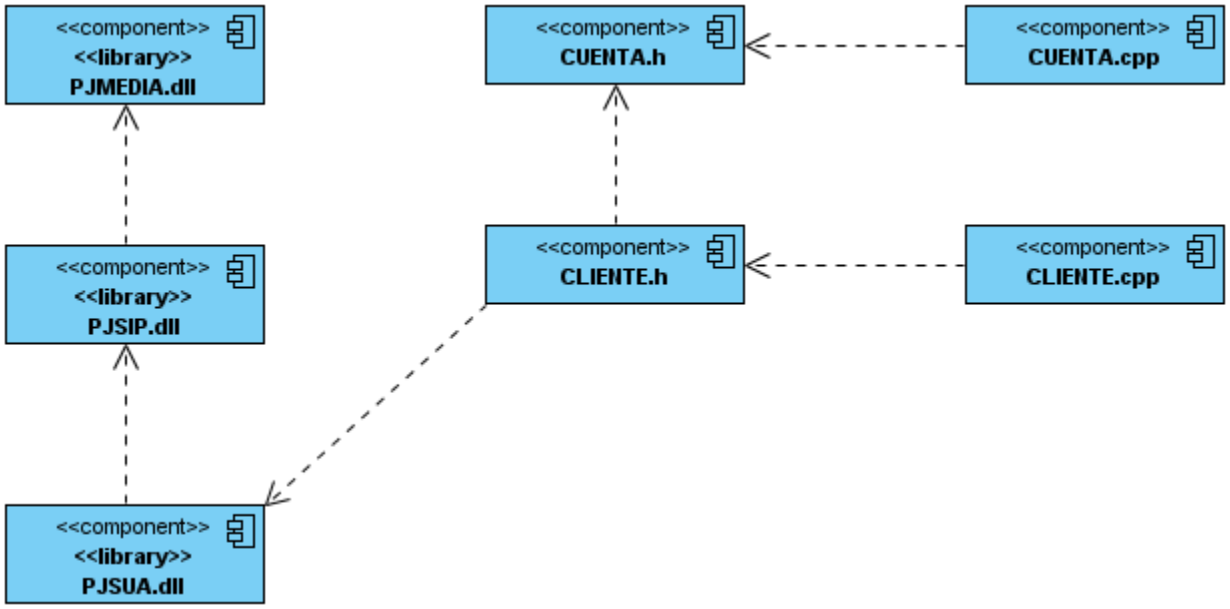


Figura 15 Diagrama de componente del subsistema Cliente.

4.7 Diagrama de despliegue.

El diagrama de despliegue para este sistema incluye la representación de una PC que realiza la función de servidor que es donde se instalará el Asterisk y una o más PCs Cliente conectadas al Servidor por medio del protocolo UDP, los clientes no se comunican directamente entre sí cuando van a transmitir audio o vídeo, ya que el servidor se encarga de re direccionar los paquetes.

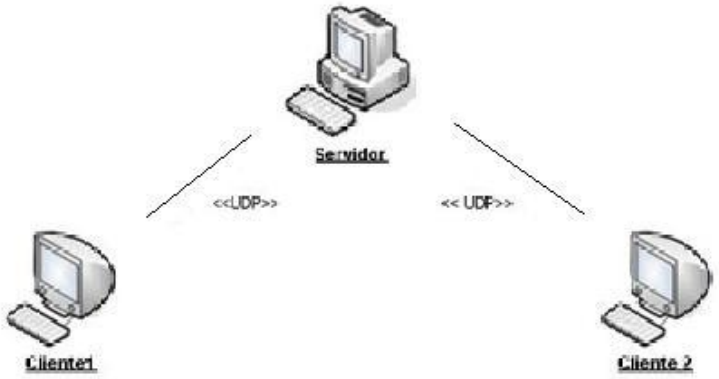


Figura 16 Diagrama de Despliegue.

4.8 Reglas de codificación.

A continuación se especifica el estándar de codificación a utilizar para el desarrollo del módulo. Los atributos y métodos serán escritos en español. Además, es importante especificar que los atributos y métodos usados que pertenezcan a alguna de las bibliotecas utilizadas, seguirán el mismo estándar de codificación usado por los desarrolladores de dicha biblioteca.

