

Universidad de las Ciencias Informáticas

Facultad 5



***SISTEMA DE VISUALIZACIÓN REMOTA
PARA EL PROYECTO VISMEDIC.***

**Trabajo de diploma para optar por el título de
Ingeniero en Ciencias Informáticas.**

Autor: Leitniz Pérez Buján

Tutor: MSc. Osvaldo Pereira Barzaga

Co-Tutores: Ing. Rubén Alcolea Nuñez

Ing. Luis G. Silva Rojas

Junio 2012

"Dicen que los pesimistas ven el vaso medio vacío; los optimistas, en cambio, lo ven medio lleno. Los ingenieros, por supuesto, ven que el vaso es el doble de grande de lo que sería necesario"

Bob Lewis

DECLARACIÓN DE AUTORÍA

Declaramos ser los 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 firmamos la presente a los _____ días del mes de _____ del año _____.

Leitniz Pérez Buján

Autor

MSc. Osvaldo Pereira Barzaga

Tutor

Ing. Rubén Alcolea Nuñez

Ing. Luis Guillermo Silva Rojas

Co-Tutores

DATOS DE CONTACTO

Tutor:



Nombre: Osvaldo Pereira Barzaga.

Institución: Universidad de las Ciencias Informáticas.

Título: MSc. en Informática Aplicada.

Categoría Docente: Profesor Instructor.

Correo electrónico: opereira@uci.cu

Graduado de la UCI, con cinco años de experiencia en el tema de la Gráfica Computacional y líder de un proyecto de Realidad Virtual en la Universidad de las Ciencias Informáticas.

Co-Tutores:



Nombre: Rubén Alcolea Nuñez.

Institución: Universidad de las Ciencias Informáticas.

Título: Ingeniero en Ciencia Informática.

Categoría Docente: Recién Graduado.

Correo electrónico: ralcolea@uci.cu

Graduado de la UCI, con un año de experiencia en el tema de la Gráfica Computacional y profesor de un proyecto de Realidad Virtual en la Universidad de las Ciencias Informáticas.



Nombre: Luis Guillermo Silva Rojas.

Institución: Universidad de las Ciencias Informáticas.

Título: Ingeniero en Ciencia Informática.

Categoría Docente: Recién Graduado.

Correo electrónico: lgsilva@uci.cu

Graduado de la UCI, con un año de experiencia en el tema de la Gráfica Computacional y profesor de un proyecto de Realidad Virtual en la Universidad de las Ciencias Informáticas.

AGRADECIMIENTOS

A mi gran familia...

*Gracias a **mi familia en Cienfuegos**, siempre pendiente de mí, por todos los años juntos, por el amor incondicional. A mis padres por siempre estar a mi lado en todo momento, por darme apoyo y amor. Son todo para mí, por ellos es que me supero cada día, soy mejor persona, los admiro, los amo.*

*Gracias a **mi familia en La Habana**, que me apoyado en toda la carrera, me han dado amor y me han acogido siempre.*

*Gracias a **mi familia en Ciego de Ávila**, pocos momentos de mi vida que hemos compartido pero siempre han estado pendientes de mí y de una forma u otra me han dado su cariño.*

*Gracias a **mi familia en la UCI**, esa tropa de amigos y compañeros que hemos compartido, coincidido, interpuestos en el camino en diferentes años de la carrera y los que venían ya de antes: en los estudios, en la beca, en la misión; a todos les estoy agradecido.*

*Gracias a **mi familia en Venezuela**, esos nuevos amigos, compañeros y mi novia en especial. Allí compartimos alegrías, trabajos, tristezas, me hice mejor persona, me hice profesional y encontré el más grande amor.*

DEDICATORIA

A mis padres.

Porque fueron mis primeros maestros, mis guías, mis educadores, quiénes me enseñan el camino y lo recorren conmigo. Si cada día soy más profesional es para que se sientan orgullosos de mí y sepan que no les he fallado, nunca lo haré.

A mi hermano.

Porque todos los niños tienen un héroe y el mío es mi hermano; siempre he admirado sus logros y he seguido sus pasos en lo profesional y familiar.

A mi abuela.

Porque tengo la dicha de tener a mis padres a mi lado pero también siempre he tenido a mi abuela y siempre he querido que se sienta orgullosa de mí y que vea todo lo que he logrado en mi vida, donde ha aportado su granito, su amor incondicional.

A mi familia, mis amistades, mi novia.

Porque han estado ahí cuando se necesita de su apoyo, compañía, amor.

RESUMEN

Las imágenes médicas adquiridas por modalidades de Tomografía Axial Computarizada (TAC) o Resonancia Magnética (RM) son cada vez más usadas como método de diagnóstico no invasivo. La gran cantidad de imágenes que se obtienen del estudio realizado por estas modalidades hacen que los sistemas informáticos para su análisis, procesamiento, transmisión y visualización requieran de computadoras con elevadas prestaciones de hardware. Por otro lado, se necesita además que dichos sistemas brinden la posibilidad a los especialistas médicos de realizar un diagnóstico colaborativo de determinado estudio; aun cuando no se encuentra situado geográficamente en el mismo lugar (tele-diagnóstico). En este trabajo se propone un sistema de visualización remota basado en arquitectura cliente-servidor para su integración al sistema de visualización médica tridimensional Vismedic. El sistema propuesto se implementa bajo protocolo de comunicación para aplicaciones en tiempo real, complementado con el algoritmo de compresión de imágenes implementado. De esta forma se optimiza el uso de los recursos de red y se logra una visualización a distancia de estructuras anatómicas.

Palabras clave:

Compresión, cliente, servidor, red, visualización médica, visualización remota.

ÍNDICE

INTRODUCCIÓN..... 1

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA..... 4

1.1 SISTEMAS DE VISUALIZACIÓN BASADOS EN REPRESENTACIÓN REMOTA..... 4

 1.1.1 *Sistema de representación a distancia: IBRAC..... 4*

 1.1.2 *Protección de gráficos 3D vía representación remota: ScanView..... 5*

 1.1.3 *Sistema de representación remota: RemoteVIS..... 5*

 1.1.4 *Combinación de visualización local y remota: NetVisMed..... 6*

 1.1.5 *Consideraciones del epígrafe..... 6*

1.2 REDES..... 7

 1.2.1 *El Protocolo de Internet (IP)..... 7*

 1.2.2 *Protocolos de nivel superior..... 9*

 1.2.2.1 TCP: Protocolo de Control de Transferencias..... 9

 1.2.2.2 UDP: Protocolo de Datagrama de Usuario..... 10

 1.2.2.3 RTP: Protocolo de Transporte en Tiempo Real..... 10

 1.2.3 *Técnicas de transmisión..... 11*

 1.2.4 *Arquitecturas de comunicación..... 12*

1.3 BIBLIOTECAS DE COMUNICACIÓN POR RED..... 13

 1.3.1 *Biblioteca JRTPLib..... 13*

 1.3.2 *Biblioteca HawkNL..... 14*

 1.3.3 *Biblioteca DataReel..... 14*

 1.3.4 *Biblioteca Libtcp++..... 15*

 1.3.5 *Biblioteca Zoidcom..... 15*

1.4 MÉTODOS DE COMPRESIÓN DE IMÁGENES..... 16

 1.4.1 *Método de compresión RLE..... 17*

 1.4.2 *Método de compresión LZW..... 17*

 1.4.3 *Método de compresión Codificación Huffman..... 17*

 1.4.4 *Método de compresión JPEG sin pérdida..... 18*

1.5 RENDIMIENTO Y RETARDO EN LA RED..... 18

1.6 CONSIDERACIONES GENERALES..... 20

CAPÍTULO 2. SOLUCIÓN TÉCNICA..... 21

2.1 ARQUITECTURA DE COMUNICACIÓN: CLIENTE-SERVIDOR..... 21

2.2	BIBLIOTECA DE COMUNICACIÓN: JRTPLIB.	22
2.3	ALGORITMO DE COMPRESIÓN/DESCOMPRESIÓN: VARIANTE RLE.	23
2.3.1	<i>Flujo del algoritmo de compresión.</i>	24
2.3.2	<i>Flujo de algoritmo de descompresión.</i>	25
2.4	CÁLCULO DEL RETARDO EN LA RED.	25
2.5	METODOLOGÍAS Y HERRAMIENTAS DE DESARROLLO.....	26
2.5.1	<i>Metodología de desarrollo de software.</i>	26
2.5.2	<i>Herramientas de desarrollo.</i>	27
2.5.3	<i>Lenguaje de modelado.</i>	27
2.5.4	<i>Lenguaje de programación.</i>	28
2.6	CONSIDERACIONES GENERALES.	28
CAPÍTULO 3. DESCRIPCIÓN DEL SISTEMA.....		29
3.1	MODELO DE DOMINIO.	29
3.2	CAPTURA DE REQUISITOS.	30
3.2.1	<i>Requisitos funcionales.</i>	30
3.2.1.1	Requisitos funcionales del servidor.	30
3.2.1.2	Requisitos funcionales del cliente.....	31
3.2.2	<i>Requisitos no funcionales.</i>	31
3.2.2.1	Requisitos no funcionales del servidor.	31
3.2.2.2	Requisitos no funcionales del cliente.	32
3.3	MODELO DE CASOS DE USO DEL SISTEMA.	33
3.3.1	<i>Actores del sistema.</i>	33
3.3.1.1	Actores del sistema servidor.....	33
3.3.1.2	Actores del sistema cliente.....	34
3.3.2	<i>Diagrama de casos de uso del sistema.</i>	34
3.3.2.1	Diagrama de casos de uso del sistema servidor.	34
3.3.2.2	Diagrama de casos de uso del sistema cliente.....	35
3.3.3	<i>Descripción de casos de uso del sistema.</i>	35
3.3.3.1	Descripción de casos de uso del sistema servidor.	35
3.3.3.2	Descripción de casos de uso del sistema cliente.	41
3.4	DISEÑO DEL SISTEMA.....	46
3.4.1	<i>Diagrama de clases de diseño del servidor.</i>	47
3.4.2	<i>Diagrama de secuencia de diseño del servidor.</i>	47
3.4.3	<i>Diagrama de clases de diseño del cliente.</i>	49

3.4.4	<i>Diagrama de secuencia de diseño del cliente.</i>	49
3.4.5	<i>Diagrama de despliegue.</i>	50
3.5	CONSIDERACIONES GENERALES.	51
CAPÍTULO 4. IMPLEMENTACIÓN Y VALIDACIÓN DE LOS RESULTADOS.		52
4.1	IMPLEMENTACIÓN.	52
4.1.1	<i>Diagrama de componentes.</i>	52
4.2	VALIDACIÓN DE LOS RESULTADOS.	54
4.2.1	<i>Tasa de compresión.</i>	54
4.2.2	<i>Flujo de información a través de la red.</i>	55
4.2.3	<i>Calidad de presentación de las imágenes.</i>	57
4.3	CONSIDERACIONES GENERALES.	58
CONCLUSIONES		59
RECOMENDACIONES		60
BIBLIOGRAFÍA		61
ANEXOS		63
GLOSARIO		66

ÍNDICE DE FIGURAS

Figura 1: Modelo 4 capas de TCP/IP.	9
Figura 2: Tipo de transmisiones de red.	12
Figura 3: Arquitecturas de comunicación.	13
Figura 4: Comparación entre la compresión con pérdida y sin pérdida.	16
Figura 5: Tipos de retrasos fijos en sistemas de red.	19
Figura 6: Ejecución del algoritmo de compresión sin pérdida. Variante RLE.	24
Figura 7: Ejecución del algoritmo de descompresión sin pérdida. Variante RLE.	25
Figura 8: Modelo de dominio.	29
Figura 9: Diagrama de casos de uso del sistema servidor.	34
Figura 10: Diagrama de casos de uso del sistema cliente.	35
Figura 11: Diagrama de paquetes del sistema.	46
Figura 12: Diagrama de clases de diseño del servidor.	47
Figura 13: Diagrama de secuencia del caso de uso Establecer Conexión.	47
Figura 14: Diagrama de secuencia del caso de uso Interactuar con el Modelo.	48
Figura 15: Diagrama de secuencia del caso de uso Desconectar Servidor.	48
Figura 16: Diagrama de clases de diseño del cliente.	49
Figura 17: Diagrama de secuencia del caso de uso Establecer Conexión con el Servidor.	49
Figura 18: Diagrama de secuencia del caso de uso Interactuar con la Escena.	50
Figura 19: Diagrama de secuencia del caso de uso Desconectar Cliente.	50
Figura 20: Diagrama de despliegue.	51
Figura 21: Diagrama de componentes sistema servidor.	53
Figura 22: Diagrama de componentes sistema cliente.	53
Figura 23: Tráfico de la red en el sistema cliente-servidor.	56
Figura 24: Modelo en el servidor (izquierda) e imagen en el cliente (derecha).	57
Figura 25: IBRAC Java Applet at http://deimos.usc.edu/~jyoon/javaj/ibrac.html	63
Figura 26: Visor cliente ScanView.	64
Figura 27: ScanView. Cliente en baja resolución (izquierda) y servidor en alta resolución (derecha).	64
Figura 28: Sistema de visualización remota: RemoteVis.	65
Figura 29: Sistema de visualización remota: NetVisMed.	65

ÍNDICE DE TABLAS

Tabla 1: Sistemas de visualización remota.	7
Tabla 2: Actores del sistema servidor.	33
Tabla 3: Actores del sistema cliente.	34
Tabla 4: Descripción del caso de uso Establecer Conexión.	35
Tabla 5: Descripción del caso de uso Interactuar con el Modelo.	36
Tabla 6: Descripción de la Sección Rotar hacia Izquierda-Derecha. Caso de uso Interactuar con el Modelo.	37
Tabla 7: Descripción de la Sección Rotar hacia Arriba-Abajo. Caso de uso Interactuar con el Modelo.	37
Tabla 8: Descripción de la Sección Acercar-Alejar. Caso de uso Interactuar con el Modelo.	38
Tabla 9: Descripción del Caso de uso Procesar Petición.	38
Tabla 10: Descripción de la Sección Solicitar Conexión. Caso de uso Procesar Petición.	39
Tabla 11: Descripción de la Sección Ejecutar Comando. Caso de uso Procesar Petición.	39
Tabla 12: Descripción de la Sección Solicitar Desconexión. Caso de uso Procesar Petición.	39
Tabla 13: Descripción del caso de uso Desconectar Servidor.	40
Tabla 14: Descripción del caso de uso Establecer Conexión con el Servidor.	41
Tabla 15: Descripción del caso de uso Procesar Dato Recibido.	41
Tabla 16: Descripción de la Sección Desconectar del Servidor. Caso de uso Procesar Dato Recibido.	42
Tabla 17: Descripción de la Sección Procesar Imagen Recibida. Caso de uso Procesar Dato Recibido.	42
Tabla 18: Descripción de la Sección Procesar Comando Recibido. Caso de uso Procesar Comando Recibido.	43
Tabla 19: Descripción del caso de uso Interactuar con la Escena.	43
Tabla 20: Descripción de la Sección Rotar hacia Izquierda-Derecha. Caso de uso Interactuar con la Escena.	44
Tabla 21: Descripción de la Sección Rotar hacia Arriba-Abajo. Caso de uso Interactuar con la Escena.	44
Tabla 22: Descripción de la Sección Acercar-Alejar. Caso de uso Interactuar con la Escena.	45
Tabla 23: Descripción del caso de uso Desconectar Cliente.	45
Tabla 24: Tasa de compresión de imágenes.	55

INTRODUCCIÓN

El mundo de la informática se encuentra en constante evolución, en especial en los últimos quince años. Cada vez son más las áreas donde la informática juega un papel muy importante e incluso fundamental. Tal auge es consecuencia directa del aumento de la capacidad de los sistemas de computación. Los equipos pueden realizar más tareas y de mayor complejidad. Es por ello que diferentes áreas científicas se han apoyado en las facilidades que brindan las tecnologías de ordenadores para el desarrollo de aplicaciones y sistemas. Con el avance de las ciencias médicas y el uso de las mismas en tecnologías más avanzadas, los sistemas de visualización tridimensional con fines de diagnósticos se han hecho muy populares entre los especialistas de diferentes esferas quirúrgicas.

Como complemento, las herramientas informáticas y los métodos de tratamiento, análisis y visualización de imágenes digitales, han resultado de gran utilidad para el desarrollo de aplicaciones médicas de diversos tipos. Estas aplicaciones se emplean en la medicina para facilitar y permitir un aprovechamiento adecuado de la cuantiosa información disponible sobre las imágenes de los pacientes. Sin embargo, la frecuente integración de aplicaciones de visualización y reconstrucción tridimensional (3D), conjunto con la gran cantidad de imágenes que se pueden obtener del estudio realizado a un paciente; hacen que se requieran de computadoras con elevadas prestaciones de hardware. Por otro lado, los usuarios de estos sistemas demandan de forma creciente que las aplicaciones con funcionalidades de visualización tridimensional, permitan el diagnóstico colaborativo de patologías.

Cuba, inmersa en la producción de software para el mercado nacional e internacional, no está exenta de esta situación. Ejemplo de ello es el desarrollo de aplicaciones en la rama de la visualización médica con fines de diagnóstico, como el proyecto de visualización científica Vismedic en la Universidad de las Ciencias Informáticas (UCI). En aras de lograr productos de alta calidad y competitividad en el mercado, atendiendo a las dificultades que puede representar para Cuba la adquisición de tecnologías a fines con los requerimientos de los sistemas ya mencionados, se trazan estrategias y soluciones que contribuyan a continuar el desarrollo de estos software. El sistema que se desarrolla actualmente en Vismedic, demanda para su puesta en funcionamiento y despliegue, de un módulo de red para visualización remota que permita el trabajo colaborativo de especialistas como aumento de funcionalidades a la aplicación. La situación problemática antes expuesta conduce al siguiente **problema científico**: *¿Cómo lograr la funcionalidad de diagnóstico colaborativo en el sistema Vismedic?* Este problema guiará la realización del presente trabajo de diploma estableciéndose como **objeto de estudio** la *visualización de modelos*

*tridimensionales a través de representación remota y enmarcando el **campo de acción** en la visualización de modelos tridimensionales a través de representación remota basado en la transmisión de imágenes.*

Con el propósito de dar solución al problema propuesto se determina como **objetivo general**: *Desarrollar un sistema de red que permita la visualización remota de modelos tridimensionales.*

En correspondencia con el problema científico planteado se trazaron **tareas investigativas** que contribuyen al cumplimiento del objetivo planteado. A continuación se relacionan las tareas:

1. Elaboración del marco teórico a partir de la bibliografía existente actualmente relacionada con el desarrollo de sistemas de visualización remota para la selección de las técnicas adecuadas para su implementación.
2. Selección de la biblioteca de transmisión de datos que permita el flujo de información por la red.
3. Selección de la técnica de compresión de imágenes más idónea para reducir el volumen de datos de las imágenes procesadas.
4. Descripción del diagrama de diseño de clases, paquetes y sub-paquetes del sistema de transmisión de imágenes por red para un mejor entendimiento entre los procesos clientes y servidor.
5. Implementación de un algoritmo de compresión sin pérdida para disminuir el volumen de datos de las imágenes procesadas.
6. Elaboración de un sistema de red que permita la visualización remota para dotar de la funcionalidad de diagnóstico colaborativo a la aplicación Vismedic.
7. Validación del sistema propuesto para evaluar los resultados del algoritmo de compresión de imágenes y la calidad de presentación de la escena.

La **idea a defender** en el presente trabajo de diploma es que con la implementación de un sistema de red cliente-servidor se logrará la visualización remota de modelos tridimensionales en múltiples estaciones de trabajo. Además se pretende dotar a la aplicación Vismedic de la funcionalidad de trabajo colaborativo.

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. Entre los métodos de trabajo científico utilizados se destacan los siguientes:

Métodos teóricos:

- **Hipotético – Deductivo** para la elaboración de la idea a defender en la investigación.
- **Histórico – lógico** para el estudio crítico de los trabajos anteriores relacionados con el tema.

- **Analítico – Sintético** para descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución de la propuesta.

Métodos empíricos:

- **Consulta de fuentes de información:** Se utiliza para la consulta de las fuentes bibliográficas durante la investigación.
- **Experimental:** Se aplica para comprobar el correcto funcionamiento de la aplicación obtenida.
- **Observación:** Se emplea para constatar los resultados visuales alcanzados y determinar la influencia de estos sobre el rendimiento de la computadora.

