Universidad de las Ciencias Informáticas

Facultad 6

Título: Colector de datos para el servidor streaming distribuido "ALLFRY´S"

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Leosmel Zayas Castillo

Tutor: Ing. Yoandrys S. Pacheco Jeréz

La Habana, 1 de julio de 2011

"Año 53 de la Revolución"

"La ciencia se compone de errores, que a su vez, son los pasos hacia la verdad."

Julio Verne

Dedicatoria

A mis padres.
Por haberme dado todo lo que soy como persona, por haberme
inculcado los principios, los valores, la perseverancia y
el empeño, acompañado siempre de una gran dosis de amor,
Sin pedir nunca nada a cambio.
A mi hermana.
Por su cariño incondicional.
A mis abuelos
Por tanta compresión y afecto que siempre me han dado.
A mi familia
Por apoyarme siempre y ayudarme a alcanzar mis metas.
A mis amigos.
Quiero dedicárselo también a mis viejas amistades
y a todos mis compañeros de la universidad
con los que he compartido todos estos años.

Agradecimientos

A la Revolución y Fidel por darnos la oportunidad de superarnos y ser mejores personas cada día.

A mi madre, a mi padre y a mi hermana por darme el apoyo necesario para llevar a cabo esta tarea. A mis abuelos y demás familiares que de una forma u otra han contribuido con la culminación de este proceso.

A mis amigos de la UCI, que son demasiado grandes para mí, de veras. Me han apoyado desde los primeros años de universidad de manera incondicional y han colaborado para que estos cinco años pasaran como una hermosa experiencia:

Pedro, Anyer, Jose, Hi, David, Yelina, Lisandra, Ivelin, Adaily, Sol, Krysna, Yelen, Magalys, Jorge, Mailin, Liuver, Lisbet, Rafael y Grechin.

A mis amigos de la secundaria y vocacional, que aunque están lejos de aquí he sentido siempre su apoyo: Yanssel, Yulien, El bolo, Denia, Fidel, Jorge, David, Javier.

A mi tutor Yoandrys Pacheco por todo el apoyo y dedicación a este trabajo, las buenas recomendaciones y por ser un amigo más.

A los integrantes del tribunal: Yusnier Valle, Lisbeth Lopez, Rafael Lorente y Frank Torres por ayudar en mi desarrollo profesional con sus señalamientos.

A los tesistas del proyecto que juntos nos enfrentamos a diversos problemas y pudimos darle solución: Frank, Yassel, Leydis, Rene y Anyer.

A todos mis profesores que durante estos años han aportado sus conocimientos para mi formación.

A todos los que me brindaron su apoyo gracias.

Leosmel Zayas Castillo.

Declaración de Autoría

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

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

_____ _____

Autor: Leosmel Zayas Castillo **Tutor:** Ing. Yoandrys S. Pacheco Jeréz

Datos de contacto

Tutor: Ing. Yoandrys S. Pacheco Jeréz

***E-mail*:** ypachecoj@uci.cu

- Ingeniero en Ciencias Informáticas, Universidad de Ciencias Informáticas, 2007.
- Vinculado al Proyecto Descomtec durante 2 años.
- Profesor del Departamento de Ciencias Básicas e Ingeniería de *Software*, Facultad 6.

Autor: Leosmel Zayas Castillo

***E-mail*:** lzcastillo@estudiantes.uci.cu

- Vinculado al Proyecto Operación Verdad durante 1 año.
- Vinculado al Proyecto Manugas durante 2 años.
- Prestó servicios al Proyecto SIPP durante 1 año.
- Vinculado al Proyecto Descomtec durante 2 años.
- Alumno Ayudante del Departamento de Ciencias Básicas durante 4 años, Facultad 6.

Opinión del tutor

Título: Colector de datos para el servidor de *streaming* distribuido "ALLFRY´S"

Autor: Leosmel Zayas Castillo

El tutor del presente Trabajo de Diploma considera que durante el período que se evalúa el estudiante mostró las cualidades que a continuación se detallan. Ha demostrado independencia, responsabilidad y laboriosidad en correspondencia con las tareas trazadas para el desarrollo de la investigación.

La calidad científica técnico del trabajo realizado cumple con los requisitos establecidos. Los resultados obtenidos demuestran profesionalidad en la investigación, lo que se evidencia mediante el cumplimiento de las tareas y objetivos propuestos demostrando en cada una de las etapas, buena preparación y dominio del tema, el diplomante muestra buen dominio de los conceptos actuales enmarcados en su campo de acción, además de haber consultado una amplia bibliografía, la cual permitió realizar profundos análisis y llegar a conclusiones que proveen al trabajo de un alto valor científico. El diplomante demuestra dominio, logrando la implementación del componente propuesto para el servidor de *streaming* distribuido del centro GEYSED.

El documento presentado tiene una estructura adecuada, hace un uso correcto del lenguaje y refleja de manera clara y concisa todas las etapas desarrolladas en la investigación. El trabajo contiene resultados que poseen valor para ser presentados en eventos y talleres científicos relacionados con el tema.

Por todo lo anteriormente expresado considero que el estudiante está apto para realizar el ejercicio de defensa del trabajo de diploma y propongo que se le otorgue al Trabajo de Diploma la calificación de __5__ puntos.

Ing. Yoandrys S. Pacheco Jeréz

_____ _____

Firma Fecha

Opiniones y avales

UCI
Universidad de las Ciencias Informáticas

Universidad de las Ciencias Informáticas
Federación Estudiantil Universitaria

JCE

Otorga el presente:

Reconocimiento

FEU

A: **Diseño e implementación de un colector de datos de un servidor de streaming distribuido**

Por haber obtenido:

Relevante

"El conocimiento nos hace responsable.."
Ernesto Ché Guevara

Dado a los 27 días del mes de mayo de 2011
"Año 53 de la Revolución"

UCI CONSEJO DE LA FEU

Lester González López
Presidente FEU-UCI

Ángel Vega García
Vicerrector de Investigación y Postgrado

Resumen

La presente investigación surge con el propósito de llevar a cabo eficientemente la gestión de los procesos asociados a la tecnología *streaming*. En el mismo se detalla elementos del estudio realizado para la implementación de un componente que permita la transmisión de flujos de videos a través de las redes de datos. Este componente fue desarrollado haciendo uso de herramientas libres para la implementación de sus funcionalidades, dentro de las que se encuentran la realización de streaming en vivo y bajo demanda para los sistemas de *software* que se desarrollan en el centro GEYSED, de la Universidad de las Ciencias Informáticas. Se investigó acerca de los colectores de datos para servidores de *streaming* distribuidos en el mundo, su evolución hasta la actualidad y en general las tendencias, normas y principios que rigen esta tecnología, arrojando que no existía un componente implementado que permitiera la realización de las funcionalidades propuestas para el componente. Durante el proceso de desarrollo de la investigación se obtuvo como resultado un componente capaz de realizar el flujo de *streaming* por diferentes protocolos: UDP, RTP y RTSP, así como formatos de videos entre los que se destacan AVI, MPEG, entre otros. Las transmisiones que efectúa son bajo demanda y en vivo. Además, se utilizó en su implementación un lenguaje de programación C++ que es cercano al *hardware* y posibilitó una mejor distribución de los recursos. También se evitaron recargas en los servidores de almacenamiento al atender múltiples peticiones concurrentemente.

Palabras clave: colector, *streaming*, servidor *streaming*

Abstract

This research emerged with the purpose of carrying out efficiently the management of processes associated with streaming technology. Here are detailed elements of the study for the implementation of a component that allows the transmission of video streams across data networks. This component was developed using free tools for implementation of its functions, within which are found the realization of live streaming and on demand for *software* systems that are developed in the center GEYSED of the University of Computer Science. We investigated about the data collectors distributed streaming server in the world, its evolution to the present and the general trends, rules and principles governing the technology, it was found that there was not a component implemented to able the realization of the functionalities proposed for the component. During the development of this research it was obtained as a result a component capable of realize streaming flow trough different protocols: UDP, RTP and RTSP, and video formats that stand between AVI, MPEG and others. Transmissions are made on demand and live. Moreover, during its implementation was used a programming language C + + which is close to the *hardware* that enabled a better distribution of resources. Also avoid freights storage on servers to serve multiple requests concurrently.

Keywords: collector, *streaming*, *streaming* server

Índice de Contenido

Índice de Figuras

Índice de tablas

INTRODUCCIÓN

El hombre desde sus inicios recibía recónditas impresiones visuales del medio que lo rodeaba y la dinámica que comportaba su peligrosa existencia diaria, de aquí derivó su extraordinario conocimiento, relaciones y la necesidad de comunicarse con sus semejantes. Con el transcurso del tiempo, perfeccionó su cerebro, que hizo posible el pensamiento abstracto y la expresión a través del lenguaje, la cual abre una etapa en la comunicación más sofisticada. Por otra parte, con el desarrollo de las civilizaciones y de las lenguas surgió también la necesidad de comunicarse a distancia de forma regular, con el fin de facilitar el comercio y el intercambio entre naciones.

La información se transmitía en sus inicios a través de las noticias que se difundían por vía oral, por carta o por anuncio, pero la necesidad de notificar a un personal más amplio dio surgimiento a los medios de comunicación de masas, sistemas a través de los cuales se ofrece información a un gran público. La historia de los medios de comunicación está muy ligada al desarrollo de la tecnología, el desarrollo económico y tecnológico de la sociedad. La prensa es el medio de comunicación de masas más antiguo, a continuación surgió el cartel, y el *cómic*. Además, la telegrafía dio lugar al surgimiento de la radio experimentación; junto con la misma surgen los radios aficionados. La historia del desarrollo de la televisión ha sido en esencia la historia de la búsqueda de un dispositivo adecuado para explorar imágenes. Las primeras emisiones públicas de televisión las efectuó la BBC en Inglaterra en 1927 y en Estados Unidos la CBS y NBC en 1930. (Ríos, 2007)

De ahí en adelante el desarrollo de la televisión como medio de comunicación ha sido impresionantemente acelerado. Por los años 40, llega la etapa de la televisión en color y, tras ella, la internacionalización del medio y de sus contenidos. Esta vocación internacional impulsó el lanzamiento de los primeros satélites de comunicación y de otras tecnologías como la distribución de señal televisiva por cable. Hoy día la tecnología de las emisiones televisivas se encuentra en revolución. Una de las más difundidas actualmente es la Televisión Digital Terrestre (TDT), que permite eliminar los ruidos, la imagen doble e interferencias y transmite cuatro canales digitales en el mismo espacio en el que se emite un solo canal analógico. También ha evolucionado la televisión con el surgimiento de la Televisión Digital de Alta

Definición (HDTV), una importante innovación y que proporciona una calidad de imagen superior a la (TDT), incorporando sonido 5.1. (Díaz, 2005)

La televisión conjuntamente con el desarrollo de Internet es la razón fundamental por la que se dice que se está viviendo en la "Era de la Información". Con Internet consolidada como un importante medio de comunicación la transmisión de video a través de redes de telecomunicaciones han llegado a convertirse en un sistema asiduo de comunicación producto del crecimiento masivo que ha supuesto Internet en estos últimos años. Se emplea para ver películas o comunicarse, pero también para dar clases remotas, videoconferencia, distribución de TV, video bajo demanda y para distribuir multimedia en Internet.

Inicialmente, la reproducción de contenido multimedia a través de Internet necesariamente implicaba tener que descargar completamente el archivo contenedor al disco duro local. Como los archivos de audio y especialmente los de video tienden a ser enormes, resultan demasiado pesados para el ancho de banda con que cuentan la mayoría de los usuarios provocando que la descarga sea una operación muy lenta. Este proceso permitía a los clientes tener acceso a estos archivos pero no en tiempo real, debido a la necesidad de uso en el presente es que aparece la tecnología *streaming*[1]. La tecnología de *streaming* se utiliza para aligerar la descarga y ejecución de audio y vídeo en la web, ya que permite escuchar y visualizar los archivos mientras se están descargando. (Quintero, y otros, 2006)

El funcionamiento de esta tecnología ocurre de la siguiente manera. El cliente se conecta al servidor *streaming* donde se encuentra publicado el *stream*[2] y éste le empieza a enviar dicho flujo al usuario. Posteriormente el consumidor comienza a recibir el fichero y construye un *buffer*[3] donde empieza a guardar la información. Cuando se ha llenado el *buffer* con una pequeña parte del archivo, el cliente lo empieza a visualizar y a la vez continúa con la descarga. El sistema está sincronizado para que el archivo pueda ser reproducido cuando se descarga, de modo que, cuando el archivo acaba de descargarse el fichero también ha acabado de visualizarse. Si en algún momento la conexión sufre descensos de velocidad se utiliza la información que hay en el *buffer*, de modo que se puede soportar un poco ese

[1] **Streaming: Consiste en la distribución de audio o video por Internet.**
[2] **Stream: Corriente, chorro, fluir, arroyo.**
[3] **Buffer: Es una porción de memoria dedicada a almacenar datos, para posteriormente enviarlos hacia un dispositivo externo o recibirlos desde un dispositivo externo.**

descenso. Si la comunicación se corta demasiado tiempo, el *buffer* se vacía y la ejecución del archivo se cortaría también hasta que se restaurase la señal. (Alvarez, 2003)

El proyecto educacional cubano ha priorizado la enseñanza de la computación como una habilidad necesaria para el progreso socioeconómico, en tal sentido, la inclusión de esta disciplina en todos los niveles educacionales es una realidad perceptible. Por este motivo principal es por el cual surge al calor de la batalla de ideas la Universidad de las Ciencias Informáticas (UCI) puntera informática de este país.

La nación cubana, un país bloqueado por el imperio, nunca ha descartado la posibilidad de desarrollarse de acuerdo con sus posibilidades y no ha estado exenta de las transformaciones que han surgido a raíz de Internet y por ello la informatización de la sociedad cubana es hoy prioridad de la dirección del Partido y del Estado.

En la actualidad existen diversos *software* para hacer *streaming* entre los cuales se destacan los servidores *streaming* de características centralizadas los cuales provocan sobrecargas en la red en ocasiones al no poder atender múltiples peticiones concurrentemente lo cual no sucede en los servidores de arquitecturas distribuidas porque evitan estas recargas en los servidores de almacenamiento. Los servidores centralizados en cuanto a modularidad poseen un nodo, disponibilidad baja y una escalabilidad restringida a diferencia de los sistemas distribuidos que tienen varios nodos, la disponibilidad alta además de una escalabilidad ilimitada que permite un mejor trabajo. Los servidores *streaming* distribuidos están conformados por varios módulos con características específicas que en su conjunto permiten el buen funcionamiento de dichos servidores.

En la (UCI) existe el Centro de Desarrollo Geoinformática y Señales Digitales (GEYSED) de la Facultad 6, el cual entre sus funciones desarrolla aplicaciones para mejorar la transmisión de televisión por la web haciendo uso de los servidores *streaming*. En la actualidad los que se han utilizado en el centro han sido centralizados y han implementado las funcionalidades principales de realizar *streaming* bajo demanda y en directo pero la alta concurrencia y aumento del consumo de ancho de banda de clientes en diferentes períodos de tiempo provocan recargas e inestabilidad en la red.

Por otra parte los servidores *streaming* que se utilizan actualmente en el departamento presentan restricciones las versiones libres con respecto a las propietarias, en cuanto a los protocolos de transmisión, formatos de videos a transmitir, y otros aspectos relacionados con la optimización, balanceo de carga y concurrencia de usuarios atendiendo al ancho de banda. Es por ello que surge la necesidad de la creación de un servidor streaming que se adapte a las necesidades de los productos del departamento, e implemente sus soluciones atendiendo a otros elementos, los cuales en la propuesta del servidor streaming que se propone está basada en una arquitectura distribuida. Dicho servidor se conforma por varios componentes que indistintamente se encargan de realizar funcionalidades por separados y que al unificarlas conforman los procesos relacionados con el streaming y el almacenamiento de medias. Es por ello que se propone el desarrollo de un componente Colector de datos el cual se encarga de conformar las medias a transmitir de acuerdo con las peticiones de los usuarios, para posteriormente realizar el flujo de streaming de acuerdo con los parámetros de la transmisión.

Según lo planteado anteriormente el **problema de la investigación** queda constituido de la siguiente forma: ¿Cómo realizar la selección de paquetes para la conformación de las medias a transmitir atendiendo a las peticiones de los clientes? Enmarcándose en las necesidades planteadas el **objeto de estudio es**: Tecnologías libres para el desarrollo de colectores de datos de un servidor de *streaming* distribuido y queda delimitado como **campo de acción:** Las características y funcionalidades que proveen las tecnologías libres para el desarrollo de colectores de datos para servidores de *streaming* distribuidos. En vista a resolver el problema de la investigación descrito se hace necesario, desarrollar un componente colector de datos para servidores de *streaming* distribuidos el cual figura como el **objetivo general.**

La **idea a defender es:** La implementación de un colector de datos permitirá un uso eficiente del *hardware* utilizado en los servidores de *streaming* distribuidos evitando recargas en servidores de almacenamiento.

Se definieron un grupo tareas que serán efectuadas durante la investigación, estas son las siguientes:

- ✓ Caracterización de los procesos relacionados con los colectores de datos para servidores de *streaming* distribuidos.
- ✓ Identificación de los estándares nacionales e internacionales utilizados para el desarrollo de colectores de datos para servidores de *streaming* distribuidos.