- ✓ **Nombre de los ficheros**

Regla:

Se nombrarán los ficheros .h y .cpp de la siguiente manera:

Cuenta .h

- ✓ **Clases**

Regla:

Las clases se nombrarán de la siguiente manera:

class Cliente

- ✓ **Constantes**

Regla:

Los nombres de constantes se escriben con letras en mayúscula. En caso de estar compuesta por más de una palabra, éstas se separan por un carácter "_".

#define PORT 5060

#define SIP_USER "3001"

- ✓ **Variables**

Regla:

Las variables se escriben con minúsculas y las variables de la clase Cliente enpezarán con “c_”, las variables con nombres compuestos serán separadas por un carácter “_”

static char* c_sip_domain;

char* c_ip;

- ✓ **Parámetros (argumentos) de métodos**

Regla:

Los parámetros se escriben con letras minúsculas y sin ningún carácter especial.

Cliente::Cliente(QWidget* parent)

- ✓ **Métodos miembros de clases**

Regla:

Comienzan con mayúscula y si son compuestos son unidos comenzando con mayúscula cada nueva palabra:

void Cliente::Llamar()

- ✓ **Comentarios**

Regla*:*

Se usan los comentarios en línea para facilitar la comprensión del código, sobre todo en procedimientos complejos, para comentarios con más de una línea se usan los caracteres “/*” y “*/”:

```
// Esto es un comentario.

/*  Comentario

*   de más de

*/  una línea
```

4.9 Conclusiones del Capítulo.

Al terminar este capítulo queda definido el diseño completo del módulo de red. En el diagrama de clases del diseño se describió la estructura del subsistema cliente que conforma el módulo. Además, fue mostrada la interacción del conjunto de objetos que participan en el desarrollo del subsistema Cliente con los diagramas de secuencia correspondientes. A partir del diseño de clases se crearon los componentes físicos que se traducen en ficheros .h y .cpp correspondientes a la implementación en C++.

CONCLUSIONES

Con el desarrollo de este trabajo se mejoró el módulo de red que se va a acoplar a las prácticas de laboratorio virtuales del proyecto PROLAVI. El módulo cumple con las necesidades actuales del proyecto, asegurando los procesos de conexión y transmisión de sonido en tiempo real entre sus usuarios. Para el desarrollo del módulo se utilizó la arquitectura Cliente-Servidor, definiendo y encapsulando las responsabilidades en un subsistema Cliente que se encargará de interactuar con el servidor Asterisk, para este subsistema se llevó a cabo un proceso de desarrollo obteniéndose resultados tangibles. El resultado obtenido del desarrollo del subsistema Servidor fue un servidor funcional basado en Asterisk, el cual se ejecuta en una computadora, una vez que esté corriendo atiende las peticiones de conexión y procesa todos los paquetes de los clientes. Al desarrollar el subsistema Cliente se obtuvieron un conjunto de clases que pueden ser utilizadas para la transmisión de audio por una aplicación. Para mostrar un ejemplo de cómo una aplicación puede utilizar las clases de este subsistema se elaboró una aplicación sencilla donde se muestra las funcionalidades que se brindan y el uso de la interfaz que provee el subsistema. Para la transmisión de audio se utilizó la biblioteca PJSUA, esta permite inicializar componentes con los cuales se pueden manipular la tarjeta de sonido de la computadora para la reproducción o entrada de audio. Las funciones de transmisión de audio del módulo han sido probadas produciéndose la entrada del audio desde un micrófono, de una computadora a otra.

Al acoplar el módulo desarrollado a las prácticas de laboratorio PROLAVI se asegura la comunicación de los usuarios involucrados en prácticas de laboratorio conjuntas por lo que el objetivo principal de este trabajo se ha cumplido parcialmente, al no desarrollar las funcionalidades relacionadas con el vídeo.

RECOMENDACIONES

- ✓ Realizar un estudio profundo del PBX Asterisk con el fin de incluir más funcionalidades al módulo de audio para darle un mayor número de opciones a los usuarios.
- ✓ Hacer un estudio profundo de las bibliotecas utilizadas para desarrollar las funcionalidades del módulo relacionado con la transmisión de vídeo en tiempo real.
- ✓ Integrar el módulo con el desarrollado anteriormente y agregarle las funcionalidades de vídeo para lograr un sistema de transmisión de datos que incluya las tres funcionalidades.