El presente Trabajo de Diploma consta de cuatro capítulos:

- **Capítulo 1. Fundamentación Teórica:** En este capítulo se exponen algunos sistemas de visualización a través de representación remota desarrollados en el mundo, para así comprender conceptos fundamentales que sirven de referentes para la programación de tales sistemas. También se caracterizan diferentes bibliotecas de red, así como métodos y arquitecturas de comunicación. Además se describen distintos métodos de compresión de imágenes y se hace un análisis del rendimiento de la red en sistemas de este tipo.
- **Capítulo 2. Solución Técnica:** En este capítulo se especifica el medio y modo de comunicación por red y la biblioteca de red que más se ajusta a los requerimientos del trabajo, que deberá ser tratada para ofrecer una solución al problema científico de la investigación y cumplimentar el objetivo. Se desarrolla un algoritmo de compresión de imágenes implementado para reducir el volumen de información a transmitir por la red y se analizan los factores que influyen en el rendimiento de la red.
- **Capítulo 3. Descripción de la Solución:** Durante este capítulo se describe el sistema desde la perspectiva de Ingeniería de Software, usando el Proceso Unificado de Desarrollo como metodología. Se presenta el modelo de dominio del problema, se realiza la captura de requisitos y el modelo de Casos de Uso del Sistema. Posteriormente se muestra el Diagrama de Clases del Diseño y los Diagramas de Secuencia correspondientes a los Casos de Uso.
- **Capítulo 4. Implementación y validación de los resultados:** En este capítulo se abordan los temas relacionados con la implementación del sistema, el mismo se basa en el trabajo desarrollado en los capítulos anteriores. Posteriormente se toman tipos de pruebas para validar los resultados alcanzados, fundamentalmente relacionados con el rendimiento del sistema y la calidad de la representación.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA.

En el presente capítulo, se caracterizan distintos sistemas de visualización remota implementados en el mundo que se apoyan en los conocimientos que a continuación se tratan. 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, haciendo énfasis en sus características para la selección más adecuada a las necesidades y características del problema a resolver. Además se detalla sobre algunos métodos de compresión de imágenes que permiten reducir redundancia en los datos a manipular. También se toma en cuenta el manejo de algunas terminologías para el cálculo del rendimiento en la red.

1.1 Sistemas de visualización basados en representación remota.

Los recientes avances en Internet y los gráficos por computadora estimulan el uso intensivo y desarrollo de los gráficos 3D en la red de redes. Los sistemas de visualización remota, por las altas prestaciones y facilidades que brindan, como la optimización, centralización y seguridad de recursos, se han convertido en un pilar importante dentro de las aplicaciones con prestación a distancia.

Representación remota en general significa visualización interactiva de conjuntos de datos tridimensionales a través de una red. Ella está jugando un papel clave en las tendencias de visualización en el futuro. A continuación se describen algunos sistemas que implementan dichas prestaciones.

1.1.1 Sistema de representación a distancia: IBRAC.

Para aumentar la eficiencia de los sistemas usando gráficos 3D en la web, el método presentado: Aceleración y Compresión de Renderizado Basado en Imágenes (IBRAC, acrónimo en inglés de *Image-Based Redering Acceleration and Compression*) utiliza renderizado previo y transmisión de imágenes para acelerar el procesamiento y compresión de las nuevas imágenes de la escena sintética [1]. El método de renderizado intrínsecamente calcula una imagen residual, sobre la base de una tolerancia de error especificada que equilibra la calidad de la imagen en función del tiempo de cálculo y ancho de banda. La codificación y decodificación utiliza el mismo algoritmo, por lo que la transmisión de la imagen residual sólo consiste en datos importantes sin direcciones o compensaciones. La relación de compresión por imagen es un factor de dos a diez mejor que *mpeg2* en casos de prueba. La eficiencia del servidor y el cliente en general, aumenta con la complejidad de la escena o el tamaño de los datos, desde que el tiempo de renderizado es ante todo, en función del tamaño de la imagen.

1.1.2 Protección de gráficos 3D vía representación remota: ScanView.

Con la investigación de técnicas para la protección de contenidos gráficos 3D, se desarrolló un sistema de visualización remota conveniente para compartir archivos de modelos 3D, mientras se protege la geometría de la extracción no autorizada. El sistema consta de un visor cliente, que incluye versiones de baja resolución de los modelos, y un servidor de renderizado que procesa y retorna las imágenes de modelos de alta resolución de acuerdo a las solicitudes del cliente. El servidor implementa una serie de defensas para protegerse de los ataques a la reconstrucción 3D, como la vigilancia y la limitación de solicitudes. Se consideran varios tipos posibles de ataques de reconstrucción en un servidor de renderizado y examinan la forma en que estos ataques pueden ser defendidos sin comprometer excesivamente la experiencia interactiva para los usuarios no malintencionados [2]. La herramienta que se hace alusión es: *Scanview*, un sistema cliente-servidor de renderización para la visualización de modelos complejos; donde la aplicación cliente es un visor de libre disposición, a través del cual se puede seleccionar un modelo simplificado y establecer la posición y orientación en la pantalla de la computadora. La respuesta (*feedback, en inglés*) se proporciona en forma de imágenes renderizadas del modelo simplificado. La calidad de estas imágenes es limitada, pero es suficiente para la navegación.

1.1.3 Sistema de representación remota: RemoteVIS.

Cuando los sistemas de visualización requieren de un elevado nivel de realismo de las imágenes que generan, los modelos utilizados en los mismos demandan elevados costos de memoria para su almacenamiento debido a la gran cantidad de información geométrica y elevados requerimientos de hardware gráfico; en dependencia del realismo y nivel de detalle que requiera su visualización. Es útil almacenar estos modelos en un servidor y transferirlos a un cliente remoto según la demanda de visualización. En muchos entornos dinámicos, esta es la única opción disponible. Es por ello el diseño e implementación de *RemoteVIS: A Remote Rendering System* [3]. El objetivo del sistema es proporcionar la mejor calidad de visualización, dada la capacidad y los parámetros de conexión del cliente. El sistema está diseñado para adaptarse a una amplia gama de clientes y velocidades de conexión. Esto puede ir desde una representación basada en la geometría que combina los niveles de detalle y la visibilidad escogida para una representación basada en imágenes en clientes con baja capacidad. Los gráficos del cliente y capacidades de almacenamiento, la velocidad del movimiento del espectador, y el ancho de banda y la latencia se tienen en cuenta para ello [4]. En una pasada de representación remota, el entorno virtual incluye todos los modelos, texturas y otros datos que se almacenan en el servidor. El cliente sólo recibe las piezas necesarias del entorno virtual que el cliente está viendo sin tener que descargar todo el

entorno virtual. Uno de los principales cuellos de botella del sistema es el ancho de banda de red entre el servidor y el cliente. El sistema optimiza la calidad de representación y el detalle de la malla del mundo basado en el ancho de banda. La representación basada en imágenes es útil en los casos de ancho de banda muy bajo y/o cuando el cliente no posee un acelerador de hardware gráfico.

1.1.4 Combinación de visualización local y remota: NetVisMed.

En ocasiones es posible combinar técnicas de visualización local y remota para la representación de volumen. El sistema NetVisMed consta de un cliente JAVA y un servidor basado en C++ y *OpenInventor*. En el cliente, primero un conjunto de datos obtenidos por el escáner se carga en una herramienta que permite inspeccionar los cortes en direcciones axiales, coronales y sagitales. Una subregión se puede seleccionar y localmente visualizar utilizando un visor de Java3D que utiliza texturas alineadas para la visualización. La prestación de alta calidad está disponible mediante la transferencia de datos de volumen para la aplicación servidor que utiliza el hardware de mapeado de texturas 3D. El servidor renderiza en una memoria intermedia, entonces los datos de imagen se comprimen y se transfieren al cliente. El cliente descifra los datos y muestra las imágenes. Los eventos del mouse y la interfaz gráfica de usuario se envían al servidor y son procesados por *OpenInventor* que renderiza cuando sea necesario [5]. Este sistema demuestra cómo utilizar PCs de escritorios locales y disponibles de forma remota para la visualización interactiva de datos de imágenes tomográficas. Muestra las diferentes posibilidades de combinar estas capacidades locales, el uso de técnicas de representación a distancia e híbrida. Los resultados son muy prometedores y muestran cómo combinar de manera efectiva la interactividad y visualización local de alta calidad proporcionada por los servidores especializados de forma remota.

1.1.5 Consideraciones del epígrafe.

Como se puede apreciar en la caracterización de distintos sistemas de visualización remota, la tendencia es al desarrollo de sistemas basado en transmisión de imágenes. La protección de la geometría es un elemento fundamental, por lo que la transmisión de la imagen o la imagen residual en algunos casos, brinda los servicios y potencialidades necesarias para evitar la extracción no deseada de modelos tridimensionales. Otro elemento a tener en cuenta es la capacidad de representación remota posibilitando sistemas colaborativos para trabajo conjunto.

En la siguiente tabla se puede evidenciar en resumen los elementos más comunes de los sistemas antes caracterizados. Donde (+) indica la disponibilidad y (-) la no disponibilidad de la característica indicada.

Tabla 1: Sistemas de visualización remota.

Aplicación	Basada en imagen	Basada en geometría	Híbrido	Colaboración
IBRAC	+	-	-	+
ScanView	+	-	-	-
RemoteVis	+	+	+	+
NetVisMed	+	+	+	+

1.2 Redes.

Una red de computadoras o red informática es un conjunto de equipos (computadoras y/o dispositivos) conectados por medio de cables, señales, ondas o cualquier otro método de transporte de datos, que comparten información (archivos), recursos (CD-ROM, impresoras, y otros) y servicios (acceso a internet, e-mail, chat, juegos) [6]. También constituyen un pilar importante en el desarrollo de sistemas de visualización a distancia. Estos sistemas utilizan, para su funcionamiento, diferentes protocolos implementados en distintas capas de comunicación. Además son desarrollados bajo técnicas de transmisión y arquitecturas de red.

1.2.1 El Protocolo de Internet (IP).

El Protocolo de Internet es un protocolo que se utiliza en el modelo TCP/IP. El modelo TCP/IP fue diseñado originalmente para ser usados en la ARPANET, una red militar en la década de 1960. Red que nació para convertirse en el Internet como la conocemos hoy en día. En comparación con el modelo OSI [7] no es una gran diferencia en la forma en que el modelo llegó a la existencia. El modelo OSI fue el primero cuidadosamente diseñado, y más tarde otros protocolos fueron diseñados para ajustar el modelo. El modelo TCP/IP, sin embargo, se originó en el sentido contrario. En primer lugar los protocolos han sido diseñados para cumplir los requisitos del Departamento de Defensa de los EE.UU. Más tarde, estos protocolos se han descrito y esta descripción, es el modelo de referencia. Esto significa que el modelo TCP/IP no se ajusta realmente a otra cosa que las redes TCP/IP [8]. Otro punto acerca de TCP/IP es que el diseño en capas no se sigue de forma muy estricta. Hay algunas violaciones a este principio en el modelo. A pesar de estos argumentos, el modelo TCP/IP es muy popular y ampliamente utilizado. En

contraste con el modelo OSI, que tiene siete capas, el modelo TCP/IP sólo tiene cuatro, como muestra la Figura 1. A Continuación una descripción de estas capas:

- **Capa de host a red:** Es la capa más baja del modelo. A veces también se llama la capa de enlace o la capa de interfaz de red. El único requisito que se da por el modelo es que esta capa debe ser capaz de transmitir y recibir los datagramas IP de la capa por encima de la red [8]. La capa tiene un poco la misma función que las capas física y de enlace de datos en el modelo OSI. Esto significa que esta capa por lo general sólo es capaz de enviar datos a los hosts que están conectados con el mismo medio.
- **Capa de Internet:** Corresponde a la capa de red en el modelo de referencia OSI. Su función es llevar los paquetes desde el origen al destino, a través de diferentes tipos de redes en caso necesario. Esta, sin embargo, no garantiza que los paquetes lleguen o que su orden se mantenga. El servicio que ofrece esta capa se denomina por tanto un servicio de mejor esfuerzo (*best-effort*, en inglés) ya que no se tiene noción de una conexión [8]. Los paquetes que se intercambian son llamados datagramas de Internet y el protocolo que se utiliza se llama Protocolo de Internet.
- **Capa de transporte:** Contiene un mecanismo de asignación de nombres para identificar los diferentes *hosts*, pero todavía tiene que haber alguna forma de diferenciar entre los procesos que están usando la red. Esto se hace en la capa de transporte por el uso de un número de puerto. Esta capa es similar a la de transporte del modelo OSI. El modelo TCP/IP tiene dos principales protocolos de capa de transporte [8]. Uno de ellos es el Protocolo de Control de Transmisión (TCP, siglas en inglés); el otro protocolo es el Protocolo de Datagramas de Usuario (UDP, siglas en inglés).
- **Capa de aplicación.** Las funciones que realiza esta capa difieren de los ofrecidos por las otras, debido a que como no existe una capa superior, no ofrece servicios. Al igual que en el modelo OSI, la capa de aplicación contiene los protocolos de aplicaciones de redes [8]. Entre ellas se encuentran las aplicaciones de terminal virtual (protocolo TELNET), servicios de transferencia de archivos (protocolo FTP) y correo electrónico (protocolo SMTP).

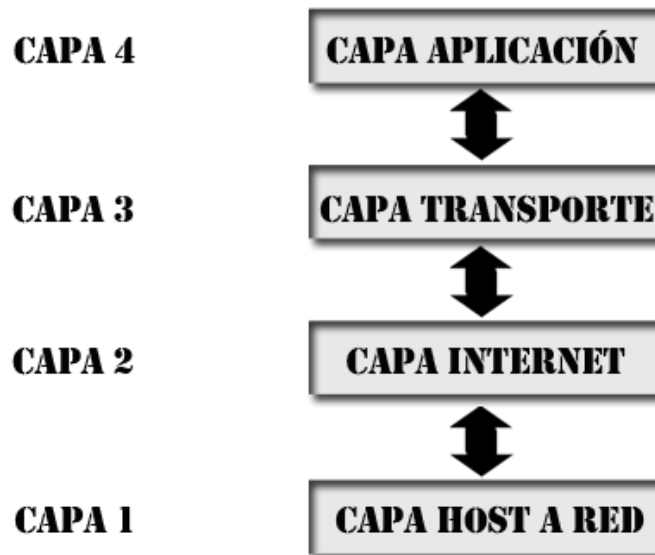


Figura 1: Modelo 4 capas de TCP/IP.

1.2.2 Protocolos de nivel superior.

Los protocolos de red 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. 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 UDP y TCP. Cada uno de estos protocolos ofrece un determinado tipo de servicio que las aplicaciones pueden utilizar para comunicarse a través de redes.

1.2.2.1 TCP: Protocolo de Control de Transferencias.

Este protocolo transforma el servicio no fiable basado en paquetes de la capa de Internet en un flujo de bytes fiable. El protocolo es diseñado para la comunicación entre dos sistemas, por lo que sólo es compatible con unidifusión. Este protocolo une la capa de aplicación a la capa de red, además se asegura que los datos tengan el tamaño adecuado, que se coloquen correctamente en paquetes y en el orden adecuado cuando se reciben, de allí que sea un protocolo fiable y orientado a la conexión. Este protocolo requiere mayor ancho de banda, es más lento que UDP y utiliza paquetes más grandes [8]. Para ofrecer este tipo de servicio, el módulo de TCP tiene que realizar un gran trabajo. En primer lugar, una conexión tiene que ser creada, de tal manera que sea más o menos segura: el módulo debe asegurarse de que las

conexiones no se puede establecer por accidente; por ejemplo debido a la duplicación de los paquetes. Se debe tener cuidado para descartar datagramas duplicados y corregir el orden de llegada, si es necesario. También debe haber algún tipo de mecanismo para hacer frente a la pérdida de paquetes.

1.2.2.2 UDP: Protocolo de Datagrama de Usuario.

Las aplicaciones que no requieren la funcionalidad que proporciona TCP, puede usar UDP. Para transmisión de datos, el módulo UDP simplemente pasa un encabezado UDP seguido por el de datos a la capa Internet que envía el datagrama en su camino. Esto significa que al igual que la propia IP, UDP es un servicio de mejor esfuerzo. No hay garantías sobre la entrega, los datagramas se pueden reordenar y duplicar. Dado que el servicio que ofrece UDP es casi idéntico al servicio de IP en sí, es posible que las aplicaciones para enviar datagramas UDP a una dirección de multidifusión y recibir datagramas UDP de un grupo de multidifusión. UDP es un protocolo no orientado a la conexión (como IP) por lo que es más rápido que TCP ya que no tiene que abrir una conexión con el receptor y no tiene que realizar ninguna corrección de error [8]. 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.

1.2.2.3 RTP: Protocolo de Transporte en Tiempo Real.

RTP establece un servicio “extremo a extremo” de entrega de datos en tiempo real, como audio y video interactivo. Los servicios incluyen la carga útil como tipo de identificación, numeración de secuencia, marcas de tiempo y el monitoreo de entrega. Las aplicaciones RTP suelen ejecutarse sobre UDP para hacer uso de su multiplexación y servicios de suma de comprobación; ambos protocolos contribuyen parte de la funcionalidad del protocolo de transporte. Sin embargo puede ser utilizado con otras adecuadas redes subyacentes o protocolos de transporte. También soporta transferencia de datos a varios destinos mediante la distribución de multidifusión si las facilita la red subyacente. RTP no proporciona ningún mecanismo para asegurar la entrega oportuna o proporcionar otra calidad de garantías de servicio, sino que se basa en servicios de bajo nivel para hacerlo. No garantiza la entrega o impide la entrega fuera de orden, ni asume que la red subyacente es fiable y la entrega de paquetes en secuencia. Se destina a ser maleable para proporcionar la información requerida por una aplicación en particular, a menudo se integra en el proceso de solicitud en lugar de ser aplicado como una capa separada [9]. A diferencia de los protocolos convencionales en los que las funciones adicionales podrían ser acomodadas por hacer el protocolo más general o mediante la adición de un mecanismo de opción que requeriría el análisis, RTP

es destinado a ser adaptado a través de modificaciones y/o adiciones a las cabeceras, según sea necesario.

1.2.3 Técnicas de transmisión.

Existen diferentes técnicas de transmisión de datos (ver Figura 2), cada una tienen sus particularidades y son usadas en distintos sistemas según el propósito que se persiga. Estas técnicas se pueden dividir en distintos tipos, cada uno de ellos con características distintivas que los diferencian e identifican:

- **Unidifusión** (*Unicasting*, en inglés): En esta técnica se establece una conexión entre dos puntos a los cuales se les 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, 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 [10] [11].
- **Multidifusión** (*Multicasting*, en inglés): 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 [10] [11].
- **Radiodifusión** (*Broadcasting*, en inglés): 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 este tipo de transmisión por radiodifusión. [10] [11].

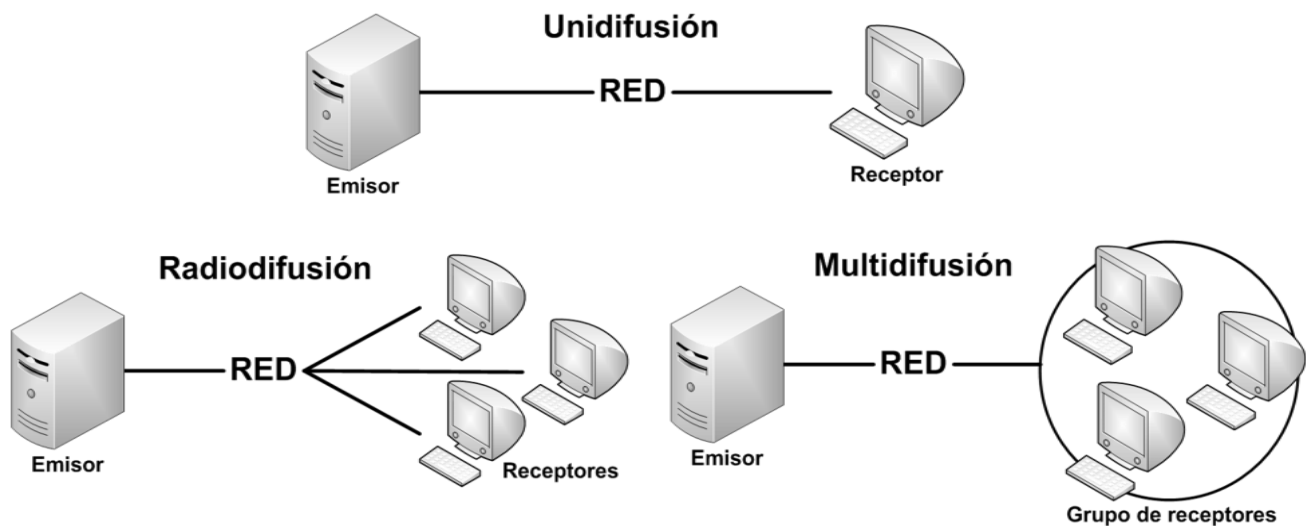


Figura 2: Tipo de transmisiones de red.