- ✓ Descripción del estado del arte de los colectores de datos para servidores de *streaming* distribuidos.
- ✓ Identificación de las funcionalidades para los colectores de datos para servidores de *streaming* distribuidos.
- ✓ Selección de la metodología de desarrollo de *software* a usar en el proceso.
- ✓ Modelado del componente de colectores de datos para servidores de *streaming* distribuidos. Confección de los artefactos necesarios.
- ✓ Implementación del componente de colectores de datos para servidores de *streaming* distribuidos.
- ✓ Confección de los datos de pruebas. Realizar proceso de pruebas al componente.

Para llegar a los resultados concretos de la investigación se hizo uso de los siguientes métodos:

Métodos Teóricos

Histórico - Lógico: Este método será de gran utilidad pues se hará un estudio acerca de la aparición de los colectores de datos para servidores de *streaming* distribuidos en el mundo, su evolución hasta la actualidad y en general todas las tendencias, normas y principios que rigen esta tecnología.

Modelado: En el desarrollo de la investigación es necesario el modelado de la arquitectura que se propone, es por eso por lo que se utiliza este método para crear modelos con vistas que favorezcan la comprensión de la misma.

Métodos Empíricos

Observación: Este método es de suma importancia, pues más allá de los métodos teóricos, este permitirá una apreciación de lo que ocurre desde el punto de vista de los investigadores para clasificar los acontecimientos de acuerdo con alguna decisión previamente tomada y para el análisis en el transcurso de la investigación.

El documento está estructurado en cuatro capítulos, que incluye todo lo relacionado con el trabajo investigativo realizado, así como las características y el diseño del sistema que se propone.

En el capítulo uno se plantean todos los elementos teóricos que sustentan la evolución, desarrollo y actualidad de los colectores de datos para servidores *streaming* distribuidos, o sea, conceptos referentes a servidores *streaming* distribuidos y las diferentes maneras en que ha evolucionado desde su surgimiento hasta el día de hoy, para brindar facilidades en el acceso de medias a través de internet, así como su desarrollo en Cuba. Esto servirá como base a los futuros planteamientos en el desarrollo de la investigación.

En el capítulo dos se realizará un análisis detallado de las tecnologías que se adecuan para la solución. Además, se investiga acerca de las diferentes metodologías existentes en el mundo así como lenguajes y herramientas de desarrollo para posteriormente según las características individuales escoger las adecuadas para realizar la aplicación.

Por otra parte, en el capítulo tres se hace referencia al análisis y diseño de la propuesta de sistema, se abordará lo relacionado al Modelo de Dominio, el levantamiento de los requisitos funcionales y no funcionales de *software*. También se realizará el Diagrama de Casos de Uso del Sistema (DCUS) y la descripción detallada de los mismos, así como también los diagramas de clases del diseño lo cual es de gran ayuda para la fase de implementación.

Por último, en el capítulo cuatro se pone en marcha el desarrollo y prueba de la solución propuesta en los capítulos anteriores. Se describe los modelos de implementación además de las pruebas realizadas. Se definen los modelos de componentes tanto de código fuente como de código ejecutable. Se describen además las pruebas realizadas al producto para verificar que todo lo antes planteado se cumpliera y la calidad del mismo.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se plantean los elementos teóricos que sustentan la evolución, desarrollo, características y tendencias actuales de los colectores de datos para servidores *streaming* distribuidos, proporciona una visión general de los conceptos referentes a servidores *streaming* distribuidos y las diferentes maneras en que ha evolucionado desde su surgimiento hasta el día de hoy, para brindar facilidades en el acceso de medias a través de internet, así como su desarrollo en Cuba.

1.2 Conceptos asociados al dominio del problema

Para un mejor entendimiento de las características de la tecnología *streaming*, su funcionamiento y su relación con los servidores *streaming* es ineludible enunciar varios conceptos asociados a la presente investigación.

- ***Streaming***

El término *streaming* (del inglés *stream* que significa corriente, arroyo, flujo, fluir) es una tecnología que permite la transferencia de contenido audiovisual por la red desde un servidor hacia sus clientes. Engloba una serie de productos y técnicas cuyo objetivo es la difusión de contenidos multimedia tales como audio y video. Este sistema de distribución se caracteriza por la visualización de los contenidos en el cliente sin la necesidad de esperar la descarga completa de un fichero en el ordenador, ya que permite escuchar y visualizar los archivos mientras se están descargando. Si no se utiliza *streaming*, para mostrar un contenido multimedia en la red, se tiene que descargar primero el archivo entero en nuestro ordenador y más tarde ejecutarlo, para finalmente ver y oír lo que el archivo contenía, por lo tanto, las emisiones en directo no se podrían realizar. (Pastor, 2007)

El componente desarrollado en esta investigación se utilizará en aplicaciones que efectúen *streaming* fundamentalmente. Este componente pertenece a un servidor de *streaming* distribuido por lo que es de gran valor conocer estos significados.

- **Servidores**

Un servidor es un tipo de *software* que realiza ciertas tareas en nombre de los usuarios. También se utiliza para referirse al ordenador físico en el cual funciona ese *software*, una máquina cuyo propósito es proveer datos de modo que otras máquinas puedan utilizar esos datos. Además, sirve información a los ordenadores que se conecten a él. Cuando los usuarios se conectan a un servidor pueden acceder a programas, archivos y otra información del servidor.

Después de notificar en qué consiste *streaming* así como servidor se explicará la unión de ambos conceptos y sus características.

- **Servidores de *streaming***

Es el elemento principal en cuanto a calidad del servicio se refiere. El servidor procesa los datos multimedia en cortos espacios de tiempo y soporta funciones de control interactivas siendo además el responsable de suministrar los servicios de audio y video en modo sincronizado. El servidor de *streaming* espera de la petición del usuario y cuando recibe una petición, el busca en el directorio apropiado el archivo solicitado. Las entregas de paquetes del servidor al usuario pueden estar sujetas a demoras conocidas como LAG (rezagarse, quedarse atrás), que dificulta el desarrollo normal y provoca desorientación e incomodidad. (Yañez, y otros, 2005)

Se puede decir que los servidores de *streaming* están formados por el *software* de *streaming* y un sistema de catalogación en caso de ser necesario y contemplan la distribución de contenidos tanto en una Intranet corporativa como en Internet.

Anteriormente se definieron generalidades del flujo de *streaming* por lo que a continuación se detallará las principales fases de este proceso.

- **Principales etapas dentro del proceso de *streaming***

Captura*:* El contenido audiovisual es capturado y convertido en datos digitales *raw data*[4], aún sin codificación y compresión.

Codificador*:* Antes de transmitir la información multimedia por la red como *streaming*, es necesario codificarla y comprimirla a un formato apropiado ya que contienen gran cantidad de redundancia espacial y temporal. Para ello se debe recurrir a herramientas denominadas genéricamente como *encoders*[5], los cuales poseen los algoritmos suficientes para permitir disminuir el tamaño de la información y así optimizar el consumo de ancho de banda.

Servidor: El servidor procesa los datos multimedia en cortos espacios de tiempo y soporta funciones de control interactivas siendo además el responsable de suministrar los servicios de audio y video en modo sincronizado. Espera de la petición del usuario y cuando recibe una petición, busca en el directorio apropiado el archivo solicitado. El flujo continuo de datos es distribuido por la red a los diferentes clientes ya sea a través de *unicast*[6], *multicast*[7] *o broadcast*[8].

Cliente: En esta etapa se reciben los datos desde la red y se envían como un flujo a la etapa de decodificación.

Decodificador: En esta etapa los datos son descodificados y descomprimidos de acuerdo con el formato contenedor y a los *codecs*[9] usados en el proceso de codificación.

Reproductor: Esta es la etapa en que el contenido multimedia regenerado a partir de los datos es desplegado en pantalla y reproducido. (Quintero, y otros, 2006)

---

[4] **Raw data: Los datos crudos (raw data), también conocidos como datos fuente o datos atomizados, son datos que no han sido procesados para ser utilizados**

[5] **Encoders: Son dispositivos que convierten una señal RGB (red, green and blue) en una señal NTSC. Son utilizados para grabar clips de video digital a una videocasete.**

[6] **Unicast: Es el envío de información desde un único emisor a un único receptor.**

[7] **Multicast:** E**nvío a ciertos destinatarios específicos.**

[8] **Broadcast: Radiado o difusión, donde los destinatarios son todas las estaciones en la red**

[9] **Codecs: Dispositivos o programas capaces de codificar y descodificar en un signo o una corriente de datos digitales.**

La distribución de los datos hacia los usuarios se envía de diferentes formas por lo que es necesario conocer estos tipos de transmisiones, las cuales se van a implementar en el componente.

- **Tipos de transmisiones**

Unidifusión (Unicast): Se basa en un proceso de envío de una información en una o más unidades de datos (datagramas IP) desde una máquina origen a una única máquina destinataria o receptor final. Por tanto, es una transmisión punto a punto con cada destinatario. Si se desea enviar la misma información y hay "n" destinatarios, habrá "n" comunicaciones punto a punto independiente o "n" copias de la misma información enviadas desde la máquina origen.

Multidifusión (Multicast): Cuando la información es distribuida a un grupo determinado de usuarios simultáneamente, se está usando el esquema de distribución *Multicast*. El servidor envía un solo flujo para todos los miembros del grupo, esto significa que se usa el mismo ancho de banda para enviar el contenido a uno o a 100 clientes. El flujo enviado está dirigido a una dirección de grupo, cada cliente debe estar configurado de forma tal que escuche la información enviada a tal grupo. Para hacer este tipo de transmisión, sin embargo, todos los *Routers*[10] en el trayecto entre el servidor y el grupo multidifusión, deben estar habilitados para esta distribución, debido a esto se realiza en redes privadas, intranet y en general en redes de área local (LAN). (Quintero, y otros, 2006)

Difusión (Broadcast): Se basa en un único proceso de envío, independientemente del número de potenciales máquinas receptoras, de una misma información en una o más unidades de datos (datagramas IP) desde un origen a todas las máquinas de una red de área local. Todo ello, sin necesidad de transmitir desde el origen una copia de la misma información, por separado, a cada una de dichas máquinas. Se resalta el hecho de que desde la máquina origen sólo se envía una vez la pertinente información y no se transmiten "n" copias de la misma aunque haya "n" destinatarios. (Ramón, 2004)

[10] **Router: Es el dispositivo conectado a la computadora que permite que los mensajes a través de la red se envíen de un punto (emisor) a otro (destinatario).**

Dentro de las funciones principales del componente están las de atender las peticiones de los clientes de acuerdo con sus características, ya que solicitan diferentes tipos de contenidos de ahí surge la importancia de conocer estas diferencias para su desarrollo posterior.

- **Tipos de contenidos**

Bajo demanda (Video on Demand, VOD): La tecnología de *Streaming* multimedia permite la visualización del contenido en el momento que el usuario desee. Cuando el video está disponible para la transmisión, este es almacenado en un servidor de *Streaming*, en ese momento el servidor está en la capacidad de manejar conexiones individuales provenientes de máquinas cliente que hagan la petición de visualización del contenido. Posteriormente el servidor comienza la entrega del flujo de bits para ser visualizado en el reproductor del cliente al otro lado de la conexión, en este punto el usuario tiene la posibilidad de controlar el flujo, debido a que en cualquier momento puede detener su ejecución, realizar un retroceso, una pausa, pasar a otra escena y demás funciones. (Sciara, 2004)

En Directo: *Streaming* en directo se refiere al flujo de contenido multimedia en tiempo real. Mediante este método se realiza una transmisión simultánea de datos a una audiencia de gran volumen. En este caso es necesario el uso de un *software* de producción que permita codificar el contenido y que tenga la capacidad de transmitirlo a un servidor desde el cual generar el flujo hacia los clientes. Un usuario que pulsa sobre un enlace a un contenido en vivo se conectará al evento en curso y, dado que está ocurriendo en tiempo real, no puede tener control sobre el contenido. La diferencia con la distribución bajo demanda es que en este caso el cliente debe escuchar el canal por el cual fluye el contenido, en este caso es común el uso de grupos *Multicast* como destino, siendo el cliente el que demuestra la intención de recibir el flujo. (Sciara, 2004)

Al inicio de la investigación se explicó la situación problemática que existía y la necesidad de proveer una solución, previamente se abordaron desigualdades de los sistemas centralizados y distribuidos que a continuación se ejemplifican un poco más.

- **Servidores centralizados**

Un sistema se dice centralizado cuando tiene un nodo que comanda a todos los demás, y estos dependen para su activación del primero, ya que por sí solos no son capaces de generar ningún proceso, por lo que el acceso a los recursos es de forma local. También se controlan más fácilmente que los descentralizados, son más sumisos, requieren menos recursos, pero son más lentos en su adaptación al contexto.

- **Servidores distribuidos**

Un sistema distribuido consiste en un conjunto de computadoras autónomas conectadas por una red y con soporte de *software* distribuido. Permite que las computadoras coordinen sus actividades y compartan los recursos de *hardware*, *software* y datos, de manera tal que el usuario percibe una única facilidad de cómputo integrada aunque esta pueda estar implementada por varias máquinas en distintas ubicaciones. En el Anexo 1 se mostrará algunas diferencias de estos tipos de servidores.

- **Características de sistemas distribuidos (Castro, 1999)**

    - Red de comunicaciones o de computadoras, denominada simplemente red.
    - Intercambio de mensajes (coordinación).
    - Meta común, compartir un recurso, brindar un servicio, o ejecutar una aplicación.

El módulo que se desarrollará está apoyado en la arquitectura basada en servicios orientados a componentes: Adaptive Distributed Multimedia Server (ADMS) por lo que es de vital importancia conocer los diferentes componentes de esta arquitectura así como sus interacciones.

- **Arquitectura de Servidor Multimedia Distribuida**

Representa un componente con base en servicio que puede ser reutilizables y con dinamismo combinado con otros componentes de la arquitectura: Adaptive Distributed Multimedia Server (ADMS), en donde cada componente cumple una tarea lógica esencial en la cadena de transmisión de medias. Entre las que se encuentran la adquisición de datos, grabación y lectura en continuo de datos, manejo de almacenamiento

de datos y control global de corriente y distribución de aplicaciones adaptativas de un servidor (ASA). Estas tareas se dividen en cuatro componentes dentro de la arquitectura ADMS, los distribuidores de datos (DDs), los manejadores de datos (DMs), los colectores de datos (DCs) y el manejador de clúster (CM).

Seguidamente se describirá el componente encargado de fraccionar los contenidos multimedia para su posterior almacenamiento y futura transmisión. Este no interactúa con el módulo que se implementará en es esta investigación.

- **Distribuidores de Datos (DD)**

Un distribuidor de datos es un componente basado en servicios, que distribuye los contenidos multimedia recibidos de un cliente o una cámara en vivo hacia los manejadores de datos, el cual es el encargado de almacenarlo. El distribuidor de datos realiza un proceso de segmentación de los contenidos en pequeños fragmentos que facilita el desarrollo de la transmisión.

A continuación se caracterizará al componente que dirige las operaciones de los demás, él es el encargado de enviar la petición al componente a desarrollar.

- **Manejador de Clúster (CM)**

El administrador del clúster es el componente central en la arquitectura ADMS. Es nombrado director del clúster, ya que gestiona el número y la ubicación de los distribuidores de datos y de los colectores de datos, que resulta en un clúster de servidores de aplicaciones dinámicas ADMS. Por otra parte, representa el punto central para el manejo de las conexiones de los clientes, aunque no atiende las solicitudes del cliente por sí mismo, las dirige adecuadamente a un distribuidor o colector. (Tusch, 2004)

Al igual que el anterior, también interactúa directamente con el componente a implementar pero en este caso particular es el encargado de proveer la dirección donde se encuentra almacenado los contenidos multimedia.

- **Manejadores de Datos (DM)**

Un Manejador de Datos es el que provee de manera eficiente el almacenamiento y la recuperación de los datos en un servidor *streaming*, después que el distribuidor de datos que es otro de los componentes con que consta el servidor *streaming* realice la operación de separación de segmentos el manejador de datos es el encargado de almacenarlo. Desde un *stream* elemental puede estar listado un sin número de información, cada manejador de datos solo almacena una porción del stream. El mismo lo organiza internamente mediante los segmentos de media almacenando, solamente guardando la información que contengan las mismas y su dirección en memoria, utilizando un *set*[11] de media para cada división del segmento, formando así un fragmento de *stream*, lo cual permite una mayor eficiencia en el almacenamiento. A la hora de recuperar dichos segmentos va hacer por una petición de los colectores de datos que va hacer la petición al manejador, el cual va a buscar donde lo tiene almacenado mediante el nombre y la dirección para así devolverlo en segmento. (Tusch, 2004)

Por último, se brindan detalles del componente que se implementará en este trabajo.

- **Colectores de datos (DC)**

El colector (recopilador) de datos es un componente del servidor *streaming* distribuido que realiza la operación inversa al distribuidor de datos. Este recolecta las unidades de bandas de un paquete de medias con la configuración apropiada del manejador de datos, recoge estos paquetes de diferentes orígenes de datos, después envía las unidades en secuencia de *buffer* hacia los clientes. También pueden incorporar un componente de almacenamiento en caché, lo que reduce las latencias de cliente al inicio y el consumo de ancho de banda, además evita las recargas en los servidores de almacenamiento. En particular, el recopilador de datos puede desempeñar el papel de un servidor *proxy*, que se asigna dinámicamente para servir a un grupo de clientes por el manejador del clúster. Esto constituye una diferencia con respecto a los *proxy* habituales, que son seleccionados por los propios clientes.