BIBLIOGRAFIA

1. **Corp., Microsoft.** MSDN Virtual Labs. [En línea] 2010. [Citado el: 18 de Enero de 2010.] http://msdn.microsoft.com/en-us/aa570323.aspx.

2. **Universidad de Univalle, Cali, Colombia.** Servicio de Videoconferencia . [En línea] 2009. [Citado el: 18 de Enero de 2010.] http://dintev.univalle.edu.co/info_general/vi_conferencia.htm.

3. **Unicorn, Electrónica.** VoIP (Voz sobre IP). [En línea] 2008. [Citado el: 20 de Enero de 2010.] http://www.unicrom.com/Tel_VoIP.asp.

4. **Foro, VoIP.** Protocolos VoIP. [En línea] 2010. [Citado el: 15 de Mayo de 2010.] http://www.voipforo.com/.

5. **CENIP.** Que es el protocolo SIP. [En línea] 2007. [Citado el: 20 de Enero de 2010.] http://www.cenip.com.ar/index.php?option=com_content&task=view&id=18&Itemid=25.

6. **Commons, Scientic.** RTP: A Transport Protocol for Real-Time Applications. [En línea] 2009. [Citado el: 7 de Febrero de 2010.] http://en.scientificcommons.org/42880116.

7. **J Van Meggelen, J Smith, L Madsen.** Asterisk: the future of telephony. [En línea] 2007. [Citado el: 18 de Febrero de 2010.] http://www.google.com/books?hl=es&lr=&id=vtQxJ3oSm64C&oi=fnd&pg=PR11&dq=asterisk&ots=LV0dC-Hj35&sig=R1iKDyDTWo-qneHqgX41r-S4vTA#v=onepage&q&f=false.

8. **HAWKSOFT.** Hawk Network Library. [En línea] 2005. [Citado el: 5 de Marzo de 2010.] http://www.hawksoft.com/hawknl/.

9. **Software, JENKINS.** Manual Raknet. [En línea] 2008. [Citado el: 9 de Marzo de 2010.] http://www.jenkinssoftware.com/raknet/manual/index.html.

10. **DATAREEL.** Open Source Home Page. [En línea] 2008. [Citado el: 12 de Marzo de 2010.] http://www.datareel.com/.

11. **SOFTPEDIA.** Libtcp++ 0.1.2. [En línea] 2006. [Citado el: 15 de Marzo de 2010.] http://linux.softpedia.com/get/Programming/Libraries/libtcpplusplus-12490.shtml.

12. **ZOIDCOM.** Zoidcom network library. [En línea] 2006. [Citado el: 20 de Febrero de 2010.] http://www.zoidcom.com/.

13. **Prijono, Benny.** PJSIP - Open Source SIP Stack. [En línea] 2010. [Citado el: 10 de Junio de 2010.] http://www.pjsip.org/pjsip/docs/html/index.htm.

14. —. PJMEDIA and PJMEDIA-CODEC. [En línea] 2010. [Citado el: 10 de Junio de 2010.] http://www.pjsip.org/pjmedia/docs/html/index.html.

15. —. PJSUA Manual Page. [En línea] 2010. [Citado el: 10 de Junio de 2010.] http://www.pjsip.org/pjsua.htm.

16. **Garin, Hiebert.** OpenAL Programmer's Guide. [En línea] 2007. [Citado el: 14 de Marzo de 2010.] http://worldspace.berlios.de/openal/tut_0/openal00.html.

17. **ACM.** On the use of RTP for monitoring and fault isolation in IPTV. [En línea] 2008. [Citado el: 28 de Mayo de 2010.] http://portal.acm.org/citation.cfm?id=1820599.

18. **Ohrn, Tim Taylor & Marina.** Gestión de Redes para la red de distribución. *Revista ABB.* [En línea] 2009. [Citado el: 22 de Abril de 2010.] http://library.abb.com/global/scot/scot271.nsf/veritydisplay/b64f7da9efbba619c125762c002f00cc/$File/45-49%203M901_SPA72dpi.pdf.

19. **Baena, Javier Martínez.** Introducción a SDL. [En línea] 2007. [Citado el: 19 de Abril de 2010.] http://decsai.ugr.es/~jfv/intro_sdl.pdf.