1.2.4 Arquitecturas de comunicación.

Las arquitecturas de red se pueden organizar de acuerdo a 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. Éstas son las siguientes:

- **Arquitectura punto a punto:** 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 de área local [10] [11].
- **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. La comunicación directa entre los nodos receptores no existe. El servidor controla todos los envíos realizados y 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 [10] [11].

- **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 [10] [11].

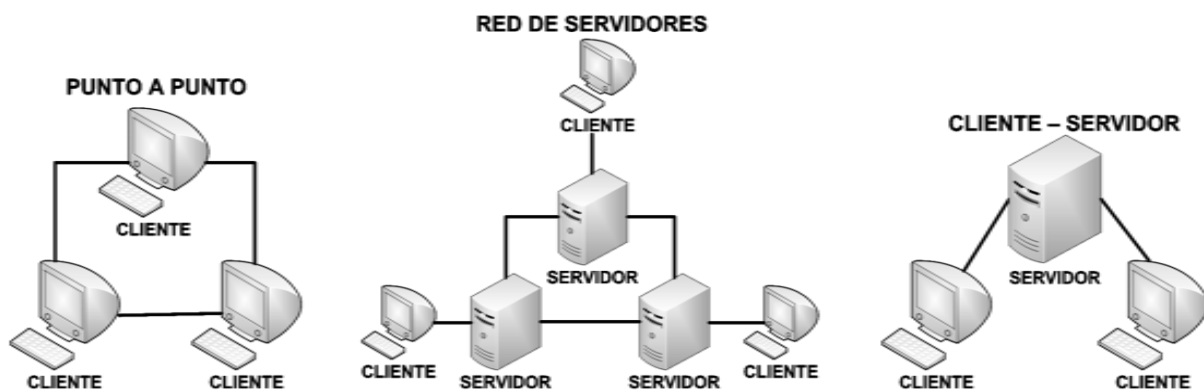


Figura 3: Arquitecturas de comunicación.

1.3 Bibliotecas de comunicación por red.

En el mundo existen diversas bibliotecas de clases y algoritmos implementados con el fin de satisfacer las demandas de los desarrolladores de sistemas de visualización a distancia, transmisión de datos, juegos en línea, etc. Estas se apoyan en protocolos de redes que permiten el flujo de datos entre cómputos y/o sistemas de cómputos. A continuación se describen algunas bibliotecas de red (*network library*, en inglés).

1.3.1 Biblioteca JRTPLib.

La biblioteca JRTPLIB, que significa “*Jori’s RTP Library*” está escrita en C++ usando un enfoque orientado a objetos [12]. La biblioteca hace que sea más fácil enviar y recibir paquetes RTP. El usuario puede seleccionar cualquier número de destinos para los datos. La multidifusión se puede utilizar para la distribución eficiente de los datos. La funcionalidad de RTCP es completamente manejado internamente por la biblioteca. La estructura de la biblioteca es muy modular, lo que hace que el código fuente sea fácil de entender y fácilmente extensible. El uso de las funciones de *socket* estándar, hace que la biblioteca se pueda usar en una amplia variedad de plataformas. La biblioteca es deseable para aplicaciones en tiempo

real. En la actualidad, la biblioteca se sabe que funciona en las siguientes plataformas como: *MS-Windows, Linux, FreeBSD, Solaris, HP-UX, VxWorks*. La API de la biblioteca consta de cuatro partes: Clase *RTPSession*, Clase *RTPSourceData*, Clase *RTPpacket* y estructuras para almacenar la información de eventos [13]. La clase *RTPSession* es la parte central de la biblioteca. A través de esta clase, el usuario puede seleccionar destinos, enviar paquetes, la encuesta de datos entrantes, etc. Esta información se almacena en una instancia de la clase *RTPSourceData*. Los paquetes recibidos se pasan al usuario como instancias de la clase *RTPpacket*.

1.3.2 Biblioteca HawkNL.

HawkNL es una API de red orientada a juegos, es libre, de código abierto, liberada bajo la licencia *GNU LGPL (GNU Library General Public License)*. Se considera de bajo nivel ya que trabaja bastante cercano a los *sockets*, tiene funciones sobre *Berkeley/Unix Sockets* y *WinSock*. También incluye soporte para distintos sistemas operativos y para múltiples transportes de red; cuenta con grupos de *sockets*, un temporizador de alta exactitud, estadísticas de los *sockets* y macros para leer y enviar datos en paquetes. Ha sido probada en *Windows, Linux, IRIX, AIX, BSDs, Mac OS* [14]. 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.

1.3.3 Biblioteca 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. Esta biblioteca produce una rápida ejecución en programas compilados y ofrece poderosas capacidades para la programación. 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. La biblioteca fue producida por trabajo independiente, bajo contrato y fue liberado al público 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 hacerla una biblioteca robusta. En 2005 el código se puso bajo el escrutinio para producir un código fuente estable y seguro que permitiera su uso en complejos sistemas comerciales. Entre las ventajas se tiene 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, portable y constituye un acercamiento a la programación para redes y bases de datos [15].

1.3.4 Biblioteca Libtcp++.

Libtcp++ es una biblioteca de clases escrita en C++ que facilita la creación de clientes y servidores TCP/IP. Esta biblioteca contiene tres clases, *TcpClient*, *TcpServer* y *TcpRuleSet*. *TcpServer* ha construido un método para la detección de las capacidades de conexión de una computadora “peer” (computadora conectada al resto de los nodos, puede enviar y recibir mensajes directamente), y un control de acceso basado en IP para regular la funcionalidad del servidor. La clase de *TcpClient* brinda en la función *connect()* un parámetro *timeout* que resulta útil para los escaneos de puertos de host y para otras situaciones en las que se esté tratando de conectar a host apagados o a puertos protegidos por reglas de cortafuegos (firewall, en inglés). 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 [16] [17] [18] [19]. Brinda el código libre y se pueden realizar las modificaciones necesarias que estime el usuario.

1.3.5 Biblioteca Zoidcom.

Zoidcom es una biblioteca de red de alto nivel libre para uso no comercial. 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 [20]. Entre las ventajas 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 tamaño requerido. Tiene una distribución dinámica y limitación del ancho de banda según las necesidades y envía paquetes UDP a destinos arbitrarios.

- **Sincronización de estado de los objetos automáticamente:** Sincroniza los tipos de datos entero, booleano y flotante; de forma automática realiza interpolación entre las actualizaciones de los estados. Implementa en solo minutos replicadores de tipos de datos y solamente envía la información que cambia realmente.

1.4 Métodos de compresión de imágenes.

La capacidad de trabajo con imágenes y su almacenamiento en los sistemas informáticos van en aumento, por lo que se dispone de muchos más píxeles. De ahí que se mantenga el uso y el interés por la evolución de técnicas de compresión de datos. La compresión en realidad, consiste en sustituir una cadena de datos por otra más corta. Ciertos métodos son irreversibles (*Lossy Data Compression*, en inglés): la información original solo se recupera aproximadamente, ya que se descarta una parte de los datos. También existe la compresión sin pérdida [21] (*Lossless Data Compression*, en inglés): que permite la reconstrucción exacta de la información original.

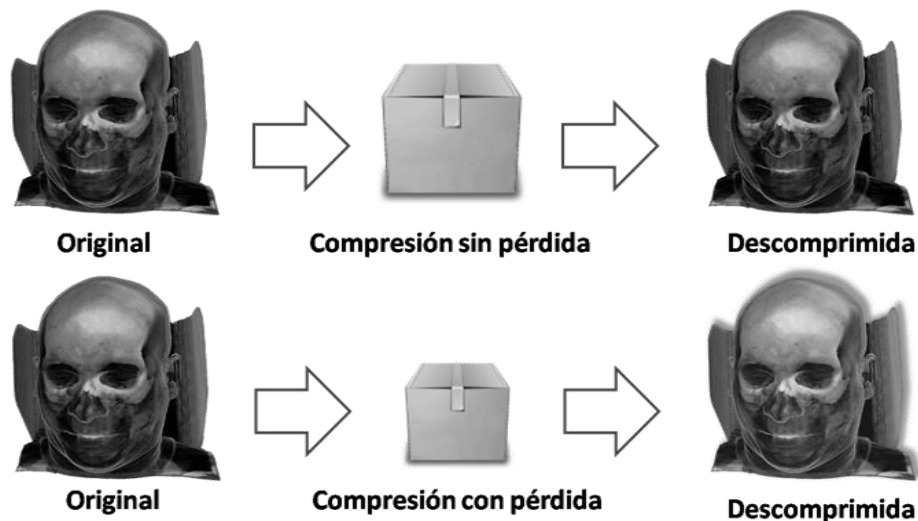


Figura 4: Comparación entre la compresión con pérdida y sin pérdida.

Como se observa en la Figura 4, la compresión con pérdida muestra disminución de la información (menos calidad visual) la cual genera que la imagen original no puede ser recuperada del todo. Para los sistemas de visualización con altas prestaciones que se desea, los métodos por compresión sin pérdida son idóneos para mantener la calidad de las imágenes procesadas. A continuación se caracterizan algunos métodos de compresión sin pérdida.

1.4.1 Método de compresión RLE.

El método de compresión *RLE* (*Run Length Encoding*, a veces escrito *RLC* por *Run Length Coding*, que significa Código de Manejo de Longitud) es utilizado por muchos formatos de imágenes (*BMP*, *PCX*, *TIFF*, *DICOM*). Este se basa en la repetición de elementos consecutivos, donde secuencias de caracteres idénticos pueden ser codificadas como un campo de cuenta, además de un identificador del carácter repetido. Este enfoque es eficaz en imágenes gráficas, no tiene prácticamente ningún valor en el texto y tiene un valor moderado en los archivos de datos [21]. El problema con la codificación RLE son los símbolos mezclados con otros datos, a la hora de distinguir los campos de conteo de símbolos normales que pueden tener el mismo patrón de bits. Este problema tiene varias soluciones, pero cada uno tiene sus desventajas [22]. Por ejemplo, un símbolo especial podría ser utilizado para marcar en cada serie de caracteres, lo cual está bien para el texto ASCII, pero no para los patrones de bits arbitrarios como los de enteros binarios. Normalmente, tres caracteres son necesarios para marcar cada serie de caracteres, por lo que esta codificación no sería utilizada para cadenas de tres o menos caracteres.

1.4.2 Método de compresión LZW.

La técnica, llamada *Lempel-Ziv. Welch* (*LZW*) de codificación [21], asigna las palabras de código de longitud fija a las secuencias de longitud variable de los símbolos fuente, pero no requiere ningún conocimiento a priori de la probabilidad de ocurrencia de los símbolos que se codificarán. La compresión LZW se ha integrado en la corriente principal con variedad de formatos de imágenes de archivo, incluyendo el formato de intercambio gráfico (*GIF*), el formato de archivo de imagen etiquetada (*TIFF*) y el documento formato portátil (*PDF*). La codificación LZW es conceptualmente muy simple. En el de proceso de codificación, se construye un libro de códigos o "diccionario" que contiene los símbolos de origen para ser codificado. Para las imágenes monocromáticas de 8 bits, las primeras 256 palabras del diccionario se asignan a los valores en escala de grises de 0, 1, 2,..., 255. Como el codificador secuencial examina los píxeles de la imagen, las secuencias de colores de nivel gris que no están en el diccionario se colocan en determinados algoritmos [22].

1.4.3 Método de compresión Codificación Huffman.

De la teoría de la fuente de codificación de Shannon, se sabe que una fuente puede codificarse con una longitud media del código cerca de la fuente de entropía. En 1952, D. A. Huffman inventó una técnica de codificación para producir el más corto promedio de longitud de código dado el símbolo de la fuente y la probabilidad asociada de ocurrencia de los símbolos. Los códigos que se generan con esta técnica se

conocen como códigos de Huffman [23]. El código de Huffman enuncia que la longitud de cada código no es idéntica para todos los símbolos: se asignan códigos cortos a los símbolos más utilizados con más frecuencia (los que aparecen más a menudo), mientras que los símbolos menos frecuentes reciben códigos binarios más largos [21] [22]. La expresión código de longitud variable (*VLC*) se utiliza para indicar este tipo de código porque ningún código es el prefijo de otro. De este modo, la sucesión final de códigos con longitudes variables serán en promedio más pequeña que la obtenida con códigos de longitud constante.

1.4.4 Método de compresión JPEG sin pérdida.

Uno de los más populares estándares de compresión de imágenes fijas es el estándar JPEG. En él se definen tres sistemas diferentes de codificación:

1. Sistema de codificación línea base (*baseline*, en inglés) con pérdida, que se basa en la transformada discreta del coseno (*DCT*, siglas en inglés) y es adecuado para la mayoría de las aplicaciones de compresión.
2. Sistema de codificación extendido para una mayor compresión, mayor precisión o aplicaciones de progresiva reconstrucción.
3. Sistema independiente de codificación sin pérdidas para la compresión reversible.

Este sistema independiente de compresión JPEG sin pérdida, se basa en los principios de la codificación predictiva. Dado que los píxeles adyacentes en una imagen típica están altamente correlacionados, es posible extraer una gran cantidad de información sobre un píxel dado el valor de los píxeles vecinos. Codificación predictiva [22] es un método simple para la reducción de la redundancia espacial. En este método, un valor de píxel se prevé por un conjunto de píxeles adyacentes previamente codificados usando un modelo de predicción adecuado. Para un modelo de predicción ideal, el valor previsto del píxel puede ser igual al valor real. El uso de un modelo de predicción efectivo, que puede predecir el valor del píxel, está muy cerca de su valor real y por lo tanto el error de la predicción puede ser muy pequeño.

1.5 Rendimiento y retardo en la red.

Cuando los métodos de reserva de recursos están soportados en los enrutadores (*routers*, en inglés), los retrasos de transmisión probablemente puede mantenerse lo suficientemente bajo como para satisfacer la restricción de demora total de 200 ms. Pero en este momento, los enrutadores que se utilizan actualmente, en general, no tienen esa capacidad. Cuando los datos se transmiten siempre hay una cantidad mínima de retraso debido a la capacidad de los enlaces a lo largo de la cual los datos viajan.

Pero la parte más importante de la demora de transmisión es por lo general debido a la cola de paquetes en los enrutadores. Este retraso es muy variable y depende tanto del número de enrutadores a lo largo de la ruta y la carga de los mismos. No es posible hacer una afirmación general sobre el retraso de transmisión en las redes IP, aunque en un solo sentido los retrasos de transmisión rara vez tienden a superar los 100 ms [24] .

El retardo total se puede clasificar en dos tipos [25]:

- La demora fija. Esta es la demora total que siempre está presente debido a la capacidad de enlace de almacenamiento en búfer, etc.
- El retardo variable. Este es el componente de retardo que es causado por la cola de paquetes en enrutadores, congestiones, etc.

Los componentes que se muestran en la Figura 5 determinan la cantidad de tiempo fijo para un sistema en entornos virtuales. Por "normal", el componente de procesamiento de 3D, simplemente se puede dejar fuera. Los retrasos por **compresión** y **descompresión** se introducen por medio de algoritmos de compresión y descompresión de datos, respectivamente. El retardo por **transmisión** es el retardo que está presente debido a la capacidad de enlace. El retardo por **procesamiento 3D** es el retraso causado por los algoritmos que generan la escena tridimensional para su uso en entornos virtuales. Finalmente, el retraso por **almacenamiento en búfer** es el retardo que se introdujo artificialmente para compensar el ruido; este retraso se podría establecer de forma manual o de forma automática.

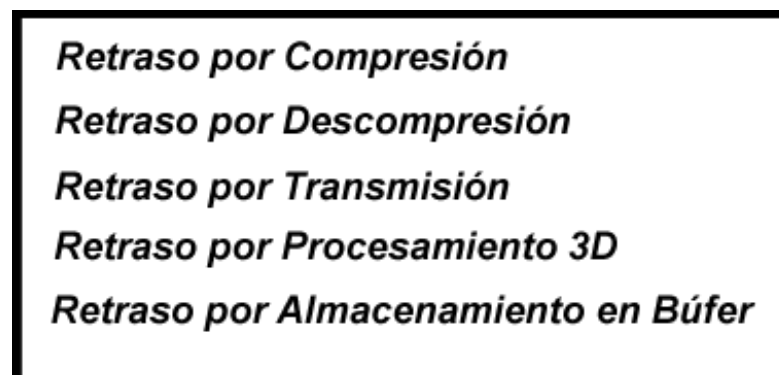


Figura 5: Tipos de retrasos fijos en sistemas de red.

1.6 Consideraciones generales.

En el desarrollo del capítulo se caracterizaron diferentes sistemas de visualización por medio de representación remota logrando caracterizar los elementos fundamentales para el desarrollo del sistema propuesto. También se analizaron distintas arquitecturas de comunicación y métodos de transmisión en el uso de redes, donde se destaca, para aplicaciones como la deseada, la arquitectura cliente-servidor implementada bajo protocolo RTP por sus prestaciones en aplicaciones en tiempo real. En el caso de los sistemas por transmisión de imágenes es necesario aplicar algoritmos de compresión para reducir el volumen de datos que se procesan en las escenas tridimensionales. Se pudo constatar que el algoritmo de compresión sin pérdida RLE, es ideal para reducir la redundancia en los datos que se manipulan, por su fácil manejo y rápido procesamiento de datos. Con la utilización de todos estos elementos antes mencionados, es posible el desarrollo de sistemas de visualización remota teniendo en cuenta las influencias de los procesos que intervienen e introducen retardo en la red donde se implementen.

CAPÍTULO 2. SOLUCIÓN TÉCNICA.

En el presente capítulo se explica el procedimiento del algoritmo de compresión y descompresión de imágenes, implementado para la disminución de la redundancia en los datos a transmitir. Se describen las herramientas y métodos utilizados para la implementación del sistema de red que permite la conexión entre los clientes y el servidor para el intercambio de información y la visualización a partir de la transmisión por la red. Además se especifican herramientas como tipo de red, arquitectura, biblioteca de red, lenguaje de programación y metodologías usadas para dar solución al objetivo de la investigación.

2.1 Arquitectura de comunicación: Cliente-Servidor.

La arquitectura cliente-servidor brinda las potencialidades necesarias para dar solución al problema de la investigación. En el modelo cliente-servidor propuesto se pueden encontrar las siguientes características:

- El cliente y el servidor pueden actuar como una sola entidad y como entidades separadas, realizando actividades o tareas independientes.
- El servidor brinda servicios a múltiples clientes en forma concurrente.
- Los cambios que se realizan en el cliente o en el servidor, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.

El área de interés para la presente investigación es la prestación remota de servicios donde la interfaz gráfica de usuario (*GUI*, siglas en inglés) está completamente en el cliente; la aplicación y los datos están en el servidor. Esto tiene como ventaja el aprovechamiento de la GUI y la Red de Área Local (*Local Área Network (LAN)*, en inglés). Aunque no se descarta que las aplicaciones puedan ser complejas de desarrollar, los programas de la aplicación siguen en el *host* y existe alto volumen de tráfico en la red que puede hacer difícil la operación de aplicaciones muy pesadas.

En la arquitectura propuesta se necesita la conexión de múltiples clientes al servidor haciendo pequeñas consultas con necesidad de rápidos tiempos de respuestas. Es por ello que se utilizó UDP como protocolo para la transmisión de paquetes por medio de multidifusión (el envío a todos los suscriptores). Por tanto para el sistema que se desarrolla es ideal para lograr tiempos de respuestas aceptables en la visualización de modelos tridimensionales a través de la red.

2.2 Biblioteca de comunicación: JRTPLib.

Esta biblioteca proporciona una implementación del protocolo RTP, especificado en el capítulo anterior. Se organiza en torno al concepto de una sesión, lo que corresponde a la noción de una sesión RTP. JRTPLib es una biblioteca general de RTP. Una de las características más importantes de esta biblioteca es que la funcionalidad del Protocolo de Control en Tiempo Real (*RTCP*, siglas en inglés) se maneja internamente por la biblioteca. El usuario no tiene que construir, ni transmitir los paquetes RTCP por sí mismo, pero todavía se puede inspeccionar los datos entrantes. Esto permite que el usuario de la biblioteca se concentre en los actuales datos en tiempo real, sin dejar de tener pleno acceso a la información proporcionada por calidad de servicio RTCP. En el código fuente de la biblioteca, sólo se usan las funciones estándar de *socket* de Berkeley, que están disponibles en una amplia variedad de plataformas, esto se suma a la portabilidad de la biblioteca que es multiplataforma. También destacar que está escrita en C++, con programación orientada a objeto.

El uso de la biblioteca en el caso del servidor se basa en establecer los parámetros necesarios para la creación de una sesión. El servidor estando en ejecución y dando servicio de multidifusión, el cliente de igual manera establece sus parámetros y crea una sesión conectándose al servidor para recibir y enviar paquetes. Aquí se muestran algunos de los parámetros y procedimientos necesarios para lograr la conexión así como el envío y recepción de paquetes:

- Se crea una sesión ***RTPSession session***; Para la mayoría de las aplicaciones basadas en RTP, la clase *RTPSession* es la que se debe usar.
- Se crea ***RTPSessionParams sessionparams***; Describe los parámetros para ser utilizados por una instancia *RTPSession*. Tenga en cuenta que la Unidad Marca de Tiempo Propio (*Own Timestamp Unit*, en inglés) se debe establecer en un número válido, de lo contrario la sesión no se puede crear.
- Se crea ***RTPUDPV4TransmissionParams transparams***; Representa los parámetros utilizados por UDP a través de los componentes de transmisión de IPv4.
- ***SetOwnTimestampUnit (doubletsunit)***; perteneciente a ***RTPSessionParams***. Establece la unidad de marca de tiempo para nuestros propios datos. La unidad de registro de la hora se define como un intervalo de tiempo en segundos, dividido por el número de muestras en ese intervalo. Dado que este valor se establece inicialmente en un valor no válido, el usuario debe configurar este a un valor permitido para poder crear una sesión.

- **SetMaximumPacketSize (MAX_PACKET_SIZE);** perteneciente a **RTPSessionParams**. Establece el tamaño máximo permitido de paquetes para la sesión.
- **SetPortbase (BASE_PORT);** perteneciente a **RTPUDPV4TransmissionParams**. Establece el puerto base **RTP**. Este tiene que ser un número par.
- **SetControlTrafficFraction (double frac);** perteneciente a **RTPSessionParams**. Establece la fracción del ancho de banda de sesión para ser usado para control de tráfico.
- **Create (sessionparams,&transparams);** Crea la **session** con los parámetros **sessparams** y **transParams**. Si **transParams** es NULL, los valores predeterminados para el transmisor solicitado serán utilizados.

Ya queda todo listo para la interacción entre el servidor y los clientes que escuchan por multidifusión. Otros procedimientos y rutinas son utilizados internamente para garantizar el envío y recepción de paquetes. El más importante y utilizado en estos procesos es la función:

- **SendPacket (const void *data, size t len, uint8 t pt, bool mark, uint32 timestampinc).** Envía el paquete con los datos de carga útil (*payload*, en inglés), que tiene longitud **len**. Se utiliza el tipo de carga útil **pt**, **mark** de marcador y después de que el paquete ha sido construido, la fecha y hora (*timestamp*, en inglés) se incrementará en **timestampinc**.

Esta función que lleva el paquete es quien viaja por la red con los mensajes, comandos y datos necesarios para la interacción y conectividad deseada. Como se ha visto hasta ahora el trabajo con RTP es basado en UDP por la rapidez de este protocolo de transmisión de datos en tiempo real.

2.3 Algoritmo de compresión/descompresión: Variante RLE.

Visto en el capítulo anterior algunos algoritmos de compresión sin pérdida más destacados en la actualidad, dada la robustez de implementación, buena tasa de compresión e interpretación de los datos dentro del volumen de información, el método de compresión sin pérdida RLE es ideal para reducir la cantidad de datos a manipular. La implementación de una variante de este algoritmo proporciona la compresión deseada, puesto que las imágenes que se procesan son obtenidas como cadena de caracteres, en este caso cadena lineal de bytes que forman los píxeles de la imagen. Es por ello que se hace referencia a la compresión de píxeles de forma lineal. El funcionamiento del algoritmo tiene como idea principal reemplazar cadenas de igual información como 'aaaa' o 'bbbb' ('a' y 'b' se refiere a un píxel de componente RGBA) con un contador diciendo cuántas repeticiones consecutivas existen. La idea es

bastante simple, si se tiene 'aaaa' da como salida 'a', y luego un byte '3', lo que significa que hubo tres píxeles más del mismo tipo. En este primer esquema existe un gran problema, 'aaaa', por supuesto, puede ser comprimido, pero ¿qué pasa con 'ab'? será comprimido como a, 0, b, 0 y esto no es una buena idea, porque va a ampliar los datos en lugar de comprimirlos, entonces se es más preciso. Ejemplo: 'aaaa' se codifica como 'aa2' y 'ab' se codifica como 'ab'. Este esquema es lo suficientemente bueno para las necesidades que se tienen, ya que solo se comprime si existe repetición de más de 2 píxeles consecutivos. También a la hora de descomprimir los datos es más cómodo puesto que al encontrarse con dos elementos iguales pasa a localizar el contador ya que sigue el patrón de compresión y así con toda la cadena.

2.3.1 Flujo del algoritmo de compresión.

La Figura 6 muestra la representación gráfica del proceso de compresión de la variante RLE implementada y se procede de la siguiente manera:

- Se almacena el primer píxel.
- Se compara el primer píxel con el segundo.
 - Si son iguales.
 - Se cuenta cuantos más de los que siguen son iguales.
 - Se almacena la cantidad que indique el contador.
 - Si no.
 - Se toma el segundo como primero.
- Se repite hasta el final.

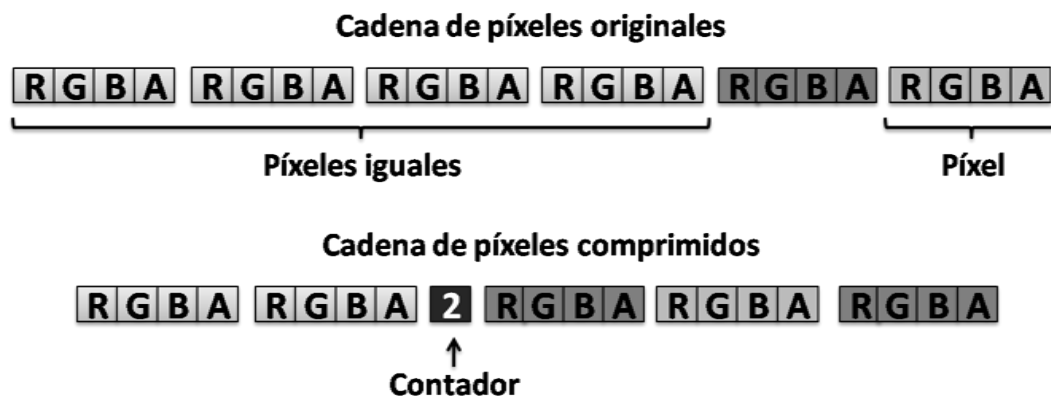


Figura 6: Ejecución del algoritmo de compresión sin pérdida. Variante RLE.

2.3.2 Flujo de algoritmo de descompresión.

El proceso de descompresión de la imagen se puede observar en la Figura 7 y se procede de la siguiente manera:

- Se almacena el primer píxel.
- Se compara el primer píxel con el segundo.
 - Si son iguales.
 - Se almacena el segundo.
 - Se almacena cuantas copias del primero indique el contador.
 - Si no.
 - Se toma el segundo como primero.
- Se repite hasta el final.

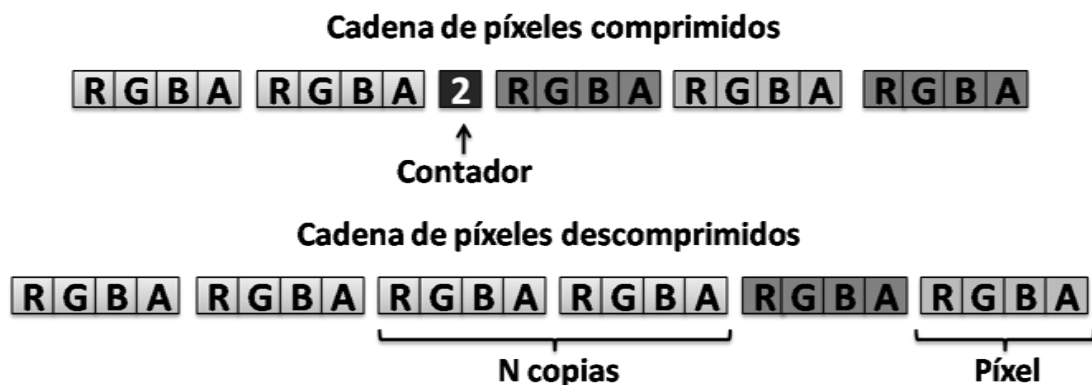


Figura 7: Ejecución del algoritmo de descompresión sin pérdida. Variante RLE.

2.4 Cálculo del retardo en la red.

En el capítulo anterior, en el epígrafe 1.5 se definieron los factores que influyen en el retardo de la manipulación de datos a través de la red. Desglosando por cada uno de los elementos que pueden agregar retardo en la red se puede apreciar la influencia de cada uno:

- **Retardo por compresión y descompresión.** El proceso de compresión y descompresión de imágenes tiene una complejidad en ambos casos de $O(n)$, y se corresponde con el recorrido de n elementos linealmente estructurados cuyos tiempos de procesamientos traen un retraso mínimo que se considera despreciable.

- **Retardo por transmisión.** Este depende de la infraestructura de red sobre la que se esté ejecutando el sistema. Aunque es funcional sobre redes LAN e inalámbricas (*wireless*, en inglés), mientras mejor sea la conexión y la velocidad de la misma, así como las capacidades de los enrutadores y conmutadores (*switches*, en inglés); está demostrado en la bibliografía consultada que los tiempos de retraso no superan los 200 ms.
- **Retardo por procesamiento 3D.** Este retraso no se consideró para el desarrollo del sistema, pues este no depende de las escenas que se manipulen ni de la complejidad de las mismas. Este proceso no se tiene en cuenta para el cálculo del rendimiento de la red.
- **Retardo por almacenamiento en búfer.** En el caso propuesto se fijó para los envíos entre los fragmentos que componen la imagen, tiempos de espera menores a los 2ms. Esto hizo posible no sobrecargar la capacidad de enlace en la red donde se esté trabajando.

El retardo total en la aplicación propuesta está determinado por el medio de comunicación que se utilizó para la transmisión. Para disminuir este retardo, se introdujo el proceso de compresión y descompresión de las imágenes y se fijó el tiempo de espera entre paquetes por almacenamiento en búfer.

2.5 Metodologías y herramientas de desarrollo.

Para la realización del presente trabajo fue necesario el empleo de una metodología de desarrollo de software y de algunas herramientas que asistieron el proceso de creación de los diagramas y la programación de la solución propuesta.

2.5.1 Metodología de desarrollo de software.

Como metodología de desarrollo de software se escogió el Proceso Unificado de Desarrollo (*RUP*). Esta robusta metodología está preparada para guiar el desarrollo de prácticamente todo tipo de software. Dentro sus principales características se encuentran:

- **Dirigido por casos de uso.** Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, estos se obtienen durante el modelado del negocio. El proceso de desarrollo de software avanza a través de una serie de flujos que parten de los casos de uso, se puede afirmar que estos proporcionan un hilo conductor y una guía para todo el proceso [26].
- **Centrado en la arquitectura.** La arquitectura muestra la visión común del sistema completo y describe los elementos del modelo que son más importantes para su construcción, los cimientos del

sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. La arquitectura debe diseñarse para que el software evolucione, no solo en su desarrollo inicial, sino también a lo largo de las futuras generaciones [26].

- **Iterativo e incremental.** RUP propone que cada proyecto se desarrolle en fases y que cada fase se desarrolle en iteraciones, donde cada iteración resulta en un incremento del proceso de desarrollo, lo cual se realiza de forma planificada y culmina con el cumplimiento del punto de control trazado en la fase [26].

2.5.2 Herramientas de desarrollo.

El Visual Paradigm es una herramienta que permite modelar casos de uso y procesos de negocio que son la base de los requisitos funcionales de cualquier aplicación. A continuación se describen sus principales características:

- Presenta licencia gratuita y comercial.
- Soporta aplicaciones web.
- Disponible en varios idiomas.
- Fácil de instalar y actualizar.
- Compatible entre versiones.
- Entorno gráfico amigable para el usuario.
- Disponible en múltiples plataformas (*Windows/Linux/Mac OS X*).

2.5.3 Lenguaje de modelado.

UML es un lenguaje de modelado visual que se usa para especificar, construir, documentar y visualizar artefactos de un sistema de software. El mismo está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Este lenguaje de modelado 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. UML incluye conceptos semánticos, notación y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos.

2.5.4 Lenguaje de programación.

Como lenguaje de programación se utilizó C++, lenguaje por excelencia para las aplicaciones de realidad virtual que hace uso eficiente del paradigma de Programación Orientada a Objetos (POO). Permite un excelente control de la memoria y una buena administración de los recursos de la computadora. Dentro de las principales ventajas que presenta el lenguaje se encuentran:

- **Difusión:** al ser uno de los lenguajes más empleados en la actualidad, posee gran número de usuarios y tiene una excelente bibliografía.
- **Versatilidad:** es un lenguaje de propósito general, se puede emplear para resolver cualquier tipo de problema.
- **Portabilidad:** se encuentra estandarizado, por tanto, el mismo código fuente puede ser compilado en diferentes plataformas.
- **Eficiencia:** es uno de los lenguajes más rápidos en tiempo de ejecución.
- **Herramientas:** existen gran cantidad de compiladores, depuradores y bibliotecas de clases basadas en este lenguaje.

2.6 Consideraciones Generales.

En el presente capítulo se propuso la arquitectura cliente-servidor por ser robusta, escalable y multiplataforma. La cual permite la centralización de recursos, donde un servidor brinda servicios a múltiples clientes de forma simultánea. Aplicando la técnica de transmisión por multidifusión para lograr el envío de paquete a todos los receptores. En la arquitectura propuesta se necesita de pequeñas consultas al servidor con necesidad de tiempos de respuestas aceptables, por lo que se utilizó UDP y RTP como protocolos para la transmisión de paquetes. Se tuvo en cuenta los elementos que introducen retardo de tiempo, determinando el de mayor impacto. El algoritmo de compresión/descompresión de imágenes implementado, se explica y se detalla, y viene encaminado a disminuir la cantidad de paquetes a enviar por imágenes influyendo en el alivio de la carga en la red. Además para el desarrollo del sistema propuesto se utilizó RUP como metodología de desarrollo, UML como lenguaje de modelado y *Visual Paradigm* como herramienta CASE (Computer Aided Software Engineering, traducción en español: Herramientas de Ingeniería de Software Asistida por Computadora.). Queda de esta manera, seleccionadas e implementadas todas las partes necesarias para desarrollo del sistema.

CAPÍTULO 3. DESCRIPCIÓN DEL SISTEMA

En el presente capítulo, se tendrá como tema fundamental el análisis y diseño de la aplicación. Se define el modelo de dominio a utilizar y se hará la captura de requisitos, que será utilizada posteriormente en la elaboración del modelo de casos de uso. Se definirán los actores del sistema y se especificará cada caso de uso obtenido a partir de la captura de requisitos. Se modelarán diferentes diagramas de secuencia del diseño con el objetivo de proporcionar una comprensión más profunda del sistema que se desea desarrollar.

3.1 Modelo de Dominio.

El modelo de dominio es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema, conceptos del mundo real en lugar de componentes de software [27]. En el presente trabajo no es necesario realizar un complejo modelo de negocio, el modelo de dominio es suficiente para modelar todo el flujo del entorno de trabajo.

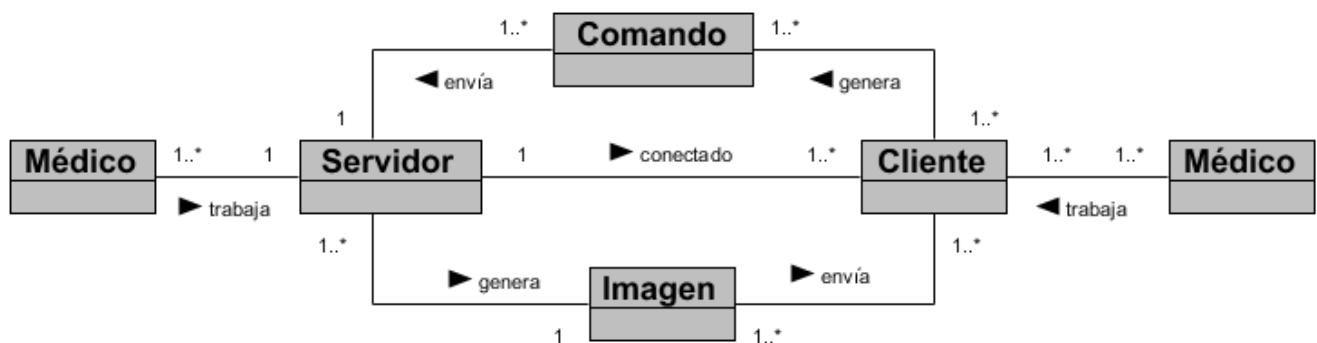


Figura 8: Modelo de dominio.

El **médico** especialista es aquel que esté adecuadamente capacitado en el manejo de los métodos de imagenología diagnóstica.

Las **imágenes** constituyen el resultado del estudio orientado por el médico al paciente a través del TAC y MR.

El sistema **cliente** es el programa que envía peticiones a un servidor y espera respuestas para seguir trabajando.

El sistema **servidor** es un programa que ofrece un servicio que se puede obtener en una red. En su forma más simple, es un programa que acepta peticiones a través de la red y que en función de esas peticiones ofrece una respuesta.

Un **comando** es la acción que se realiza y genera un dato que responde a una orden determinada.

3.2 Captura de requisitos.

Los requisitos son condiciones o capacidades que debe tener un sistema o un componente de un sistema para satisfacer un contrato, norma, especificación u otro documento formal, facilitando el entendimiento entre clientes y desarrolladores. La captura de requisitos es de vital importancia en todo sistema. Estos pueden ser funcionales y no funcionales. Los requisitos funcionales son los que especifican las funcionalidades que debe de tener el sistema, mientras que los requisitos no funcionales especifican las características que desea el cliente que tenga el sistema. Los siguientes epígrafes tienen como objetivo identificar los requisitos identificados para la solución propuesta.

3.2.1 Requisitos funcionales.

Los requisitos funcionales representan las funcionalidades del sistema. Estos son transformados en casos de uso y modelados posteriormente. Los siguientes requisitos responden a las funcionalidades que el sistema debe tener una vez concluida su implementación.

3.2.1.1 Requisitos funcionales del servidor.

El sistema servidor debe permitir:

RF1. Establecer Conexión.

RF2. Interactuar con el Modelo.

RF2.1. Rotar hacia Izquierda-Derecha.

RF2.2. Rotar hacia Arriba-Abajo.

RF2.3. Acercar-Alejar.

RF3. Procesar Petición.

RF3.1. Solicitar Conexión.

RF3.2. Ejecutar Comando.

RF3.3. Solicitar Desconexión.

RF4. Desconectar Servidor.

3.2.1.2 Requisitos funcionales del cliente.

El sistema cliente debe permitir:

RF1. Establecer Conexión con el Servidor.

RF2. Procesar Dato Recibido.

RF2.1. Desconectar del Servidor.

RF2.2. Procesar Imagen Recibida.

RF2.3. Procesar Comando Recibido.

RF3. Interactuar con la Escena.

RF3.1. Rotar hacia Izquierda-Derecha.

RF3.2. Rotar hacia Arriba-Abajo.

RF3.3. Acercar-Alejar.

RF4. Desconectar Cliente.

3.2.2 Requisitos no funcionales.

Los requisitos no funcionales representan aquellos atributos que debe tener el sistema, pero que no son una funcionalidad específica. Por ejemplo requisitos de facilidad de uso, fiabilidad, eficiencia, portabilidad.

Además el sistema debe mantener las reglas del negocio definidas en el proyecto Vismedic para la aplicación en desarrollo [28] [29]. Con la particularidad de que el sistema debe estar conectado a una red de área local.

3.2.2.1 Requisitos no funcionales del servidor.

Los requisitos no funcionales que debe tener el sistema servidor son los siguientes:

RNF1. Software

- Se debe de tener instaladas las bibliotecas de OpenGL y Glew para uso de técnicas implementadas con Shader.
- Se debe de tener instalada la biblioteca JRTPLib para la redes.

RNF2. Hardware

- El modelo de microprocesador será Intel Pentium IV a 3.0 GHz o superior.
- La memoria RAM será de 1GB o superior.

- La tarjeta gráfica usada será NVidia GeForce 9800 GT de 512 MB o superior.
- Conectividad LAN o inalámbrica superior a los 100 MB/S.

RNF3. Seguridad

- El sistema está protegido antes ataques a la red gracias a la técnica multidifusión que solo envía mensajes a los clientes conectados. Un cliente se desconecta cuando se deja de recibir mensajes de este. Si los mensajes recibidos no tienen el formato definido por el sistema propuesto estos son ignorados.

RNF4. Disponibilidad.

- El servidor se puede ejecutar en el momento que se desee, ejecutando la función de conexión (establece parámetros, escucha peticiones y envía respuestas).

RNF5. Interfaz externa

- La interfaz de usuario debe ser sencilla y amigable para permitir al usuario una rápida y cómoda interacción con las funcionalidades del sistema.

RNF6. Soporte

- Se brindará soporte para los sistemas operativos Windows XP o superior.

RNF7. Diseño e Implementación

- Se empleará como lenguaje de programación C++ y como ambiente de desarrollo Visual Studio 2008.

3.2.2.2 Requisitos no funcionales del cliente.

Los requisitos no funcionales que debe de tener el sistema cliente son los siguientes:

RNF1. Software

- Se debe de tener instaladas las bibliotecas de *OpenGL* y *JRTPLib*.

RNF2. Hardware

- Cliente Ligerero o superior.
- Conectividad LAN o inalámbrica superior a los 100 MB/S.

RNF3. Seguridad

- El sistema está protegido antes ataques a la red gracias a la técnica multidifusión que solo recibe mensajes de el servidor al que está conectado. Si los mensajes recibidos no tienen el formato definido

por el sistema propuesto estos son ignorados. Si no se recibe la totalidad de mensajes que forman el volumen de dato, se desecha y se capturan los del próximo conjunto de datos.

RNF4. Disponibilidad.

- El cliente solo funciona si existe un sistema servidor ejecutándose, prestando servicios.

RNF5. Interfaz externa

- La interfaz de usuario debe ser sencilla y amigable para permitir al usuario una rápida y cómoda interacción con las funcionalidades del sistema.

RNF6. Soporte

- Se brindará soporte para los sistemas operativos *Windows XP* o superior.

RNF7. Diseño e Implementación

- Se empleará como lenguaje de programación C++ y como ambiente de desarrollo Visual Studio 2008.

3.3 Modelo de casos de uso del sistema.

En este epígrafe se identificarán los actores del sistema, así como los casos de uso. Además se hará la descripción textual de los casos de uso del sistema, permitiendo así una comprensión más precisa de la lógica del funcionamiento.

3.3.1 Actores del sistema.

Los actores del sistema son entidades externas al sistema que guardan una relación con este y que le demandan una o más funcionalidades. Esto incluye a los operadores humanos, pero también incluye a todos los sistemas externos.

3.3.1.1 Actores del sistema servidor.

Tabla 2: Actores del sistema servidor.

Actores	Justificación
Médico Especialista	Interactúa con el sistema servidor durante la ejecución de los requisitos funcionalidades RF1, RF2 y RF4.

3.3.1.2 Actores del sistema cliente.

Tabla 3: Actores del sistema cliente.

Actores	Justificación
Médico Especialista	Interactúa con el sistema cliente durante la ejecución de los requisitos funcionales RF1, RF3 y RF4.

3.3.2 Diagrama de casos de uso del sistema.

El diagrama de casos de uso del sistema (DCUS) representa gráficamente los casos de uso y su interacción con los actores.

3.3.2.1 Diagrama de casos de uso del sistema servidor.

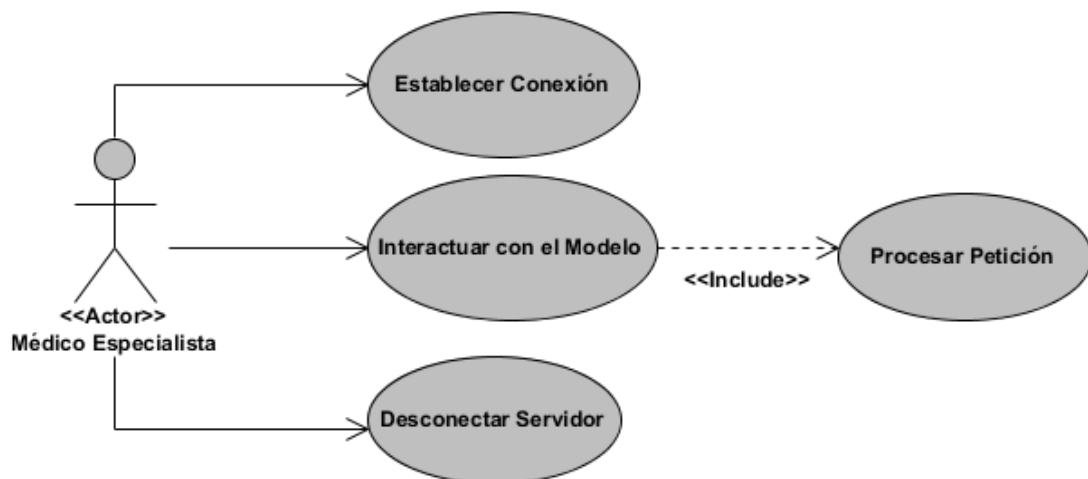


Figura 9: Diagrama de casos de uso del sistema servidor.

3.3.2.2 Diagrama de casos de uso del sistema cliente.

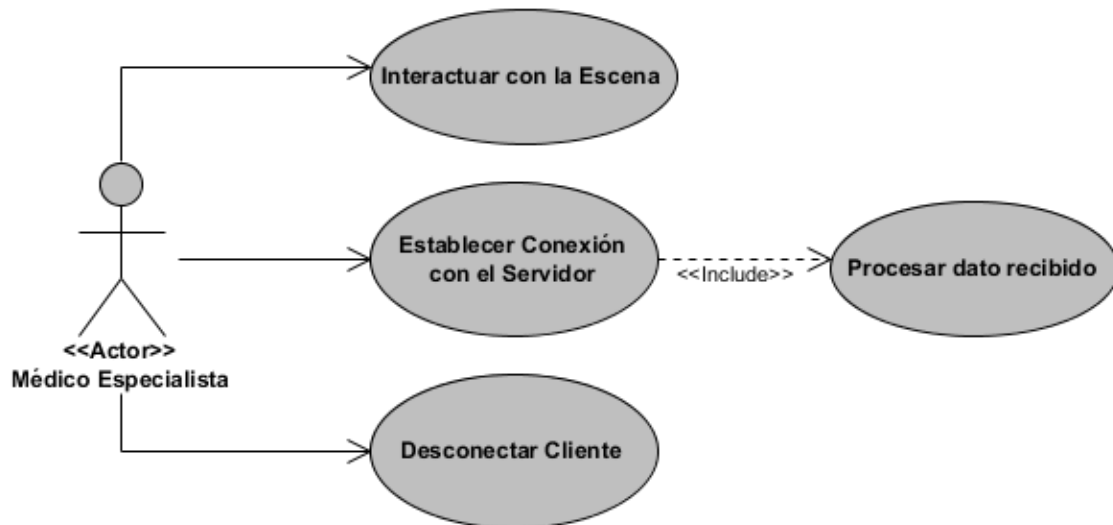


Figura 10: Diagrama de casos de uso del sistema cliente.

3.3.3 Descripción de casos de uso del sistema.

Cada caso de uso posee una descripción de las acciones que realizará el sistema como respuesta a las peticiones del usuario. A continuación se relacionan las tablas correspondientes a las descripciones de los casos de uso detectados y se argumentan los flujos operacionales de cada uno.

3.3.3.1 Descripción de casos de uso del sistema servidor.

Tabla 4: Descripción del caso de uso Establecer Conexión.

Caso de Uso:	Establecer Conexión
Actores:	Médico
Propósito:	Establecer conexión del servidor.
Resumen:	Con la conexión, el servidor comienza a recibir solicitudes de conexión, mensajes con peticiones y visualiza el modelo 3D.
Referencia:	RF1.
Precondición:	N/A
Flujo Normal de Eventos:	

Actividades:	Respuesta del Sistema:
1. Ejecuta el sistema servidor.	<p>1.1 El Sistema abre una conexión a partir de su dirección IP y un puerto de red prefijado.</p> <p>1.2 Establece los parámetros de conexión prefijados en el sistema.</p> <p>1.3 Comienza a escuchar peticiones de clientes conectados a la red.</p> <p>1.4 Visualiza el modelo cargado de fichero.</p>
Postcondiciones:	Se establece la conexión del servidor listo para dar servicio.
Prioridad:	Crítica.

Tabla 5: Descripción del caso de uso Interactuar con el Modelo.

Caso de Uso:	Interactuar con el Modelo.
Actores:	Médico
Propósito:	Movimiento del modelo para una exploración por la escena.
Resumen:	Con el movimiento con las teclas direccionales más otras del teclado y el uso del botón izquierdo del mouse, el médico manipula los movimientos permisibles de la escena virtual.
Referencia:	RF2.
Precondición:	Que la conexión esté establecida
Flujo Normal de Eventos:	
Actividades:	Respuesta del Sistema:
1. Utiliza el botón izquierdo del mouse o las flechas direccionales del teclado combinándolos con movimientos (izquierda, derecha, arriba, abajo) y las teclas “+” y “-“.	<p>1.1 Realiza un conjunto de acciones que se detallan a continuación:</p> <ul style="list-style-type: none"> a) Si el actor emplea la tecla direccional izquierda o derecha: consultar Sección Rotar hacia Izquierda-Derecha. b) Si el actor emplea la tecla direccional arriba o abajo: consultar Sección Rotar hacia Arriba-Abajo. c) Si el actor sostiene el botón izquierdo del mouse y arrastra: consultar Sección hacia Izquierda-Derecha y Sección hacia Arriba-Abajo.

	<p>d) Si el actor emplea las teclas “+” y “-”: consultar Sección Acercar-Alejar.</p> <p>1.2 Se actualiza la escena, posicionando el modelo según los eventos recibidos.</p> <p>1.3 Envía la nueva imagen generada a los clientes.</p>
Postcondiciones:	Se representó el modelo en la escena con la nueva posición y orientación.
Prioridad:	Crítica.

Tabla 6: Descripción de la Sección Rotar hacia Izquierda-Derecha. Caso de uso Interactuar con el Modelo.

Sección:	Rotar hacia Izquierda-Derecha
Flujo Normal de Eventos:	
Actividades:	Respuesta del Sistema:
1. Realiza las operaciones que permite el botón izquierdo del mouse arrastrando hacia la izquierda o derecha y las teclas direccional izquierda o derecha del teclado.	1.1 Recibe los eventos correspondientes y rota el modelo. O sea rotación hacia la izquierda y la derecha
Postcondiciones:	Se representó el modelo en la nueva posición.
Prioridad:	Crítica.

Tabla 7: Descripción de la Sección Rotar hacia Arriba-Abajo. Caso de uso Interactuar con el Modelo.

Sección:	Rotar hacia Arriba-Abajo
Flujo Normal de Eventos:	
Actividades:	Respuesta del Sistema:
1. Realiza las operaciones que permite el botón izquierdo del mouse arrastrando hacia la arriba o abajo y las teclas direccional arriba o abajo del teclado.	1.1 Recibe los eventos correspondientes y rota el modelo. O sea rotación hacia atrás (arriba) y hacia delante (abajo).
Postcondiciones:	Se representó el modelo en la nueva posición.
Prioridad:	Crítica.

Tabla 8: Descripción de la Sección Acercar-Alejar. Caso de uso Interactuar con el Modelo.

Sección:	Acercar-Alejar
Flujo Normal de Eventos:	
Actividades:	Respuesta del Sistema:
1. Utiliza las teclas “+” y “-” teclado.	1.1 Recibe los eventos correspondientes, acerca (“+”) o aleja (“-”) el modelo representado en la escena.
Postcondiciones:	Se representó el modelo en la nueva posición.
Prioridad:	Crítica.

Tabla 9: Descripción del Caso de uso Procesar Petición.

Caso de Uso:	Procesar Petición.
Actores:	Sistema
Propósito:	Solventar las peticiones recibidas de los clientes.
Resumen:	Cada petición recibida es procesada, generando la respuesta adecuada a cada solicitud.
Referencia:	RF3.
Precondición:	Que la conexión este establecida.
Flujo Normal de Eventos:	
Actividades:	Respuesta del Sistema:
1. Recibe la petición.	<p>1.1 Realiza un conjunto de acciones que se detallan a continuación:</p> <ul style="list-style-type: none"> a) Si la petición recibida es de conexión: consultar la Sección Solicitar Conexión. b) Si la petición recibida es un comando: consultar la Sección Ejecutar Comando. c) Si la petición recibida es de desconexión: consultar la Sección Solicitar Desconexión. <p>1.2 Se actualiza el sistema servidor con la solicitud recibida.</p>
Postcondiciones:	Se responde a la solicitud recibida.

Prioridad:	Crítica.
------------	----------

Tabla 10: Descripción de la Sección Solicitar Conexión. Caso de uso Procesar Petición.

Sección:	Solicitar Conexión.
Flujo Normal de Eventos:	
Actividades:	Respuesta del Sistema:
1. Recibe un mensaje con la petición de solicitar conexión de una estación de trabajo cliente al servidor.	1.1 Adiciona el cliente (a través de su dirección <i>IP</i>) a la lista multidifusión. 1.2 Envía la lista de comandos actualizados a los clientes.
Postcondiciones:	La lista de multidifusión queda actualizada con la incorporación del nuevo cliente.
Prioridad:	Crítica.

Tabla 11: Descripción de la Sección Ejecutar Comando. Caso de uso Procesar Petición.

Sección:	Ejecutar Comando.
Flujo Normal de Eventos:	
Actividades:	Respuesta del Sistema:
1. Recibe un mensaje que constituye un comando a ejecutar en la escena sobre el modelo que se procesa en el servidor.	1.1 Se aplica el comando al sistema servidor. 1.2 Se actualiza el modelo en la escena. 1.3 Se envía la nueva imagen de la escena a la lista multidifusión de clientes. 1.4 Se envía el comando procesado a la lista multidifusión de clientes.
Postcondiciones:	Se representó el modelo en la escena con la nueva representación según el comando recibido.
Prioridad:	Crítica.

Tabla 12: Descripción de la Sección Solicitar Desconexión. Caso de uso Procesar Petición.

Sección:	Solicitar Desconexión.
Flujo Normal de Eventos:	

Actividades:	Respuesta del Sistema:
1. Recibe un mensaje de desconexión del servidor.	1.1 Se elimina el cliente (su dirección IP) de la lista multidifusión de clientes.
Postcondiciones:	La lista de multidifusión queda actualizada con la eliminación del cliente en cuestión.
Prioridad:	Crítica.

Tabla 13: Descripción del caso de uso Desconectar Servidor.

Caso de Uso:	Desconectar Servidor.
Actores:	Médico
Propósito:	Terminar la conexión existente del sistema servidor.
Resumen:	El servidor conectado procesando solicitudes se desconecta de la red cerrando las conexiones con sus clientes asociados y la propia.
Referencia:	RF4.
Precondición:	Que la conexión este establecida.
Flujo Normal de Eventos:	
Actividades:	Respuesta del Sistema:
1. Cerrar el Sistema Servidor.	1.1 Envía un mensaje de desconexión a los clientes asociados. 1.2 Cierra todas las conexiones establecidas: con los clientes y el puerto en uso. 1.3 Cierra el sistema servidor.
Postcondiciones:	Se cierra el sistema servidor. Los clientes no reciben más paquetes del servidor y este queda inhabilitado para procesar solicitudes.
Prioridad:	Crítica.

3.3.3.2 Descripción de casos de uso del sistema cliente.

Tabla 14: Descripción del caso de uso Establecer Conexión con el Servidor.

Caso de Uso:	Establecer Conexión con el Servidor.	
Actores:	Médico	
Propósito:	Conectar el cliente al servidor que este corriendo.	
Resumen:	Con la introducción de la dirección <i>IP</i> donde corre el sistema servidor se establece la conexión del cliente con este, quedando listo para la interacción entre estos.	
Referencia:	RF1.	
Precondición:	N/A	
Flujo Normal de Eventos:		
Actividades:	Respuesta del Sistema:	
1. Teclea la dirección <i>IP</i> del servidor con el formato decimal (Ejemplo: 10.8.67.156) y presionar entrar (tecla <i>ENTER</i>).	1.1 Al teclear el <i>IP</i> y presionar <i>ENTER</i> se envía la solicitud de conexión al servidor. 1.2 Se establecen los parámetros de conexión. 1.3 Se establece la conexión con el servidor agregándolo a su dirección destino para el envío y recepción de paquetes.	
Postcondiciones:	Se establece la conexión con el Servidor.	
Prioridad:	Crítica.	

Tabla 15: Descripción del caso de uso Procesar Dato Recibido.

Caso de Uso:	Procesar Dato Recibido.	
Actores:	Sistema	
Propósito:	Procesar todos los mensajes recibidos desde el servidor.	
Resumen:	Los paquetes enviados por el servidor, son recibidos y procesado según la información que dispongan.	
Referencia:	RF2.	
Precondición:	Que la conexión este establecida.	

Flujo Normal de Eventos:	
Actividades:	Respuesta del Sistema:
1. Recibe el dato por la red proveniente del servidor.	<p>1.1 Realiza un conjunto de acciones que se detallan a continuación:</p> <ul style="list-style-type: none"> a) Si el actor recibe un mensaje de servidor desconectado: consultar Sección Desconectar del Servidor. b) Si el actor recibe un mensaje de dato de imagen: consultar Sección Procesar Imagen Recibida. c) Si el actor recibe un mensaje de comando: consultar Sección Actualizar Comando. <p>1.2 Se actualiza la escena con los nuevos parámetros recibidos.</p>
Postcondiciones:	Se representa la escena con la actualización de sus parámetros.
Prioridad:	Crítica.

Tabla 16: Descripción de la Sección Desconectar del Servidor. Caso de uso Procesar Dato Recibido.

Sección:	Desconectar del Servidor.
Flujo Normal de Eventos:	
Actividades:	Respuesta del Sistema:
1. Recibe un mensaje de desconexión del servidor.	1.1 Se cierra la conexión con el servidor.
Postcondiciones:	El cliente no recibe más mensajes por la red.
Prioridad:	Crítica.

Tabla 17: Descripción de la Sección Procesar Imagen Recibida. Caso de uso Procesar Dato Recibido.

Sección:	Procesar Dato Recibido
Flujo Normal de Eventos:	
Actividades:	Respuesta del Sistema:

1. Recibe un mensaje con dato de la nueva imagen a mostrar en la escena.	1.1 Se acumulan todos los mensajes de la imagen. 1.2 Se reconstruye la imagen recibida. 1.3 Se representa la imagen en la escena.
Postcondiciones:	Se actualiza la escena con la nueva imagen recibida.
Prioridad:	Crítica.

Tabla 18: Descripción de la Sección Procesar Comando Recibido. Caso de uso Procesar Comando Recibido.

Sección:	Procesar Comando Recibido.
Flujo Normal de Eventos:	
Actividades:	Respuesta del Sistema:
1. Recibe un mensaje correspondiente a un comando.	1.1 Actualiza el comando correspondiente en la lista de comandos.
Postcondiciones:	Queda actualizada la lista de comandos del cliente.
Prioridad:	Crítica.

Tabla 19: Descripción del caso de uso Interactuar con la Escena.

Caso de Uso:	Interactuar con la Escena.
Actores:	Médico
Propósito:	Movimiento de mouse y teclado para una exploración de la escena.
Resumen:	Con el movimiento con las teclas direccionales más otras del teclado y el uso del botón izquierdo del mouse, el médico manipula los movimientos permisibles de la escena virtual.
Referencia:	RF3.
Precondición:	Que la conexión este establecida.
Flujo Normal de Eventos:	
Actividades:	Respuesta del Sistema:
1. Utiliza el botón izquierdo del mouse o las flechas direccionales del teclado combinándolos con movimientos (izquierda,	1.1 Realiza un conjunto de acciones que se detallan a continuación:

derecha, arriba, abajo) y las teclas “+” y “-“.	<p>a) Si el actor emplea la tecla direccional izquierda o derecha: consultar Sección Rotar hacia Izquierda-Derecha.</p> <p>b) Si el actor emplea la tecla direccional arriba o abajo: consultar Sección Rotar hacia Arriba-Abajo.</p> <p>c) Si el actor sostiene el botón izquierdo del mouse y arrastra: consultar Sección Rotar hacia Izquierda-Derecha y Sección Rotar hacia Arriba-Abajo.</p> <p>d) Si el actor emplea las teclas “+” y “-“: consultar Sección Acercar-Alejar.</p> <p>1.2 Se envía un mensaje con el comando generado de la acción realizada.</p>
Postcondiciones:	Se ejecuto un comando y se envió al servidor.
Prioridad:	Crítica.

Tabla 20: Descripción de la Sección Rotar hacia Izquierda-Derecha. Caso de uso Interactuar con la Escena.

Sección:	Rotar hacia Izquierda-Derecha
Flujo Normal de Eventos:	
Actividades:	Respuesta del Sistema:
1. Realiza las operaciones que permite el botón izquierdo del mouse arrastrando hacia la izquierda o derecha y las teclas direccional izquierda o derecha del teclado.	<p>1.1 Recibe los eventos correspondientes y rota la escena por su eje Vertical. O sea rotación hacia la izquierda y la derecha.</p> <p>1.2 Se genera un mensaje con la acción realizada (comando ejecutado).</p>
Postcondiciones:	Se ejecutó una acción en el cliente y se envió al servidor el comando.
Prioridad:	Crítica.

Tabla 21: Descripción de la Sección Rotar hacia Arriba-Abajo. Caso de uso Interactuar con la Escena.

Sección:	Rotar hacia Arriba-Abajo
Flujo Normal de Eventos:	
Actividades:	Respuesta del Sistema:
1. Realiza las operaciones que permite el	1.1 Recibe los eventos correspondientes y rota el modelo

botón izquierdo del mouse arrastrando hacia la arriba o abajo y las teclas direccional arriba o abajo del teclado.	por su eje Horizontal. O sea rotación hacia atrás (arriba) y hacia delante (abajo). 1.2 Se genera un mensaje con la acción realizada (comando ejecutado).
Postcondiciones:	Se ejecutó una acción en el cliente y se envió al servidor el comando.
Prioridad:	Crítica.

Tabla 22: Descripción de la Sección Acercar-Alejar. Caso de uso Interactuar con la Escena.

Sección:	Acercar-Alejar
Flujo Normal de Eventos:	
Actividades:	Respuesta del Sistema:
1. Utiliza las teclas “+” y“-” teclado.	1.1 Recibe los eventos correspondientes, acerca (“+”) o aleja (“-”) el modelo representado en la escena. 1.2 Se genera un mensaje con la acción realizada (comando ejecutado).
Postcondiciones:	Se ejecutó una acción en el cliente y se envió al servidor el comando.
Prioridad:	Crítica.

Tabla 23: Descripción del caso de uso Desconectar Cliente.

Caso de Uso:	Desconectar Cliente.
Actores:	Médico
Propósito:	Terminar la conexión existente con el sistema servidor
Resumen:	Se cierra la conexión con el servidor, no se recibe más paquetes por la red. Cierre del cliente.
Referencia:	RF4.
Precondición:	Que la conexión este establecida.
Flujo Normal de Eventos:	
Actividades:	Respuesta del Sistema:

1. Cerrar el sistema cliente.	1.1. Envía un mensaje de desconexión al servidor 1.2 Cierra todas las conexiones establecidas: puerto de red, dirección <i>IP</i> del servidor. 1.3 Cierra el sistema cliente.
Postcondiciones:	El servidor no envía más paquete al cliente. Se cierra el sistema cliente.
Prioridad:	Crítica.

3.4 Diseño del sistema.

En el flujo de trabajo de diseño, la elaboración de los diagramas de clases de diseño juega un papel fundamental, estos muestran las clases finales para la realización de los casos de uso modelados con anterioridad. Los diagramas de clases de diseño muestran las clases con sus atributos y métodos y la forma en que se relacionan entre sí. En este flujo también se realizan los diagramas de interacción que muestran la comunicación y las relaciones entre los objetos, con el objetivo de dar cumplimiento a los requerimientos. En la Figura 11 se muestra el diagrama de paquetes general del sistema. Para obtener un sistema modular, el mismo se ha dividido en dos paquetes fundamentales: Sistema Servidor y Sistema Cliente.

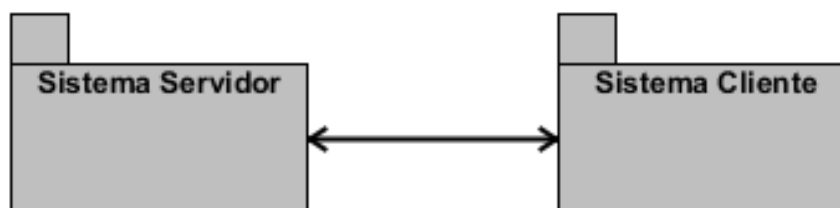


Figura 11: Diagrama de paquetes del sistema.

3.4.1 Diagrama de clases de diseño del servidor.

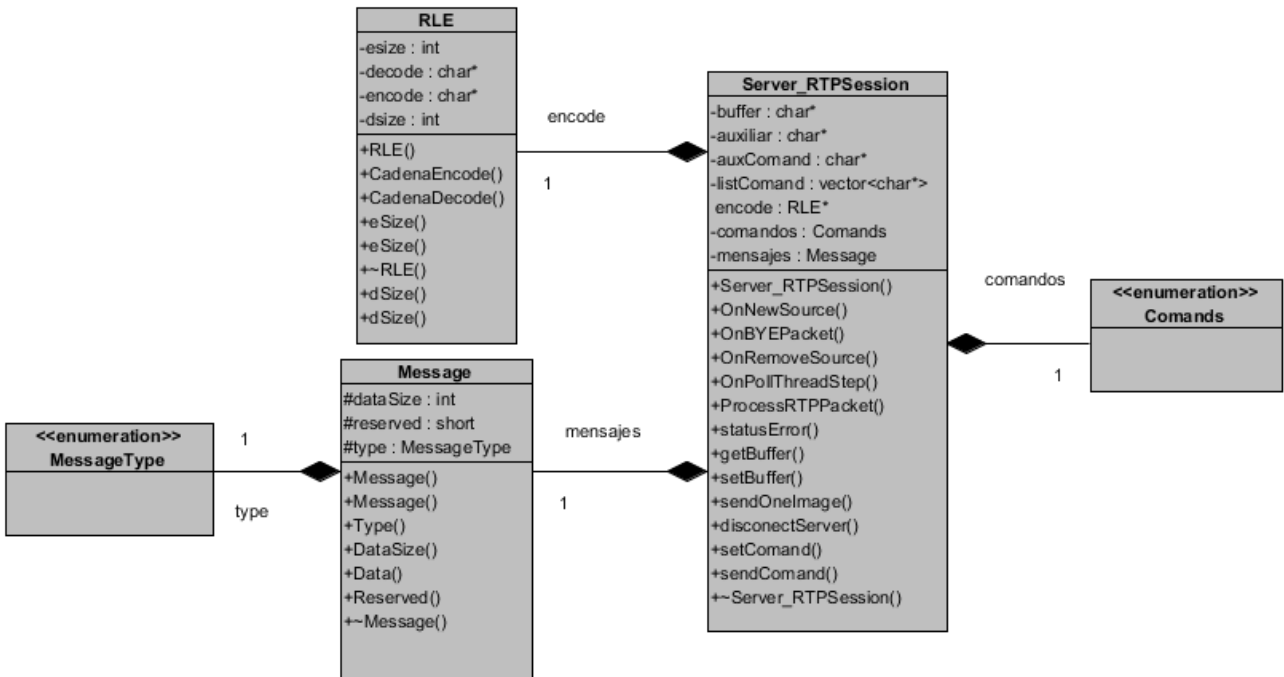


Figura 12: Diagrama de clases de diseño del servidor.

3.4.2 Diagrama de secuencia de diseño del servidor.

Los siguientes **diagramas de secuencia de diseño del servidor** incluyen los **diagramas de secuencias** de las **secciones** del **caso de uso Procesar Dato Recibido** del **sistema cliente** dada la interconexión de ambos sistemas.

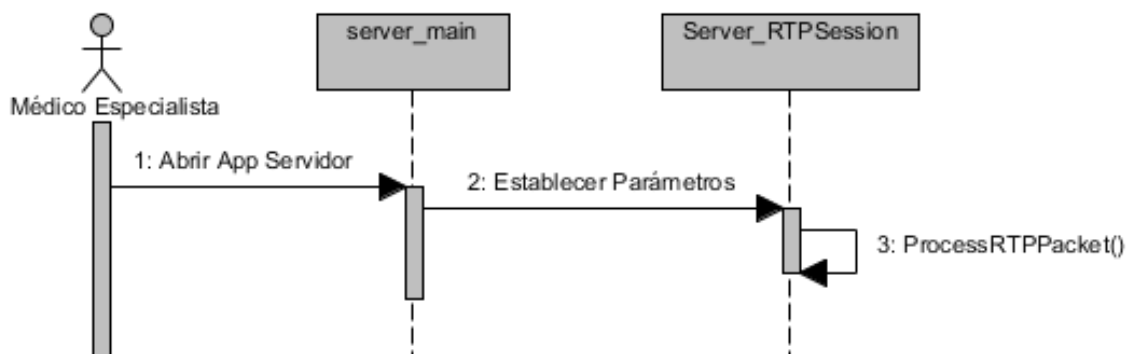


Figura 13: Diagrama de secuencia del caso de uso Establecer Conexión.

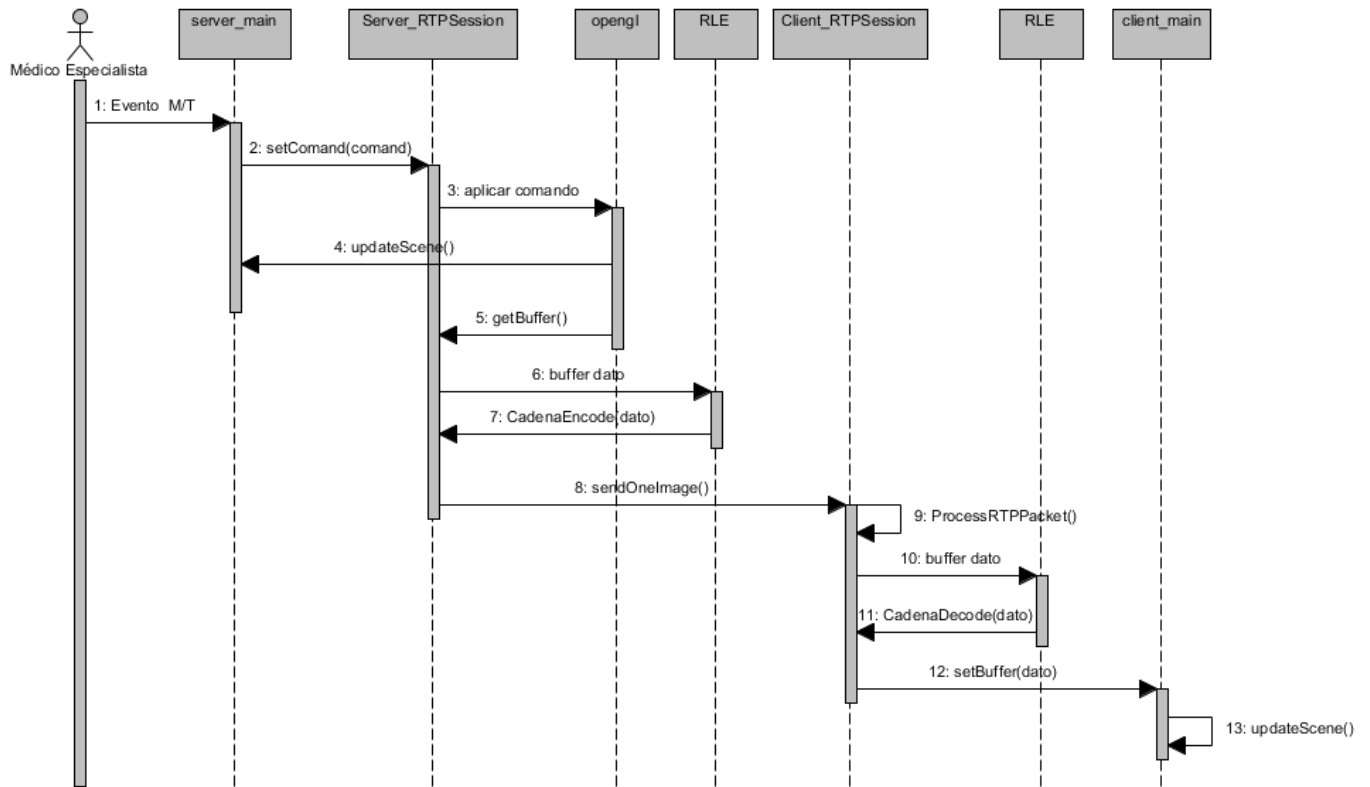


Figura 14: Diagrama de secuencia del caso de uso Interactuar con el Modelo.

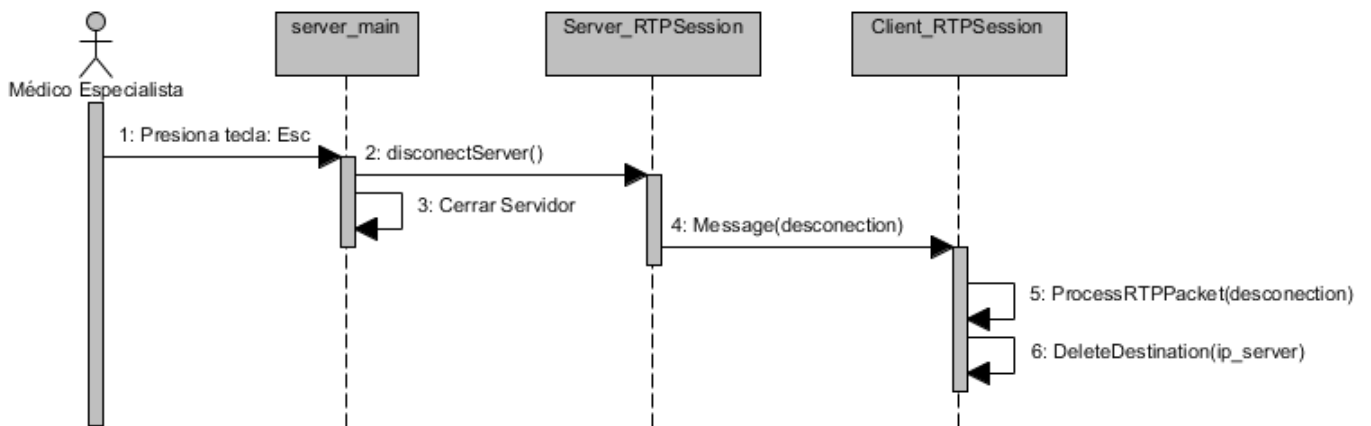


Figura 15: Diagrama de secuencia del caso de uso Desconectar Servidor.

3.4.3 Diagrama de clases de diseño del cliente.

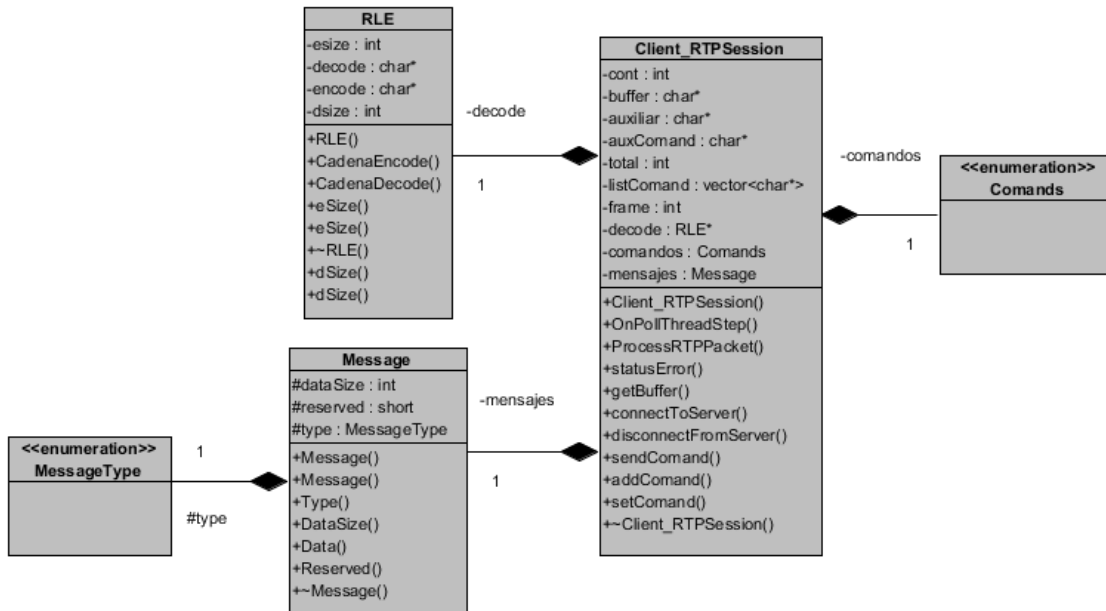


Figura 16: Diagrama de clases de diseño del cliente.

3.4.4 Diagrama de secuencia de diseño del cliente.

Los siguientes diagramas de secuencia de diseño del cliente incluyen los diagramas de secuencias de las secciones del caso de uso Procesar Petición del sistema servidor dada la interconexión de ambos sistemas.

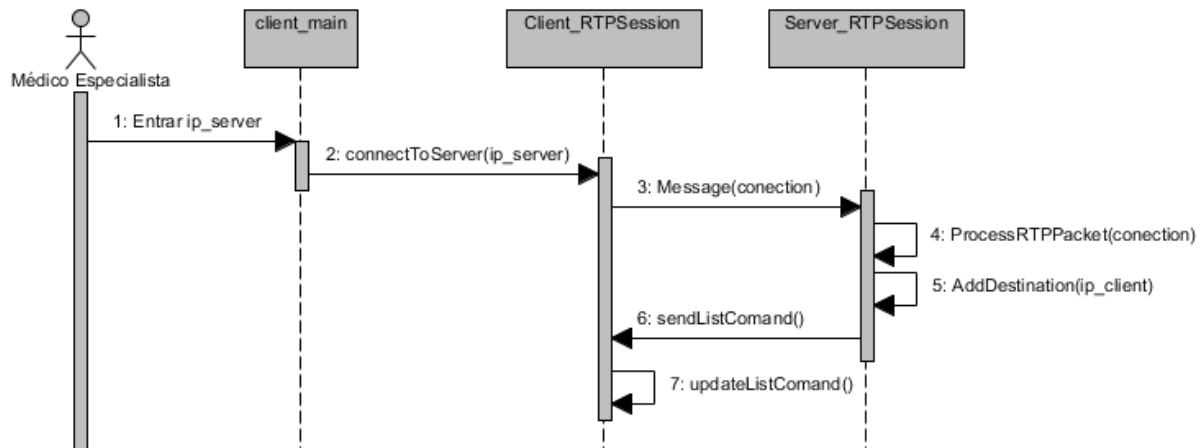


Figura 17: Diagrama de secuencia del caso de uso Establecer Conexión con el Servidor.

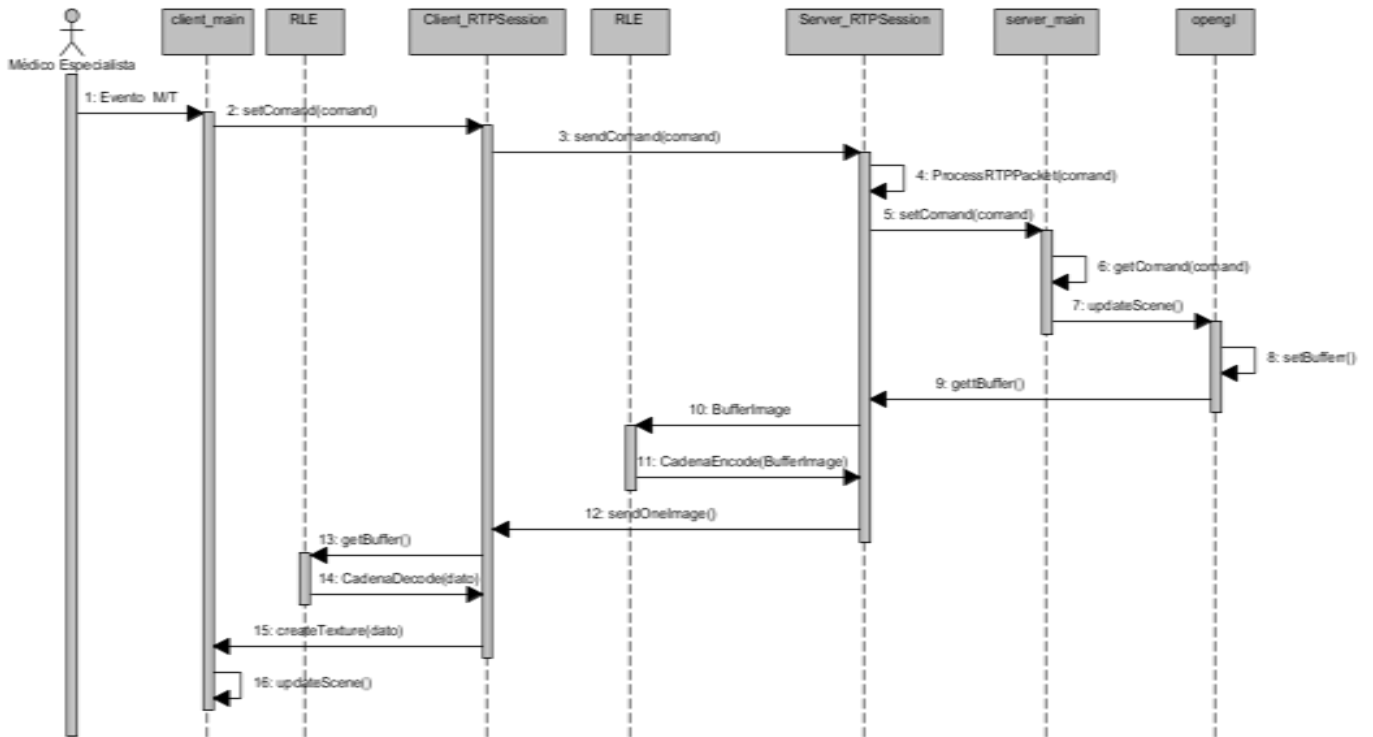


Figura 18: Diagrama de secuencia del caso de uso Interactuar con la Escena.

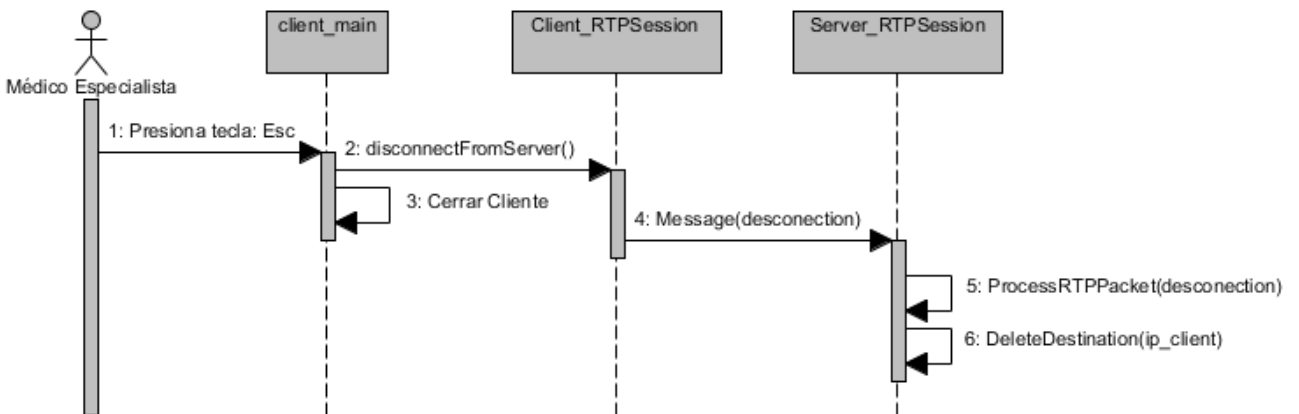


Figura 19: Diagrama de secuencia del caso de uso Desconectar Cliente.

3.4.5 Diagrama de despliegue.

El modelo de despliegue es un modelo de objeto que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre nodos de cómputo. Se utiliza como entrada

fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño [30].



Figura 20: Diagrama de despliegue.

3.5 Consideraciones Generales.

En el capítulo se definió el modelo de dominio a utilizar y se hizo una captura de los requisitos (funcionales y no funcionales) más importante en el desarrollo del sistema, de esta manera quedó elaborado el modelo de casos de uso. Se especificó cada caso de uso obtenido a partir de los requisitos con la descripción de cada uno para una mejor comprensión de los procesos. Se especifica el actor Médico Especialista tanto para el cliente como para el servidor. Como parte del diseño del sistema se modelaron los diagramas de clases y de secuencias, lográndose un mejor entendimiento del cliente y el servidor. Además se especifica el diagrama de despliegue como distribución física del sistema implementado. Con todos los elementos desarrollados se logra el entendimiento del sistema cliente-servidor, se reflejan las clases que intervienen, la lógica de los procesos y la interconexión del sistema.

CAPÍTULO 4. IMPLEMENTACIÓN Y VALIDACIÓN DE LOS RESULTADOS

En el presente capítulo se abordarán los temas de resultados de la implementación del sistema, dado el trabajo realizado en los capítulos anteriores. Se modelará el diagrama de componentes utilizado y se harán algunas pruebas para validar los resultados del algoritmo de compresión de imágenes implementado y medir aspectos como rendimiento y calidad de las imágenes obtenidas a partir de su transmisión por la red.

4.1 Implementación.

En esta etapa se realiza la implementación de las clases y objetos en ficheros fuente, binarios y ejecutables. Como resultado se obtiene un sistema ejecutable que incluye todas las funcionalidades propuestas en la captura de requisitos funcionales. La estructura de todos estos modelos forma el modelo de implementación.

4.1.1 Diagrama de componentes.

Un componente representa una parte física del sistema, por ejemplo, una biblioteca de clases, un ejecutable, una tabla, etc., que engloba la implementación de un grupo de clases del diseño. Cada componente define una interfaz que describe su funcionalidad y forma de empleo. El diagrama de componentes, permite conocer a los desarrolladores y clientes la estructura física que tiene el sistema y cómo se relacionan sus partes. El sistema propuesto está formado por clientes y un servidor, ambos tienen sus funcionalidades específicas por ellos se realizan sendos diagramas de componentes.

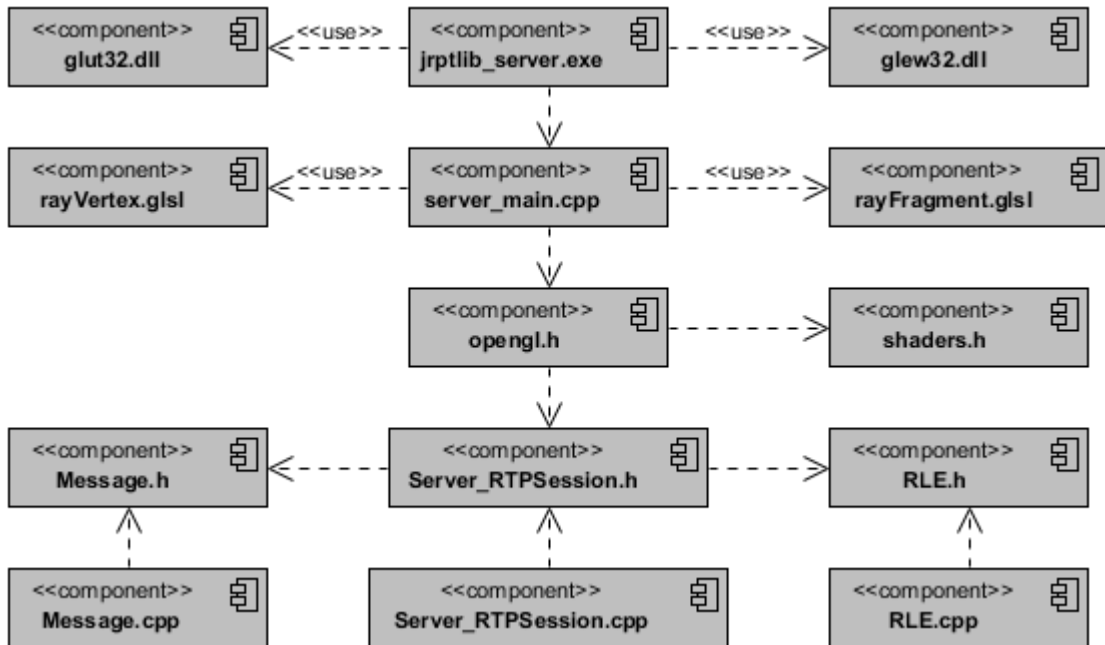


Figura 21: Diagrama de componentes sistema servidor.

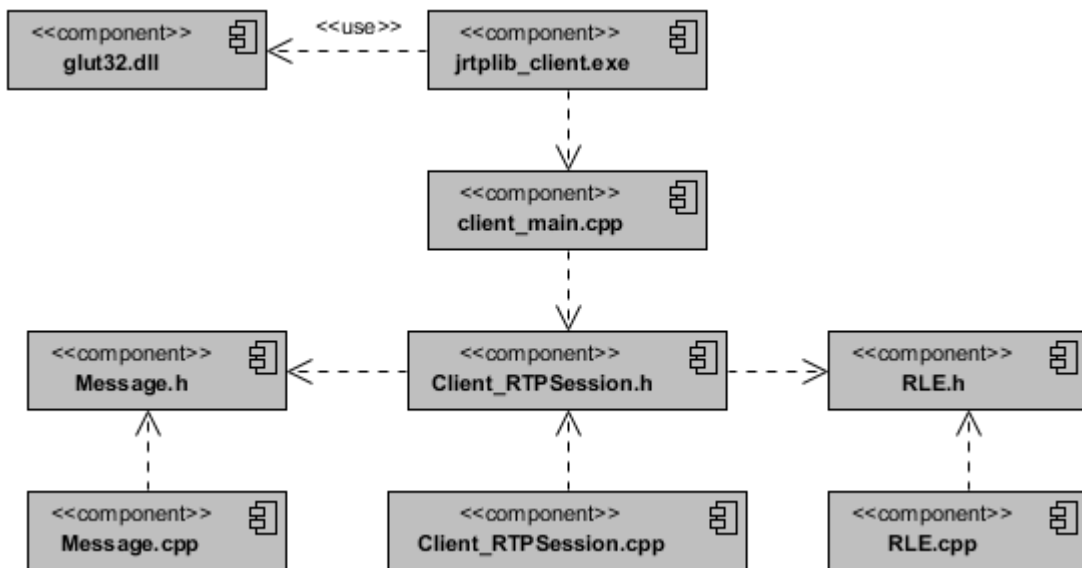


Figura 22: Diagrama de componentes sistema cliente.


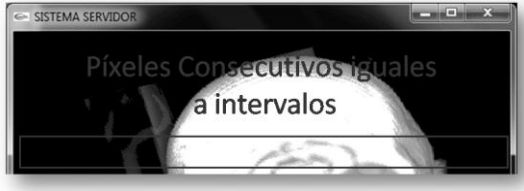
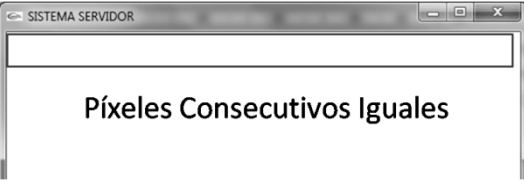
4.2 Validación de los resultados.

Esta sección muestra los principales resultados alcanzados con la implementación del sistema propuesto. Se tomaron en cuenta diferentes criterios para la validación del sistema como la compresión lograda con el algoritmo RLE implementado. El cuello de botella en aplicaciones de este tipo está en la capacidad del medio de comunicación, por ello se tiene en cuenta la cantidad de información transmitida, la velocidad de conexión y la calidad de las imágenes obtenidas en los clientes. Para la validación se utilizaron máquinas con las siguientes prestaciones: para el servidor se utilizó una computadora con un procesador Intel Core2 Quad Q6600 y una tarjeta gráfica NVidia 9800 GT con 512 MB de RAM para video y para los clientes se utilizó computadoras con procesador Pentium IV y 1 GB de RAM. En todos los casos se utilizó una red cableada de 100 Mb/s. El modelo tridimensional utilizado en las pruebas fue una cabeza humana cargada de fichero *.raw con dimensiones de 256x256x256 representada bajo técnica de simple *raycasting* en la escena del servidor. A continuación se presentan los resultados alcanzado con los sistemas servidor y clientes.

4.2.1 Tasa de compresión.

La siguiente tabla muestra la cantidad de información de las imágenes obtenidas a partir de la escena tridimensional del servidor, en ella se evidencia la **tasa de compresión (TC)** dada la complejidad de la imagen (mayor o menor cantidad de píxeles consecutivos iguales) a transmitir. La biblioteca de red JRTPLib permite el máximo de envío de paquetes de 64 Kb por lo que se define como máximo de información por paquete a enviar 60 Kb; de esta manera no se sobrecarga al límite las capacidades de la red y se evita la pérdida de paquetes. La **cantidad de símbolos inicial (CSI)** representa la información obtenida a partir de la imagen visualizada en la escena tridimensional. La **cantidad de símbolos final (CSF)** es la información luego de la compresión de la imagen con el algoritmo implementado. Dada la capacidad de envío por paquete, la **cantidad de paquetes (CP)** da el aproximado de cantidad de paquetes a enviar por cada imagen. Tomando en cuenta que la **dimensión de la escena (DE)** es 512x512 y que dada la complejidad de la misma se puede obtener la **compresión (Comp.):** ninguna, mínima, promedio y máxima. Esta tabla muestra la efectividad del algoritmo RLE en una escena donde el modelo se dispuso en distintas posiciones (distintos niveles de profundidad) para lograr diferentes tipos de secuencias de píxeles y así obtener las distintas tasas de compresión.

Tabla 24: Tasa de compresión de imágenes.

DE	CSI	CSF	CP	TC	Comp.	Ejemplo ilustrativo.
512x512	1048576	1048576	~17	0%	Ninguna	Imagen con menos de 3 píxeles consecutivos iguales.
512x512	1048576	761942	~12	~25%	Mínima	
512x512	1048576	521441	~8	~50%	Promedio	
		319696	~5	~70%		
512x512	1048576	214488	~3	~80%	Máxima	

De esta manera se puede observar como a mayor sea la compresión menor es la cantidad de paquetes a enviar por imagen. Las tasas de compresión son aceptables con promedios desde 50 a 70 por ciento de disminución de los datos de las imágenes. La compresión “Ninguna” y “Máxima” no tienen valor alguno en la práctica, en realidad siempre se obtienen compresiones medias y en escenas complejas en cuanto a la cantidad de píxeles consecutivos iguales. Los resultados obtenidos son alentadores y constituyen un pilar fundamental en la transmisión de datos por la red.

4.2.2 Flujo de información a través de la red.

Probado sobre una red cableada a 100 MB/s de velocidad, los resultados alcanzados se pueden ver de la siguiente forma. La cantidad de bytes por segundos (*B/sec.*) que envía el servidor es la suma de la cantidad que reciben los clientes conectados. Generalmente los clientes reciben la misma cantidad de información, aunque se da el caso en que se pierden paquetes en la red. Esto no es alarmante puesto que el protocolo sobre el que se implementó la aplicación (RTP) no cuenta con mecanismos de control para evitar que durante el flujo de los paquetes ocurran pérdidas de los mismos. No obstante el control para

que lleguen íntegramente los datos al cliente son manejados por la propia aplicación, evitando la presentación de información incorrecta (ejemplo una imagen incompleta). Una información importante transmitida por la red son los comandos que se generan aunque carecen de gran volumen de información y se transmiten con mejor facilidad. A continuación un ejemplo del tráfico en la red, representado un servidor y cuatro clientes:



Figura 23: Tráfico de la red en el sistema cliente-servidor.

En la figura se puede observar como el sistema tiene conectado los cuatros clientes, el modelo que se está visualizando en el servidor, se refleja en la cantidad de *B/sec* enviados (*send*, en inglés) y recibidos (*receive*, en inglés) por el servidor. Los datos enviados desde el servidor a todos los clientes representan los paquetes que conforman las imágenes obtenidas y actualización de comandos. Por otro lado lo recibido refleja la cantidad de información que ocupan los comandos generados en los clientes y enviados al servidor. Todo esto demuestra el uso de la multidifusión, la arquitectura cliente-servidor y que este sistema depende de la capacidad de enlace y ancho de banda de la red. Este último elemento importante en sistemas de transmisión de datos en tiempo real por una red.

4.2.3 Calidad de presentación de las imágenes.

El sistema servidor procesa el volumen de datos con aceleración gráfica, generando la visualización de la escena. La información de la escena (conjunto de píxeles) es transmitida con un procesamiento previo de compresión de la imagen obtenida y el envío a los sistemas clientes, en la cantidad de paquetes necesarios. Una vez recibida la imagen, se descomprime los datos y se crea la textura correspondiente. Esta imagen es mostrada en el cliente (simulando la visualización del modelo anatómico) lográndose la visualización remota del modelo tridimensional procesado por la GPU. Las imágenes generadas en el servidor tienen dimensiones de 512x512, formato *RGBA* y 1048576 (512x512x4) símbolos. Las imágenes recibidas en clientes tienen la misma dimensión, formato y cantidad de símbolos; por tanto igual nivel de detalle y calidad que las generadas en el sistema servidor. En la Figura 24, la imagen de la izquierda muestra el modelo renderizado en el servidor y la imagen de la derecha es obtenida en el proceso de renderizado, compresión, transmisión, descompresión y representación en el cliente.

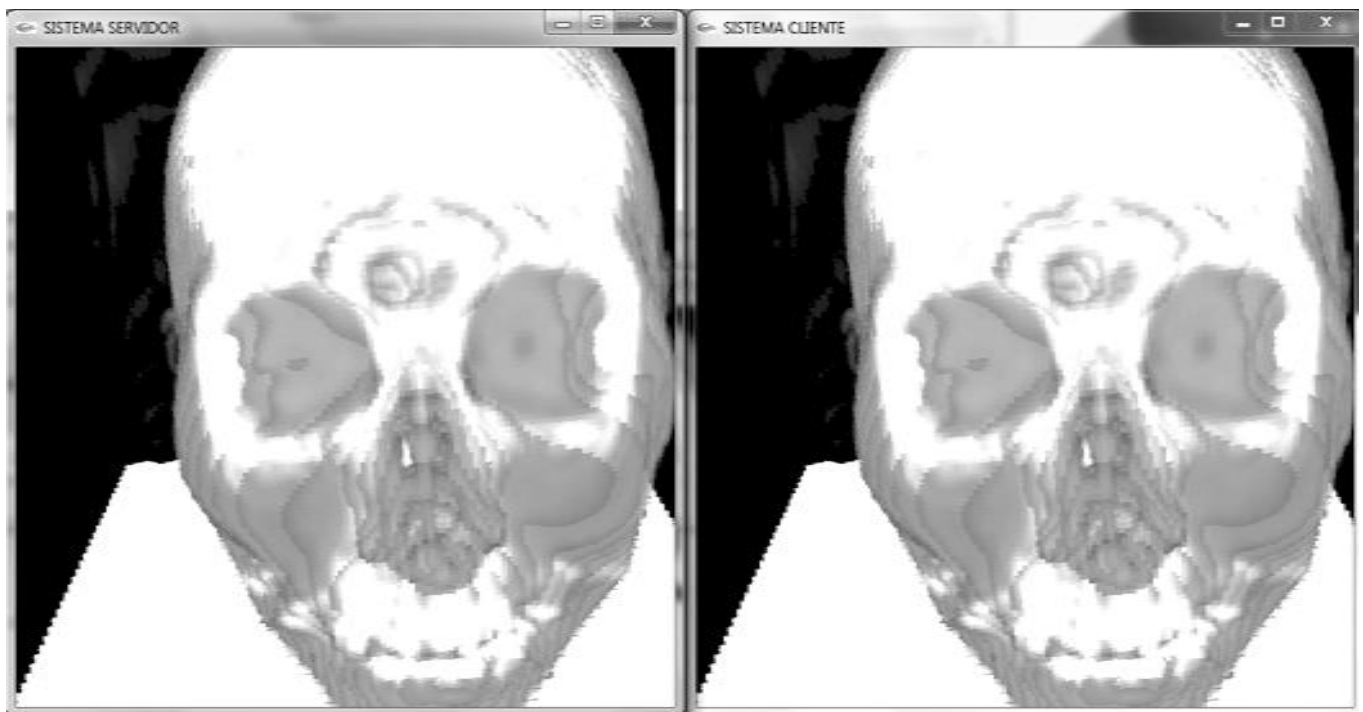


Figura 24: Modelo en el servidor (izquierda) e imagen en el cliente (derecha).

4.3 Consideraciones generales.

Analizados los factores que influyen en el proceso de visualización remota, se pudo constatar la importancia de estos. La compactación de los datos es de vital importancia porque disminuye la redundancia y minimiza la cantidad de paquetes enviados a través de la red. Quedó demostrado que el algoritmo implementado logra buenas tasas de compresión, sin comprometer la velocidad de procesamiento. En el sistema desarrollado, con técnica de multidifusión, los datos son enviados a una lista de subscriptores (clientes) al mismo tiempo, la misma información. Esto evita que viajen paquetes libres y sin destinatario (solo reciben paquetes los clientes conectados). El Servidor se encuentra a la escucha tanto de conexiones, desconexiones como de aplicación de alguna modificación en la escena. La respuesta de éste es inmediata a toda la lista multidifusión, sin importar la cantidad de miembros. La cantidad de paquetes enviados desde el servidor dependen exclusivamente de la capacidad de procesamiento y no del canal de comunicación. En sistemas de visualización en tiempo real con constante interacción no se debe sacrificar la calidad de las imágenes. La calidad debe mantenerse en todo momento para lograr la misma visualización de la escena tridimensional de servidor y las imágenes 2D recibidas y representada en los clientes.

CONCLUSIONES

Con la realización del presente trabajo se desarrolló un sistema de red basado en arquitectura cliente-servidor que responde al objetivo propuesto. La utilización de la biblioteca JRTPLib permitió la transmisión de datos en tiempo real entre los sistemas que intervienen en la solución.

Se utilizó el algoritmo RLE para comprimir los datos enviados a través de la red debido a su alta tasa de compresión para imágenes con mayor redundancia de los datos. Su uso permitió reducir la cantidad de paquetes necesarios para transmitir información desde un servidor hacia todos los clientes.

Se logró la visualización remota a partir de la transmisión de imágenes para más de una estación de trabajo simultáneamente. Estos resultados sientan las bases para incorporar la funcionalidad de trabajo colaborativo al proyecto Vismedic.

RECOMENDACIONES

- Continuar con el desarrollo del sistema con la integración en software de visualización de escenas tridimensionales que requieran presentaciones a distancia (remoto) simultáneamente por más de una aplicación.
- Implementar mecanismos para otorgar control de los comandos a un cliente determinador por solicitud al servidor.

BIBLIOGRAFÍA

- 1 *Web-Based Remote Rendering with IBRAC*.2000
2. **Koller, D, et al.** *Protected interactive 3d graphics via remote rendering*. 2004.
3. **Deb, Soumyajit.** *RemoteVIS: A Remote Rendering System*. 2003.
4. **Deb, Soumyajit and Narayanan, P J.** *RemoteVis: Remote Visualization of Massive Virtual Environments*.
5. **Engel, K, et al.** *Combining Local and Remote Visualization Techniques for Interactive Volume Rendering in Medical Applications*.
6. **McMahon, Richard A.** *Introducción a las redes*. Universidad de Huston : s.n.
7. **Feldgen, María.** *Sistemas Distribuidos*. s.l. : Facultad de Ingeniería Universidad de Buenos Aires (FIUBA). Vol. Modelo de Referencia OSI.
8. **Liesenborgs, Jori.** *Voice over IP in networked virtual Environments*. 2000. Vol. 2: The Internet Protocol.
9. **Schulzrinne, H, et al.** *RTP: A Transport Protocol for Real-Time Applications*. 2003.
10. **Smed, et al.** *Algorithms and Networking for Computer Games*. University of Turku : s.n., 2006.
11. **Rodríguez, Carlos and Gómez, Yordanis.** *Arquitectura de red para la confección de videojuegos Multijugadores*. Habana : s.n., 2008.
12. Jori's page. [Online] Mayo 2012. <http://research.edm.uhasselt.be/~jori/page/index.php?n=CS.Jrtplib>.
13. **Liesenborgs, Jori.** *Voice over IP in networked virtual Environments*. 2000. Vol. 8: JRTPLIB.
14. HAWKSOFT. *Hawk Network Library*. [Online] <http://www.hawksoft.com/hawkn/>.
15. DATAREEL. *DataReel Open Source Home Page*. [Online] Mayo 2012. <http://www.datareel.com/>.
16. Sasha's Linux Applications and Utilities. [Online] Mayo 2012. <http://www.asksasha.com/download.html>.
17. Novell Shareware. *LIBTCP++*. [Online] Mayo 2012. <http://www.novellshareware.com/info/libtcp--.html>.
18. SOFPEDIA. *LIPTCP++*. [Online] Mayo 2012.
<http://linux.softpedia.com/get/Programming/Libraries/libtcpplusplus-12490.shtml>.
19. Programming 4 Us. *LIPTCP++*. [Online] Mayo 2012. <http://mscerts.programming4.us/es/319838.aspx>.
20. ZOIDCOM. *Zoidcom network library*. [Online] Junio 2012. <http://www.zoidcom.com/>.
21. **Sandoval, Mario.** *Algoritmo de Compresión de Imágenes de Alta Resolución Sin Perdidas*. 2008.
22. **Interscience, Wiley.** *Image Processing Principles and Applications*. 2005.
23. **Gonzalez, Woods.** *Digital Imagen Processing*. Vol. 8 Image Compression.
24. **Hall, Prentice and Black, U.** *Voice over IP*. 2000.
25. **Larson, E and Nikola, S.** *Voice technologies for IP and Frame Relay Networks*.

26. **Rumbaugh, James, Booch, Grady and Jacobson, Ivar.** *El Proceso Unificado de Desarrollo de Software.* 2000.
27. *Rational Unified Process.* s.l. : IBM Corporation, 2006.
28. **Alcolea Nuñez, Rubén.** *Módulo de Iluminación para Visualización Directa de Volumen.* 2011.
29. **Silva Rojas, Luis Guillermo.** *Visualización Directa de Volumen para Endoscopias Virtuales .* 2011.
30. **Larman, C.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : s.n., 1999.

ANEXOS

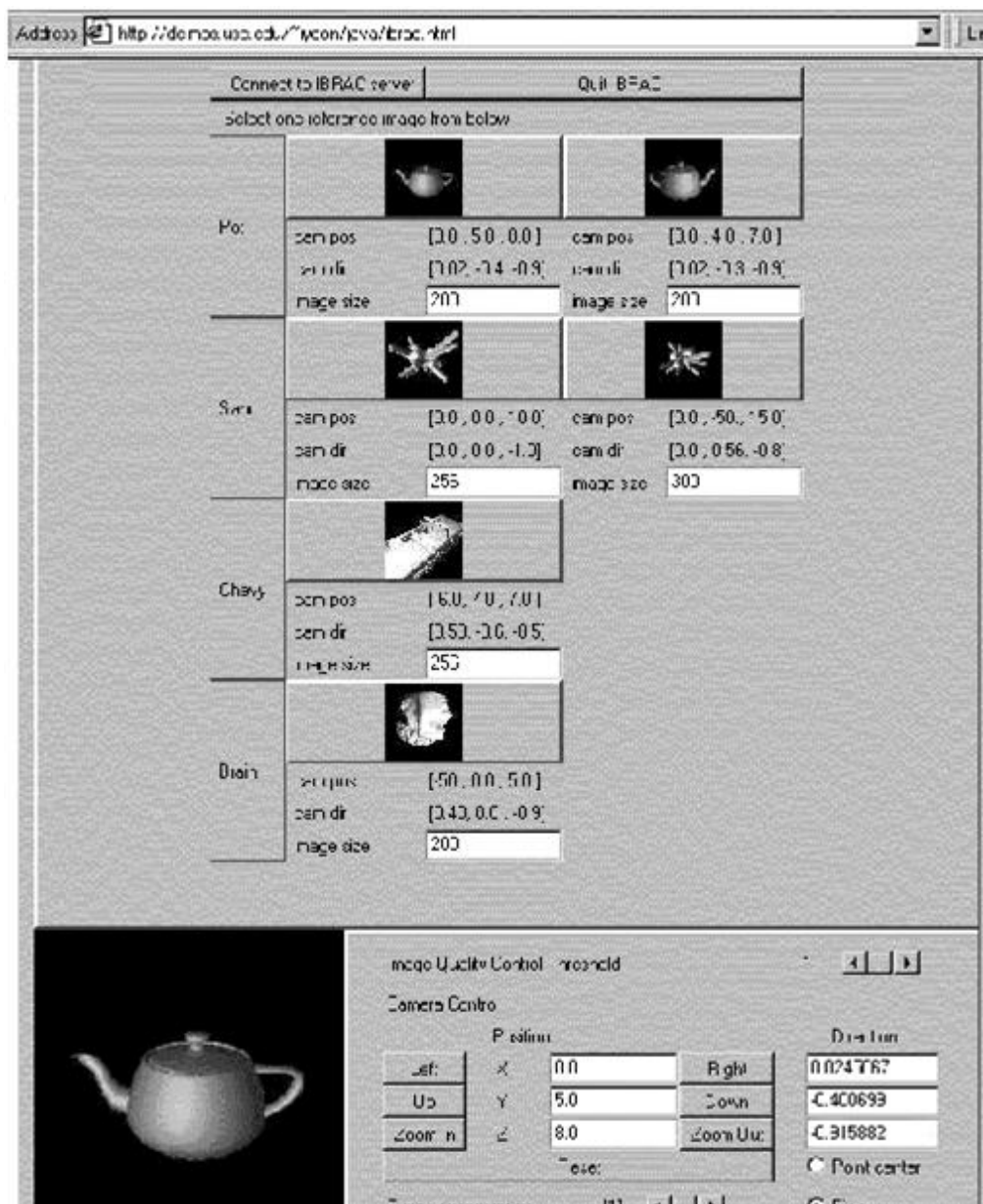


Figura 25: IBRAC Java Applet at <http://deimos.usc.edu/~iyoon/javai/ibrac.html>.

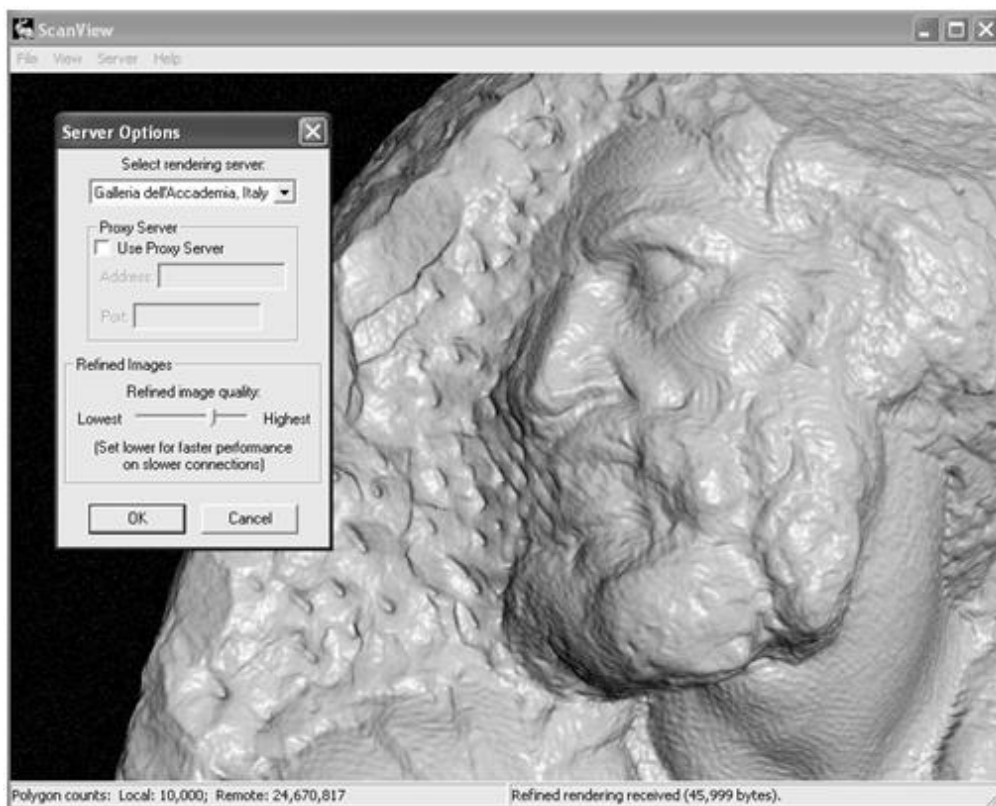


Figura 26: Visor cliente ScanView.

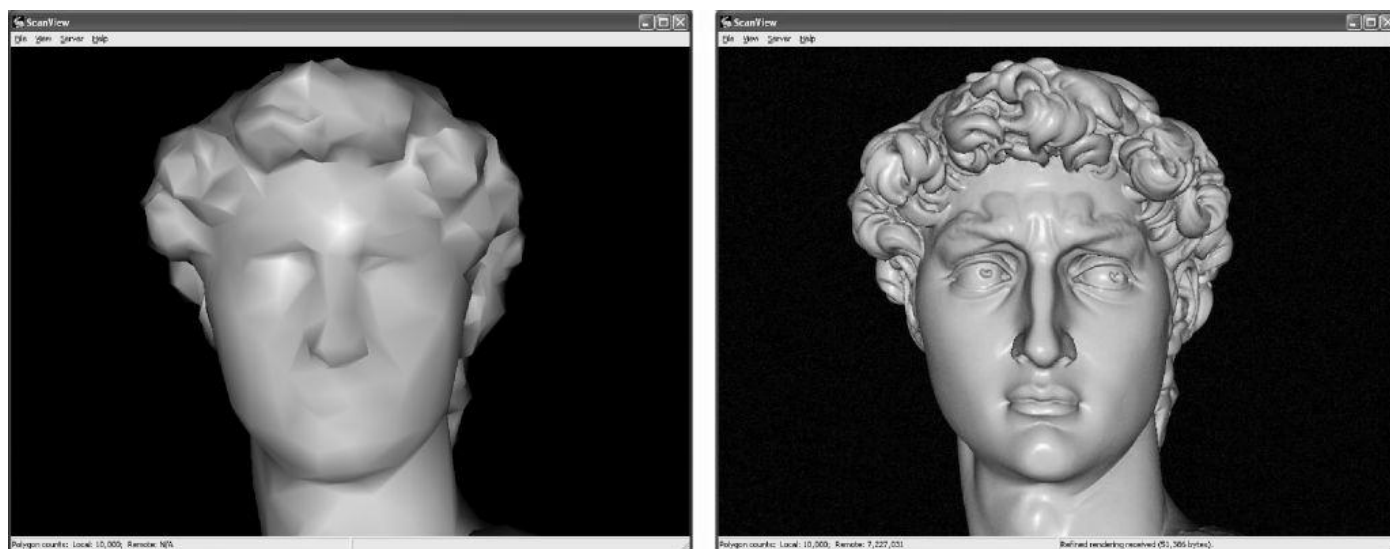


Figura 27: ScanView. Cliente en baja resolución (izquierda) y servidor en alta resolución (derecha).



Figura 28: Sistema de visualización remota: RemoteVis.

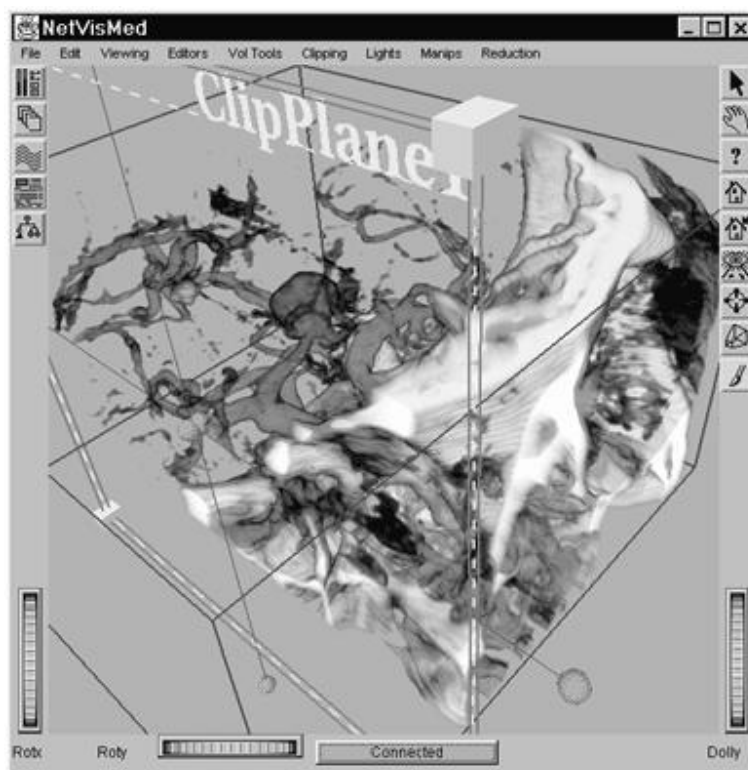


Figura 29: Sistema de visualización remota: NetVisMed.

GLOSARIO

A

Algoritmo: Es una lista que, dado un estado inicial y una entrada, propone pasos sucesivos para arribar a un estado final obteniendo una solución.

C

Ciente: Un programa que envía peticiones a un servidor y que espera una respuesta para seguir trabajando.

G

GPU: Unidad de procesamiento gráfico.

H

Hardware: Componentes físicos de una computadora o de una red (a diferencia de los programas o elementos lógicos que los hacen funcionar).

I

Imagen: Figura, representación, semejanza y apariencia de algo. En computación es formada por la unión de $M \times N$ píxeles (imagen 2D) o vóxeles (imagen 3D).

L

Librerías: En Inglés *library*, cuando se habla de ordenadores, refiere al conjunto de rutinas que realizan las operaciones usualmente requeridas por los programas. Las librerías pueden ser compartidas, lo que quiere decir que las rutinas de la librería residen en un fichero distinto de los programas que las utilizan. Los programas enlazados con bibliotecas compartidas no funcionarán a menos que se instalen las bibliotecas o librerías necesarias.

P

Píxel: Abreviatura de "*picture element*". Es la mínima unidad de información dentro de una imagen bidimensional.

R

Render: Proceso de obtención de imágenes por computadora.

RGB: (del inglés *Red, Green, Blue*, “Rojo, Verde, Azul”). Composición del color en términos de la intensidad de los colores primarios con que se forma: rojo, verde y azul.

RGBA: (del inglés *Red, Green, Blue, Alpha*, “Rojo, Verde, Azul y Alfa”). Es básicamente el uso del modelo de color *RGB* con información extra. El canal alfa es normalmente usado como el canal opacidad. Si un píxel tiene un valor de 0% en su canal alfa será totalmente transparente, mientras que con un valor de 100% se mostrará totalmente opaco.

S

Servidor: Cualquier programa que ofrece un servicio que se puede obtener en una red. En su forma más simple, es un programa que acepta peticiones a través de la red y que en función de esas peticiones ofrece una respuesta.

T

Tarjeta gráfica: Es una tarjeta de circuito impreso encargada de transformar las señales eléctricas que llegan desde el microprocesador en información comprensible y representable por la pantalla del ordenador.

Tridimensional: Se dice de lo que se desarrolla en las tres dimensiones espaciales, altura, anchura y profundidad.