Un enfoque interesante es la integración de funcionalidades dentro de los colectores de datos. Una puerta de enlace realiza la codificación a fin de adaptarse a propuesta de la red o a las limitaciones del

[11] **Set: Plato cinematográfico o televisivo.**

dispositivo cliente. Puede por ejemplo, reducir la resolución de un vídeo que se envía al equipo de un cliente con una pequeña resolución de la pantalla. Los colectores de datos incorporan otras funcionalidades y es un ejemplo particularmente atractivo para la combinación de las adaptaciones ofensivas y defensivas. Se utiliza la adaptación ofensiva para la entrada al lugar adecuado, y el uso de la adaptación defensiva para cumplir con las capacidades de clientes determinados**.** En la fig. 1 se muestra las relaciones entre los diferentes componentes. (Tusch, 2004)

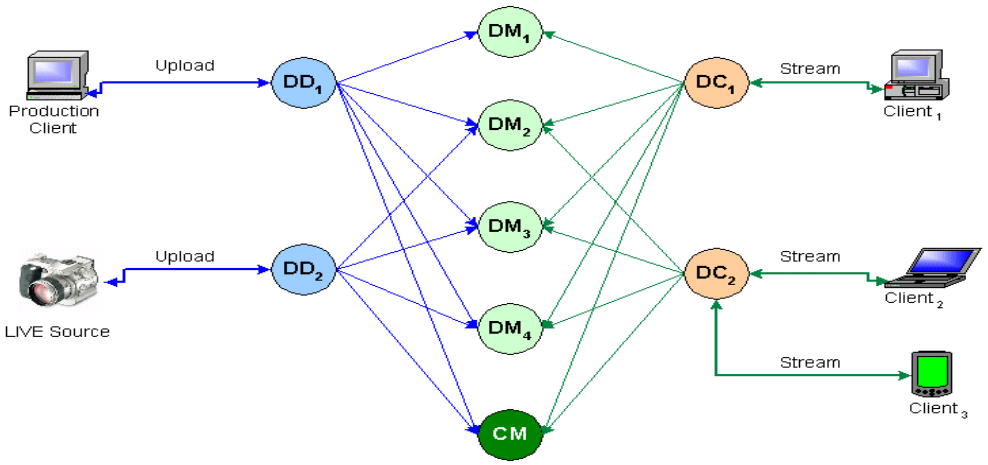

Fig. 1. Sistema de ADMS orientado a los servicios de componentes. (Tusch, 2004)

1.3 Situación problemática.

Con el transcurso de los años se ha experimentado un considerado avance en la velocidad de transmisión de las redes de computadoras, lo cual, apoyado en las actuales capacidades de procesamiento de los ordenadores, permite que las aplicaciones de video y sonido sobre redes IP sean más populares y más fáciles de implementar que hace algunos años atrás. Otro de los elementos que ha influido en el incremento de este tipo de aplicaciones es la aparición de la tecnología *Streaming*, al dar la posibilidad de visualizar y escuchar un archivo mientras se está descargando, en otras palabras, permite ver y oír en tiempo real vídeos y audios.

Para poder poner a disposición de los clientes (aplicaciones que usen los *Streaming*) el flujo de datos (*Streaming*) es necesario un servidor *streaming*, los que se encargan de realizar una serie de operaciones

sobre los datos que se van a transmitir. Actualmente se cuenta con una gran variedad de servidores *streaming* tanto de *software* libre como propietario, ambos presentan un buen prestigio en el mundo. Los de *software* libre poseen una gran cantidad de funcionalidades que garantizan la calidad en el servicio que brindan, pero sus soluciones no abarcan protocolos de transmisión y formatos de videos necesarios para los clientes, constituyendo esto una necesidad para poder alcanzar la soberanía tecnológica que defiende el país a la hora del desarrollo de aplicaciones que hagan uso de este recurso.

En la Universidad de las Ciencias Informáticas existe el Centro de Desarrollo Geoinformática y Señales Digitales (GEYSED) de la Facultad 6, el cual posee varios proyectos en el Departamento de Señales Digitales y entre sus funciones desarrolla aplicaciones para mejorar la transmisión de televisión por la web a través de los servidores *streaming*. En la actualidad los que han desarrollado han sido de características centralizadas y han implementado las funcionalidades principales de realizar *streaming* bajo demanda y en directo pero la alta concurrencia y aumento del consumo de ancho de banda de clientes en diferentes períodos de tiempo provoca recargas y desequilibrio en la red, por lo que surge la necesidad de implementar un servidor con características distribuidas.

1.4 Análisis de otras soluciones existentes.

Se realizó una investigación en búsqueda de soluciones anteriores al componente (DC) que se desarrollará y no se encontró información al respecto. Los servidores de *streaming* en su mayoría son aplicaciones que no interactúan con el cliente directamente, sino a través de plataformas las cuales funcionan de intermediarias entre clientes y el servidor con características particulares en su implementación, lo que las diferencias de las demás. Existen diferentes plataformas, aplicaciones y servidores de *streaming* de las cuales se abordará las similitudes y diferencias con respecto al componente colector de datos.

- **Real Server**

Es el *software* de Real para la distribución de contenidos vía *streaming* tanto en directo como en diferido. Es una solución que ofrece RealNetworks para la producción y distribución de contenido multimedia. El *software* es multiplataforma y puede ser instalado tanto en Windows como en sistemas Unix (Linux y

Solaris). Se puede instalar una versión de demo que permite la conexión de hasta 25 usuarios simultáneos.

Características:

- ✓ Facilidad de instalación y configuración.
- ✓ Multiplataforma en Servidor, codificación y visualizadores.
- ✓ Se apoya en estándares (SMIL, RTP/RTCP, RTSP).
- ✓ Soporte *multicast* (real y simulado).
- ✓ Soporta protocolos de sesión como SAP/SDP para su comunicación con las herramientas típicas de videoconferencia *multicast*.
- ✓ Herramientas de control y estadísticas de uso (JAVA).
- ✓ Control de acceso (por dirección IP).
- ✓ Soporta formatos como: AVI, MOV, QT, MP3, RealVideo8, MPEG – 1, MPEG – 2.

El proceso de configuración de las técnicas del *splitting*[12], *multicasting*[13] o *caching*[14] mejoran el funcionamiento del Real Server. La técnica del *splitting* permite configurar los servidores de Real como transmisores o espejos de los contenidos en directo, de forma que los usuarios se conecten a dichos contenidos a su servidor más cercano. Con ello se logra que los usuarios se conecten con más garantías a los contenidos y se descarga el ancho de banda necesario en el servidor central y en cada uno de los incluidos en la infraestructura. (Bedrina, 2002)

Además, está técnica se puede observar cuando desde un servidor principal (en el que se está realizando una retransmisión en directo), reparte la señal a un segundo *Splitter*[15]que, a su vez, la reparte a otros tres y, al mismo tiempo, a otros tres más individuales. De esta forma, se tendrían 6 servidores de borde a los que se conectarán los usuarios directamente usando Internet. Evidentemente, este acceso no se produce de forma automática. En las páginas de acceso a este contenido habrá que indicar a qué servidor se tiene que acceder. Una forma, la más obvia, es que existan páginas diferentes dependiendo de la región donde

[12] **Splitting: Es una práctica que permite el hendimiento, desdoblamiento y repartición de un flujo de información**
[13] **Multicasting: Es una técnica de transmisión a través de Internet que permite enviar información a varios lugares al mismo tiempo.**
[14] **Caching: Esta técnica permite ocultar y esconder información.**
[15] **Splitter: Divisor.**

se acceda. Otra solución posible y fácil, es programar un filtrado de la IP del usuario para así encaminarlo a su servidor más cercano.

Esta técnica es válida sólo para retransmisiones en directo. Para los contenidos bajo demanda es necesario programar o contratar algún servicio de *caching* que reparta dichos contenidos por en dicha infraestructura. Esto no lo hace Real Server por sí mismo. Para interconectar mediante *splitting* los servidores se pueden usar tres tipos distintos de protocolos: UDP *Unicast*, UDP *Multicast* y TCP. UDP *Multicast* es la más eficaz, pero debe conocer su arquitectura de red para saber cuál de ellas puede utilizar.

Existen varias partes a configurar para la transmisión, entre ellos está introducir el punto de montaje desde el que se van a servir los streams[16]en vivo. Los clientes accederán al contenido de la forma: rtsp: //servidor/punto_de _montaje/nombre_del_stream.rm.

- **IceCast**

Dentro del *software* libre se encuentra esta opción de servidor para hacer *streaming*. Icecast es un proyecto para *streaming* de medios mantenido por La Fundación Xiph.org. Es muy versátil, ya que los nuevos formatos se pueden agregar relativamente fáciles y soporta estándares abiertos para comunicación e interacción. Actualmente existen solo dos versiones de este servidor: la estable (solo soporta mp3, funciona muy bien) y la que está en desarrollo, (soporta mp3, ogg y *relay*[17] de otros servidores).También el servidor Icecast soporta en sus últimas versiones *streams* Ogg Vorbis, MP3, Ogg Speex, Ogg FLAC, Ogg Theora y AAC. El servidor Icecast tiene una funcionalidad similar al programa propietario de servidor de medios Shoutcast de Nullsoft y es compatible con este. Puedes enviar los archivos de audio a Icecast utilizando el *software* Simplecast.

Existen diversos programas para enviar el *streaming* hacia el servidor IceCast algunos de ellos son:

[16] **Streams: Arroyos, Corrientes.**
[17] **Relay: Retransmitir, pasar.**

- ✓ DarkIce: Es muy estable y permite tanto ogg como mp3 por lo que no es necesario tener una máquina dedicada para hacer esta tarea, ya que solo se pierde 3KB/s de conexión y la entrada de línea.
- ✓ MuSE (Múltiple *Streams* Engine): Permite mezclar diversas entradas de audio (otros servidores de *streaming*, ficheros mp3 y listas de mp3 (.pl (*playlist*[18])) y la entrada de micro; será adaptado para que soporte ogg. Además, permite cambiar el *bitrate*[19]del *stream* en el aire, sin tener que reiniciar las conexiones. Pero la versión actual es muy inestable y se segmenta con facilidad.

➢ Helix Universal Server

RealNetworks, fabricante de los códec RealVideo y RealAudio, ha lanzado un servidor multiplataforma, multiformato, y recientemente, de código libre. Se trata del versátil RealNetworks Helix Server. Es considerado un versátil servidor de archivos y *streaming* multimedia. Está disponible en versiones para Windows, GNU/Linux y Solaris. Esto permite que los usuarios de un sistema operativo no tengan que migrar toda su plataforma a otro sistema operativo. (Marín, y otros, 2009)

Algunas características de este servidor *streaming* son: liderazgo indiscutible en la industria de distribución de multimedia por internet, sencillez de uso y administración 100% de mejora en rendimiento sobre la versión 8 del RealServer arquitectura de distribución avanzada para redes públicas y empresariales soporte de codificación redundante soporte de formatos Real, Windows Media, y QuickTime, entre otros.

Helix Universal Server también es una garantía para la transmisión en directo, además posee características muy novedosas y ventajosas entre las que se encuentra el que pueda usar Sistemas Redundantes de Producción, en otras palabras es capaz de utilizar varias fuentes de codificación lo que prácticamente desaparece la posibilidad de falla del servidor casi asegurando la conexión en cualquier circunstancia. Este servidor también cuenta con la opción *archiving*[20] que consiste en grabar las emisiones en directo mientras se están transmitiendo, lo que permite la transmisión posterior del archivo bajo demanda, para el servicio que se pretende brindar esta característica es un regalo porque posibilitará el

[18] **Playlist: Lista de reproducción.**
[19] **Bitrate: Taza de bits.**
[20] **Archiving: Archivo**

ver la programación de los canales en vivo y en caso de no poder verlo pues posibilitará el verlo en el archivo previamente grabado, y este servidor está pensado para formatos móviles lo que lo convierte el que más llega a los usuarios.

- **Videolan Server**

Es un servidor dedicado que funciona bajo GNU/Linux, Windows y Mac OS X. Puede realizar *Streaming* de los formatos MPEG 1, 2 y 4 almacenados en disco duro o CDROM, además de Dvds localizado en lector DVD o copiado a disco duro, y videos en vivo sobre una red mediante tarjeta de codificación MPEG en *Unicast* o *multicast*, pero también puede transmitir canales digitales de satélite con Tarjeta de satélite (DVB-S) y canales digitales de televisión terrestre con tarjeta terrestre (DVB-T). Posee Soporte IPv4/6.

VLS está dotada de una interfaz de línea de comandos y de una interfaz Telnet pero no tiene una interfaz gráfica. Desde el punto de vista de un usuario, VLS puede ser dividido en varios tipos de componentes incluye: Gestor, entradas y salidas. El gestor controla el modo en el que son enviados los flujos a través de una interfaz de administración, se le puede decir al gestor que comience, pare, suspenda, o reinicie los diferentes programas. Se puede también conseguir la lista de todos los programas disponibles en la tabla de programas. El gestor consigue esta tabla del fichero de configuración de VLS (vls.cfg), por lo que no puede ser modificado una vez que VLS arranca. (Torres, 2009)

Una salida recibe un flujo de un conversor enviándolo a un destino dado (red o fichero). Actualmente existen dos tipos de salidas soportadas: Network y file. Por el momento, VLS solo soporta una salida por flujo, por lo que no se puede enviar al mismo tiempo a una red y escribir a un fichero. La salida de red es bastante configurable, se puede elegir que interfaz de red se quiere utilizar para especificar las direcciones IP de origen y destino. Actualmente, existen diferentes modos de controlar VLS, se puede usar la línea de comandos para proporcionarle argumentos en el arranque y mediante Telnet para arrancar/parar/pausar el envío siempre que se quiera.

Archivos capaces de leer: UDP/RTP *unicast* o *multicast*, HTTP, FTP, MMS, DVD, VCD, SVCD, CD Audio, DVB (Solamente en Windows y GNU/Linux).

Formatos con audio y video incrustados: 3GP, ASF, AVI, FLV, MKV, QuickTime, MP4, Ogg, OGM, WAV, MPEG-2 (ES, PS, TS, PVA, MP3), AIFF, Raw audio, Raw DV, FLAC.

Formatos de video: Cinepak, DV, H.263, H.264/MPEG-4 AVC, HuffYUV, Indeo, MJPEG, MPEG-1, MPEG-2, MPEG-4 Part 2, Sorenson, H.263 (vídeos de YouTube), Theora, VC-1, VP5, VP6, WMV Subtítulos: DVD, SVCD, DVB, OGM.

Formatos de audio: AAC, AC3, ALAC, AMR, DTS, DV Audio, FLAC, MACE, MP3, QDM2/QDMC, RealAudio, Speex, Screamtracker 3/S3M, TTA, Vorbis, WMA.

- **Darwin *streaming* server (DSS)**

Es la versión de código abierto para los servidores *streaming* QuickTime de Apple's, dicha tecnología permite enviar *stream* a los clientes mediante la Internet usando los protocolos estándares de la industria RTP y RTSP. Basándose en el mismo código base que QuickTime *Streaming* Server, Darwin *Streaming* Server proporciona un alto nivel de personalización y corre en una variedad de plataformas que le permiten manipular el código para satisfacer sus necesidades.

Darwin *Streaming* Server es un proyecto pensado por desarrolladores que necesitan difundir el *streaming* QuickTime y los videos MPEG – 4 en plataformas alternativas como Windows, GNU/Linux, y Solaris, o que cuando estos desarrolladores necesitan extender o modificar el código *streaming* existente en el servidor según sus necesidades. Permite hacer "*streaming*" de forma increíblemente sencilla, simplemente poner los videos en el directorio especificado y conectar los clientes. Darwin *Streaming* Server puede ser utilizado para transmitir *Unicast*, *Multicast* y *Relay*, esta última es muy interesante pues el servidor escucha la información entrante y la reenvía a uno o más destinos, estos pueden estar ubicados en distintos puntos geográficos sintonizando la misma señal. Los usuarios también pueden sintonizar transmisiones en directo o pregrabadas, o pueden acceder al contenido multimedia bajo demanda.

Darwin *Streaming* Server tiene dos tipos de autenticación que le permiten controlar el acceso al contenido multimedia, básico y clasificado, por defecto el servidor usa la autenticación clasificada como la más segura. La autenticación no controla el acceso a una transmisión desde un servidor *relay*, es

responsabilidad del administrador del servidor *relay* controlar el acceso al contenido transmitido (*relayed*) por su servidor. (Rodríguez, 2003)

El problema es que solo permite hacerlo para unos determinados formatos y por ejemplo DivX, no es uno de ellos. Por tanto, si se quiere disponer de ellos se deberá convertir los videos. Los formatos que acepta DSS son ficheros QuickTime (con extensión mov habitualmente) y MPEG – 4 (con diferentes posibles extensiones como avi, 3gp u otras). Una opción muy sencilla para convertir videos es usando el mismo QuickTime. (Torres, 2009)

Entre los formatos multimedia que puede transmitir se encuentra el estándar MPEG – 4, que pueden transmitirse directamente sin ser convertido a archivo .mov que es el formato propietario de Apple. Darwin no puede transmitir archivos MP3 bajo demanda pero si puede transmitir estos a clientes que soportan *streaming* MP3 vía HTTP, tales como iTunes, WinAmp y RealPlayer. El *hinting* (indicar o sugerir) crea un track (pista) para cada track multimedia en el archivo que le dice al Darwin *Streaming* Server cómo y cuándo entregar cada trama multimedia. El *hinting* también permite el uso de nuevos códecs sin necesidad de actualizar el servidor.

- **YouTube**

YouTube[21] es un sitio web que permite a los usuarios subir, bajar, ver y compartir vídeos digitales a través de internet. Es fácil de usar y, además, gratuito. Para ver los vídeos o enviarlos a otras personas no es necesario registrarse, aunque sí para colocarlos en la página. Es uno de los servicios popularmente más conocido y usado por la comunidad social de Internet. Fue fundado en febrero de 2005, aunque actualmente es propiedad de Google, desde su compra, el 10 de octubre de 2006 por 1.650 millones de dólares. El servicio que brinda YouTube presenta las siguientes características: incorpora buscador, un sistema de calificación de los vídeos. Permite a los usuarios calificar los vídeos de 1 a 5 estrellas; el sistema guarda, genera y muestra estadísticas del uso del vídeo (número de visualizaciones, comentarios, añadido a favoritos, y relación de sitios que enlazan el vídeo), además de otras funcionalidades que se le

[21] **Disponible en http://www.youtube.com**

han ido incorporando a media que pasa el tiempo. El sitio aloja una variedad de video clips de películas y programas de televisión, videos musicales, y vídeos caseros.

YouTube se apoya en el reproductor en línea basado en Adobe Flash para servir su contenido. Es muy popular gracias a la posibilidad de alojar vídeos personales de manera sencilla. Los servidores durante el proceso de "subida" (upload) del video se encargan de convertir este en dicho formato. Los formatos en los que se envía el vídeo son: MPEG, AVI, MOV, los utilizados por videocámaras y cámaras integrada en los teléfonos móviles y algunos otros. Para poder difundir toda la información que se encuentra alojada en el sitio, utiliza muchos servidores, dentro de ellos se puede destacar el uso del Darwin *Streaming* Server que lo utiliza para transmitir videos a móviles.

- **Digimeld**

Es un portal de distribución de vídeo y también de *streaming*, la cual posee una herramienta de vídeo de HD la cual se basa en la transferencia distribuida. Ésta hace hincapié en la computación distribuida, aquella que reparte o distribuye partes de su procesamiento entre varios nodos que forman parte de una red. Todos los usuarios de Digimeld son receptores y transmisores a la vez. Los expertos afirman que el funcionamiento de este tipo de redes ayuda en gran manera a disminuir el tráfico web y a mejorar la calidad de los contenidos que son distribuidos al llevarla a HD, solucionando dos problemas a la vez (Mashinsky, 2007)

Digimeld ha contado con la colaboración de la NASA, que ha puesto parte de su red NASA TV a disposición de este proyecto para su prueba. Como test final, el lanzamiento del satélite GeoEye de Google fue visto por más de 100.000 personas de manera simultánea y en calidad HD, sin problemas de intermitencias. (Klew, 2008)

- **Inter-nos**

Inter-Nos[22] es un sitio web que nace de un trabajo de diploma de un estudiante de la Universidad de las Ciencias Informáticas (UCI), el cual surge por la necesidad de distribuir contenidos audiovisuales por la

[22] **Disponible en http://inter-nos.uci.cu**

red en dicha Universidad, donde existe gran cantidad de computadoras distribuidas en diferentes áreas. El principal propósito que tiene este sitio es entretener, informar y educar a los usuarios que hagan uso de él. En el portal se publican las teleclases docentes, permitiendo a los estudiantes y profesores acceder a ellas desde cualquier lugar dentro de la Universidad, en cualquier momento y cuantas veces estime necesario, permitiendo personalizar la actividad docente de los estudiantes y trabajadores.

Para este sitio poder brindarle a los usuarios el servicio que ellos soliciten, se apoya en el servidor de *streaming* de la Microsoft el Windows Media Server 9, el cual brinda la posibilidad de acceder a los recursos multimedia en el momento en el que se está transmitiendo (*broadcasting*[23]) o a la información que se encuentra digitalizada y almacenada en el servidor (VOD). Para poder transmitir en vivo necesitara instalar Windows Media Encoder[24] en su ordenador.

- **Flumotion**

Otra solución encontrada es el Flumotion en su versión comercial, el cual es un servidor de *streaming* compatible con los formatos más utilizados en Internet (Flash, Windows Media, MP3, H264, AAC, AAC+…) que incluye la adquisición en su diseño descentralizado. Flumotion *Streaming* Server permite una gran escalabilidad y su administración es sencilla. Este *software* utiliza una tecnología homogénea que funciona sobre GNU/Linux.

Flumotion es multiformato, es decir, que es capaz de leer y convertir para su posterior reproducción en internet diferentes formatos, tanto para emisiones en Vivo como para video Bajo Demanda. Ahorra tiempo y dinero, simplifica la gestión y, además, puede responder con el formato más apropiado en cada momento. (Pegaz, y otros, 2006)

La versión libre transmite por el protocolo http y soporta H264, pero no posee muchas de las funcionalidades de la versión privada.

[23] **Broadcasting: Se refiere a transmisión, emisión o difusión de un flujo.**
[24] **Windows Media Encoder: Compresor para el formato Windows Media.**

1.5 Conclusiones parciales

Del estudio realizado se obtuvo que no existe alguna versión anterior al componente que se va a implementar, pero de los sistemas estudiados se detectó que poseen funcionalidades y características similares. De manera general se puede afirmar que el componente colector de datos presenta como función principal realizar el *streaming* y esta tarea la realizan todas las aplicaciones anteriores de diferentes formas. Es por ello que se hizo un estudio de las arquitecturas que utilizan y se pudo observar que existen diversos *software* que realizan *streaming* en su mayoría servidores *streaming,* quienes a su vez son utilizados en plataformas las cuales son las encargadas de interactuar con los usuarios.

Por tanto, se determinaron de cada aplicación las semejanzas y diferencias con respecto al componente y los inconvenientes que presenten, se corregirán con la implementación del componente para un mejor desarrollo de las funcionalidades del componente propuesto. Tal es el caso que el Real Server posee una técnica de *splitting,* la cual posibilita que los usuarios se conecten a visualizar los contenidos pero a través del servidor más cercano, lo que posibilita reducir el consumo de ancho de banda.

En la universidad también se han realizado trabajos relacionados con la tecnología *streaming* como es el caso de Inter-nos pero presenta inconvenientes ya que solo soporta formato .wmv porque se apoya en el servidor de la Microsoft Windows Media Server 9. Por último, se está utilizando igualmente el Flumotion *streaming* server en su versión libre que posee compatibilidad con algunos de los formatos utilizados en internet.

Con la investigación realizada se profundizó en el conocimiento de la tecnología *streaming*, se llegaron a diferentes criterios fundamentales para el desarrollo posterior del componente. También se encontraron funciones dentro de los sistemas distribuidos que son similares a la que se desea desarrollar. Se investigaron acerca de los diferentes formatos de video y protocolos que se transmiten en la actualidad, así como sus ventajas y desventajas para escoger los adecuados para la aplicación que se desarrollará. Además, se analizaron los tipos de transmisiones que existen y su uso en los diferentes *software* estudiado para implementarlos en el componente.

CAPÍTULO 2: TENDENCIAS Y TECNOLOGIAS A UTILIZAR.

2.1 Introducción

En este capítulo se realizará un análisis detallado de las tecnologías que se adecuan para la solución, se investiga acerca de las diferentes metodologías existentes en el mundo así como lenguajes de programación y herramientas de desarrollo para posteriormente según las características individuales escoger las adecuadas para la implementación de la aplicación. Esto servirá como base a los futuros planteamientos en el desarrollo de la investigación.

2.2 Metodologías

- **Proceso Unificado de Desarrollo (RUP).**

Es una metodología tradicional que hace mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos, además se adapta fácilmente a las condiciones del proyecto mediante su configuración previa a aplicarse.

Es un proceso de *software* genérico que puede ser utilizado para una gran cantidad de tipos de sistemas de *software* y que define un conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto asignando tareas y responsabilidades para asegurar la producción de *software* de muy alta calidad que satisfaga las necesidades de los usuarios finales, dentro de un calendario y presupuesto predecible. (Jacobson, y otros, 2000)

En su modelación se define como sus principales elementos:

Trabajadores (¿Quién?): Definen el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo.

Actividades (¿Cómo?): Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.

Artefactos (¿Qué?): Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

Flujo de actividades (¿Cuándo?): Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo.

El Proceso Unificado tiene dos dimensiones. (Jacobson, y otros, 2000)

Un eje horizontal que representa el tiempo y muestra los aspectos del ciclo de vida del proceso a lo largo de su desenvolvimiento. Un eje vertical que representa las disciplinas, las cuales agrupan actividades de una manera lógica de acuerdo con su naturaleza.

1) Modelación del negocio
2) Requerimientos.
3) Análisis y diseño
4) Implementación.
5) Prueba.
6) Despliegue.
7) Administración del proyecto.
8) Administración de configuración y cambios.
9) Ambiente.

RUP posee tres características principales: (Jacobson, y otros, 2000)

a) Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos.

b) Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que 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.

c) Interactivo e incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto.

- **Programación Extrema (XP)**

Es una de las metodologías de desarrollo de *software* más célebre en la actualidad, utilizada para proyectos de corto plazo, con un breve equipo y cuyo plazo de entrega es muy rápido. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, ya que es uno de los requisitos para llegar al éxito del proyecto.

Una de las herramientas más importantes en la metodología XP es el desarrollo orientado a pruebas, que utiliza las pruebas unitarias como eje de todo desarrollo. (Calderón, y otros, 2007)

Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelanta a los clientes en algo hacia el futuro, se pueda hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara para obtener los posibles errores.

Lo primordial en este tipo de metodología es:

La comunicación, entre los usuarios y los desarrolladores, la simplicidad, al desarrollar y codificar los módulos del sistema, la retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales. (Calderón, y otros, 2007)

XP se define especialmente para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Roles XP.

Los roles principales de acuerdo con la propuesta original de Beck son:

- ✓ Programador.
- ✓ Cliente.
- ✓ Encargado de pruebas (Tester).
- ✓ Encargado de seguimiento (Tracker).
- ✓ Entrenador (Coach).
- ✓ Consultor.
- ✓ Gestor (Big boss).

- ➢ **Microsoft Solution Framework (MSF)**

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

También proporciona prácticas comprobadas para planificar, crear y poner en marcha soluciones exitosas. En oposición a la metodología prescriptiva, MSF proporciona una estructura flexible y escalable para conocer las necesidades de una organización o equipo encargado de un proyecto de cualquier tamaño. La orientación de MSF consiste en proporcionar principios, modelos y disciplinas para manejar personas, procesos y elementos de tecnología con los que se encuentra la mayoría de los proyectos.

- ➢ **Características de MSF**

- ✓ *Adaptable:* es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.

- ✓ *Escalable:* puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas o más.
- ✓ *Flexible:* es utilizada en el ambiente de desarrollo de cualquier cliente.
- ✓ *Tecnología Agnóstica:* porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el Modelo de Aplicación. (Richard Mueller, 2005)

- ✓ *Modelo de Arquitectura del Proyecto*
- ✓ *Modelo de Equipo*.
- ✓ *Modelo de Proceso*.
- ✓ *Modelo de Gestión del Riesgo*.
- ✓ *Modelo de Diseño del Proceso*.
- ✓ *Modelo de Aplicación*.

- ➢ **Valoración de la metodología escogida.**

Después de estudiar lo referente a estas metodologías de desarrollo se escoge RUP debido a que aporta todos los elementos para desarrollar aplicaciones grandes y que requieren de mucha documentación, XP en cuanto a estos aspectos le resta importancia a la documentación y al lenguaje de modelado y por su parte MSF le resta importancia a la selección de las tecnologías que en el presente trabajo tiene prioridad por estar destinado a realizarse en *software* libre. También constituye una metodología adaptable al proyecto, utilizada para el análisis, implementación y documentación de sistemas a través del UML[25] (Unified Modeling Language), que implementa el Paradigma Orientado a Objetos.

[25] **UML: Lenguaje Unificado de Modelado**

Se escoge RUP además teniendo en cuenta la magnitud del proyecto que hace que las partes a entregar deban quedar bien documentadas internamente, la cantidad de miembros del equipo, el tiempo con que se dispone para su culminación, lo distante que se encuentran desarrolladores y clientes geográficamente.

Teniendo en cuenta la necesidad de una posterior integración con las aplicaciones que actualmente se desarrollan en el departamento de Señales Digitales principalmente los otros componentes del servidor de *streaming* distribuido, RUP sería la metodología que más se ajusta para solucionar esta situación.

Por lo anteriormente dicho, entre otros aspectos, RUP fue la metodología seleccionada para el desarrollo del proyecto.

2.3 Herramientas de modelo visual.

- **Herramientas Case.**

Las Herramientas CASE (Computer - Aided *Software* Engineering: Ingeniería de *Software* Asistida por Ordenador) son básicamente diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de *software* reduciendo el coste de las mismas partiendo de tiempo y dinero.

Las herramientas CASE son un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases de proceso.

- **Rational Rose Enterprise**

Rational Rose Enterprise es el producto más completo de la familia Rational Rose de IBM. Todos los productos Rational Rose, como el *Rose Data Modeler* para el diseño visual de bases de datos, el *Rose Modeler*, para el análisis y la arquitectura de *software*, entre otros, tienen soporte para modelado UML. Es un *software* privativo, potente para el ambiente de modelado que soporte la generación de código a partir de modelos en Ada, ANSI C++, C++, CORBA, Java™/J2EE™, Visual C++ y *Visual Basic*. Como todos los demás productos Rational Rose, proporciona un lenguaje común de modelado para el equipo que facilita la creación de *software* de calidad con mayor rapidez. (Innova, 2007)

- **Visual Paradigm.**

Constituye una herramienta CASE que utiliza UML, como lenguaje de modelado. Está disponible en varias ediciones: Enterprise, Professional, Community, *Standard*, Modeler y Personal, que permite escoger la más idónea para el usuario según su necesidad.

Soporta UML 2.1, permite realizar ingeniería tanto directa como inversa, tiene la capacidad de crear el esquema de clases a partir de una base de datos y crear la base de datos a partir del esquema de clases. Es una herramienta colaborativa, soporta múltiples usuarios trabajando sobre el mismo proyecto y brinda la posibilidad de que los mismos vean los cambios hechos en tiempo real; genera la documentación del proyecto automáticamente en varios formatos como Web o Pdf, y permite el control de versiones. Es multiplataforma y muy útil para la generación de código fuente en PHP. Por las múltiples ventajas con que cuenta, que es muy fácil de instalar y actualizar, que cuenta con una comunidad que brinda soporte técnico, y que es una herramienta de *software* libre, se selecciona Visual Paradigm con la herramienta para realizar el modelado visual del presente trabajo.

También es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El *software* de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. (International, 2004)

La versión 6.4 incluye un paquete de aplicaciones ofreciendo varias herramientas que facilitan el desarrollo de proyectos de cualquier dimensión.

- **Fundamentación de la herramienta propuesta: Visual Paradigm 6.4**

Al concluir el estudio realizado de estas herramientas, se escoge esta para el modelado del sistema, por las facilidades que brinda para el diseño UML del ciclo de vida de un proceso de desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Visual Paradigm es además completamente compatible con la metodología que se decidió utilizar, ofrece fluidez en la generación de la

documentación del *software* que se está desarrollando. Todo esto favorece un buen desarrollo del producto por lo cual se obtendrá una mayor calidad en el *software*.

También presenta soporte para la realización de la ingeniería inversa y generación de código. Es multiplataforma y mejora el tiempo de construcción de aplicaciones, posibilitando el modelado de todo tipo de diagramas de clases. Facilita la codificación desde diagramas así como el desarrollo de documentación de una aplicación. Además de poseer una amplia bibliografía tanto en materiales audiovisuales como documentación digital.

- **Framework QT 4.7.0. Características**

Un framework es una estructura conceptual y tecnológica de soporte definida, normalmente, con artefactos o módulos de *software* concretos, con base en la cual otro proyecto de *software* puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de *software* que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

Las principales ventajas de la utilización de un framework son:

1. El desarrollo rápido de aplicaciones. Los componentes incluidos en un framework constituyen una capa que libera al programador de la escritura de código de bajo nivel.

2. Los frameworks son los paradigmas de la reutilización.

3. El uso y la programación de componentes que siguen una política de diseño uniforme. Un framework orientado a objetos logra que los componentes sean clases que pertenezcan a una gran jerarquía de clases, lo que resulta en bibliotecas más fáciles de aprender a usar.

Qt es un framework para el desarrollo de aplicaciones multiplataforma. Algunas de sus características son:

- ✓ Compatibilidad multiplataforma con un sólo código fuente
- ✓ *Performance* de C++
- ✓ Disponibilidad del código fuente
- ✓ Excelente documentación
- ✓ Arquitectura lista para plugins.

Como ventajas tiene el hecho de tener una buena velocidad, debido a que está escrito en C++ y tener un muy buen diseño orientado a objetos. Además de ser multiplataforma y de *software* libre. También, soporta los lenguajes: C# /*.NET* Lenguajes (Mono), Python: PyQt y PySide, Ada, Pascal, Perl, PHP y Ruby. Es distribuido bajo los términos de GNU Lesser General Public License (y otras), Qt es *software* libre y de código abierto.

2.4 Herramientas de desarrollo.

➢ Lenguajes de programación

Un lenguaje de programación es una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen un lenguaje informático. Un lenguaje de programación permite a un programador especificar de manera precisa sobre qué datos una computadora debe operar, cómo deben ser estos almacenados y transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. (Aguilar, 2003)

➢ Los lenguajes de programación se pueden clasificar en 3 grupos según su nivel de trabajo:

Lenguajes de máquina: Es el único que entiende directamente la computadora, utiliza el alfabeto binario que consta de los dos únicos símbolos 0 y 1, denominados bits (abreviatura inglesa de dígitos binarios). Fue el primer lenguaje utilizado en la programación de computadoras, pero dejó de utilizarse por su dificultad y complicación, siendo sustituido por otros lenguajes más fáciles de aprender y utilizar, que además reducen la posibilidad de cometer errores. (Aguilar, 2003)

Lenguajes de bajo nivel: Proporciona poca o ninguna abstracción del microprocesador de un ordenador. Consecuentemente es fácilmente trasladado a lenguaje de máquina. En general se utiliza este tipo de lenguaje para programar controladores (*drivers*). (Aguilar, 2003)

Lenguajes de alto nivel: Logran la independencia del tipo de máquina y se aproximan al lenguaje natural.

A continuación se caracterizarán algunos de los lenguajes de alto nivel más usados en la actualidad.

- **Java**

Java es un lenguaje dinámico simple, orientado a objetos, distribuido, interpretado, robusto, de arquitectura neutra, portable, de alto rendimiento y multihilo. De apariencia similar a C++, pero sin sobrecarga de operadores, herencia múltiple, y aritmética de punteros. Tiene recolector de basura. Primitivas de sincronización basadas en los monitores que se encuentran en Cedar y Mesa. Diseñado para ser seguro, con técnicas de autenticación basada en cifrado de clave pública. Interpretado con una pila a base de máquina virtual. Una interfaz de red permite que el código compilado pueda ser enviado a través de Internet y ejecutado remotamente, lo que permite a los usuarios añadir sus programas a páginas web. (García de Jalón, y otros, 2002)

- **C++**

El C++ es a la vez un lenguaje procedural (orientado a algoritmos) y orientado a objetos. Como lenguaje procedural se asemeja al C y es compatible con él, aunque ya se ha dicho que presenta ciertas ventajas. Como lenguaje orientado a objetos se basa en una filosofía completamente diferente, que exige del programador un completo cambio de mentalidad. Las características propias de la Programación Orientada a Objetos (Object Oriented Programming, u OOP) de C++ son modificaciones mayores que sí que cambian radicalmente su naturaleza.

Proporciona nuevas características útiles en diversos contextos como la sobrecarga de operadores. Su expresividad es elevada, pues la sintaxis de clases y objetos permite manipular convenientemente diversas estructuras de datos y operaciones; las excepciones permiten procesar de un modo claro los casos de errores. Es un lenguaje muy difundido y popular por lo que suele ser usado para resolver

diferentes tipos de problemas, desde los más simples hasta los más complejos, su código se puede compilar en diversas plataformas (García de Jalón, y otros, 1998)

- **C#**

Microsoft C # es un lenguaje de programación diseñado para la construcción de una amplia gama de aplicaciones empresariales que se ejecutan en *.NET* Framework. Una evolución de Microsoft C y C + +, C # es simple, moderno, seguro y orientado a objetos. El código C # se compila como código administrado, lo que significa que utiliza los beneficios de los servicios de Common Language Runtime. Estos servicios incluyen el idioma interoperabilidad, la recogida de basura, la mejora de la seguridad, el apoyo y la mejora de versiones.

C # es introducido como Visual C # en Visual Studio. *NET* suite. Soporte para Visual C # incluye plantillas de proyecto, diseñadores, páginas de propiedades, asistentes de código, un modelo de objetos, y otras características del entorno de desarrollo. La biblioteca de programación de Visual C # es el de. *NET* Framework. (Corporation, 2000)

- **Python**

Python es un lenguaje de programación dinámico orientado a objetos que pueden utilizarse para diversas formas de desarrollo de *software*. Ofrece un fuerte apoyo para la integración con otros idiomas y herramientas, posee una gran gama de bibliotecas, y su aprendizaje puede ser en poco tiempo. Python se puede ejecutar en diferentes sistemas operativos como Windows, GNU/Linux, Mac OS X y algunos teléfonos móviles. Este lenguaje también ha sido portado a Java y *.NET*. Python es distribuido bajo una licencia de código abierto que hace que sea libre de usar, incluso para productos comerciales. (Fundation, 1990)

- **Fundamentación del lenguaje propuesto: C++**

Se escoge este lenguaje de programación después de haber realizado un estudio detallado y profundo de las características que poseen cada uno de ellos. Posteriormente se llega a la conclusión de que el lenguaje de programación con mejores opciones para el desarrollo de la aplicación es C++, ya que tiene características que lo exaltan por encima de los demás al ser muy didáctico, además en el caso de Java

las aplicaciones resultantes son muchas más lentas que las derivadas de C++ y tienen dependencia de la máquina virtual, por lo que su uso no es el más adecuado a los propósitos finales del componente y en el caso de Python al ser un lenguaje interpretado las aplicaciones finales se harían más lentas.

Por otra parte, para desarrollar el sistema propuesto se hace necesaria la utilización eficiente de los recursos de *hardware* de la computadora. El sistema debe ser lo más ligero posible dado el consumo de procesamiento y memoria que requerirán sus funcionalidades, por esas razones es seleccionado C++, ya que es un lenguaje versátil y potente.

Además de las funcionalidades que brinda es el lenguaje nativo del framework de desarrollo Qt y debido a las ventajas que presenta el framework hace más dinámico el desarrollo del componente Colector de Datos. Actualmente, puede compilar y ejecutar código de C, ya viene con librerías para realizar esta labor que mejoran el progreso de implementación.

2.5 Librerías para hacer streaming

El término librería se utiliza para referirse a un conjunto de módulos objeto .obj / .o (resultados de compilación) agrupados en un solo fichero que suele tener las extensiones .lib, .bpl .a, .dll, etc. Estos ficheros permiten tratar las colecciones de módulos como una sola unidad, y representan una forma muy conveniente para el manejo y desarrollo de aplicaciones grandes, además de ser un concepto muy fértil para la industria del *software*, ya que permiten la existencia de las librerías de los propios compiladores y de un mercado de utilidades y componentes adicionales.

- **Tipos**

En lo que respecta al lenguaje C++, existen dos tipos fundamentales de librerías: estáticas y dinámicas, que aunque comparten el mismo nombre genérico "librería", utilizan mecanismos distintos para proporcionar su funcionalidad al ejecutable.

En ambos casos es costumbre, que junto a las librerías propiamente dichas (ficheros .lib, .a, .dll etc), se incluya un fichero .h denominado "de cabecera", porque es tradición utilizar las primeras líneas del programa para poner las directivas #include que los incluirán en el fuente durante la fase de pre proceso.

Este fichero contiene las declaraciones de las entidades contenidas en la librería, así como las macros y constantes predefinidas utilizadas en ella, de forma que el programador solo tiene que incluir el correspondiente fichero .h en su aplicación para poder utilizar los recursos de la librería en cuestión. Este sistema tiene la ventaja adicional de que proporciona al usuario la información mínima para su uso. Es decir, la "interfaz" de las funciones o clases que utilizará. En el caso de funciones esto se concreta en el prototipo; en el caso de clases, en la especificación de sus métodos y propiedades públicas.

- **Librerías estáticas**

Denominadas también librerías-objeto, son colecciones de ficheros objeto (compilados) agrupados en un solo fichero de extensión .lib, .a, etc. junto con uno o varios ficheros de cabecera (generalmente .h).

Durante la construcción de la aplicación, el preprocesador incluye en los archivos fuentes los ficheros de cabecera. Posteriormente, durante la fase de enlazado, el *linker*[26] incluye en el ejecutable los módulos correspondientes a las funciones y clases de librería que hayan sido utilizadas en el programa, de forma que el conjunto entra a formar parte del ejecutable. De ahí su nombre: librerías enlazadas estáticamente.

Dejando aparte consideraciones de comodidad y rapidez, el resultado de utilizar una de tales librerías no se diferencia en nada al que puede obtenerse escribiendo en la fuente las funciones o clases correspondientes y compilándolas como un módulo más de la aplicación.

- **Librerías dinámicas**

Otra forma de añadir funcionalidad a un ejecutable son las denominadas librerías de enlazado dinámico, generalmente conocidas como DLLs, acrónimo de su nombre en inglés ("Dynamic Linked Library"). Estas librerías se utilizan mucho en la programación para el SO Windows. Este Sistema contiene un gran número de tales librerías de terminación .DLL, aunque en realidad pueden tener cualquier otra terminación .EXE, .FON, .BPI, .DRV, etc. Cualquiera que sea su terminación, de forma genérica se refiere a ellas como DLLs, nombre por el que son más conocidas.

[26] **Linker: Enlazador**

- **Diferencias entre las librerías**

Las diferencias más relevantes de las librerías dinámicas respecto a las estáticas son fundamentalmente dos:
Las librerías estáticas quedan incluidas en el ejecutable, mientras las dinámicas son ficheros externos, con lo que el tamaño de la aplicación es mayor en el primer caso que en el segundo. Esto puede ser de capital importancia en aplicaciones muy grandes, ya que el ejecutable debe ser cargado en memoria de una sola vez.
Las librerías dinámicas son ficheros independientes que pueden ser invocados desde cualquier ejecutable, de modo que su funcionalidad puede ser compartida por varios ejecutables. Esto significa que solo se necesita una copia de cada fichero de librería (DLL) en el Sistema. Esta característica constituye la razón principal de su utilización, y es también origen de algunos inconvenientes, principalmente en sistemas como Windows en los que existen centenares de ellas.

- **LibVLC**

Esta librería esta escrita en C, posee una variedad de funcionalidades que facilitan el flujo de streaming por la red. Brinda la posibilidad de realizar el *streaming* por diferentes protocolos. Además, se puede efectuar *streaming* bajo demanda y en vivo. También soporta varios formatos de video los cuales son muy utilizados en internet.

- **Libgstreamer**

Es una librería del framework Gstreamer. Este framework multimedia libre multiplataforma escrito en el lenguaje de programación C y usa la biblioteca GObject.

Permite crear aplicaciones audiovisuales, como de vídeo, sonido, codificación, etc. Con el uso de esta librería se puede reproducir música o realizar tareas más complejas como mezclar audio y vídeo para su transmisión. Soporta varios formatos de video y diferentes protocolos para realizar el flujo de *streaming*.

- **Fundamentación de la librería propuesta: libVLC 1.1.4.1.**

Se escoge esta librería por las características que brinda las cuales son de ayuda para el desarrollo del componente. Las diferentes particularidades que posee proporcionan la realización del *streaming* la cual es la función principal de la aplicación a través de diferentes tipos de transmisiones y de protocolos que proveen un buen servicio a los clientes. Por otra parte, se analizó otra librería como libgstreamer pero esta presenta inconveniente cuando provee el flujo de *streaming*, ya que esta lo realiza por protocolos como TCP[27] y DCCP[28] los cuales no se recomiendan usarlos en la transmisión de archivos multimedia por la red porque en caso de un error en la transmisión provocará que se replique ese flujo en la red. Además, la libvlc transmite por RTP[29], RTSP[30] y UDP[31], que son los recomendados para la transmisión de archivos de audio y video, pero la libgstreamer realiza el flujo de *streaming* por algunos de estos protocolos pero envía el audio y video por diferentes puertos lo que provoca que los reproductores no lo capturen unidos.

2.6 Entornos de Desarrollo Integrado (Integrated Development Environment)

Un Entorno de Desarrollo Integrado (IDE) es un conjunto de herramientas de desarrollo de *software* para programadores. Dentro de las partes fundamentales que lo integran se encuentra el editor de código, compilador, depurador y un constructor de interfaz gráfica de usuario. Los IDE son creados para desarrollar aplicaciones generalmente en un sólo lenguaje de programación. Sin embargo, hay algunos en los que se puede desarrollar en más de un lenguaje, como son el caso de Netbeans, Eclipse y Visual Studio por citar algunos.

- **Eclipse**

Eclipse es una comunidad Open Source (***código abierto***), cuyos proyectos se centran en la construcción de una plataforma de desarrollo abierta formada por marcos extensibles, herramientas y rutinas para crear, desplegar y gestionar *software* en todo el ciclo de vida. El IDE es compatible con varios de los sistemas operativos más conocidos como son, todas las versiones de Windows a partir del 98, GNU/Linux

[27] **TCP: Protocolo de control de transmisión**
[28] **DCCP: Protocolo de control de congestión de datagramas**
[29] **RTP: Protocolo de transporte de tiempo real**
[30] **RTSP: Protocolo de flujo de datos en tiempo real**
[31] **UDP: Protocolo de datagramas de usuario**

y Mac OS X. Es un IDE muy usado en la actualidad por su portabilidad y por la gran cantidad de lenguajes en los que se puede desarrollar por lo que brinda un gran servicio de soporte a través de la web. (Eclipse, 1995)

- **Principales ventajas**

Mediante plugins se le pueden añadir multitud de funcionalidades, además es libre y tiene una amplia comunidad que ofrece considerable documentación sobre el trabajo con él.

- **Principal Desventaja**

Entre sus principales desventajas está que consume gran cantidad de recursos del ordenador lo cual no es despreciable cuando se trata de aplicaciones que utilizarán funcionalidades del sistema operativo que ralentizan la máquina.

- **Netbeans**

El entorno de desarrollo Netbeans es una herramienta libre y gratuita que permite a los programadores escribir, compilar, depurar y ejecutar programas. Especialmente fue desarrollado para el lenguaje Java pero perfectamente se puede utilizar en otros como C/C++, Ruby, PHP, PERL, entre otros. NetBeans IDE es fácil de instalar y de uso instantáneo y se ejecuta en varias plataformas incluyendo Windows, GNU/Linux, Mac OS X y Solaris. Existe además un número importante de módulos que posibilitan la extensión de este IDE (Netbeans, 2000)

Aparte de la filosofía de distribución y desarrollo que respalda a NetBeans, el IDE ofrece a los desarrolladores numerosas ventajas, en la creación de nuevas aplicaciones multiplataforma.

- **Microsoft Visual Studio**

Es un entorno de desarrollo integrado para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y *Visual Basic .NET*, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma *.NET* (a partir de la versión *NET* 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

Visual Studio 2010 es la versión más reciente de esta herramienta, acompañada por *.NET* Framework 4.0. Hasta ahora, uno de los mayores logros de la versión 2010 de Visual Studio ha sido el de incluir las herramientas para desarrollo de aplicaciones para Windows 7, tales como herramientas para el desarrollo de las características de Windows 7.

Entre sus más destacables características, se encuentran la capacidad para utilizar múltiples monitores, así como la posibilidad de desacoplar las ventanas de su sitio original y acoplarlas en otros sitios de la interfaz de trabajo. Además de esto, aparece una edición que compila las características de todas las ediciones comunes de Visual Studio: Professional, Team Studio, Test, conocida como Visual Studio Ultimate.

- **QT Creator**

Qt es un conjunto de librerías multiplataforma diseñadas principalmente para desarrollar interfaces gráficas de usuario, aunque es posible desarrollar una aplicación completa haciendo sólo uso de estas librerías. Qt utiliza el lenguaje de programación C++ de forma nativa y es utilizada principalmente en KDE, un entorno de escritorio para sistemas como GNU/Linux o FreeBSD, entre otros.

Qt Creator es un IDE (Entorno de Desarrollo Integrado) que se emplea para programar en C++ usando las librerías de Qt. Este IDE es distribuido bajo tres tipos de licencias: Qt Commercial Developer License, Qt GNU LGPL v. 2.1, Qt GNU GPL v. 3.0 y está disponible para las plataformas: Linux, Mac OSX; Windows, Windows CE, Symbian y Maemo.

- **Principales características de Qt Creator:**
  - ✓ Posee también una GUI integrada y diseñador de formularios.
  - ✓ Ayuda sensible al contexto integrado.

- ✓ Resaltado y auto-completado de código.
- ✓ Soporte para refactorización de código.

- **Fundamentación de la herramienta propuesta: QT Creator 2.0.1.**

Entre las características principales esta que posee un editor avanzado para C++. Esta herramienta es de las más utilizadas en los proyectos del centro de GEYSED por lo que existe una gran capacitación para facilitar la ayuda. Posee ventajas con respecto a otros IDE como son herramientas para la administración, construcción de proyectos y depurador visual. Una amplia gama de métodos abreviados de teclado y navegación están disponibles para ayudar a acelerar el proceso de desarrollo de su aplicación. También permite la integración de manera sencilla con muchas librerías implementadas en C como la "libvlc", la cual será utilizada para realizar el proceso de *streaming* para el consumo de los clientes. El mismo cuenta con una amplia documentación incluyendo ejemplos sobre el trabajo con la mayoría de sus funcionalidades. Uno de los principales elementos que inclinaron la balanza en favor de este IDE es que el mismo es *software* libre con licencia LGPL.

2.7 Middlewares orientados a objetos

Es un hecho que las aplicaciones actuales necesitan interactuar con el «mundo exterior». La mejora en las comunicaciones, las nuevas necesidades de los usuarios y otros muchos factores han contribuido a que los sistemas de información actuales ofrezcan servicios en red y funcionen en un entorno distribuido.

Los middlewares son plataformas que proporcionan soporte al desarrollo de aplicaciones distribuidas y abstraen, en mayor o menor medida, de la tecnología y protocolo de red. El objetivo ideal de cualquier middleware es ofrecer una visión abstracta al programador sobre la tecnología de red que se utiliza para la comunicación real de las aplicaciones. (Cleto, 2010)

Existen gran variedad de middlewares en el mercado. En las siguientes secciones se presentarán algunos de los más utilizados.

- **ZeroC Ice.**

Internet Communications Engine (Ice) es un middleware orientado a objetos con licencia GPL construido por la empresa ZeroC. Existen componentes básicos en el middleware Ice para realizar las funciones mínimas necesarias. Como se dijo anteriormente, estos componentes son dependientes del propio producto Ice, pero es posible encontrar estructuras similares en otros middlewares orientados a objetos.

Cliente y servidor

Los conceptos de cliente y servidor en Ice varían poco en lo fundamental: el cliente requiere funcionalidad que el servidor proporciona. No obstante, y debido a los conceptos anteriores, cabe destacar que los servidores Ice necesitan los adaptadores de objetos para poder añadir los sirvientes necesarios para proporcionar el servicio. Por su parte, los clientes únicamente necesitan un *proxy* al servidor para poder solicitar la funcionalidad requerida.

- **Licencias**

Los programas de ordenador suelen distribuirse con licencias propietarias o cerradas. Estas licencias son intransferibles y no exclusivas, es decir, no eres propietario del programa, sólo tienes derecho a usarlo en un ordenador o tantos como permita expresamente la licencia y no puedes modificar el programa ni distribuirlo. ICE es *software* libre y está liberado bajo la licencia GNU/GPL.

La licencia GPL o General Public License, desarrollada por la FSF o Free *Software* Foundation, es completamente diferente. Se puede modificar el programa para adaptarlo a lo que se desee. Como inconveniente presenta que se debe facilitar siempre con el programa binario el código fuente, es decir, el programa de forma que pueda ser leído por un programador. Los programas propietarios o cerrados, solo se distribuyen en binario, listos para ejecutarse en el ordenador.

- **CORBA.**

Uno de los middlewares más extendidos actualmente es el conocido como Common Object Request *Broker* Architecture (CORBA). Se trata de un estándar del Object *Management* Group (OMG) en el que se

describe un modelo de comunicación de un sistema distribuido. El OMG es el encargado de definir la API, funcionalidad, servicios y arquitectura que una implementación de CORBA debe proporcionar.

La arquitectura cliente-servidor de CORBA es muy similar a la de Ice. Interface Definition Language (IDL) es el lenguaje de especificación de interfaces de CORBA. Al igual que Slice, IDL permite definir el contrato entre el cliente y servidor. Existen herramientas que traducen IDL a un lenguaje de programación concreto para que pueda ser utilizado por las aplicaciones.

El protocolo de red que utiliza CORBA se llama General Inter-ORB Protocolo (GIOP), también definido por el OMG. La implementación sobre TCP de este protocolo se conoce como Internet Inter-ORB Protocolo (IIOP). En este protocolo se definen los formatos y tipos de mensajes, las normas de comunicación entre clientes y servidores CORBA y la representación de las estructuras en los mensajes.

Una implementación de CORBA con licencia pública es The ACE ORB (TAO). Algunas otras implementaciones de CORBA con licencia propietaria son:

- ✓ Familia OpenFusion.
- ✓ Familia ORBexpress. (Cleto, 2010)

- **Java RMI.**

Java Remote Method Invocation (RMI) es un middleware creado por Sun Microsystems para aplicaciones Java. De manera general el funcionamiento es el siguiente: El servidor utiliza el rmiregistry para registrar los objetos que serán accesibles por los clientes. Utilizando el sistema de nombres de RMI (RMI's simple namingfacility), el servidor identifica a cada objeto dentro del rmiregistry. Cada objeto registrado en el rmiregistry debe implementar, al menos, la interfaz Remote para que pueda ser accesible. Esta interfaz se encuentra en la biblioteca estándar de RMI. El cliente obtiene la referencia a los objetos que el servidor público a través del rmiregistry. Finalmente, el cliente puede invocar métodos sobre los objetos remotos utilizando dicha referencia de forma muy similar a si el objeto estuviera accesible localmente. El contrato entre el cliente y el servidor se especifica utilizando interfaces Java. Por este motivo, el lenguaje de implementación de los clientes y servidores que utilicen RMI sólo puede ser Java (Cleto, 2010)

- **Fundamentación de la herramienta propuesta: ZeroC Ice.**

Además de proporcionar la funcionalidad básica de un middleware, Ice provee de diferentes servicios ya implementados para propósitos más específicos como: Canales de eventos: IceStorm es el servicio de Ice que proporciona comunicación entre clientes y servidores. Por ejemplo, existen nodos clientes que quieren recibir vídeo de un nodo servidor. Este último puede publicar el vídeo en un canal de eventos para tal fin y los clientes que quieran recibir el vídeo sólo tienen que suscribirse a dicho canal.

Despliegue: uno de los principales problemas de los sistemas distribuidos es que en los nodos debe residir el *software* de aplicación y, a veces, la tarea de distribución y actualización del *software* puede ser costosa. Para ello, Ice incluye IcePatch que permite transferir archivos de forma automática entre nodos del sistema distribuido.

2.8 Conclusiones parciales

Al finalizar este capítulo recogió el resultado de todo el estudio y la investigación relacionada a las tendencias y tecnologías actuales a desarrollar. También se logró investigar acerca de las diferentes tecnologías actuales y se compararon según las ventajas y desventajas de cada una para escoger las idóneas para el componente. Se llegó a la conclusión de que la propuesta a desarrollar utilizará como metodología de desarrollo RUP por ser adaptable; UML como lenguaje de modelado por ser de fácil asimilación y entendimiento; Visual Paradigm como herramienta case porque soporta el ciclo de vida completo del desarrollo de *software* y ser multiplataforma, C++ como lenguaje de programación por estar muy unido al *hardware* con elementos que le permiten un estilo de programación con alto nivel de abstracción; y como entorno de desarrollo Qt Creator con las librerías correspondientes al framework Qt por las funcionalidades que ofrece y ser multiplataforma.

CAPITULO 3: CONSTRUCCIÓN DE LA SOLUCIÓN PROPUESTA

3.1 Introducción

En el presente capítulo se hace referencia al análisis y diseño de la propuesta de sistema siguiendo la guía metodológica del RUP, se abordará lo relacionado al Modelo de Dominio, el levantamiento de los requisitos funcionales y no funcionales de *software*, del Diagrama de Casos de Uso del Sistema (DCUS) y la descripción detallada de los mismos, así como también los diagramas de clases del diseño lo cual es de gran ayuda para la fase de implementación.

3.2 Modelo de dominio

El Modelo de Dominio o Modelo Conceptual, permite de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio del problema. Un Modelo del Dominio es una representación de las clases conceptuales del mundo real, no de componentes *software*. Es empleado fundamentalmente cuando los flujos de información son difusos, es decir, que tengan múltiples orígenes y cuando son solo eventos o sucesos. También la imposibilidad de realizar subsistemas en el proceso de desarrollo del *software* dado por las múltiples interconexiones, es uno de los factores que influyen en la decisión de realizar un modelo de este tipo.

El modelo desarrollado no se trata de un conjunto de diagramas que describen clases de *software* u objetos de *software* con responsabilidades, sino que puede considerarse como un diccionario visual de las abstracciones relevantes, vocabulario e información del dominio. Cuando no es posible identificar claramente los procesos del negocio, la metodología RUP propone realizar un modelo de dominio, que es un subconjunto del modelo de negocio. (Loja, 2008)

3.2.1 Definición de clases del modelo de dominio

- **Manejador de Clúster.**

Componente que interactúa con el sistema para solicitar un fichero de media para un cliente determinado.

- **Servidor de Medias.**

Es el conjunto de clases encargada de almacenar todas las medias.

- **Contenido.**

Es la clase que se encargará de facilitar la gestión de los contenidos a ofrecer al usuario, o sea, los archivos que estarán disponibles para ser consumidos, por ejemplo: series, películas y música.

- **Paquete de Media.**

Es la media que se visualizará según la petición del cliente.

- **Transmisión.**

Es la clase encargada de realizar la transferencia del archivo de acuerdo con el tipo de solicitud de transmisión, puede ser en Vivo o Bajo Demanda.

En la fig. 2 se muestra las relaciones entre estas clases.

- **Diagrama de clases del modelo de dominio**

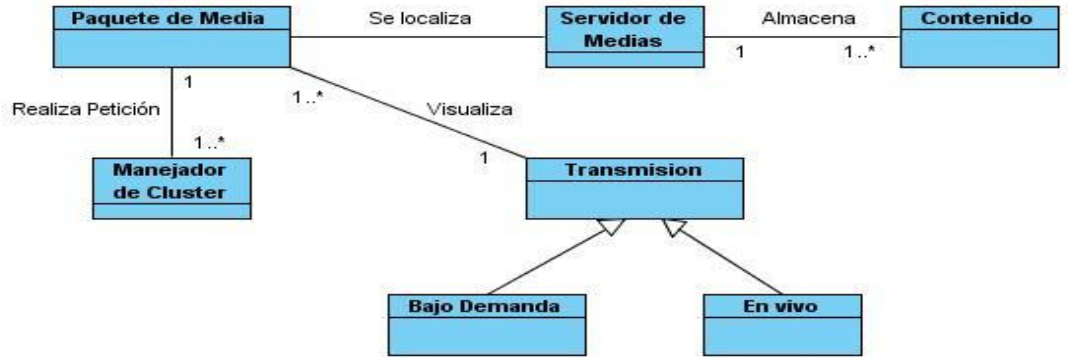

Fig. 2. Diagrama de clases del modelo de dominio.

- **Breve descripción del diagrama**

El sistema comienza a funcionar cuando el componente manejador de clúster recibe una petición de un cliente determinado, el sistema procederá a localizar la media solicitada en el servidor de medias el cual

almacena diferentes tipos de contenido como son: películas, series, documentales y otros. Después de ubicarla se efectuará un proceso a la media para que el cliente pueda visualizarlo según los parámetros definidos anteriormente.

3.3 Requerimientos

Los requisitos funcionales y no funcionales muestran las capacidades o condiciones que el sistema debe cumplir y las propiedades o cualidades que el producto debe tener, los cuales en la fase de construcción deben ser posibles de probar o verificar. Los requerimientos de *software* son las necesidades de los clientes, los servicios que los usuarios desean que proporcione el sistema de desarrollo y las restricciones en las que debe operar. (Pressman, 2000)

3.3.1 Requerimientos funcionales

Los requerimientos funcionales no son más que la determinación clara y concisa de qué debe ser capaz de hacer el sistema, éstas se corresponden con operaciones realizadas de forma oculta o condiciones extremas a determinar por el sistema. (Pressman, 2005)

- **RF1 Ubicar la media.**

Consiste en localizar los servidores que contienen la ubicación de los fragmentos de la media solicitada.

- **RF2 Conformar la media.**

El sistema ensambla la media hasta llenar el *buffer* para posteriormente comenzar el *streaming*.

- **RF3 Efectuar *streaming*.**

Cuando se ha llenado el *buffer*, el cliente lo comienza a visualizar. El sistema está sincronizado para que el archivo se pueda ver mientras se descarga si lo desea el cliente.

- **RF4 Definir parámetros de transmisión.**

El sistema debe permitir que se definan diferentes características de acuerdo con el tipo de transmisión solicitada lo cual incluye protocolos a transmitir y puertos.

3.3.2 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener, además son aspectos importantes que el producto debe cumplir para lograr un producto atractivo, usable, rápido o confiable y agradable para los usuarios. A continuación se enuncian, separados en categorías, los diferentes requisitos no funcionales que el componente debe satisfacer. (Pressman, 2005)

- **RNF 1 Eficiencia**

La velocidad de procesamiento y la respuesta del sistema, dependerán de la cantidad de información a procesar y la infraestructura tecnológica.

- **RNF 2 Requerimientos del *software***

La construcción de la aplicación funcionará bajo una arquitectura distribuida basada en clúster.

Requerimientos mínimos para los servidores:

- ✓ Sistemas operativos GNU/Linux 10.10 o superior.
- ✓ Poseer instalada la librería libVLC 1.1.4.
- ✓ Framework Qt, Licencia Qt GNU LGPL v. 2.1.
- ✓ Biblioteca Ice 3.4.0, Licencia GNU v2.

- **RNF 3 Requerimientos del *Hardware***

- Se requiere tarjeta de red.
- Memoria RAM de 1GB o superior
- Disco duro de 80GB o superior
- Procesador 3 GHz o superior.

- **RNF 4 Fiabilidad**

Disponibilidad: el sistema debe estar disponible todas las horas del día.

3.4 Descripción del sistema

Después de haber realizado un estudio de la descripción de los requisitos, quienes constituyen las funcionalidades esenciales del sistema, se le da paso a la siguiente etapa de construcción de un *software* que es la descripción del Modelo de Casos de Uso del Sistema (MCUS). El MCUS describe un sistema en cuanto a su utilización, es un modelo que contiene actores, casos de uso y las relaciones que se establecen entre ellos.

3.4.1 Definición de actores

Un actor del sistema representa un conjunto coherente de roles que juegan los usuarios. Además, él no es parte del sistema, es el encargado de inicializar los casos de usos del sistema, y es beneficiado con el resultado de los mismos. Es una entidad externa al sistema que se modela y que puede interactuar con él. Los actores representan a los seres humanos, a un *software*, *hardware* externo u otros sujetos que interactúen con el sistema que se esté especificando. (Sommerville, 2005)

Durante el desarrollo de la aplicación se definió un único actor, él mismo se describe a continuación:

Tabla 1. Descripción de los actores del sistema.

Actor	Descripción
Llamador	**Es el sistema que interactúa con el colector de datos (CD), da inicio a los casos de usos que conforman dicho componente. Este sistema realiza una serie de solicitudes las cuales deberán ser atendidas por el componente CD y así satisfacer las peticiones.**

3.4.2 Modelo de casos de usos del sistema

Un diagrama de casos de uso presenta la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios u otros sistemas. A partir de la identificación del actor que interactúa con la

aplicación, así como la recopilación del conjunto de funcionalidades escritas en forma de requerimientos, que a su vez han sido agrupados en casos de uso según sus peculiaridades, se conforma el Modelo de Casos de Uso que se representa a continuación. (Sommerville, 2005)

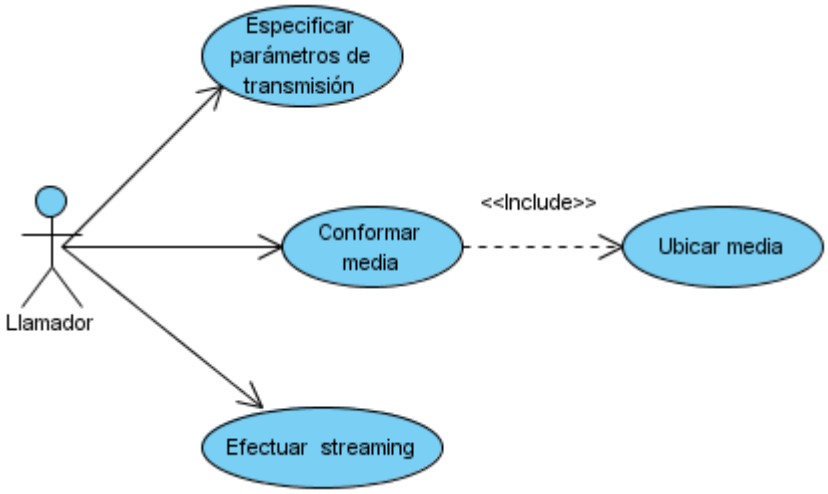

Fig. 3. Diagrama de Casos de Uso del Sistema.

3.5 Descripción textual de casos de usos del sistema

Cada uno de los casos de uso identificados presenta características particulares que para su mayor entendimiento se hace necesario describirlas textualmente. Además, permite lograr un mejor rendimiento del tiempo en la construcción del *software* es recomendable identificar los casos de uso arquitectónicamente significativos, que en la descripción textual se diferencian porque tienen la prioridad de crítico. Estos describen qué hace el sistema, no cómo lo hace. Los casos de uso se obtienen de los servicios requeridos por los usuarios finales, lo que significa que representan un buen hilo conductor como guía en el desarrollo de *software*. (Jacobson, y otros, 2000)

A través del modelado de los casos de usos se pueden observar las funciones del futuro sistema. No es realmente una aproximación a la orientación a objetos; es una forma de modelar procesos y de ayudar al cliente con las perspectivas del sistema. A continuación se brinda un resumen de los casos de usos identificados para el sistema.

- **Ubicar Media**

Tabla 2. Descripción del CU Ubicar Media.

Caso de Uso:	Ubicar Media.	
Actores:	Llamador	
Resumen	El caso de uso se lleva a cabo con el objetivo de ubicar la media para que pueda ser visualizada al usuario posteriormente.	
Precondiciones:	Se debe realizar la petición con los parámetros establecidos.	
Referencias	RF1	
Prioridad	Crítico	
Flujo Normal de Eventos		
Acción del Actor		**Respuesta del Sistema**
1. Solicita la localización de la media.		1.1 El sistema localiza los servidores donde se encuentra almacenada la media. 1.2 Devuelve la dirección para su posterior gestión.
Flujos Alternos		
Acción del Actor		**Respuesta del Sistema**
		1.1 Se emite mensaje de error porque no está disponible algún servidor donde se encuentra almacenada la media. 1.2 Se emite mensaje de error porque no existe la media.

- **Efectuar streaming**

Tabla 3. Descripción del CU Efectuar *Streaming*.

Caso de Uso:	Efectuar *Streaming*.
Actores:	Llamador
Resumen:	Cuando se confecciona el *buffer*, el cliente lo comienza a visualizar.
Precondiciones:	Se debe conformar el archivo de media.
Referencias	RF2, RF3.
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Solicita al sistema la necesidad de visualizar la media según el tipo de transmisión requerida en la petición.	1.1. El sistema comienza a transmitir la media a través de un proceso sincronizado para los clientes. Dicho flujo de acuerdo con el tipo de transmisión, el usuario podrá pausar, detener, adelantar o atrasar la reproducción.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	1.1 Se emite mensaje de error cuando el proceso de llenado del *buffer* se detiene, por lo que no se puede continuar la transmisión.

Para ver la descripción de los demás casos de uso ver Anexo 2 y 3.

3.6 El diseño de *software*

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. El modelo de diseño está muy cercano al de implementación, lo que es natural para guardar

y mantener el modelo de diseño a través del ciclo de vida completo del *software*. En el diseño se realiza el modelado del sistema, se define y confecciona su arquitectura, para que pueda soportar todos los requerimientos: funcionales y no funcionales. El objetivo de esta fase es convertir los requisitos del *software* en especificidades que describen cómo implementar el sistema. (Jacobson, y otros, 2000)

3.7 Diagrama de clases del diseño

Los diagramas de clases del diseño son una representación más concreta y detallada que los diagramas de clases del análisis, aunque también representan la parte estática del sistema, conteniendo las clases y sus relaciones. En el diseño se modela el sistema y se encuentra su forma, incluyendo la arquitectura, para que soporte todos los requisitos y otras restricciones relacionadas con el entorno de la implementación, tiene impacto en el sistema a desarrollar. Además, el modelo de diseño sirve de abstracción de la implementación del sistema y es utilizado como entrada fundamental de las actividades de implementación. Los diagramas de clases de diseño son lo más utilizados en el modelado de sistemas orientado a objetos. (Pressman, 2005)

A continuación se presentan los diagramas de clases del diseño elaborados para el sistema que se propone:

- **Diagrama de clases del diseño Ubicar Media**

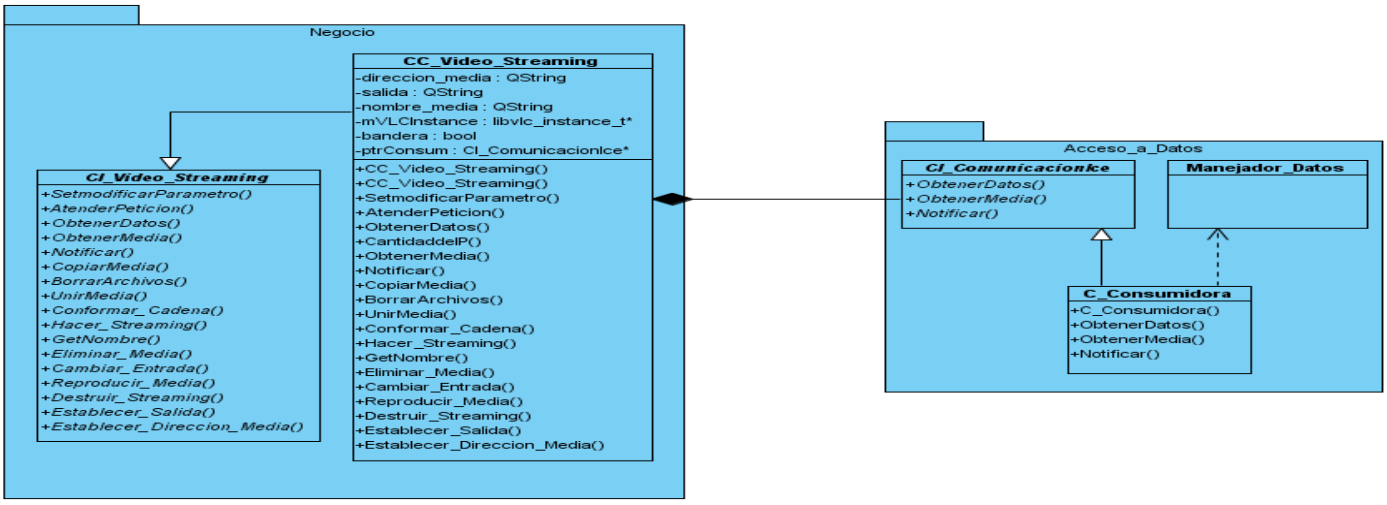

Fig. 4. Diagrama de CD Ubicar Media

- **Diagrama de clases del diseño Efectuar streaming**

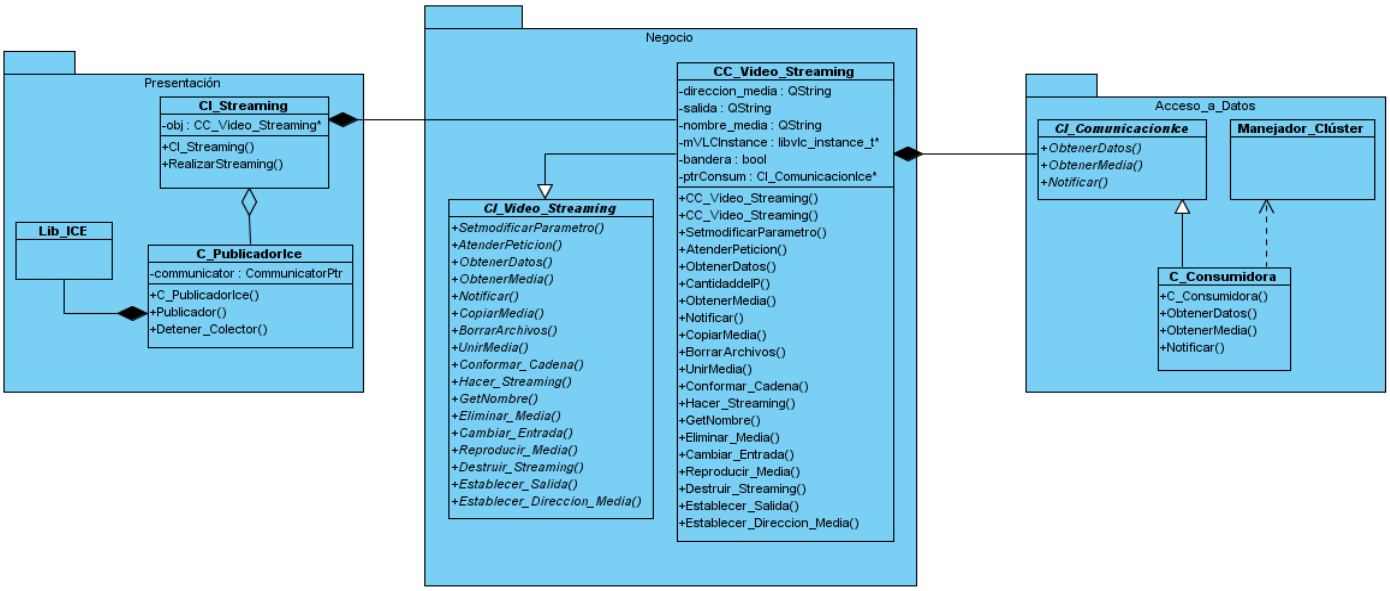

Fig. 5. Diagrama de CD Efectuar *streaming*

Para más información de los otros diagramas ver Anexo 4 y 5.

Para el diseño del componente se utilizó el estilo en tres capas. Consiste en una capa de la Presentación, otra capa de la lógica de la aplicación y otra capa del acceso a los datos. Este posee muchas ventajas, primero que nada, el estilo soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los programadores la participación de un problema complejo en una secuencia de pasos incrementales. En segundo lugar, admite optimizaciones y refinamientos. Proporciona además amplia reutilización, lo que se convierte en uno de los fuertes de esta arquitectura.

También esta arquitectura en capas se enfoca en la distribución de roles y responsabilidades de forma jerárquica proveyendo una forma muy efectiva de separación de responsabilidades. El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada.

El estilo de arquitectura basado en capas se identifica por las siguientes características:

- ✓ Describe la descomposición de servicios de forma que la mayoría de la interacción ocurre solamente entre capas vecinas.
- ✓ Las capas de una aplicación pueden residir en la misma máquina física (misma capa) o puede estar distribuido sobre diferentes computadores (n-capas).
- ✓ Los componentes de cada capa se comunican con otros componentes en otras capas a través de interfaces muy bien definidas.
- ✓ Este modelo ha sido descrito como una "pirámide invertida de re-uso" donde cada capa agrega responsabilidad y abstracción a la capa directamente sobre ella.

3.8 Patrones

Los patrones son un modelo a seguir para realizar diferentes actividades. Se ocupan de problemas recurrentes. Además, identifican y especifican abstracciones de niveles más altos que componentes o clases individuales. Por lo tanto, capturan las experiencias existentes y probadas para promover buenas prácticas ya que son a grandes rasgos una abstracción de "problema – solución". Esto hace posible un mejor entendimiento a la situación en que se encuentre algún problema que se le quiera dar solución.

3.8.1 Patrones de Diseño

Los patrones de diseño son una guía de cómo construir *software*, de cómo utilizar las clases y los objetos de forma conocida. Para la solución de este componente se tuvo en cuenta diversos patrones de diseño, los cuales son soluciones a los problemas de diseño, ya que al ser aplicados se obtiene como resultado que los diseños sean más flexibles, modulares y reutilizables.

A continuación, una breve descripción de los patrones GRASP[32], objetos utilizados en la aplicación, quienes son los que describen los principios fundamentales de la asignación de responsabilidades.

Controlador: Está diseñado para asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades. Un ejemplo de ello es la clase controladora CC_Video_Streaming que posee las funcionalidades principales de la aplicación.

[32] **GRASP : General Responsibility Assignment Software Patterns**

Alta Cohesión: Posee una alta cohesión funcional cuando los elementos de un componente (clase) colaboran para producir algún comportamiento bien definido. Es decir cuando varias clases colaboran entre sí, para dar una solución en conjunto.

Bajo Acoplamiento: Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. Este posibilita grandes ventajas ya que no afectan los cambios de otros componentes, además de ser más fáciles de entender por separado.

Experto: Este plantea que cada objeto realiza la funcionalidad de acuerdo con la información que domina, la cuestión a la hora de diseñar es asignar responsabilidades a la clase que mayor información posee para cumplir con dicha tarea.

Seguidamente una breve descripción de los Patrones GOF[33] utilizados.

Singleton: Este patrón posibilita mantener la visibilidad global o un único punto de acceso a una única instancia de una clase, en lugar de cualquier otra forma de visibilidad, haciéndolo con una referencia permanente a la misma (Larman, 2003). Este se utiliza en la creación del sistema cuando se crea una instancia de la clase C_Instance en la clase controladora, permitiendo poder tener acceso a dicha instancia en cualquier sección.

Fachada: Se utiliza para proporcionar la interfaz de una capa a la siguiente, permitiendo que solo pueda acceder a un pequeño número de funcionalidades de bajo nivel. Este patrón es utilizado cuando se crea una clase interfaz para la comunicación como es el caso de CI_Icecomunicación que se relaciona con el componente Manejador de Datos y la clase CI_Streaming que interactúa con el componente Manejador de Clúster.

Observador: Define una dependencia del tipo uno-a-muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, el observador de encarga de notificar este cambio a todos los otros

[33]**GOF: Gang of four.**

dependientes. Este patrón también se conoce como el patrón de publicación-suscripción o modelo-vista. Este patrón se utiliza en la clase C_Consumidora.

3.9 Conclusiones parciales

En este capítulo se ha tratado de manera específica los artefactos generados en el análisis correspondiente según lo planteado por RUP. Además, se abordó el proceso de construcción del Modelo de Dominio, así como su diagrama y descripción de las clases, se expuso la importancia del levantamiento de los requerimientos empleados en la construcción de un *software*, y se muestran los requisitos funcionales y no funcionales que presenta el producto, que fueron representadas mediante un Diagrama de Casos de Usos del Sistema con la descripción de todas las acciones que realizan los actores y el sistema en general. También, como aporte de la investigación realizada, se obtuvo un conjunto de artefactos en la fase de diseño, que servirán como guía para la fase de implementación del sistema. Teniendo en cuenta todas estas características se puede comenzar a implementar el sistema poniendo en práctica el cumplimiento de los requisitos tanto funcionales como no funcionales planteados en este capítulo. Se puede concluir además que el estilo arquitectónico de tres capas utilizado, posee una característica muy importante y es que las capas de una aplicación pueden residir en la misma máquina física o puede estar distribuido sobre diferentes computadoras. Por otra parte, los patrones utilizados GRASP y GOF ofrecen a los desarrolladores un vocabulario común para hablar de soluciones de software. Ellos permiten legibilidad del código ya que ayudan a escribir código más comprensible con mejores nombres para lo que se desea lograr. Además, cooperan a comunicar los objetivos de diseño entre los programadores para encontrar soluciones comunes a problemas frecuentes.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

4.1 Introducción

En este capítulo se pone en marcha el desarrollo y prueba de la solución propuesta en los capítulos anteriores. Se describe los modelos de implementación además de las pruebas realizadas. Se definen los modelos de componentes tanto de código fuente como de código ejecutable. Se describen además las pruebas realizadas al producto para verificar que todo lo antes planteado se cumpliera. Además, se tuvo muy en cuenta la escalabilidad puesto que esta es una primera versión que estará sujeta a cambios para mejorar su eficiencia y posibilitar mayor funcionalidad al usuario final.

- **Diagrama de Componentes de código fuente.**

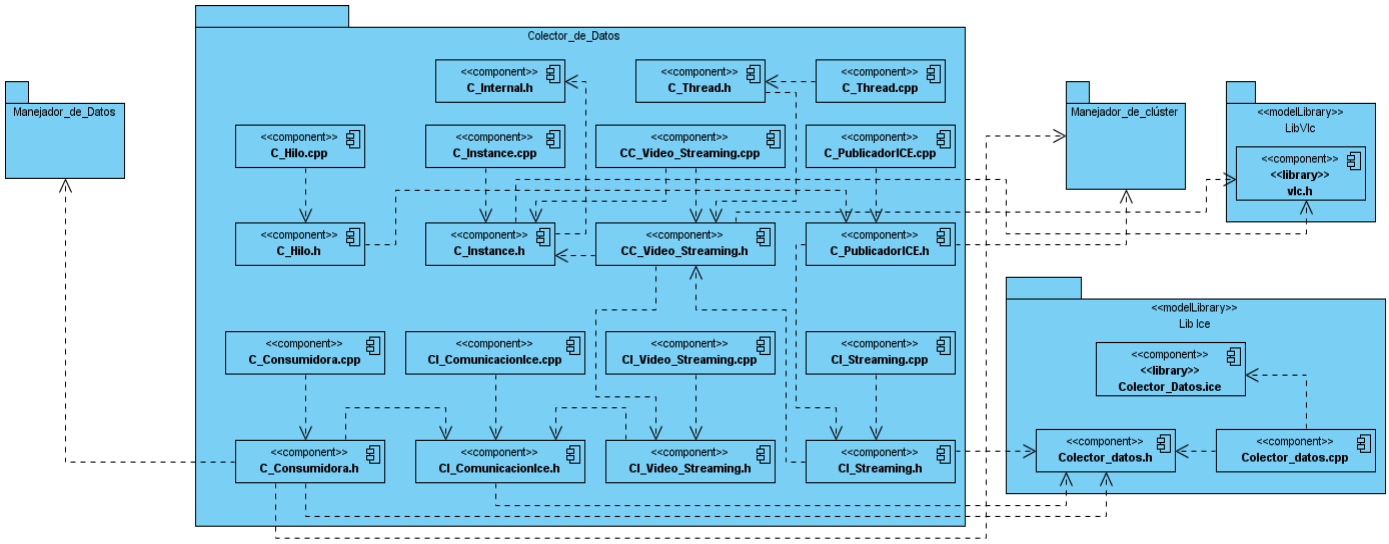

Fig. 6. Diagrama de Componentes de código fuente

CC_Video_Streaming: Es la clase principal de la aplicación y posee las funcionalidades más importantes.
C_PublicadorIce: Es la encargada de brindar algunos métodos para otros componentes consumirlos e interactuar con la aplicación.

C_Consumidora: Se encarga de la comunicación entre los componentes al consumir diferentes métodos para su posterior uso.

4.2 Pruebas al sistema

La prueba es un elemento crítico para la garantía de la calidad del *software*, es el proceso que permite verificar y revelar la calidad de este tipo de producto. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa. Básicamente es una fase en el desarrollo de *software* consistente en probar las aplicaciones construidas. Con la realización de estas pruebas se pretende encontrar y documentar los defectos que puedan afectar la calidad del sistema desarrollado y validar los requisitos que debe cumplir el mismo y a su vez verificar si estos fueron implementados correctamente.

4.2.1 Pruebas de caja blanca

Están dirigidas principalmente a las funciones internas del sistema. Para realizar estas pruebas es necesario conocer la lógica del programa. Mediante los métodos de prueba de la caja blanca, el ingeniero de *software* puede obtener casos de prueba que garanticen que:

- ✓ Ejecuten todos los bucles en sus límites y con sus límites operacionales.
- ✓ Se ejerciten las estructuras internas de datos para asegurar su validez.
- ✓ Se ejerciten por lo menos una vez todos los caminos independientes para cada módulo
- ✓ Se ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsa.

Debido a que el componente no posee una interfaz de usuario, sino una interfaz de comunicación para la interacción con los demás componentes solamente por lo que se aplicará las pruebas de caja blanca a través del camino básico. Estas pruebas se consideran entre las más importantes que se le aplican al *software*, logrando como resultado que disminuya en un gran porciento el número de errores existentes en los sistemas y por ende una mayor calidad. (Pressman, 2000)

Prueba del camino básico

Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática.

Estos casos de pruebas derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. En la figura 9 se muestra una parte del código de la clase controladora CC_Video_Sreaming que corresponde al método de Hacer_*Streaming* al cual se le realizará las pruebas de caja blanca.

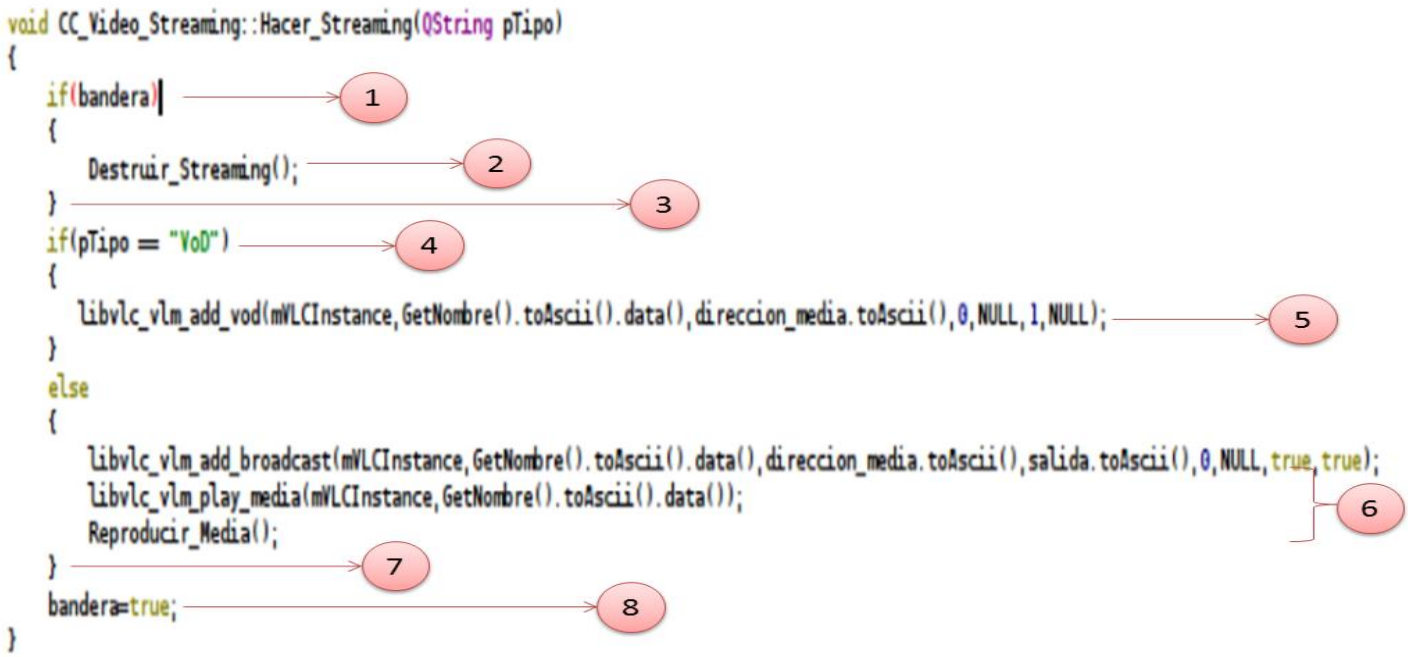

Fig. 7. Representación del código del método Hacer_*Streaming*

La figura 10 muestra el grafo de flujo representado para el código anterior:

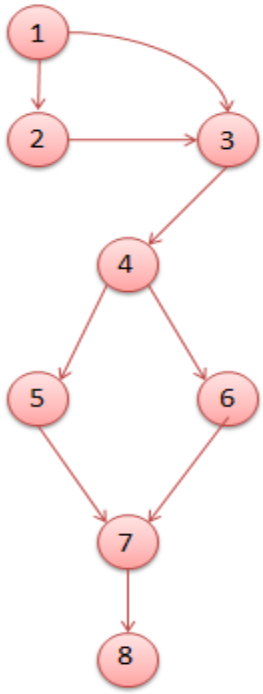

Fig.8.Grafo del flujo asociado al método Hacer_*Streaming*

Cálculo de la complejidad ciclomática:

La complejidad ciclomática es una métrica de *software* extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa. Además, da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez.

Complejidad ciclomática [V (G)] = Cantidad de Aristas [A] – Cantidad de nodos [N] + 2.

1. V (G) = A – N + 2

2. V (G) = 9 – 8 + 2
3. V (G) = 3

El cálculo realizado arrojó como resultado 3, por lo que se puede plantear que la complejidad ciclomática del código anteriormente expuesto es de 3, lo que significa que existen 4 posibles caminos por donde el flujo puede circular. En la tabla 7 se muestran los caminos básicos para posteriormente realizar los casos de pruebas.

Tabla 4. Caminos básicos del flujo.

Números	Caminos básicos
1	**1-2-3-4-5-7-8**
2	**1-3-4-5-7-8**
3	**1-2-3-4-6-7-8**
4	**1-3-4-6-7-8**

Luego de tener elaborado el Grafo de Flujos y los caminos a recorrer, se preparan los casos de prueba que forzarán la ejecución de cada uno de esos caminos. Para realizar los casos de pruebas es necesario tener en cuenta los siguientes aspectos.

- ✓ Descripción: Se realiza la entrada de datos necesaria para validar que los parámetros obligatorios no pasen nulos al método.
- ✓ Entrada: Se exponen los parámetros que son necesarios para realizar el método.
- ✓ Resultados Esperados: Se define el resultado esperado que debe devolver el método.

A continuación, en la tabla 8 se muestra el caso de prueba del primer camino básico obtenido.

Tabla 5. Caso de prueba para el camino básico número 1.

Caso de prueba para el camino básico número 1	
Camino 1	1-2-3-4-5-7-8
Descripción	Los datos de entrada cumplirán con los siguientes requisitos: Está conformada la cadena a transmitir, con el protocolo, dirección_ip y punto de montaje. Por último, es enviado por parámetro el tipo de transmisión.
Entrada	bandera = true salida = rtsp://10.34.13.222:5050/nikita101 direccion_media = /home/leo/Escritorio/Medias nombre_media = nikita101 pTipo = VoD
Resultados esperados	Se efectúa el proceso de *streaming* VoD para que el cliente pueda visualizarlo. Además, se podrán realizar diferentes acciones como pausar, detener, adelantar y atrasar la media.

Seguidamente, en la tabla 11 se muestra el caso de prueba del cuarto camino básico obtenido.

Tabla 6. Caso de prueba para el camino básico número 4.

Caso de prueba para el camino básico número 4

Camino 4	1-3-4-6-7-8
Descripción	Los datos de entrada cumplirán con los siguientes requisitos: Está conformada la cadena a transmitir, con el protocolo, dirección_ip y punto de montaje. Por último, es enviado por parámetro el tipo de transmisión.
Entrada	bandera = false salida = rtp://239.1.1.6:2350/heroes305 direccion_media = /home/leo/Escritorio/Medias nombre_media = heroes305 pTipo = Vivo
Resultados esperados	Se realiza el proceso de *streaming* en vivo para que el cliente pueda visualizarlo en tiempo real.

Los otros casos de prueba se pueden ver en los Anexos 6 y 7. Al realizar las pruebas de caja blanca a través de los diferentes caminos encontrados, se alcanzaron resultados satisfactorios ya que se obtuvieron los flujos de streaming deseados y se visualizaron en las PC clientes.

4.3 Conclusiones parciales

En el presente capítulo se presenta como quedó el sistema expresado en componentes de implementación. Se diseñó el diagrama de componentes de código ejecutable el cual brinda una visión general del código a través de las relaciones de las clases, así como del uso de librerías que facilitaron el desarrollo del componente. Además, se realizaron las pruebas de caja blanca tomando una parte de lo implementado para demostrar que la aplicación cumple con las funcionalidades previstas al inicio y arrojaron resultados satisfactorios.

CONCLUSIONES GENERALES

En la presente investigación se profundizó en los conocimientos sobre los procesos de *streaming* desde su surgimiento hasta la actualidad, procedentes de diversas fuentes en diferentes publicaciones digitales logrando organizarlos en función de la fundamentación del objetivo propuesto. Se cumplieron los objetivos trazados así como se le dio cumplimiento a las tareas propuestas posibilitando un buen desarrollo del componente.

Se logró desarrollar el componente Colector de Datos con tecnologías libres, con la característica de ser eficaz por la utilización de un lenguaje de programación cercano al *hardware* y de técnicas de programación que posibilitaron una mejor distribución de los recursos. Dicha módulo brinda al servidor *streaming* distribuido la posibilidad de hacer *streaming* completamente, disminuyendo el consumo de ancho de banda de los clientes y la concurrencia. Su función principal es de hacer *streaming,* lo realiza por diferentes tipos de transmisiones como son en vivo y bajo demanda, distribuidos a través de los protocolos más seguros en distribución de contenido multimedia por la red como son UDP, RTP y RTSP. Las pruebas realizadas a la aplicación con el objetivo de validar los requisitos funcionales determinados para su desarrollo, arrojaron resultados favorables, y reveló un sistema sólido y eficiente.

Los resultados de este trabajo posibilitan la factibilidad tecnológica de la utilización del Colector de Datos en el departamento de Señales Digitales de la UCI.

RECOMENDACIONES

Para mejorar el componente se recomienda utilizar otras librerías que faciliten la confección del *streaming* como es el caso de libgstreamer la cual posee diferentes funcionalidades que se podrían utilizar para brindar una mayor cantidad de servicios a los clientes. Por otra parte, lograr un manejo eficaz de la aplicación desarrollada se recomienda la elaboración de un manual de ayuda para el usuario, así como capacitar al personal para la utilización del *software*.

BIBLIOGRAFIA

Aguilar, Luis Joyanes. 2003.*Fundamentos de programación.Algoritmos,estructuras.* España : s.n., 2003. ISBN:8448139860.
Alex Mashinsky. 2007. Digimeld. [En línea] 2007. [Citado el: 2 de Diciembre de 2010.] http://www.digimeld.com/.
Alvarez, Miguel Angel. 2003. Desarrollo Web. [En línea] Julio de 2003. [Citado el: 12 de Diciembre de 2010.] http://www.desarrolloweb.com/articulos/482.php.
Ángel Ríos Morales. 2007. Conocimientos web. [En línea] Agosto de 2007. [Citado el: 24 de Noviembre de 2010.] http://www.conocimientosweb.net/zip/article986.html.
Calderón, Jorge Carlos Valverde Rebaza y Sarah Dámaris Amaro. 2007.*Metodologías Ágiles.* Trujillo : s.n., 2007.
Cleto Martín Angelina. 2010.*PROYECTO FIN DE CARRERA.* Real : s.n., 2010.
Denis Derivet Thaureaux. 2010. Ecured. [En línea] 2010. [Citado el: 27 de Noviembre de 2010.] http://www.ecured.cu/index.php/Televisi%C3%B3n_en_Cuba..
Eclipse, Fundation. Eclipse. [En línea] [Citado el: 26 de Noviembre de 2010.] http://www.eclipse.org.
Grupo de Soluciones Innova. 2007. Rational Rose Enterprise. [En línea] 2007. [Citado el: 30 de Noviembre de 2010.] http://www.rational.com.ar/herramientas/roseenterprise.html.
Ignacio Marín, David Melendi. 2009.*Servicios de Audio/Video.* Asturias : s.n., 2009.
Ivar Jacobson, Grady Booch y James Rumbaugh. 2000.*El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Educacion, 2000. 84-7829-036-2.
Javier A. Bedrina. 2002.*IWorld.* [En línea] 2002. [Citado el: 5 de Marzo de 2011.] http://www.idg.es/iworld/articulo.asp?id=132333.
Javier García de Jalón, José Iganacion Rodríguez. 2002.*Aprenda Java como si estuviera en primero.* San Sebastián : s.n., 2002.
—. 1998.*Aprendiendo c++ como si estuviera en primero.* San Sebastián : s.n., 1998.
Jesus Antonio Castro. 1999. Departamento de Sistemas y Computación. [En línea] 1999. [Citado el: 2 de Diciembre de 2010.] http://sistemas.itlp.edu.mx/tutoriales/sistsdist1/u1parte2.htm.
José Ramón Pérez Agüera, Rodrigo Sánchez Jiménez, Jorge Caldera Serrano. 2004.*Adaptación de tecnologías Stream y XML a centros de documentación en televisión.* Madrid : s.n., 2004.
Juan Carlos. 2010.*Programática.* [En línea] 2010. [Citado el: 18 de Febrero de 2011.] http://programatica.blogspot.com/2010/07/servidor-streaming-con-gnump3d-en.html.
Juan Pablo Quintero Ortiz, Cristian Andrey Castro Serna. 2006.*Evaluacion de servidores de streaming de video orientado a dispositivos moviles.* Medellin : s.n., 2006.
Larman, Craig. *UML y Patrones.*
—. 1999. *UML y Patrones Introducción al análisis y diseño orientado a objetos.* Mexico : s.n., 1999.
Mauricio Leonardo Hernán Venegas Morales, Aquiles Alfredo Yañez Cañas. 2005.*Transmisión de video de alta calidad a traves de redes IP.* s.l. : Valparaíso, 2005.

Microsoft Corporation. 2000. Visual Studio Developer Center. [En línea] 2000. [Citado el: 26 de Noviembre de 2010.] http://msdn.microsoft.com/en-us/library/aa287558.aspx.
Nestor Díaz. 2005. CASADOMO. [En línea] 2005. [Citado el: 25 de Octubre de 2010.] http://casadomo.com/noticiasDetalle.aspx?c=6&id=7160..
Netbeans.org. 2000. Netbeans. [En línea] 2000. [Citado el: 27 de Noviembre de 2010.] http://netbeans.org/index_es.html.
Pascal Pegaz-Paquet, Julien Moutte, Thomas Vander Stichele, Jean-Noel Saunier. 2006. Flumotion Fundation. [En línea] 2006. [Citado el: 3 de Diciembre de 2010.] http://www.flumotion.com.
Pierina Flores Pastor. 2007. SlideShare. [En línea] 2007. [Citado el: 15 de Noviembre de 2010.] http://www.slideshare.net/picitapastor/concepto-de-streamingwebcasting-y-cms.
Pressman, Roger S. 2005.*Ingeniería del Software.* s.l. : McGraw-Hill, 2005. 970-10-5473-3.
Python *Software* Fundation. 1990. Python Programming Language. [En línea] 1990. [Citado el: 27 de Noviembre de 2010.] http://www.python.org/.
Redacción VSAntivirus. 2003. VSantivirus. [En línea] 2003. [Citado el: 25 de Febrero de 2011.] http://www.vsantivirus.com/vul-dss-streaming.htm.
Richard Mueller. 2005. Microsoft Technet. [En línea] 27 de Abril de 2005. [Citado el: 20 de Noviembre de 2010.] http://technet.microsoft.com/en-us/library/bb497060.aspx.
Sciara, Daniel Rijo. 2004.*Fundamentos de Video Streaming.* Montevideo : s.n., 2004.
Sommerville, Ian. 2005.*Ingenieria del Software.* Madrid : Pearson Educacion S.A., 2005. 84-7829-074-5.
Torres, Oscar Daniel. 2009.*Evolucion y tendencia de la tecnología de streaming en internet.* Buenos Aires : s.n., 2009.
Tusch, Roland. 2004.*Arquitectura del servidor de streaming basado en la orientación a los servicios de componentes.* Austria : s.n., 2004.
Universidad Católica de Loja. 2008. [En línea] 2008. [Citado el: 9 de Enero de 2011.] http://www.utpl.edu.ec/eva/descargas/material/175/G18401.8.pdf.
Visual Paradigm International. 2004. Visual Paradigm. [En línea] 2004. [Citado el: 30 de Noviembre de 2010.] http://www.visual-paradigm.com/.
Willy Klew. 2008. Visual Beta. [En línea] 16 de Septiembre de 2008. [Citado el: 28 de Noviembre de 2010.] http://www.visualbeta.es/5640/aplicaciones-web/digimeld-video-streaming-distribuido-en-calidad-hd/.

GLOSARIO

Códec: Dispositivo o programa capaz de codificar y descodificar en un signo o una corriente de datos digitales.

Plugin: Es un módulo de *hardware* o *software* que añade una característica o un servicio específico a un sistema más grande.

Media: Película, imagen o cualquier otro material audio visual que requiere de un uso especial de equipamiento para visualizarlo.

Video: Es la tecnología de la captación, grabación, procesamiento, almacenamiento, transmisión y reconstrucción por medios electrónicos digitales o analógicos de una secuencia de imágenes que representan escenas en movimiento.

Multimedia: El término multimedia se utiliza para referirse a cualquier objeto o sistema que utiliza múltiples medios de expresión (físicos o digitales) para presentar o comunicar información.

Multiplataforma: Se utiliza el término para denominar a los programas, lenguajes de programación u otra clase de *software* que pueden brindar sus prestaciones funcionando sobre diversas combinaciones de *hardware* y *software*.

Servidor: *Software* u ordenador que provee servicios a otros programas o equipos denominados clientes.

***Streaming*:** Consiste en la distribución de audio o video por Internet.

Aplicación: Es una clase de programa informático creado para facilitar al usuario un determinado tipo de trabajo. Esto lo caracteriza frente a otros programas como los sistemas operativos, las utilidades y los lenguajes de programación.

Audio: Señal, técnica o sistema electrónico relacionado con la grabación, reproducción y transmisión del sonido que normalmente se encuentra acotado al rango de frecuencias audibles para los seres humanos.