20. **Herreros, Mariano Cebrián.** *La Web 2.0 como red social de comunicación e información.* s.l. : UNO, 2008. ISSN 1134-1629.

21. **Viscaya Guarín, Pedro & Ayala Pañuela, Omar.** Transmisión de Secuencias Codificadas de Telefonía Visual Usando RTP. [En línea] Pontificia Universidad Javeriana, Bogotá, Colombia, 2004. [Citado el: 13 de Mayo de 2010.]

22. **Herrera Pérez, Enrique.** *Tecnologías y redes de transmisión de datos.* s.l. : Editorial Limusa, 2003.

23. **Jasmin, Blanchette.** *C++ GUI Programming with Qt 4.* s.l. : Editorial Prentice Hall, 2006.

24. **Duchi Gavaldà, Jordi y Tejedor Navarro, Heliodoro.** *Lógica de videojuegos.* Barcelona : s.n., 2008.

25. **James, Jacobson Ivar & Booch Grady & Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* Santa Clara, California : s.n., 1999.

26. **Daniel., Molkentin.** *The Art of Building Qt Aplications.* San Francisco, EEUU : s.n., 2007.

27. **Christopher, Negus.** *Linux Bible.* Indianapolis, Indiana : Wiley Publishing, 2005.

28. **William, Stallings.** *Comunicaciones y redes de Computadora.* s.l. : Editorial Prentice Hall, 2000.

29. **Lavoisier, Jacques.** Speech quality of VoIP: Assessment & Prediction. [En línea] 2009. [Citado el: 30 de Abril de 2010.] http://www.lavoisier.fr/notice/fr332309.html.

30. **Goncalves, FE.** Configuration Guide for Asterisk PBX. [En línea] 2007. [Citado el: 15 de Junio de 2010.] http://www.google.com/books?hl=es&lr=&id=dr3ZZmB8N-sC&oi=fnd&pg=PA33&dq=asterisk+PBX&ots=wbq43OsDxk&sig=cT_KYPv68py6IhlbXU_IPbM9nJA#v=onepage&q&f=false.

REFERENCIAS BIBLIOGRAFICAS

1. **Corp., Microsoft.** MSDN Virtual Labs. [En línea] 2010. [Citado el: 18 de Enero de 2010.] http://msdn.microsoft.com/en-us/aa570323.aspx.

2. **Universidad de Univalle, Cali, Colombia.** Servicio de Videoconferencia. [En línea] 2009. [Citado el: 18 de Enero de 2010.] http://dintev.univalle.edu.co/info_general/vi_conferencia.htm.

3. **Unicorn, Electrónica.** VoIP (Voz sobre IP). [En línea] 2008. [Citado el: 20 de Enero de 2010.] http://www.unicrom.com/Tel_VoIP.asp.

4. **Foro, VoIP.** Protocolos VoIP. [En línea] 2010. [Citado el: 15 de Mayo de 2010.] http://www.voipforo.com/.

5. **CENIP.** Que es el protocolo SIP. [En línea] 2007. [Citado el: 20 de Enero de 2010.] http://www.cenip.com.ar/index.php?option=com_content&task=view&id=18&Itemid=25.

6. **Commons, Scientic.** RTP: A Transport Protocol for Real-Time Applications. [En línea] 2009. [Citado el: 7 de Febrero de 2010.] http://en.scientificcommons.org/42880116.

7. **ACM.** On the use of RTP for monitoring and fault isolation in IPTV. [En línea] 2008. [Citado el: 28 de Mayo de 2010.] http://portal.acm.org/citation.cfm?id=1820599.

8. **Herrera Pérez, Enrique.** *Tecnologías y redes de transmisión de datos.* s.l. : Editorial Limusa, 2003.

9. **HAWKSOFT.** Hawk Network Library. [En línea] 2005. [Citado el: 5 de Marzo de 2010.] http://www.hawksoft.com/hawknl/.

10. **Software, JENKINS.** Manual Raknet. [En línea] 2008. [Citado el: 9 de Marzo de 2010.] http://www.jenkinssoftware.com/raknet/manual/index.html.

11. **DATAREEL.** Open Source Home Page. [En línea] 2008. [Citado el: 12 de Marzo de 2010.] http://www.datareel.com/.

12. **SOFTPEDIA.** Libtcp++ 0.1.2. [En línea] 2006. [Citado el: 15 de Marzo de 2010.] http://linux.softpedia.com/get/Programming/Libraries/libtcpplusplus-12490.shtml.

13. **ZOIDCOM.** Zoidcom network library. [En línea] 2006. [Citado el: 20 de Febrero de 2010.] http://www.zoidcom.com/.

14. **Prijono, Benny.** PJSIP - Open Source SIP Stack. [En línea] 2010. [Citado el: 10 de Junio de 2010.] http://www.pjsip.org/pjsip/docs/html/index.htm.

15. —. PJMEDIA and PJMEDIA-CODEC. [En línea] 2010. [Citado el: 10 de Junio de 2010.] http://www.pjsip.org/pjmedia/docs/html/index.html.

16. —. PJSUA Manual Page. [En línea] 2010. [Citado el: 10 de Junio de 2010.] http://www.pjsip.org/pjsua.htm.

17. **Garin, Hiebert.** OpenAL Programmer's Guide. [En línea] 2007. [Citado el: 14 de Marzo de 2010.] http://worldspace.berlios.de/openal/tut_0/openal00.html.

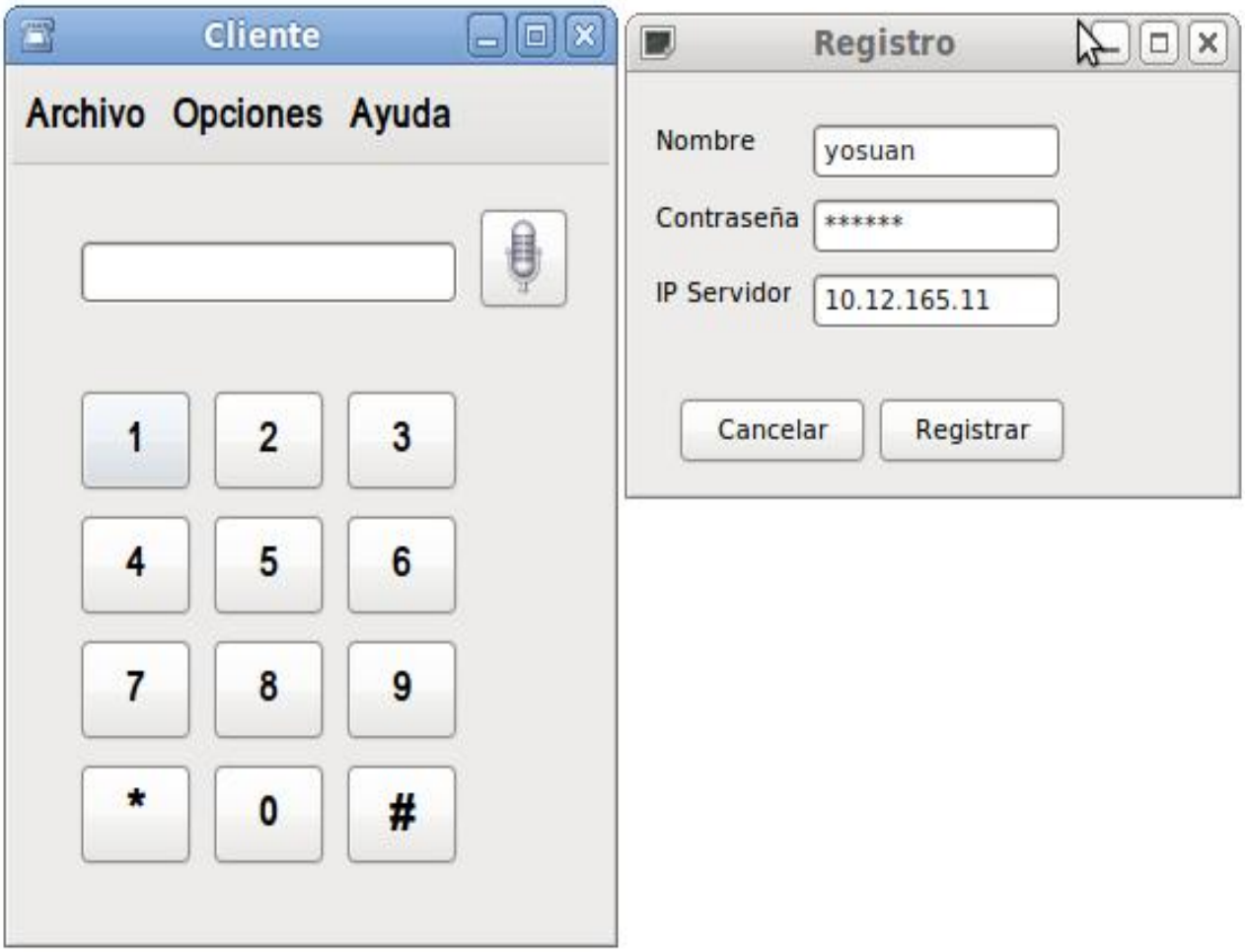
18. **J Van Meggelen, J Smith, L Madsen.** Asterisk: the future of telephony. [En línea] 2007. [Citado el: 18 de Febrero de 2010.] http://www.google.com/books?hl=es&lr=&id=vtQxJ3oSm64C&oi=fnd&pg=PR11&dq=asterisk&ots=LV0dC-Hj35&sig=R1iKDyDTWo-qneHqgX41r-S4vTA#v=onepage&q&f=false.

19. **Herreros, Mariano Cebrián.** *La Web 2.0 como red social de comunicación e información.* s.l. : UNO, 2008. ISSN 1134-1629.

20. **Baena, Javier Martínez.** Introducción a SDL. [En línea] 2007. [Citado el: 19 de Abril de 2010.] http://decsai.ugr.es/~jfv/intro_sdl.pdf.

ANEXOS

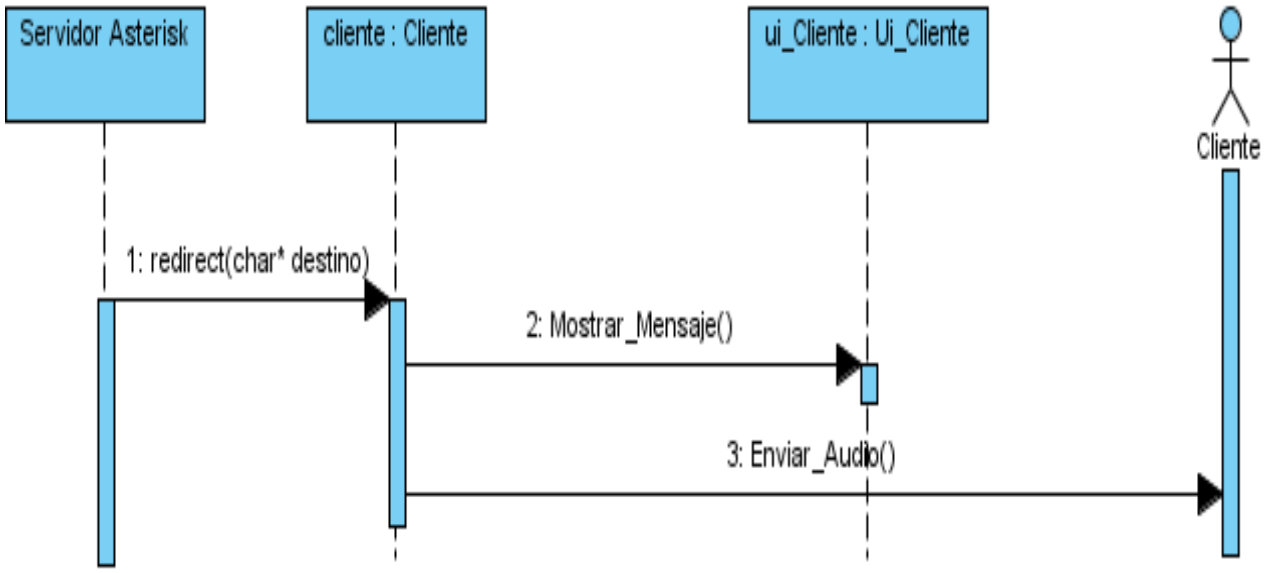
Proceso de registro:



Proceso de llamada:

Diagrama de Secuencias:

CU Recibir Audio del subsistema Cliente:



CU Desconectar del subsistema Cliente: