DEPARTAMENT D'INFORMÀTICA

VISUALIZACIÓN INTERACTIVA DE TERRENOS EXTENSOS EN ENTORNOS DISTRIBUIDOS SOBRE REDES MIXTAS P2P Y CLIENTE-SERVIDOR.

RICARDO OLANDA RODRÍGUEZ

UNIVERSITAT DE VALÈNCIA
Servei de Publicacions
2010

Aquesta Tesi Doctoral va ser presentada a València el dia 30 d'abril de 2010 davant un tribunal format per:

- Dr. Roberto Vivó Hernando
- Dr. Federico Silla Jiménez
- Dr. Luis Matey Muñoz
- Dr. César Otero González
- Dr. Juan Manuel Orduña Huertas

Va ser dirigida per:
Dr. Mariano Pérez Martínez
Dr. Marcos Fernández Marín

Dipòsit legal: V-2054-2011
I.S.B.N.: 978-84-370-7914-1
Edita: Universitat de València
Servei de Publicacions
C/ Arts Gràfiques, 13 baix
46010 València
Spain
Telèfon:(0034)963864115

VNIVERSITAT DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria

Departament d'Informàtica

TESIS DOCTORAL

VISUALIZACIÓN INTERACTIVA DE TERRENOS EXTENSOS EN ENTORNOS DISTRIBUIDOS SOBRE REDES MIXTAS P2P Y CLIENTE-SERVIDOR

Presentado por:

D. Ricardo Olanda Rodríguez

Valencia, marzo de 2010

Trabajo dirigido por:

Dr. D. Mariano Pérez Martínez

Dr. D. Marcos Fernández Marín

A Verónica y a el/la peque …

Agradecimientos

Quisiera agradecer el apoyo desinteresado recibido de todas las personas con las que he compartido estos años en el Instituto de Robótica (o IRTIC) y en el Departamento de Informática, sin cuya ayuda no podría haber llegado a este momento.

Al Dr. Mariano Pérez, por su disponibilidad a tiempo completo, sus consejos y correcciones sin los cuales no podría haber llevado a término esta tesis.

Al Dr. Marcos Fernández, que ha completado la labor de tutela de la tesis aportando su experiencia, conocimientos y transmitiéndome su confianza.

A todos los compañeros del Instituto que me han permitido sobrellevar el duro camino recorrido, en especial a Alejandro, Bibi, Lucía, Sento, Ximo, Silvia y al resto de compañeros de tupper que me han hecho desconectar un rato todos los días, a Ignacio por todos sus consejos pasados, presentes y futuros, a Toñi por su apoyo logístico y reprográfico, a los compañeros del metro, a los del futbito de los viernes y a los de las quinielas y euromillones.

A mi familia, por su apoyo y ánimos constantes.

A Verónica por estar siempre ahí, y a el/la peque por estar de camino.

Resumen

En los últimos años se ha producido una gran demanda de aplicaciones dedicadas a la visualización interactiva de terrenos en entornos distribuidos. Esta demanda ha estado motivada, entre otros motivos, por el aumento de la cantidad y precisión de la información de las bases de datos del terreno y el aumento de las líneas de conexión de ancha banda y la facilidad de acceso a las mismas.

Este tipo de aplicaciones tienen que hacer frente a varios problemas. Uno de ellos es la gestión de la ingente cantidad de información que contienen las bases de datos del terreno que emplean estas aplicaciones, la cual no puede ser representada simultáneamente en pantalla con una velocidad de refresco adecuada, ya que sobrepasa las capacidades de memoria y procesamiento de los equipos actuales. Otro problema es la transmisión de esa información desde los servidores a los equipos utilizados por los usuarios con la suficiente rapidez para que éstos puedan realizar una visualización interactiva en tiempo real de la misma. Otro de los problemas principales es la escalabilidad del sistema con el número de usuarios. A este tipo de aplicaciones se conectan usualmente un elevado número de usuarios; incluso es previsible que a medida que las bases de datos sean más precisas y las aplicaciones más completas, este número de usuarios siga aumentando. A todos ellos este tipo de aplicaciones tendrá que ofrecer una calidad de servicio adecuada.

A lo largo de esta tesis se ha diseñado un sistema de visualización interactiva de terrenos eficiente para entornos distribuidos que hace frente y soluciona estos problemas. Para ello se lleva a cabo un estudio de los problemas que presenta y sus posibles soluciones, escogiendo para la implementación de cada uno de los módulos en los que se divide el sistema, la solución que se ha considerado más adecuada. Así, para solucionar el problema asociado a la representación y visualización de la información del terreno, se ha encontrado un algoritmo que realiza de forma eficiente esta visualización, en un entorno con bases de datos distribuidas. Para solucionar el problema asociado a la transmisión y almacenamiento de la información del terreno, se ha diseñado un nuevo esquema de compresión basado en wavelets que realiza una compresión óptima de esta información permitiendo realizar la transmisión progresiva de la misma de forma óptima a través de la red. Por último, para solucionar el problema de la escalabilidad con el aumento del número de usuarios conectados, se ha definido una arquitectura híbrida basada en el funcionamiento de las arquitecturas cliente-servidor y P2P, que incorpora las ventajas de ambas arquitecturas y evita sus inconvenientes, aumentando considerablemente la escalabilidad del sistema. Todos estos elementos se han integrado perfectamente para crear un sistema de visualización interactiva de terrenos eficiente para entornos distribuidos.

Índice

Índice de Tablas

Índice de Figuras

Índice de Gráficas

Índice de Ecuaciones

Capítulo 1 – Introducción

En los últimos años, la visualización de terrenos en tiempo real ha sido un campo de investigación muy activo dentro del área de gráficos por ordenador. Las aplicaciones de este campo son muy diversas, desde la incorporación de efectos especiales en películas, a la elaboración de simuladores civiles y militares de vuelo o de conducción de vehículos, pasando por aplicaciones de cartografía, elaboración de entornos virtuales o videojuegos de toda clase de temática.

Una de las principales tareas de estas aplicaciones es la visualización de modelos virtuales de terreno realistas a velocidades de refresco interactivas. Estos modelos virtuales se almacenan normalmente en extensas bases de datos ubicadas en el sistema de visualización.

Más recientemente, el aumento de las capacidades de las líneas de conexión de banda ancha y la facilidad de acceso a las mismas, ha hecho posible que estas bases de datos no deban restringirse exclusivamente a un uso local, permitiendo el acceso remoto a las mismas a través de la red. Esta situación ha contribuido a la aparición de un conjunto de aplicaciones de visualización de terrenos en tiempo real que realizan un acceso distribuido a estas bases de datos. Como ejemplos podemos encontrar aplicaciones de visualización de mapas como Google Maps [50] o Virtual Earth [135] y aplicaciones de vuelos virtuales sobre entornos 3D como Google Earth [49] o Virtual Earth 3D [135].

1.1– Contexto y Motivación

Los avances en los dispositivos de adquisición de los datos de terrenos han provocado un aumento en la cantidad y en la precisión de la información almacenada en las bases de datos de terrenos que emplean las aplicaciones comentadas anteriormente. De modo que, a pesar de los avances tecnológicos en los ordenadores y las tarjetas gráficas actuales, todavía no es posible visualizar toda esa información sin llevar a cabo ningún tipo de gestión o adaptación de la misma.

Por lo tanto, uno de los principales problemas a los que se enfrentan estas aplicaciones es la gestión de la ingente cantidad de información que contienen esas bases de datos, la cual no puede ser representada simultáneamente en pantalla con una velocidad de refresco adecuada, puesto que sobrepasaría las capacidades de memoria y procesamiento de los equipos actuales.

Aparte, las aplicaciones que emplean bases de datos distribuidas tienen que hacer frente, además, al problema de la transmisión de la información desde el servidor de las bases de datos a los equipos de los usuarios**.** A pesar del aumento de la capacidad de las líneas de conexión, los datos a transmitir pueden llegar a colapsar estas conexiones y provocar que el usuario no obtuviera la información necesaria con la rapidez adecuada, con lo que éste no conseguiría llevar a cabo una visualización interactiva en tiempo real. Por eso, es necesario emplear el mecanismo de compresión y transmisión a través de la red más cercano a la solución óptima para toda esa información.

Este tipo de aplicaciones cuentan actualmente con un gran número de usuarios (a día de hoy, el programa de visualización de terrenos Google Earth ha superado los 400 millones de descargas) y se espera que a medida que las bases de datos sean más precisas y las aplicaciones más completas, el número de usuarios siga aumentando. Esto representará un problema importante si se desea ofrecer una calidad de servicio aceptable a todos estos usuarios.

La arquitectura de red típica empleada para este tipo de aplicaciones es la arquitectura cliente-servidor, que es la arquitectura estándar para este tipo de aplicaciones debido, principalmente, a la facilidad que aporta para la gestión y transmisión de la información. Sin embargo, esta arquitectura dispone de una escalabilidad limitada. Conforme el número de usuarios se incrementa, el número de recursos necesarios en el lado del servidor también aumenta considerablemente, lo que provoca un incremento en el coste del sistema para evitar que disminuya su rendimiento.

Actualmente están surgiendo alternativas a esta arquitectura, como las arquitecturas P2P, que son inherentemente escalables con el número de usuarios y permiten minimizar este problema. Sin embargo, estas arquitecturas tienes asociados otros problemas, como una mayor complejidad en la gestión y transmisión de la información, o la necesidad de que exista siempre un número mínimo de usuarios conectados a estas redes que dispongan de la información a compartir.

Esta necesidad, en el caso de una aplicación P2P de terrenos, no es un requisito fácil de cumplir debido a que la información que tienen que compartir los usuarios no estará formada por un reducido conjunto de datos idéntico para todos ellos, como ocurre en una aplicación P2P tradicional de descarga de ficheros, sino que cada usuario va a requerir un conjunto de datos diferente seleccionado de una extensa base de datos, lo que limitará la cantidad de información que pueden intercambiar entre sí y complicará la obtención de esa información dentro de los parámetros de tiempos exigidos por un sistema de visualización en tiempo real.

1.2 – Objetivos

El objetivo principal de esta tesis es el diseño de un sistema de visualización interactiva de terrenos eficiente para entornos distribuidos. Para ello, se intentarán resolver los problemas enunciados en el apartado anterior, así como otros que puedan surgir a lo largo del desarrollo del sistema, adaptando y mejorando aquellas partes que se puedan reutilizar para nuestro sistema de las soluciones actuales diseñadas para entornos locales.

Para solucionar el problema asociado a la representación y visualización de la información de las bases de datos de terrenos, se tratará de encontrar un algoritmo, de entre los algoritmos que se emplean habitualmente en la visualización de terrenos en tiempo real con bases de datos de terrenos locales, que se adapte de forme eficiente a un sistema de visualización con bases de datos distribuidas.

Después, para solucionar el problema asociado a la transmisión y almacenamiento de la información de terrenos, se intentará encontrar, de entre los diferentes esquemas de compresión y transmisión existentes, aquellos que se adapten mejor a nuestro sistema, prestando especial atención a los esquemas basados en wavelets, como el usado por el estándar Jpeg2000, que a priori pueden ser adecuados debido a sus propiedades innatas de multirresolución.

Para el último de los problemas enunciados, el asociado a la escalabilidad con el aumento del número de usuarios, se intentará definir e implementar una arquitectura híbrida basada en las arquitecturas cliente-servidor y P2P, con la que se pretende obtener un funcionamiento eficiente tanto con un número bajo de usuarios, como con un número elevado de los mismos. Con esta arquitectura se tratará de resolver tanto el problema de escalabilidad de las arquitecturas cliente-servidor como el problema de la necesidad de un número mínimo de usuarios conectados al sistema de la arquitectura P2P, intentando conseguir que la obtención de la información del terreno por parte de un usuario pueda realizarse dentro de los parámetros de tiempo que exige un sistema de visualización en tiempo real.

Por lo tanto, como resumen, establecemos los siguientes objetivos:

- Obtención de algoritmos de representación y visualización de terrenos en tiempo real adecuados para bases de datos distribuidas.
- Obtención de esquemas de compresión y transmisión más cercanos a la solución óptima para bases de datos de terrenos distribuidas.

- Obtención de una arquitectura de red apropiada para la visualización de terrenos en tiempo real eficiente, que funcione correctamente independientemente del número de usuarios conectados al sistema.
- Diseño de un sistema de visualización interactiva de terrenos extensos en entornos distribuidos que incorpore los mecanismos adecuados extraídos de la consecución de los objetivos anteriores.

1.3 – Metodología

Durante el diseño e implementación del sistema será necesario resolver los problemas enunciados en el apartado anterior.

Para la resolución del problema asociado a la representación y visualización de la información del terreno, primeramente se realizará un estudio de los principales algoritmos existentes en la bibliografía que nos permitirá conocer las propiedades de cada uno y hacer una primera selección de aquellos cuyas características puedan adaptarse mejor a nuestro sistema. Una vez seleccionados estos algoritmos, se llevará a cabo la implementación de los mismos para evaluar su rendimiento dentro del sistema. Para realizar esta evaluación, se modelarán un conjunto de pruebas que permitan comparar tanto la calidad visual de los resultados, medida a través de una métrica del error definida en el espacio de la pantalla, como la velocidad de refresco de los datos en pantalla que son capaces de alcanzar cada uno de estos algoritmos. Tras el análisis de los resultados de las pruebas se seleccionará para nuestro sistema aquel algoritmo que haya obtenido unas mejores prestaciones.

Para la resolución del problema asociado a la compresión y transmisión de la información de las bases de datos de terrenos, se realizará primeramente un estudio de las estructuras de organización existentes para este tipo de información que se emplean habitualmente en las aplicaciones de visualización de terrenos. A continuación, se llevará a cabo un estudio de los mecanismos empleados para la codificación y transmisión de información de terrenos que, además, soporten este tipo de estructuras de organización. Para poder evaluar estos mecanismos, se realizará una implementación de aquellos que mejor se adapten al sistema, tratando de evitar o minimizar los problemas encontrados durante su estudio. Para cada uno de ellos, se llevará a cabo una serie de pruebas de compresión, descompresión y transmisión de imágenes, donde se evaluará tanto la velocidad de transmisión como la calidad visual de las imágenes descomprimidas, empleando una métrica del error definida en el espacio de la pantalla. Finalmente, en función de los resultados de esas pruebas, se procederá a seleccionar aquellas técnicas que mejores prestaciones consigan.

Por último, para conseguir solucionar el problema relacionado con la escalabilidad con el número de usuarios y obtener, de esa forma, un sistema que funcione eficientemente tanto con un número reducido de usuarios como con un número elevado de los mismos cumpliendo las restricciones temporales exigidas a un sistema de tiempo real, se evaluarán diferentes arquitecturas de red como las arquitecturas cliente-servidor y las arquitecturas P2P y, además, se experimentará con una arquitectura híbrida entre las dos anteriores. Para realizar la evaluación de estas arquitecturas se diseñarán un conjunto de pruebas que emplearán un número variable de servidores y clientes y de las que se podrán extraer resultados de latencias y tiempos de transferencia de la información de terrenos, seleccionando, a partir de los resultados obtenidos, la arquitectura que mejor se adapte a un sistema de visualización de terrenos distribuido en tiempo real.

A la hora de evaluar el sistema nos encontramos con que, debido al elevado coste y a la complejidad que reside en la realización y evaluación de pruebas en las que tomen parte un número elevado de usuarios en un entorno real, no sólo será necesario la implementación de un sistema real, sino que además será necesario implementar un simulador que permita probar la eficiencia del sistema para estos casos. Para validar el correcto funcionamiento de este simulador, se diseñarán un conjunto de pruebas que se repetirán tanto en el simulador real, dentro de un entorno de laboratorio y con un número reducido de usuarios, como en el simulador, empleando las mismas condiciones de partida. Los resultados obtenidos permitirán validar el simulador si su comportamiento se adapta al del sistema real cuando se emplea un número reducido de clientes. Tras la validación del simulador, se llevarán a cabo con él un conjunto de pruebas emulando el comportamiento del sistema con un número elevado de clientes, y con los resultados obtenidos, se podrá estimar cuál será el funcionamiento esperado del sistema en la realidad.

1.4 – Estructura de la Tesis

La estructura de esta tesis está organizada en 7 capítulos, incluida esta introducción:

En el Capítulo 1 se describe el estado actual de todos los elementos que forman parte de un sistema de visualización interactiva en entornos distribuidos: los algoritmos de representación y visualización de los datos del terreno, los métodos para la organización jerárquica de los datos, los esquemas de compresión y transmisión de la información del terreno, y por último las arquitecturas de red empleadas, cliente-servidor y P2P.

En el Capítulo 3 se realiza un estudio comparativo de los algoritmos de representación y visualización de terrenos utilizados habitualmente, con el objetivo de averiguar cuál de ellos es el que realiza un acceso más cercano a la solución óptima para bases de datos distribuidas.

En el Capítulo 4 se realiza el diseño de un nuevo esquema de compresión basado en el estándar Jpeg2000 para realizar la compresión óptima de los datos del terreno. Además, se realiza una comparativa de este nuevo esquema con el esquema estándar Jpeg2000 con el objetivo de averiguar cuál de estos esquemas permite realizar la transmisión de la información del terreno de forma más eficiente, y consigue reducir más el tamaño de la misma.

En el Capítulo 5 se realiza el diseño de la nueva arquitectura de red mixta cliente-servidor / P2P. Además, se realiza una comparativa de la misma con la arquitectura clásica cliente-servidor para averiguar cuál de ellas proporciona una mayor escalabilidad con el número de usuarios conectados.

En el Capítulo 6 se realiza el diseño de un simulador para emular el sistema de visualización implementado en el Capítulo 5. Primeramente, se realiza la validación del funcionamiento del simulador respecto al sistema real. Posteriormente, se realizan pruebas con él para averiguar el comportamiento del mismo ante diferentes condiciones de funcionamiento.

En el Capítulo 7 se lleva a cabo un resumen con las conclusiones más relevantes del presente trabajo, las aportaciones realizadas y el trabajo futuro.

Capítulo 2 – Estado del Arte

En este capítulo se va a realizar un estudio de las diferentes soluciones existentes para cada uno de los problemas a los que hay que enfrentarse a la hora de diseñar un sistema de visualización de terrenos en tiempo real en entornos distribuidos.

Primeramente se mostrarán las técnicas empleadas para la representación de superficies de terreno en tiempo real, después se estudiará qué esquemas de organización, compresión y transmisión son los más adecuados para este tipo de información y, por último, se hará una revisión de las arquitecturas empleadas en este tipo de sistemas.

2.1 – Representación de Terrenos

2.1.1 – Introducción

La representación de terrenos en tiempo real es un área de trabajo que ha estado muy activa estos últimos años debido a su amplio campo de aplicación. Se pueden encontrar desde simuladores de vuelo para el entrenamiento de pilotos, hasta representaciones virtuales de entornos paisajísticos con un objetivo educativo, pasando por todo tipo de videojuegos. En estas aplicaciones es necesario obtener el máximo realismo posible, manteniendo unos valores altos de velocidad de refresco que permitan al usuario interactuar en tiempo real con la escena.

El aumento en la precisión y en el tamaño de las bases de datos de información del terreno supone que, a pesar de los avances tecnológicos que nos ofrecen ordenadores y tarjetas gráficas más potentes, éstos no sean suficientes para permitirnos representar toda esa información de forma simultánea en pantalla, puesto que sobrepasarían las capacidades de memoria y procesamiento de los mismos. Por lo que será necesario realizar algún tipo de gestión de toda esa información.

2.1.2 – Aplicaciones

Como se ha comentado, la representación de terrenos posee un amplio campo de aplicación. Uno de ellos es la visualización interactiva de terrenos a través de la red, donde encontramos aplicaciones como Google Earth [49], desarrollada por Google, o Virtual Earth 3D [135], desarrollada por Microsoft. Estas aplicaciones combinan información geográfica de alturas e imágenes reales tomadas por satélite, imágenes sintéticas de mapas y modelos 3D (edificios, carreteras, árboles, etc.) para mostrar una representación virtual de un lugar específico del planeta (Figura 2.1).

Las bases de datos empleadas por estas aplicaciones son muy extensas, del orden de cientos de terabytes, sin embargo, estas aplicaciones son capaces de representarlos con una gran calidad y una adecuada velocidad de refresco. Generalmente los datos empleados en estas aplicaciones son obtenidos a través de satélites, concretamente, Google Earth emplea datos de la NASA obtenidos de la "Shuttle Radar Topography Mission" (SRTM)[119], si bien estos datos contienen algunos errores y huecos que deben ser completados con otras fuentes de datos.

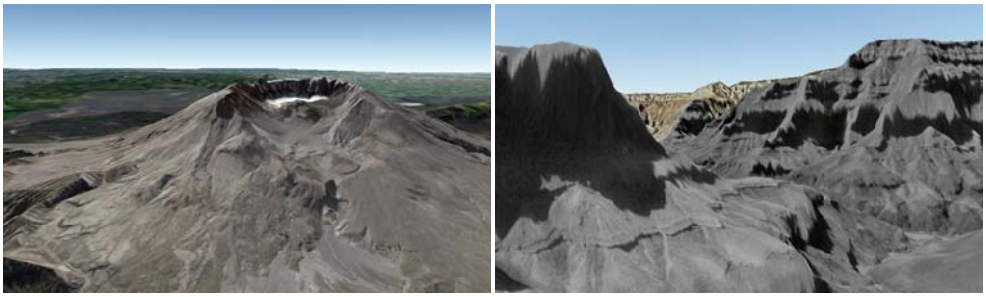


Figura 2.1 – Representaciones de superficies del terreno empleando Google Maps (Izquierda) y Virtual Earth 3D (derecha).

La resolución de los datos de texturas que se puede encontrar habitualmente navegando en Google Earth es de 15 metros/pixel, aunque cada vez existen más zonas disponibles con resolución de 1 metro/pixel, e incluso se está comenzando a introducir en las bases de datos regiones en las cuales la información de texturas tiene una resolución de 50 e incluso de 15 centímetros/pixel. En Virtual Earth 3D, en cambio, la resolución máxima que podemos encontrar para la información de texturas es de 4.5 metros/pixel, siendo la resolución media bastante inferior.

En cuanto a la resolución de las mallas de alturas, ésta es bastante inferior comparada con la de las texturas. En Google Earth, la mayor parte de la información de alturas de la superficie de terreno dispone de una resolución de 90 metros, sin embargo, se pueden

encontrar algunas zonas con 30 metros de resolución y, excepcionalmente, existen zonas con 3 metros de resolución, como es el caso del Monte Helena.

Estas dos aplicaciones incorporan algunas funcionalidades adicionales; por ejemplo, Virtual Earth 3D dispone de "Bird's Eye View" que ofrece fotografías aéreas de alta resolución de algunas ciudades y zonas de interés turístico, tomadas desde cuatro puntos de vista diferentes. Estas fotografías se superponen a la superficie del terreno proporcionando una visualización más precisa y realista. De forma parecida, Google Earth incorpora "Street View" que ofrece fotografías panorámicas de gran calidad de las calles de varias ciudades, permitiendo un paseo virtual realista a pie por las diferentes calles de estas ciudades (Figura 2.2).

Otra de las funcionalidades destacadas que poseen estas aplicaciones es la de que el usuario pueda subir información propia a las bases de datos. Por ejemplo, Google Earth permite que los usuarios incorporen modelos 3D de edificios, puentes, carreteras, etc. elaborados con el programa de modelado SketchUp [118]. Estos modelos se colocan en la escena como una capa más sobre la superficie del terreno (Figura 2.3). Microsoft también dispone de una aplicación similar llamada 3dvia [136] que permite elaborar modelos para Virtual Earth 3D.

Además de las funcionalidades anteriores, estas aplicaciones van incorporando progresivamente otras nuevas, como pueden ser la de ver la situación actual del tráfico en las carreteras, la meteorología actual e incluso la de poder sobrevolar la superficie terrestre empleando un simulador de vuelo. Todo este conjunto de características unido a la continua mejora de la precisión y de la cantidad de información de terrenos disponible, está consiguiendo un incremento continuo del número de usuarios de estas aplicaciones.

Figura 2.2 – Fotografías de alta calidad de Street View (Izquierda) y Bird's Eye View (Derecha).

Figura 2.3 – Visualización de objetos 3D en Google Earth.

Otro ejemplo de aplicaciones son las herramientas de gestión de información geográfica (SIG), que permiten la manipulación y visualización de todo tipo de información topográfica. Dentro de estas herramientas cabe destacar GVSIG [52], un proyecto de software libre desarrollado por la Consejería de Infraestructuras y Transportes de la Comunidad Valenciana que ha alcanzado un gran éxito en la comunidad internacional. Esta herramienta permite el acceso a todo tipo de información geográfica tanto de forma local como a través de servidores remotos (WMS, WCS, WFS, etc.). Además, dispone de múltiples extensiones, entre ellas la de gvSIG3D [53] que permite la visualización 3D de la información geográfica, si bien, actualmente, no está preparada para la continua descarga interactiva de datos provenientes de extensas bases de datos del terreno ubicadas de forma remota. La edición dinámica de los datos de las bases de datos remotas del terreno está fuera del objetivo de esta tesis, por lo que en ella nos centraremos en el apartado de visualización y transmisión de los mismos.

2.1.3 – Tipos de datos del terreno

Este tipo de aplicaciones emplea principalmente dos tipos de datos: las mallas de alturas, que describen geométricamente la superficie de terreno, y las texturas, que le proporcionan un aspecto más realista al terreno.

Las mallas de altura describen geométricamente la superficie del terreno. Su expresión más sencilla se denomina usualmente modelo digital de elevaciones (DEM: Digital Elevation Model) o modelo digital del terreno (DTM: Digital Terrain Model) [7], la cual consiste en una matriz bidimensional de alturas distribuidas espacialmente de forma uniforme (Figura 2.4).

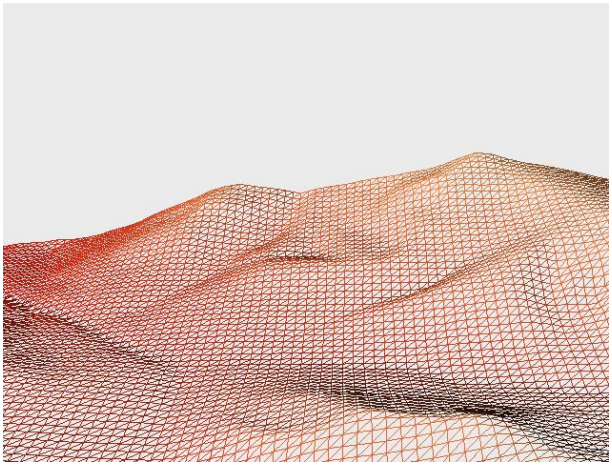


Figura 2.4 – Visualización de una DEM en perspectiva.

La DEM o DTM es la forma más sencilla y la más compacta de representar las alturas del terreno. Gracias a su regularidad, puede representarse en memoria con una simple matriz de alturas, lo que permite un acceso a los datos rápido y sencillo. Sin embargo, esta misma regularidad es su principal inconveniente, ya que le impide adaptarse de forma óptima a la rugosidad del terreno, al emplear la misma resolución para todas sus regiones.

Una alternativa a la DEM o DTM son las mallas irregulares de triángulos (TIN: Triangulated Terrain Network) [14]. Estas representaciones se adaptan normalmente mejor a las características del terreno empleando un menor número de triángulos que la primera, debido a que las áreas con pequeñas diferencias de altura se modelan con pocos triángulos, mientras que en las regiones con mayores irregularidades se emplean un número mayor de triángulos. No obstante, la mejora de la calidad obtenida empleando TINs no compensa el mayor uso de CPU que requiere su elaboración y su visualización [40]. Además, para éstas, se ha de almacenar la posición de cada punto de la malla (no sólo la altura como ocurre en las DEM), lo que implica el empleo de una mayor cantidad de memoria.

El otro tipo principal de datos de terrenos manejados en estas aplicaciones son las texturas [56], las cuales son imágenes reales del terreno extraídas a partir de fotos aéreas o fotos de satélite, que se superponen a las mallas de alturas para proporcionar una mayor sensación de realismo (Figura 2.5).

Como se ha comentado previamente, normalmente en las aplicaciones de visualización de terrenos la información de las texturas tiene un mayor nivel de resolución que la de las alturas, y por lo tanto su tamaño es mayor, por lo que generalmente se emplean mecanismos de compresión para reducir su tamaño. Actualmente, los esquemas que se emplean para la compresión de la información de las texturas del terreno están basados en la transformada

wavelet, tal como ocurre con el algoritmo de compresión Jpeg2000 [68]. En el apartado siguiente se discutirá con más profundidad toda la problemática asociada con este tipo de información.

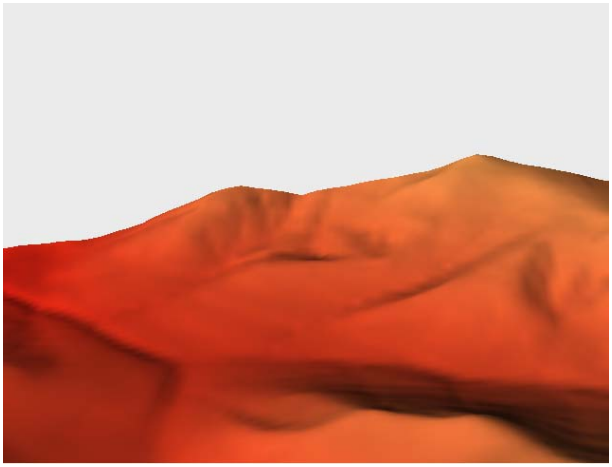


Figura 2.5 – Visualización de una textura superpuesta al DEM.

Aparte de la información de las alturas y de las texturas, también suele añadirse en las aplicaciones de visualización de terrenos una capa de información modelada como objetos 3D. Estos objetos se definen a partir de un conjunto de primitivas poligonales y de texturas con las que se intenta emular objetos del mundo real como pueden ser edificios, árboles o carreteras [109].

Estos objetos se colocan en la escena sobre la capa de la malla y la textura, y tienen como objetivo aumentar el realismo de la escena virtual. La complejidad de estos objetos no suele ser muy elevada y no suele haber un número muy alto de ellos visualizándose de forma simultánea en la escena para evitar que la velocidad de refresco disminuya. Tal como hemos comentado previamente, tanto Google Earth como Virtual Earth 3D permiten la inclusión de estos elementos en sus entornos virtuales (Figura 2.3).

2.1.4 – Técnicas de simplificación

Como se ha comentado en la introducción del capítulo, una aplicación de visualización de terrenos en tiempo real suele manejar una gran cantidad de información, la cual no puede almacenarse en la memoria y visualizarse toda ella de forma simultánea, ya que sobrepasa la capacidad de los ordenadores y de las tarjetas gráficas actuales. Por ello, para reducir su

tamaño, es necesario llevar a cabo algún tipo de gestión de toda esta información utilizando una serie de algoritmos que obtienen y combinan versiones simplificadas de las superficies del terreno en tiempo real [83].

La mayor parte de estos algoritmos definen diferentes versiones simplificadas del terreno conocidas generalmente como niveles de detalle del terreno (LOD: Level of Detail) [81], utilizando tan sólo una de ellas de forma simultánea. El objetivo de esta técnica es regular la cantidad de datos usados para representar la superficie del terreno en cada una de sus regiones, empleando una mayor o menor cantidad de datos dependiendo de las necesidades de cada área, consiguiendo de esta forma disminuir el tiempo empleado en los procesos de visualización de la escena y obtener una velocidad de refresco superior. En el caso de la visualización de terrenos, habitualmente se emplean técnicas de gestión de los niveles de detalle dependientes del punto de vista [85], donde el nivel de detalle utilizado depende de la distancia al observador, de modo que las regiones más próximas a éste se visualizarán con un nivel de resolución mayor que las que están más alejadas. Adicionalmente al uso de la distancia al observador, existen otras técnicas que se suelen emplear para la selección del nivel de detalle que se basan en la rugosidad de la superficie del terreno, permitiendo el uso de un mayor o menor nivel de detalle en función de esa rugosidad. Estás técnicas emplean diferentes métricas del error para determinar qué nivel de rugosidad es el más adecuado en cada momento.

Cuando se emplean niveles de detalle, se pueden utilizar algoritmos discretos o continuos. En los algoritmos discretos se produce un cambio brusco entre el nivel de detalle que se está visualizando y el siguiente nivel de detalle a visualizar, lo que puede provocar que el observador aprecie visualmente esta transición. Para suavizar esa transición se pueden emplear algoritmos continuos [110], donde se realiza la visualización de un conjunto de interpolaciones entre los vértices del nivel de detalle actual y los del nuevo nivel de detalle, para evitar que el usuario perciba esta transición. Estas técnicas se conocen como técnicas de morphing o geomorphing [20][83].

Representación jerárquica para una estructura regular. Quadtree y Bintree

Para implementar la técnica de gestión de los LODs en función del punto de vista sobre una estructura de malla regular, es necesario representar diferentes regiones de la malla a distintas resoluciones. Para ello se divide el terreno espacialmente en regiones cada vez más pequeñas utilizando una estructura jerárquica de árbol de regiones. Cada nivel del árbol representa una versión del terreno de diferente resolución. Se obtendrá una mayor resolución a medida que se profundice en el árbol.

Las dos estructuras jerárquicas usadas habitualmente son el quadtree y el árbol binario de triángulos o bintree [113]. El quadtree representa una región rectangular que es uniformemente dividida en cuatro cuadrantes, los cuales, a su vez, se pueden subdividir recursivamente en otros cuatro cuadrantes hasta alcanzar la resolución requerida (Figura 2.6). A la hora de visualizar estos cuadrantes se divide cada uno de ellos en triángulos. Tenemos ejemplos del uso de esta estructura en [55], [57] y [97], entre otros.

Un problema común cuando se trabaja con estas estructuras es la aparición de huecos o uniones en T que se producen cuando dos triángulos adyacentes tienen niveles de resolución diferentes (Figura 2.7). Los huecos, aparecen sobre las aristas del cuadrante cuando una región de mayor nivel de detalle introduce vértices que no pertenecen a la arista de menor nivel de detalle, y las uniones en T, se producen cuando un triángulo de mayor nivel de detalle no comparte un vértice con en el triangulo adyacente de menor nivel de detalle. Las uniones en T pueden causar diferencias visibles en la iluminación del terreno.

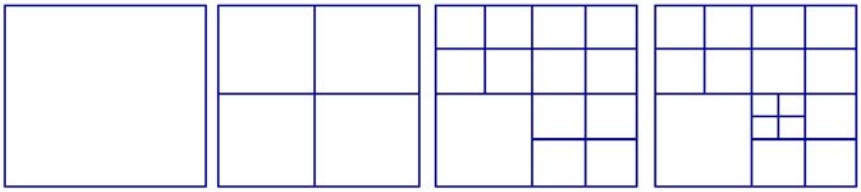


Figura 2.6 – Subdivisión recursiva de una región empleando una estructura de quadtree.

Al igual que el quadtree, el árbol binario de triángulos representa una región rectangular que es dividida uniformemente pero, en vez de en cuatro cuadrantes, esta región se subdivide en dos mitades (Figura 2.8). Generalmente, el triángulo base se define como un triángulo isósceles recto y se subdivide trazando una arista entre el vértice que contiene el ángulo recto y el punto medio de la hipotenusa del triángulo, esta técnica se conoce como la técnica de bisección por el lado más largo [81]. Como ejemplos de uso de esta estructura se pueden mencionar [36] y [82].

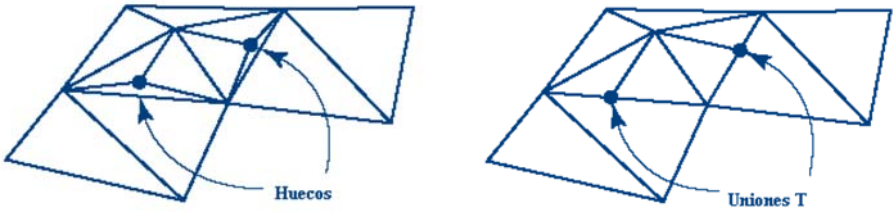


Figura 2.7 – Huecos y uniones en T

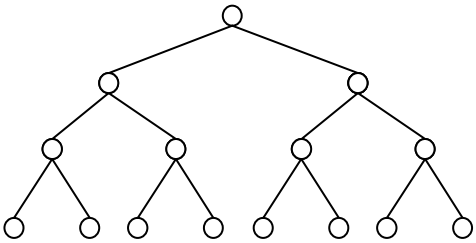


Figura 2.8 – Niveles 0-3 de un árbol binario de triángulos

Una de las principales ventajas de esta estructura respecto a la del quadtree es que evita la aparición de huecos que se producen entre regiones adyacentes de diferente resolución, debido a que la diferencia máxima de resolución entre dos triángulos vecinos será de uno (Figura 2.9).

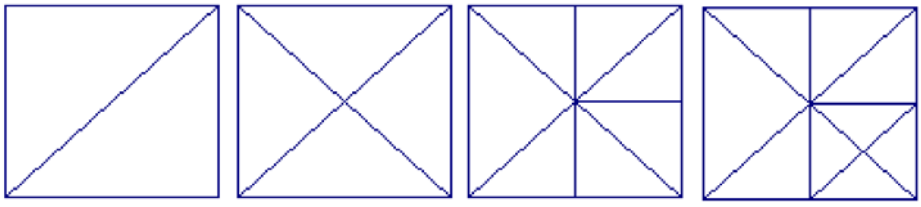


Figura 2.9 – Subdivisión recursiva de una región empleando un árbol binario de triángulos.

Recorte de la vista

Para llevar a cabo la visualización de las superficies de terreno, aparte de los niveles de detalle, existe otra técnica utilizada habitualmente para reducir la cantidad de polígonos y texturas que se envían a la tarjeta gráfica. Esta técnica se conoce como recorte de la vista (en inglés esta técnica se conoce como view-culling), y consiste en excluir de la fase de dibujado aquellas regiones de la escena que caen fuera de la pirámide de visión del observador [17].

Si se emplea una estructura de representación jerárquica como el quadtree, el proceso de recorte de la vista puede realizarse de forma rápida y eficiente. En este caso, el proceso consiste en recorrer el quadtree de arriba abajo, comenzando por el nodo raíz, comprobando si cada uno de los nodos cae o no fuera de la pirámide de visión. Si el nodo cae totalmente dentro de la pirámide de visión, tanto el cuadrante asociado a ese nodo como los de sus

nodos hijos son dibujados, deteniéndose en ese punto el recorrido de esa rama del quadtree. Si el nodo cae totalmente fuera de la pirámide de visión, su cuadrante y el de sus nodos hijos no son dibujados, deteniéndose también el recorrido de esa rama en ese nodo. Por último, si el nodo cae parcialmente dentro de la pirámide de visión, el cuadrante asociado al nodo se dibuja, pero esta vez el recorrido continúa para evaluar la visibilidad de cada uno de sus nodos hijos. Empleando esta técnica, se reduce sustancialmente la cantidad de nodos para los cuales se evalúa su visibilidad.

En la Figura 2.10 se observa el recorte de la vista de una superficie de terreno, donde las zonas más claras son las regiones que se encuentran totalmente dentro de la pirámide de visión (por lo que se dibujarán), las partes más oscuras son las que se encuentran totalmente fuera (no se dibujarán), y las restantes son las que se encuentran parcialmente dentro (también se tendrán que dibujar).

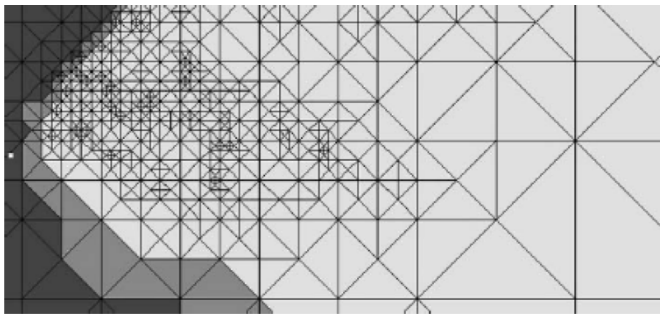


Figura 2.10 – Recorte de la vista de una superficie del terreno.

2.1.5 – Algoritmos para la Representación de Terrenos

Durante los últimos años se han desarrollado un variado conjunto de algoritmos para la simplificación de las mallas de alturas del terreno. En este apartado se va a realizar una revisión global de estos algoritmos; si se quiere obtener información más detallada, se recomienda consultar los resúmenes [29], [83], [98] y [100] .

Se puede hacer una clasificación de estos algoritmos en tres grandes grupos: algoritmos basados en triangulaciones irregulares, algoritmos basados en triangulaciones semirregulares y algoritmos basados en triangulaciones regulares:

Algoritmos basados en triangulaciones irregulares (TIN: Triangular Irregular Network).

Estos algoritmos generan aproximaciones simplificadas de la superficie del terreno empleando triángulos de cualquier forma y tamaño (Figura 2.11), proporcionando, generalmente, la mejor aproximación posible a la superficie del terreno original (DEM sin simplificar) que puede obtenerse para un determinado número de triángulos.

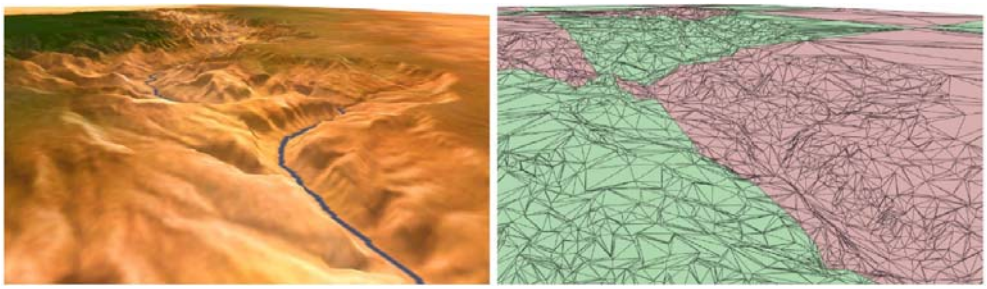


Figura 2.11 – Triangulación irregular (TIN)

Para obtener estas aproximaciones, estos algoritmos emplean diferentes métodos de simplificación como mallas progresivas, eliminación de vértices, métricas del error cuadráticas o inserción de vértices [44][45][58].

Los primeros algoritmos de este tipo se basaban en la triangulación 2D de Delaunay [31], como es el caso de los trabajos de Cignoni et al. [14] y Voigtmann et al. [131], si bien, posteriormente, aparecieron otros algoritmos que permitían una conectividad arbitraria entre triángulos, como los descritos por De Floriani et al. [28] y por Hoppe [60].

Garland y Hekbert en [44] y Lee en [78], especifican y comparan diferentes métodos para insertar y eliminar los vértices de las mallas de alturas dentro de una TIN de forma eficiente. La pirámide de Delaunay empleada en los algoritmos descritos por Davis et al. en [24] y por Voigtmann et al. en [131] es una extensión multirresolución de la triangulación de Delaunay, donde los grupos de triángulos conectados son reemplazados jerárquicamente por otros grupos con pocos triángulos.

Otros algoritmos que emplean triangulaciones jerárquicas, como las que se discuten en los trabajos de De Floriani et al. en [25], [26] y en [27], se basan principalmente en la subdivisión de triángulos para obtener diferentes versiones de la superficie del terreno. Hoppe también emplea una triangulación jerárquica en [60], donde describe un algoritmo que emplea un método de triangulación multirresolución del terreno basado en el uso de operaciones de fusión de aristas de triángulos y de división de vértices [58] para llevar a cabo la simplificación y el refinamiento de la malla.

La falta de restricciones de estos algoritmos asociados a la TIN provoca que sean computacionalmente costosos, debido a que tienen que gestionar las relaciones entre los triángulos de la malla y mantener sus dependencias durante el refinamiento o la simplificación de la misma. Además, estos algoritmos consumen más memoria que los algoritmos empleados para las mallas regulares o semirregulares, al tener que almacenar todas las coordenadas de los vértices de las mallas y no sólo el valor de la altura.

Por todo esto, estos algoritmos no se suelen emplear en aplicaciones de visualización de terrenos de tiempo real, porque solamente son capaces de representar pequeñas superficies de terreno de baja calidad a velocidades de refresco interactivas.

Para solucionar estos problemas, se desarrollaron otros algoritmos, que en vez de generar las simplificaciones de las superficies del terreno en tiempo de ejecución, las realizan en tiempo de preproceso, obteniendo bloques de primitivas precalculados formados por una gran cantidad de triángulos, que son almacenados en disco para poder ser empleados en cualquier momento a lo largo de la visualización (Figura 2.12).

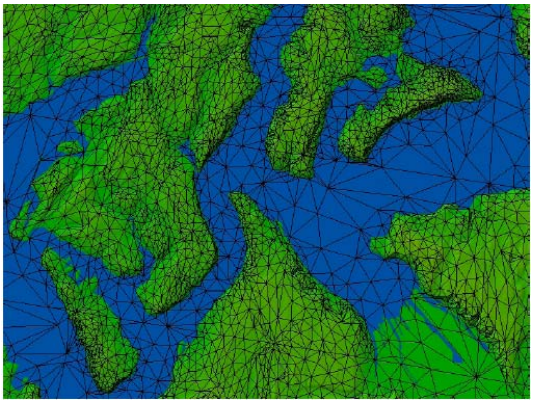


Figura 2.12 – Representación del terreno empleando bloques precalculados.

El primero de estos algoritmos que empleaba bloques precalculados es RUSTiC [103]. La idea principal de este algoritmo es reemplazar los triángulos simples del árbol binario empleado en el algoritmo de ROAM, por bloques de triángulos precalculados compuestos de un gran número de triángulos. Esta idea fue evolucionada por Levenberg en [79], donde se emplea una caché de esos bloques de triángulos en la memoria de la tarjeta gráfica para aumentar la eficiencia del algoritmo.

Lario et al. en [77] y Schneider y Westermann en [114], describen algoritmos similares que emplean quadtrees en vez de árboles binarios, estos quadtrees utilizan hiperbloques de triángulos como elemento mínimo para la simplificación de las superficies del terreno.

La misma idea propuesta en [79] y [103] se explota por Cignoni et al. en [15], donde triángulos simples de una jerarquía de árboles binarios son reemplazados por pequeños TINs llamados batches. Los batches para todos los niveles de resolución son construidos en tiempo de preproceso y se almacenan en disco de forma optimizada para llevar a cabo la representación de la superficie del terreno. En [16], los mismos autores extienden su algoritmo original para representar la superficie del terreno de todo un planeta de forma simultánea, alcanzando tiempos de refresco interactivos.

Los algoritmos que emplean bloques de geometría precalculados aprovechan las prestaciones de las tarjetas gráficas actuales para visualizar estos bloques de manera muy rápida, sin afectar a la velocidad de refresco del sistema. Este hecho, junto con el del ahorro de tiempo de proceso al no tener que realizar la triangulación de la malla de alturas en tiempo de ejecución, aumenta la eficiencia general del sistema de visualización. Sin embargo, las triangulaciones que se construyen con estos algoritmos necesitan mucho espacio de almacenamiento y requieren de frecuentes accesos a disco que pueden disminuir la velocidad de refresco del sistema de visualización.

Algoritmos basados en triangulaciones semirregulares

Estos algoritmos dividen recursivamente las mallas de alturas empleando un tipo de estructura regular predeterminada, como pueden ser triángulos, diamantes o cuadrantes, formando una jerarquía que relaciona estas estructuras con diferentes niveles de detalle (Figura 2.13). Se puede encontrar información más detallada de estos algoritmos y de las estructuras que emplean en [98] y [100].

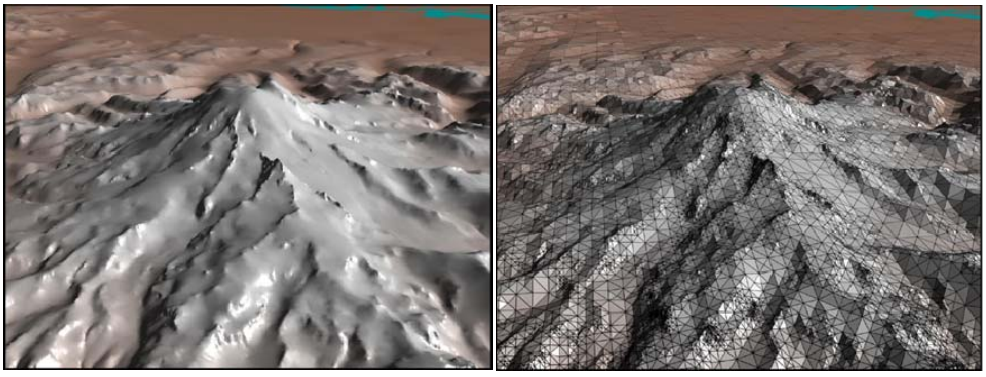


Figura 2.13 – Triangulación semirregular

Dentro de este tipo de algoritmos se encuentran aquellos que emplean estructuras jerárquicas basadas en triángulos como el definido por Evans et al. en [40], los que aplican la técnica de la bisección de triángulos por el lado más largo como el definido por Lindstrom y Pascucci en [83], los que usan árboles binarios de triángulos como los definidos por Duchaineau en [36] y Lindstrom en [81], o aquellos que emplean quadtrees restrictivos como los definidos por Pajarola en [97] y Samet en [113], o no restrictivos, como los definidos por Olanda et al. [93] y en [95].

La idea principal de estos algoritmos es construir una jerarquía multirresolución regular, con la cual, empleando procesos de refinamiento y simplificación, se obtienen simplificaciones de las superficies del terreno libres de grietas. El proceso de refinamiento se aplica a un triángulo isósceles y consiste en el refinamiento recursivo del mismo, dividiéndolo por su lado más largo, para obtener dos triángulos más pequeños. El proceso de simplificación realiza la operación contraria, fusionando pares de triángulos en uno solo.

La estructura jerárquica regular empleada en estos algoritmos permite codificar todas las dependencias de los triángulos o cuadrantes en los procesos de refinamiento y de simplificación de una forma simple y compacta. Cada uno de los nodos de esta estructura jerárquica tiene asociado un error, a partir del cual se decide que triángulos o cuadrantes han de refinarse o simplificarse en cada paso del algoritmo.

Lindstrom definió en [81] uno de los primeros algoritmos de triangulación de superficies de terreno basados en quadtrees. Este algoritmo emplea una métrica del error en el espacio de la pantalla para controlar el nivel de detalle con el que se tienen que representar cada una de las regiones del terreno. Para generar una triangulación sin huecos o grietas, este algoritmo impone una restricción en el quadtree, por la cual, la diferencia de nivel de detalle entre regiones vecinas debe ser como mucho de un nivel. Esta restricción define unas relaciones de dependencia jerárquicas entre los vértices que deben mantenerse para garantizar la ausencia de huecos.

El algoritmo ROAM [36], es conceptualmente muy parecido al anterior, aunque éste se basa en una jerarquía de árbol binario. Emplea, al igual que Lindstrom, una métrica del error en el espacio de la pantalla y, además, utiliza colas de prioridad para dirigir la triangulación de la superficie de terreno. Existen varios trabajos posteriores en los que se realizan mejoras de estos dos algoritmos, como por ejemplo el de Gerstner en [46], o los de Pajarola en [97] y en [99].

Otros algoritmos de simplificación de superficies del terreno alternativos definidos por Gross et al. en [55] y por Yusov y Turlapov en [133], se basan en el uso de la transformada wavelet. Estos algoritmos utilizan una métrica del error basada en los coeficientes wavelet obtenidos al aplicar la transformada wavelet sobre la malla de alturas del terreno.

También existen otros algoritmos como los descritos por Olanda et al. en [93] y [95], que emplean quadtrees no restrictivos, donde no se limita la variación máxima de nivel entre nodos vecinos. Estos algoritmos obtienen triangulaciones de la superficie del terreno de la misma calidad que los algoritmos que emplean quadtrees restrictivos, pero empleando una cantidad de triángulos menor. Esos triángulos adicionales utilizados en los quadtrees restrictivos se emplean únicamente para evitar la aparición de huecos o grietas. Los algoritmos basados en quadtrees no restrictivos eliminan estos triángulos, por lo que tienen que emplear otros mecanismos para evitar la aparición de esos huecos o grietas.

Estas triangulaciones semirregulares basadas en árboles binarios o en quadtrees (tanto restrictivos como no restrictivos) son, hasta el momento, uno de los modelos de triangulación que mejores resultados obtienen a la hora de construir aproximaciones de superficies del terreno dependientes del punto de vista. No obstante, estos algoritmos llevan a cabo todos sus procesos en la CPU y no aprovechan las capacidades de proceso de las GPUs de las actuales tarjetas gráficas, por lo que sería interesante realizar una adaptación de estos algoritmos para que algunos de estos procesos fueran computados por las GPUs, liberando parte de la carga de la CPU.

Un ejemplo de uso de las GPUs para este tipo de triangulaciones lo encontramos en el trabajo de Dick et al. en [33], donde se emplea una técnica de "ray-casting" basada en GPU para generar triangulaciones de forma dinámica que obtiene unas prestaciones adecuadas cuando se emplean datos de muy alta resolución, pero que cuando esa resolución es moderada, proporciona unas prestaciones menores que con los métodos tradicionales, debido al alto coste computacional de esta técnica. Otros trabajos a destacar donde se emplean técnicas de "ray-casting" para generar este tipo de triangulaciones son los de Tevs et al. en [128] y Oh et al. en [94], ambos trabajos emplean una técnica basada en una versión simplificada de un mipmap de máximos/mínimos [11][51], sin embargo, ambos trabajos presentan también un alto coste computacional que limita el tamaño máximo de los datos que pueden manejar de forma eficiente.

Algoritmos basados en triangulaciones regulares

Estos algoritmos generan aproximaciones simplificadas de la superficie del terreno independientes de la rugosidad del mismo, empleando triangulaciones regulares. Generalmente, estas aproximaciones están formadas por un elevado número de triángulos, sin embargo, su creación y visualización es sencilla y rápida.

Lossaso y Hoppe en [84] propusieron abandonar el refinamiento dependiente de la rugosidad del terreno para, en su lugar, desarrollar un marco de trabajo que alimente de forma óptima la tubería gráfica. Esta decisión se apoya en el hecho de que las prestaciones de las tarjetas gráficas actuales permiten visualizar grandes cantidades de triángulos manteniendo unas velocidades de refresco interactivas. Para ello, los autores proponen la

técnica de geometry clipmap, que genera la versión simplificada de la superficie del terreno entremezclando un conjunto de regiones regulares rectangulares centradas en la posición del observador. Cada una de estas regiones tiene un nivel de detalle distinto (Figura 2.14). El nivel de detalle empleado en cada región depende exclusivamente de su distancia al punto de vista del observador, sin emplear ninguna métrica del error dependiente de la geometría del terreno. Asirvatham y Hoppe en [3], extendieron este trabajo implementando en la GPU la mayor parte de los procesos de la técnica de geometry clipmap.

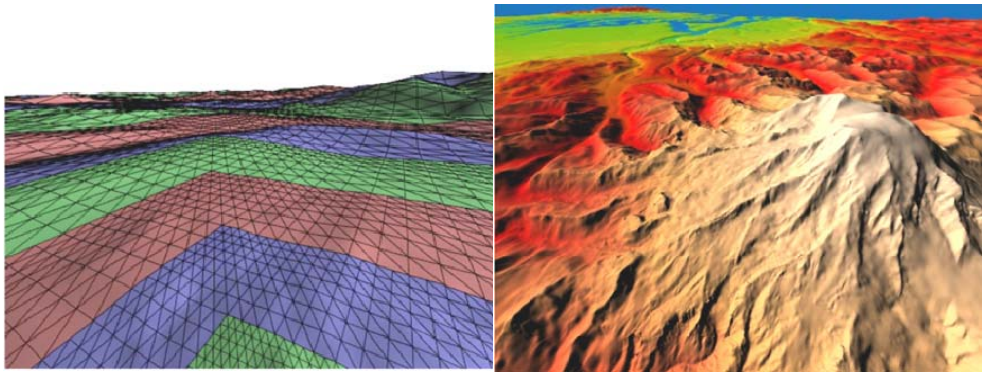


Figura 2.14 – Triangulación regular

Esta técnica proporciona varios beneficios, como por ejemplo, la posibilidad de compresión de los datos del terreno, la obtención de una velocidad de refresco de pantalla estable, el poder llevar a cabo la síntesis de nuevos niveles de detalle del terreno empleando ruido fractal o la simplicidad general del algoritmo.

El mayor punto débil de esta técnica es que para conseguir representar con alta precisión visual una extensa superficie del terreno y mantenerla a lo largo del desplazamiento del observador por la escena, se requiere el procesado y la visualización de un número elevado de triángulos. Esto puede afectar a la velocidad de refresco de la escena, sobre todo, si se emplean técnicas avanzadas de iluminación, que requieren una gran cantidad de tiempo de cálculo, o si se visualizan objetos 3D complejos en la escena.

2.2– Organización y compresión de Terrenos

2.2.1 – Introducción

Los avances en los dispositivos de adquisición de datos del terreno han permitido un aumento de la cantidad y de la precisión de la información disponible para las aplicaciones de visualización de terrenos. Los ordenadores actuales, a pesar de los avances tecnológicos, no son capaces de almacenar toda esa información en memoria y, aunque el ancho de banda de las líneas de transmisión ha aumentado considerablemente, tampoco lo ha hecho de forma suficiente para una transmisión eficiente de toda esa información.

Por ello, será necesario emplear algún esquema de compresión que permita disminuir el tamaño de toda esta información, logrando un almacenamiento y transmisión más eficiente de la misma. Además, también será necesario organizar esa información de forma eficiente. Para poder llevar a cabo todo lo anterior, en este apartado se realizará, en primer lugar, un estudio de los modelos existentes para organizar esta información. A continuación, se analizarán los diferentes esquemas válidos para la compresión y transmisión de esta información que, a su vez, sean compatibles con los modelos estudiados de organización.

2.2.2 – Organización de la información

Como se ha comentado, las aplicaciones que emplean terrenos tienen que gestionar un gran volumen de información. Las estrategias más comúnmente usadas para realizar esta gestión consisten en dividir esta información en un conjunto de rejillas de cuadrantes, donde cada una de estas rejillas hace referencia a un nivel de resolución de los datos del terreno. Con todas estas rejillas se construye una pirámide de cuadrantes (Figura 2.15).

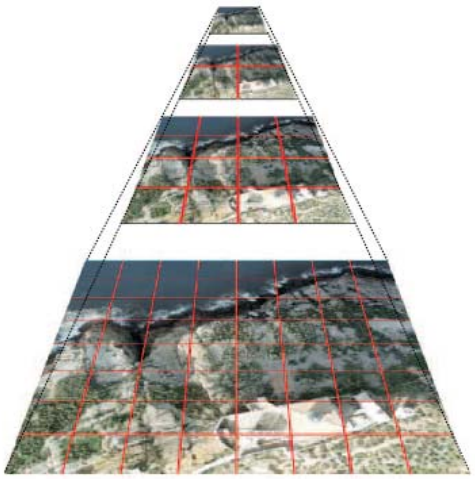


Figura 2.15 – Pirámide de cuadrantes

En la literatura encontramos que, en las primeras soluciones propuestas para la visualización de superficies del terreno, como en la descrita por Cohen et al. en [21], la información se almacenaba en disco y se particionaba en cuadrantes fijos. Rabinovich y Gotsman en [107], sugirieron una solución específica para aplicaciones cliente-servidor. En su trabajo, la información se almacena en un servidor central organizada en cuadrantes y se comprime por medio de un esquema basado en la transformada wavelet. El servidor envía progresivamente la información requerida para reconstruir los cuadrantes que intersectan con el campo de visión del usuario empleando la resolución apropiada.

Goss y Yuasa en [54] y Cline y Egbert en [19] definen las pirámides de cuadrantes. Estas pirámides son el resultado de dividir la secuencia de imágenes de diferentes niveles de resolución de una pirámide de imágenes [132] en un quadtree de cuadrantes de tamaño uniforme (Figura 2.15). Goss and Yuasa usaron este tipo de estructura para transferir texturas de modelos VRML entre servidores web y clientes. En este sistema, el cliente es el encargado de determinar dinámicamente que cuadrantes son visibles y que resolución de los mismos es la adecuada.

The FlyAway system definido por Hüttner en [65] emplea una aproximación similar, Hüttner propone una estructura llamada Mipmap Pyramid Grid o MPGrids. Esta estructura consiste en un conjunto de texturas de mipmap organizadas en una rejilla para representar una textura de alta resolución que podía ser mayor que la que soportaba la implementación de OpenGL.

Davis et al. en [24] y Cignoni et al. en [15] emplearon unas estructuras similares a las pirámides de cuadrantes para el paginado de extensas texturas de terreno. En Terra Vision II [108] también se resuelve el problema de visualización de largas bases de datos de terrenos usando una pirámide de cuadrantes para la geometría y para las texturas. Dollner et al. especifican en [34] un modelo multirresolución similar a las pirámides de cuadrantes para manejar múltiples capas de texturas. Este modelo extiende el modelo de resolución usado para los datos de geometría de los terrenos.

Además de estas soluciones, se pueden encontrar también alguna solución completamente diferente a las pirámides de cuadrantes, como es el caso de clipmap [124]. Clipmap proporciona un mecanismo hardware para soportar el uso de texturas extensas. Sin embargo, estas técnicas están restringidas a bases de datos locales funcionando sobre ordenadores de altas prestaciones gráficas.

Otro trabajo interesante es el desarrollado por Date et al. en [22], donde se realiza un control del nivel de detalle para las texturas y para la geometría empleando wavelets, sin embargo, no se emplea compresión, por lo que los datos a transmitir son demasiado grandes. Además, sólo se utiliza la transformada wavelet Haar, que produce modelos multirresolución de baja calidad.

2.2.3 – Compresión de la Información

Como se ha comentado previamente, las aplicaciones de visualización de terrenos en tiempo real deben gestionar bases de datos extensas. Para disminuir en lo posible los recursos necesarios para almacenar toda esta información, es necesario emplear algún mecanismo de compresión que a su vez permita reducir el ancho de banda necesario para transmitir toda esta información a través de la red. Esta compresión aportará un sobrecoste a la hora de procesar la información debido al proceso de compresión y descompresión que debe llevarse a cabo.

Como también se ha comentado a lo largo de este capítulo, este tipo de aplicaciones emplea principalmente dos tipos de datos: las mallas de alturas, que describen geométricamente la superficie de terreno, y las texturas, que se sobreponen a las alturas proporcionando un aspecto más realista al terreno. Estos tipos de datos, aunque diferentes, guardan bastantes similitudes entre ellos. Los dos se pueden representar como una matriz que almacena un valor, alturas en el caso de las mallas de alturas y componentes de color en el caso de las texturas. Además, una malla de alturas puede codificarse como una textura donde los valores de color son en realidad valores de alturas, por lo que las técnicas empleadas para la

compresión de texturas pueden aplicarse de forma parecida para comprimir los datos de las alturas.

La Transformada Wavelet en la compresión de imágenes

Hoy en día, los esquemas de compresión basados en la transformada wavelet han demostrado ser los más eficientes a la hora de comprimir imágenes. De entre las ventajas que han llevado a la transformada wavelet a ocupar un lugar preferente en la compresión de imágenes se puede destacar [102]:

- La transformada wavelet decorrelaciona casi perfectamente los datos, concentrando la mayor parte de la información en un conjunto reducido de coeficientes. Esto permite a los algoritmos basados en esta transformada ofrecer excelentes tasas de compresión, especialmente cuando hay pérdidas. Es más, cuando las tasas de compresión son especialmente altas, los resultados obtenidos por la transformada wavelet suelen ser bastante superiores al del resto de esquemas conocidos, entre ellos la versión de Jpeg que utiliza la transformada discreta del coseno [67] (DCT = Discrete Cosine Transform). De hecho, la última versión del formato de compresión Jpeg, conocido como Jpeg2000 [68], sustituye la DCT por la transformada wavelet.

- La transformada wavelet es computacionalmente más eficiente que el resto de transformadas utilizadas en la compresión de imágenes (como la transformada rápida de Fourier o la transformada discreta del coseno). Mientras el coste de la transformada wavelet es lineal con el número de puntos, para el resto de transformadas suele ser como mínimo de N log N (Siendo N el número de puntos).

- Gracias al análisis en multi-resolución es posible realizar una reconstrucción progresiva, transmitiendo primero una versión de baja resolución de la imagen, para posteriormente ir transmitiendo información para refinar esa imagen. Además, el empleo de un esquema progresivo no provoca un aumenta del volumen de la información a transmitir.

Para conocer con más detalle las propiedades de la transformada wavelet, se recomienda revisar el estudio que Daubechies realiza en [23], donde se analiza en profundidad las propiedades y características de los principales tipos de transformada wavelet. Además, para conocer en detalle los algoritmos de compresión basados en wavelets que se aplican a la compresión de imágenes se recomienda el resumen realizado por Sudhakar et al. en [122].

Compresión de las mallas de alturas

Recientemente, varios autores se han concentrado en combinar métodos de compresión de datos con esquemas multirresolución, para reducir la cantidad de ancho de banda de

transmisión y de memoria necesarias para las aplicaciones de visualización de terrenos. Las técnicas que emplean la división de los datos en cuadrantes, suelen usar compresores estándar 2D para comprimir de forma independiente cada cuadrante, lo que limita la tasa de compresión que se puede alcanzar al no aplicarse la compresión sobre todos los datos. Se han propuesto una gran cantidad de algoritmos de compresión que son capaces de proporcionar altas tasas de compresión, transmisión progresiva de la información y acceso espacial aleatorio. Jpeg2000 [68] es un estándar que soporta todas estas características.

Como se ha comentado en el apartado anterior, Rabinovich y Gotsman en [107] y Date et al. en [22] emplean la transformada wavelet para codificar la información del terreno. Gioa et al. en [47] elaboran un conjunto de algoritmos basados en la transformada wavelet, que pueden ser empleados para la reconstrucción de superficies de geometría por una aplicación de visualización de terrenos que utilice una arquitectura cliente-servidor.

Kim and Ra proponen en [74] una visualización de superficies de terrenos a través de la red que emplea la transformada wavelet para comprimir la geometría, si bien se especifica que también podría emplearse el mismo esquema para las texturas. Este trabajo emplea un modelo de quadtree restrictivo que es actualizado en tiempo real mediante la transmisión y descompresión de los coeficientes wavelets obtenidos en la compresión. Sin embargo, la adaptación por vértices realizada por este método limita la eficiencia de la GPU, y los métodos de proceso por bloques especificados necesitan del acceso a múltiples bloques de forma simultánea para llevar a cabo la reconstrucción de la aproximación.

Lossaso y Hoppe, con el esquema de geometry clipmaps [84], organizan los datos de alturas del terreno en una pirámide multirresolución, donde los residuos entre niveles son comprimidos empleando un compresor que permite un rápido acceso a cualquier región de los datos [87]. El almacenamiento de las alturas de forma comprimida, y la reconstrucción en tiempo de ejecución de los datos de las normales y de los colores (usando un simple mapeado de color de altura), proporciona una representación muy compacta. Sin embargo, el esquema piramidal empleado limita la adaptabilidad, funcionando esta técnica mejor con campos de visión amplios que contengan una geometría casi plana. Además, es necesario emplear tiempo de CPU o de GPU para mezclar los diferentes niveles de clipmap y así evitar la aparición de grietas entre ellos.

Clasen y Hege proponen en [18] una extensión de este trabajo para el uso de conjuntos de datos esféricos. Esta propuesta tiene el problema de necesitar realizar en la GPU costosas computaciones trigonométricas para cada vértice. Deb and Narayanan han propuesto en [30] una versión de geometry clipmap para la transmisión de información a través de la red, aunque sólo permite un pequeño número de clientes por servidor y cada servidor tiene que controlar constantemente el estado de cada cliente.

Hwa et al. apuntan en [66] que usando una jerarquía 4-8, los cuadrantes rectangulares asociados a cada diamante también pueden ser comprimidos empleando métodos de compresión estándar 2D de imágenes.

Gobbetti et al. desarrollan en CBDAM [48], un algoritmo que emplea "wavelet lifting" [70] para la compresión eficiente de los datos. En este marco de trabajo, la reducción de resolución de los diamantes se asocia con el análisis wavelet, mientras que el refinamiento se corresponde con la síntesis wavelet. Esta aproximación, emplea métricas del error para controlar la calidad de la simplificación, pero requiere el almacenamiento de dos matrices de coeficientes por diamante y un proceso de construcción en dos pasos en vez de uno.

Una aproximación similar ha sido realizada por Bettio et al. en [5], donde proponen un proceso eficiente para incorporar la compresión de alturas y de información de textura dentro del marco de trabajo de BDAM [15]. Pero, a diferencia de la solución anterior, en este trabajo se emplea la "transformada wavelet lifting estándar" [75] y se propone una implementación más rápida utilizando aritmética de enteros, más adecuada para un entorno de red con clientes heterogéneos.

Otros trabajos a destacar son Royan et al. en [111], que utiliza la transformada wavelet para la codificación y transmisión de la geometría de los terrenos, Lin et al. en [80], que emplea la transformada wavelet que incorpora Jpeg2000 [68] para la compresión y transmisión de objetos geométricos 3D, y Dick et al. en [32] que presentan un esquema de compresión para la geometría de mallas de quadtree restrictivas, que reduce el número de accesos a disco y la transferencia de información entre la CPU y la GPU.

Existen otros muchos autores que han explorado el problema de crear representaciones comprimidas de los datos de geometría, pero en la mayoría de los casos, se centran en conseguir altas tasas de compresión en vez de centrarse en algoritmos rápidos más adecuados para la visualización en tiempo real de la representación comprimida. Un resumen de este tipo de soluciones puede encontrarse en [2] y en [101].

2.3 – Arquitectura de Transmisión

2.3.1 – Introducción

Como ya se ha comentado anteriormente, las aplicaciones de visualización de terrenos suelen emplear una gran cantidad de información, que puede alcanzar en ocasiones un tamaño de varios terabytes. Sin embargo, cada usuario generalmente sólo va a visualizar una pequeña parte de esa información. Por lo tanto, no es adecuado que el usuario tenga una copia local de toda la información, ya que supondría un desperdicio de los recursos de almacenamiento, además de los costes de adquisición y mantenimiento de toda esa información.

Lo más conveniente es que toda la información se almacene en una base de datos remota, donde cada usuario sólo acceda y almacene de forma local la información que necesita. Emplear bases de datos remotas proporciona, además, otras ventajas como disponer de un mayor control de la información, pudiendo restringir el acceso a cierta información, o realizar modificaciones y añadir nueva información de manera que los usuarios pueden acceder inmediatamente a las mismas. Además, se pueden emplear varias bases de datos remotas distribuidas con información replicada, de forma que a las ventajas anteriores hay que añadir las de fiabilidad y disponibilidad de los datos.

La arquitectura típica empleada en las aplicaciones de visualización de terrenos en tiempo real a través de la red es la arquitectura cliente-servidor. En esta arquitectura, toda la información del terreno se encuentra almacenada en un servidor remoto y se transmite para su visualización a cada cliente atendiendo a las acciones de los usuarios.

El servidor puede garantizar un cierto tiempo de respuesta siempre y cuando el número de usuarios sea limitado. Sin embargo, cuando el número de clientes simultáneos aumenta sustancialmente, el tiempo de respuesta del servidor aumenta también de forma sustancial, debido a que el incremento en la carga del servidor provoca que éste no pueda responder a los clientes dentro de unos márgenes de tiempo aceptables. La única forma, a priori, de solucionar este problema con esta arquitectura, es añadir más servidores que hagan frente a este incremento de carga, con el consiguiente aumento del coste en la adquisición de los equipos y en su mantenimiento.

Otra posible forma de resolver este problema es empleando arquitecturas P2P. Estas arquitecturas son escalables por naturaleza con el número de clientes. En esta arquitectura todos los clientes actúan a su vez como servidores, enviando la información que tienen almacenada a los clientes que la soliciten. Por ello, el aumento del número de clientes conectados supone también el aumento del número de potenciales servidores, de forma que

la carga general del sistema se mantiene bastante estable y se suele distribuir equitativamente entre todos los clientes.

2.3.2 – Arquitectura Cliente–Servidor

Dentro del marco de la visualización de terrenos en tiempo real con bases de datos distribuidas, cuando la arquitectura empleada es la arquitectura cliente-servidor, todos los datos del terreno se encuentran almacenados en uno o varios servidores y se envían de forma progresiva a los clientes según las necesidades de cada uno (Figura 2.16). Popescu y Codella definen en [104] dos formas en las que los clientes y los servidores pueden colaborar para intercambiar información: el modelo "push" y el modelo "pull".

En el modelo "push" es el servidor el que se encarga de averiguar qué información necesita cada cliente. Para ello, el servidor realiza un seguimiento del movimiento del punto de vista representado en cada cliente. Este modelo tiene varios problemas. El primero es el aumento de la carga del servidor por cada nuevo cliente, que afecta claramente a la escalabilidad del modelo. Otro problema es el continuo intercambio de información entre el cliente y el servidor para que este último disponga del punto de vista del cliente actualizado, con el consiguiente aumento del ancho de banda necesario conforme aumenta el número de clientes. Además, necesita algún mecanismo para que el servidor conozca, en cada momento, el contenido de la cache del cliente, y así, evitar el envío de información redundante que ya posea éste. Debido a estos problemas, este esquema no suele emplearse para aplicaciones de visualización de terrenos en tiempo real.

En el modelo "pull", es el cliente el que se encarga de comunicar al servidor qué información del terreno necesita en cada instante. Para ello, el cliente inicia la comunicación con el servidor enviándole peticiones con los datos necesarios en cada momento. En este modelo, el servidor asume un papel más pasivo devolviendo al cliente únicamente la información requerida. Este es el modelo que se suele emplear en las aplicaciones de visualización de terrenos en tiempo real. El principal problema que presenta este modelo es la escalabilidad. Las aplicaciones en tiempo real necesitan obtener una respuesta dentro de un intervalo de tiempo reducido. A medida que aumenta el número de clientes que piden información al servidor, la carga de trabajo del servidor aumenta, llegando un instante en el que no puede devolver dicha información dentro del tiempo requerido. Como se ha comentado antes, la única solución posible que existe es la de añadir nuevos servidores, con el consiguiente coste económico que ello supone.

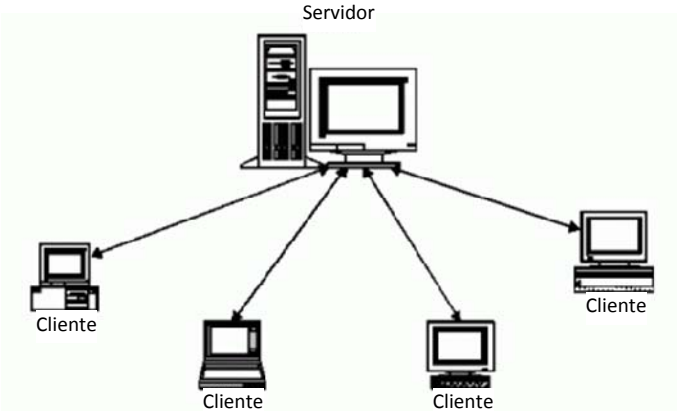


Figura 2.16 – Arquitectura Cliente-Servidor

2.3.3 – Arquitectura P2P

Tal como se comentó en el apartado 2.1.2 para el caso de Google Earth y Virtual Earth 3D, el número de usuarios de las aplicaciones de visualización de terrenos en tiempo real está aumentando constantemente. Dadas las limitaciones físicas a las que está sujeta la arquitectura cliente-servidor para dar calidad de servicio a tantos usuarios, recientemente están apareciendo soluciones que usan la arquitectura P2P dentro de este tipo de aplicaciones.

Las arquitecturas P2P son de uso común en múltiples campos; sin embargo, en el área de la visualización de terrenos en tiempo real es una arquitectura que apenas se ha utilizado hasta el momento. Un campo relacionado con los gráficos donde estas arquitecturas P2P son ampliamente utilizadas es en el de los entornos virtuales distribuidos (DVE: Distributed Virtual Environments), donde muchos usuarios comparten información e interactúan sobre el mismo escenario virtual [116]. Los participantes en este tipo de entornos intercambian entre sí, sobre todo, información relacionada con la localización de los usuarios dentro del mundo virtual y la localización de determinados objetos dentro del mundo con los que los usuarios pueden interactuar. La propia información del mundo virtual no suele intercambiarse, ya que suele encontrarse replicada en la máquina de cada participante.

Como ejemplo sirva los juegos en red multiusuario tales como Quake [106] y Everquest [41], en los cuales todos los datos de texturas y terrenos que emplea el juego se almacenan en el disco local del usuario, y la única información que se intercambia es la posición de los

jugadores y de los personajes del juego, así como la ubicación de los objetos con los que los jugadores pueden interactuar, tales como armas o tesoros.

El caso de las aplicaciones de representación de terrenos es algo distinto, ya que lo que se intercambia son datos de texturas y terrenos, que como se ha comentado previamente, suponen una gran cantidad de información, por lo que no resulta adecuado almacenarla de forma local para cada usuario y se suele repartir entre todos los nodos P2P.

Cuando se emplea una arquitectura P2P para intercambiar datos de terrenos extensos, el principal problema que se plantea radica en confeccionar una lista adecuada de nodos que puedan suministrar la información necesaria para cada nodo en cada instante. Se puede suponer, a priori, que si los puntos de vista de dos usuarios están situados en una posición similar, la información del terreno que requerirán ambos será también similar, de forma que gran parte de la información que necesita uno de ellos puede ser que ya esté disponible en la caché del otro usuario.

A priori, una estrategia podría ser confeccionar una lista con los nodos que estén más próximos en el mundo virtual. Sin embargo, ocurre que, tal como se muestra en la Figura 2.17, generalmente no existe relación entre la localización de un usuario en la escena y la localización física de su nodo en la red.

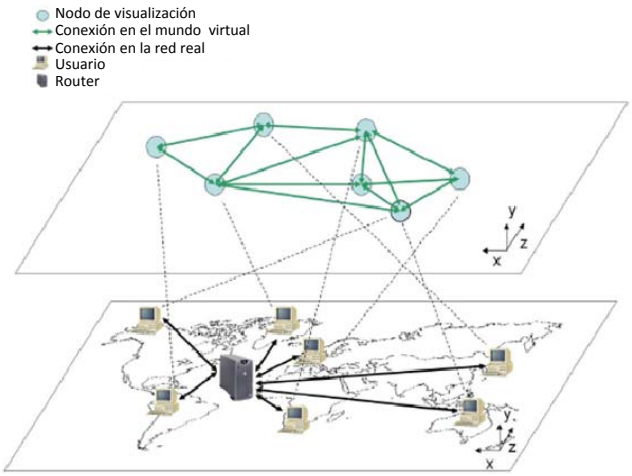


Figura 2.17 – Relación arquitectura P2P: Ubicación física real – Ubicación en la escena

Por lo tanto, hay que tener en cuenta que las conexiones entre los nodos P2P próximos espacialmente en la escena virtual, pueden ser eficientes desde el punto de vista de que pueden disponer de mucha información en común que intercambiar, sin embargo, las características de la red (por ejemplo, la latencia y el ancho de banda) de los clientes pueden ser muy diferentes. Por ello, las aproximaciones empleadas en DVEs como Solipsis [43][73], VON [61][62] o VoroGame [9], donde los nodos se interconectan según la cercanía de los usuarios en el entorno virtual utilizando una estrategia de localización 2D, puede no resultar una buena solución en un entorno de visualización de terrenos en tiempo real 3D.

Hu et al. en [62] y en [63] diseñan un sistema para seleccionar las conexiones entre nodos de un sistema DVE empleando criterios de localización espacial en la escena y características de las conexiones de red, con el objetivo de intercambiar información relacionada con los objetos 3D que componen la escena virtual (son objetos simples, no datos relacionados con terrenos). Parte de las ideas diseñadas para este sistema pueden extrapolarse a la representación de terrenos. En este sistema, para cada petición se elabora una lista de clientes formada exclusivamente por los clientes que se encuentran dentro de su área de interés, calculada en función de su posición dentro del entorno virtual. Una vez obtenida esa lista, se pregunta a cada cliente si dispone de un objeto concreto. Con ello se obtiene una nueva lista de la cual se seleccionan los vecinos de forma aleatoria.

Una vez seleccionado el vecino, el cliente realiza las peticiones de forma secuencial. En respuesta a estas peticiones, el cliente puede obtener o no la información requerida (el vecino puede no tener los datos, o estar muy cargado y decidir no responder a la petición). Si no se recibe la información, el cliente pedirá la información a otro nodo de la lista. Como miembro de esta lista se incluye al servidor, que contiene todos los objetos de la escena. Sin embargo, sólo se pedirá información al servidor cuando el cliente que realiza la petición sea el que se encuentra más cerca del objeto dentro de la escena. Uno de los problemas de este mecanismo es que no se puede asegurar un tiempo de respuesta máximo, requisito necesario para cualquier aplicación en tiempo real, ya que no se puede saber, a priori, cuándo se va a obtener una respuesta válida.

Dentro de este esquema, los clientes cuentan con una caché de tamaño triple al tamaño de almacenamiento supuestamente necesario para contener los objetos incluidos en su área de interés. Con ello, la caché también albergará información relacionada con los objetos ubicados en anteriores regiones de la escena por las que se haya desplazado el cliente, aumentando la posibilidad de intercambio de información con otros nodos.

Hu et al. en [64] han desarrollado un prototipo en el que aplicando todo lo anterior, han conseguido obtener un sistema con una mayor escalabilidad con el número de clientes conectados que empleando una arquitectura cliente-servidor tradicional.

Cavagna et al. en [12], emplean un mecanismo similar al anterior para obtener los objetos del mundo virtual. Primeramente se establecen tres tipos de nodos, nodos de memoria, que almacenan la información, nodos de visualización, que son los nodos finales que realizan la visualización de la escena, y nodos de conectividad, que se encargan de elaborar las listas de conexiones de los nodos.

Un nodo obtendrá información de los otros nodos que dispone en su lista. La selección del nodo al que realizar la petición se lleva a cabo evaluando una serie de parámetros, entre los que se encuentran, el tiempo que tarda un nodo en servir los datos, la estimación de los datos disponibles en el nodo dependiendo de su punto de vista, el ancho de banda asignado al nodo para responder y el número de nodos a los que ha servido ese nodo. Este trabajo, al igual que el anterior, está dirigido a la localización de objetos virtuales dentro de un entorno virtual, en lugar de a la transmisión de información del terreno.

Royan et al. en [111] extiende el trabajo de Cavagna et al. [12] para visualizar entornos virtuales extensos que incorporan modelos 3D de edificios. Los mismos autores emplean en [112] mpeg4 como método de compresión y transmisión de la información de la escena.

Otro trabajo relacionado con los anteriores es el de Sung et al. en [123], en el cual se especifica un estrategia diferente para seleccionar los nodos de los que obtener información de objetos 3D. Los autores proponen que cada nodo envíe periódicamente a sus nodos vecinos los nuevos objetos de que dispone, de forma que cuando necesiten obtener un determinado objeto sepan a qué nodo o nodos pueden acudir. Definen los nodos vecinos de un nodo como aquellos que se encuentran dentro de su área de interés. A la hora de seleccionar el nodo del que descargar el objeto, se divide el área de interés de cada nodo en niveles circulares concéntricos a los que se les asigna una mayor prioridad conforme más cerca está el nivel del centro del área de interés. De entre todos los nodos vecinos, se escoge para pedir el objeto aquel nodo donde el objeto se encuentra en un nivel de mayor prioridad. En caso de que haya más de un nodo vecino con el mismo nivel de prioridad para ese objeto, se escoge el nodo de forma aleatoria.

En la literatura sólo aparece PeerTR [134] como una arquitectura específica para aplicaciones de visualización de terrenos en tiempo real, desmarcándose de los DVEs. Esta arquitectura emplea una solución mixta en la que existen servidores y clientes, donde los clientes también pueden intercambiar información con otros clientes.

PeerTR emplea una pirámide de cuadrantes para organizar la información. Estos cuadrantes son independientes entre sí y se dividen en bloques independientes por niveles. A la hora de descargar información, no reutilizan la información descargada previamente, lo que supone una descarga de información adicional que como mínimo será de un 33%, tal y como se demuestra en [96].

Cada cliente tiene una lista de clientes vecinos a los que puede pedir información. Esta lista se crea por cercanía física en la red, por lo tanto, compondrán esta lista, entre otros, aquellos clientes que pertenezcan a la misma red local. A la hora de realizar una petición, siempre se seleccionan primero de la lista los vecinos que pertenecen a la red local del cliente. Sólo se pedirá información fuera de estos vecinos cuando ninguno de ellos disponga de la información requerida. De entre los vecinos de la red local, se selecciona aquel cuyo número medio de bloques de información enviada por unidad de tiempo sea mayor. Se eliminará de la lista el vecino que obtenga el valor más bajo de este parámetro, substituyéndolo por un nuevo cliente recibido del servidor.

Para conocer la información de que dispone cada cliente, éstos se intercambian mensajes cada cierto tiempo con la información almacenada en su caché. Después, cuando un cliente va a realizar una petición, selecciona de entre los clientes que disponen de la información deseada, aquel cuyo ancho de banda es mayor y tiene suficiente tiempo disponible para responder.

En este trabajo también se implementa un mecanismo de tolerancia a errores por medio del cual, cuando un cliente se desconecta o pierde la conexión, las peticiones dirigidas a ese cliente se reenvían al servidor, el cual se encuentra como un elemento más en las listas de todos los clientes.

2.4 – Conclusiones

Después de llevar a cabo el estudio de la situación actual de los diferentes campos que entran a formar parte del diseño de un sistema de visualización de terrenos en tiempo real en entornos distribuidos, se van a elegir las siguientes tecnologías:

- La gran cantidad de datos del terreno que se han de manejar, hace necesario emplear mecanismos que permitan visualizar esta información con una velocidad de refresco interactiva. Para realizar esta visualización se emplean algoritmos que llevan a cabo la triangulación de las mallas del terreno. Basándonos en el estudio realizado de estos algoritmos se decide llevar a cabo la implementación de varios de ellos basados en triangulaciones semirregulares, ya que son hasta el momento, los que permiten obtener mejores aproximaciones de la malla del terreno, debido a que ajustan la resolución a la rugosidad de la malla. Dentro de este tipo de algoritmos se decide implementar los que a priori proporcionan unos mejores resultados, que son ROAM [37](en su versión 2.0), SOAR [82][83] y un algoritmo basado en quadtrees no restrictivos que hemos elaborado [93] (NRQT: None Restricted Quadtree Triangulation).

Además, también se decide implementar un algoritmo basado en triangulaciones regulares, Geometry Clipmap [84], ya que se quiere comprobar si los avances de las tarjetas gráficas actuales son suficientes para que este algoritmo obtenga mejores resultados que los algoritmos que se basan en triangulaciones semirregulares, a pesar de que no adapta la triangulación a la rugosidad de la malla, sino que dibuja el mayor número posible de triángulos manteniendo una velocidad de refresco interactiva.

No se decide implementar ningún algoritmo basado en triangulaciones irregulares porque su complejidad suele provocar la obtención de tiempos de refresco elevados que no son válidos para sistemas de tiempo real.

Los algoritmos seleccionados se implementarán intentando hacer el mejor uso posible de las nuevas funcionalidades de las tarjetas gráficas actuales, como son los vertex buffers o vertex arrays. Se realizará una evaluación de los distintos algoritmos para determinar cuál de ellos se ajusta mejor a nuestro sistema.

- Esta gran cantidad de datos del terreno también es necesario almacenarla y transmitirla, por lo tanto, hay que emplear algún esquema de compresión que permita disminuir el tamaño de toda esta información, logrando un almacenamiento y transmisión eficiente de la misma. Además, también se necesita organizar esa información de forma eficiente. En la revisión del estado del arte, se ha visto que la estructura de organización más empleada y que mejor se ajusta al sistema que se va a desarrollar es la pirámide de cuadrantes, por lo tanto, se decide emplear esta estructura para organizar la información del terreno. De esta revisión, también se puede extraer que los esquemas basados en la transformada wavelet son, a priori, los más adecuados para realizar la compresión y transmisión de la información. Por ello, se plantea estudiar el uso del estándar Jpeg2000 [68], que emplea esta transformada, para evaluar sus propiedades y su comportamiento. Además, también se tendrá que estudiar la compatibilidad de este esquema con el modelo de organización de la pirámide de cuadrantes.

- También es necesario usar una arquitectura de red que sea adecuada para transmitir toda esa información. Del estudio realizado sobre las arquitecturas se puede intuir que la arquitectura cliente-servidor no va a ser adecuada para nuestro sistema, puesto que al no ser demasiado escalable con el número de usuarios, a medida que estos aumentan la calidad de servicio disminuye drásticamente, siendo el coste de mantener esa calidad de servicio elevado. Sin embargo, se llevará a cabo la implementación de esta arquitectura para validar esta intuición.

También parece que la arquitectura P2P va a ser más adecuada para obtener un sistema que sea altamente escalable y de bajo coste, debido a que el aumento del número de nodos conectados al sistema, aunque da lugar a una mayor carga global del sistema, esta carga se distribuye entre los nodos.

Sin embargo, debido a la ingente cantidad de datos del terreno que suelen manejar las aplicaciones de visualización de terrenos, y ante el requerimiento de un tiempo de respuesta adecuado para tiempo real, tampoco parece conveniente emplear una arquitectura P2P pura, debido a que, generalmente, un nodo no va a tener acceso en cada momento a toda la información existente disponible, ya que ésta se encontrará repartida entre todos los nodos y éstos no estarán continuamente conectados al sistema. Por lo tanto, no se va implementar una arquitectura P2P pura, puesto que para solventar ese problema, inicialmente habría que distribuir toda la información del terreno entre los nodos, proceso que, en una situación real de uso, no se puede llevar a cabo.

En su lugar, se implementará una arquitectura mixta cliente-servidor / P2P donde los usuarios puedan obtener información de otros usuarios y además, puedan acudir también a un servidor general que contenga toda la información. Esta arquitectura, a priori, mantendrá la propiedad de escalabilidad de la arquitectura P2P, y asegurará la disponibilidad de toda la información para cada usuario gracias al empleo de servidores.

Capítulo 3 – Comparativa de los algoritmos para la representación de terrenos

En el capítulo anterior se ha hecho un breve repaso de los algoritmos más empleados para la representación de terrenos en tiempo real. En este capítulo se van a seleccionar para su implementación y evaluación aquellos que, a priori, pueden adaptarse mejor al caso particular que se plantea en esta tesis, el de la visualización de terrenos extensos en entornos distribuidos, con el fin de seleccionar uno que pueda adaptarse de forma óptima a las características particulares que se presentan para este tipo de esquemas.

Los algoritmos basados en triangulaciones regulares o semirregulares parecen ser los que mejor se adaptan a este esquema. Su regularidad permite la división de las mallas del terreno en cuadrantes que pueden organizarse en una pirámide multirresolución de cuadrantes. Una vez construida la pirámide, se puede realizar una transmisión progresiva e independiente de distintas partes del terreno a diferentes resoluciones con las que construir, para cada vista, una aproximación del terreno, cosa que es complicado de conseguir con los algoritmos basados en triangulaciones irregulares.

Las triangulaciones regulares o semirregulares, además, suelen ser las más empleadas en la visualización de terrenos en general, ya que permiten obtener unas aproximaciones precisas de las superficies del terreno manteniendo unas velocidades de refresco interactivas; algo que es difícil de conseguir con las triangulaciones irregulares, debido a la complejidad que supone la generación de esas aproximaciones, y debido a las altas necesidades de memoria y almacenamiento que requieren.

Por ello, en esta tesis se va a optar por implementar para su evaluación dentro del sistema sólo algoritmos basados en triangulaciones regulares y semirregulares. En particular, los elegidos serán ROAM (en su versión 2), SOAR, NRQT y Geometry Clipmap.

A continuación se va a explicar el funcionamiento de los algoritmos elegidos; después se especificará la metodología que se va a emplear para llevar a cabo la evaluación de los mismos, y por último, se realizará una comparativa de estos algoritmos para determinar cuál de ellos es el más adecuado para el sistema a desarrollar en esta tesis.

3.1 – Algoritmos implementados

3.1.1 – ROAM

ROAM es uno de los algoritmos más populares para la representación de terrenos [36]. Este algoritmo emplea una estructura jerárquica de árbol binario de triángulos donde cada nodo representa un triángulo isósceles de la malla (Figura 2.8).

Este algoritmo modifica la resolución de cualquier parte de la malla empleando operaciones de división y de mezcla de triángulos. La operación de división de triángulos, permite dividir recursivamente cualquier triángulo de la malla en dos nuevos triángulos más pequeños, aumentando la resolución de la malla del terreno donde se realiza dicha división. La operación contraria, de mezcla, permite unir dos triángulos para obtener uno mayor, disminuyendo la resolución de la malla en la parte donde se realiza dicha unión.

La división de cada triángulo se realiza mediante la bisección de éste por su lado más largo, añadiendo un nuevo vértice en el punto medio de la hipotenusa, dando así lugar a dos nuevos triángulos isósceles (Figura 3.1).

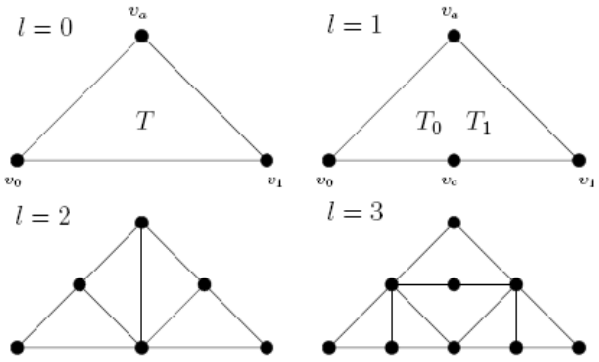


Figura 3.1 – Niveles 0-3 de una división en triángulos de un árbol binario de triángulos

Para que la operación de división no genere grietas en la superficie resultante, el triángulo a dividir (T en la Figura 3.2) y su triángulo vecino en el lado de la hipotenusa (T_B en la Figura 3.2) deben pertenecer al mismo nivel de resolución, formando una estructura de diamante (Figura 3.2). En caso contrario, será necesario dividir previamente el triángulo vecino (T_B en la Figura 3.3) antes de poder llevar a cabo la división, ya que si no, la altura del nuevo vértice surgido de la división de T (Figura 3.3) podría ser diferente a la altura interpolada del lado del triángulo vecino, creando una grieta en la superficie del terreno que podría tener un efecto visual negativo muy acentuado. Esta división se debe extender de

forma recursiva a cada uno de los nuevos vecinos del triángulo a dividir para evitar nuevas grietas en sus fronteras, lo que provoca una cascada de operaciones de división que puede incrementar de forma sustancial el coste computacional del algoritmo (Figura 3.3).

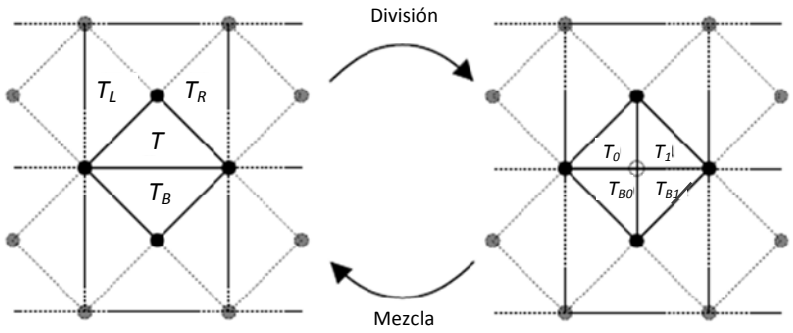


Figura 3.2 – Operaciones de división y mezcla. Relación de vecindad del triángulo T.

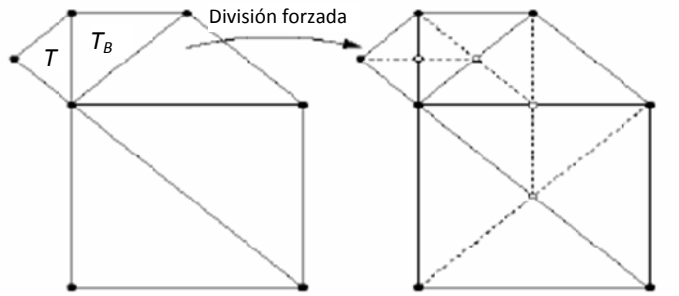


Figura 3.3 – División forzada para evitar grietas.

Para paliar el efecto negativo que tiene la cascada de operaciones en la velocidad de refresco, se limita el número de operaciones de división y mezcla llevadas a cabo en cada iteración del algoritmo. Una de las técnicas que se utiliza para ello es definir dos colas de prioridad, una para la operación de división y otra para la de mezcla, y en cada iteración, sólo se realiza la división y mezcla de los triángulos con mayor prioridad.

Este algoritmo fue originalmente diseñado para trabajar con bases de datos locales, por lo que presenta algunos problemas cuando se emplean bases de datos remotas. Cada operación de división se realiza de forma individual y sin relación, a priori, con otras divisiones

realizadas en iteraciones previas o posteriores del algoritmo. Esto hace que no se pueda predecir de una forma exacta las necesidades futuras de información a partir de las operaciones de división que se están realizando en ese instante y que, por lo tanto, no se pueda adelantar la descarga de la nueva información asociada a estas divisiones, sino que ésta debe realizarse para cada nuevo vértice de forma individualizada.

ROAM 2

ROAM 2 [37] es una implementación de código libre mejorada del algoritmo ROAM que actualmente se encuentra en desarrollo. Esta implementación tiene como objetivo aprovechar mejor las innovaciones del hardware gráfico que han aparecido desde la definición inicial del algoritmo ROAM, para aumentar su eficiencia respecto a éste.

Los objetivos iniciales que se plantearon en el nuevo algoritmo de ROAM 2 para mejorar las prestaciones del algoritmo original fueron:

1) Emplear diamantes en lugar de triángulos como estructura de datos principal.
2) Emplear técnicas de vertex arrays en vez de triángulos simples.
3) Aplicar un mecanismo de división-unión a las texturas, en lugar de clipmapping.
4) Utilizar técnicas de paginado y transmisión progresiva para la geometría y las texturas del terreno.
5) Emplear iluminación por píxel.
6) Generar geometría de forma procedural.
7) Proporcionar recorte de la vista de la escena por omisión.
8) Emplear compresión wavelet.

Hay que decir que el código disponible en la red del algoritmo ROAM2 es muy básico y sólo incluye la implementación completa de la primera parte, mientras que el resto de partes están implementadas parcialmente o de forma poco eficiente.

Implementación Evaluada

Se ha decido utilizar para su evaluación la implementación del algoritmo de ROAM en su versión más actual (ROAM 2). Para ello, se ha empleado como base el código disponible en la página oficial del algoritmo [37]. Se ha modificado ligeramente esta implementación para que el conjunto de triángulos generados por este algoritmo se almacene en un vertex array que permita realizar una transmisión más rápida y eficiente de los datos a la tarjeta gráfica, así como una visualización posterior más rápida.

3.1.2 – SOAR

Este algoritmo desarrollado por Lindstrom y Pascucci emplea, al igual que ROAM, la técnica de bisección por el lado más largo a la hora de dividir los triángulos de la malla, aunque a diferencia de ROAM, no utiliza colas de prioridad, sino que realiza el refinamiento del terreno utilizando exclusivamente un recorrido descendente de la estructura jerárquica que almacena la información de los triángulos [82][83] (Figura 3.4).

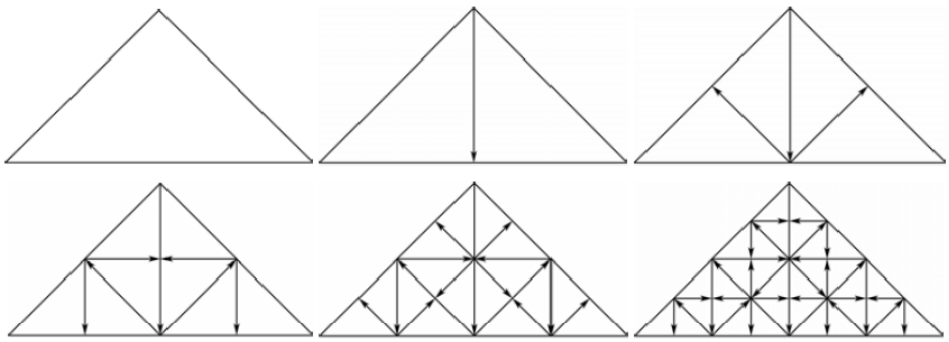


Figura 3.4 – Refinamiento de triángulos en SOAR

Esta estructura jerárquica se denomina grafo acíclico dirigido (DAG: Directed Acyclic Graph) donde cada nodo tiene asociado de cero a cuatro hijos y de cero a dos padres. Una arista dirigida en el DAG representa una arista de bisección desde el ápice del triángulo hasta el centro de la hipotenusa (Figura 3.5). La relación establecida entre padres e hijos permite evitar la aparición de huecos o grietas en las aproximaciones realizadas forzando la inclusión de los vértices de los nodos padres antes de insertar los vértices de los nodos hijos.

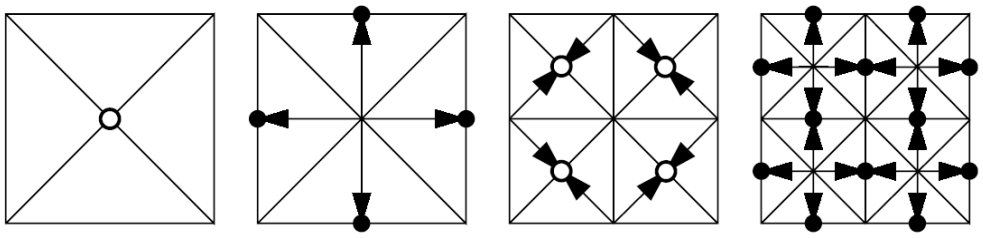


Figura 3.5 – División de los primeros niveles del DAG.

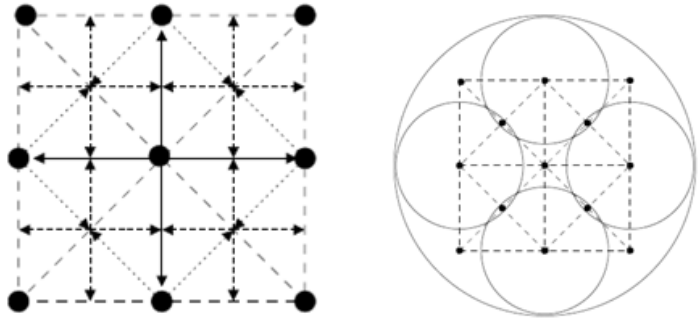


Figura 3.6 – DAG y Jerarquía de esferas envolventes

Además de la información de las coordenadas de cada vértice, un nodo en el DAG contiene el valor del error asociado a cada vértice y una esfera que envuelve espacialmente ese vértice.

El error asociado a cada vértice permite seleccionar el nivel de detalle adecuado para cada región del terreno. Los autores de este algoritmo explican que existen múltiples métricas que pueden emplearse para el cálculo de ese error, tanto métricas del error definidas en el espacio del objeto, como métricas del error definidas en el espacio de la pantalla. Para seleccionar el nivel de detalle adecuado se recorre el DAG de forma recursiva partiendo del nodo raíz. Durante el recorrido del grafo, se realiza una proyección dependiente del punto de vista del error asociado a cada vértice, que ha sido calculado en el espacio del objeto, obteniendo el valor del error para cada vértice en el espacio de la pantalla. Ese error se compara con un valor umbral predeterminado, y si este error es menor que ese umbral, el recorrido del grafo termina en ese vértice.

Las esferas envolventes contenidas en cada nodo del DAG permiten realizar el recorte de vista de la escena. La esfera envolvente de cada padre en el DAG es lo suficientemente grande para cubrir las esferas envolventes de sus hijos (Figura 3.6), de forma que, cuando se detecta que la esfera envolvente de un nodo cae fuera del campo de visión del observador, no es necesario procesar ni ese nodo, ni sus respectivos hijos.

Una característica importante de SOAR es que el empleo del DAG y de la jerarquía de esferas envolventes permite realizar en un solo recorrido de este DAG, tanto el proceso de refinamiento de la superficie de terreno, como el proceso de triangulación y el de recorte de la vista.

Otra característica interesante es que la triangulación se puede representar con una única tira de triángulos, lo cual es idónea para aprovechar las nuevas prestaciones de las tarjetas gráficas actuales, al facilitar el uso de vertex arrays para almacenar esas tiras y enviarlas a la GPU en el menor tiempo posible.

Al igual que en el caso de ROAM, este algoritmo fue diseñado originalmente para trabajar con bases de datos locales, por lo que no está, a priori, preparado para trabajar con aplicaciones con bases de datos remotas debido, entre otras cosas, a que la descarga de la información de cada nuevo vértice se debe realizar de forma individualizada, sin posibilidad de agruparla con la información de los vértices que se necesitarán en el futuro (no posee un mecanismo para predecir cuáles serán estos vértices).

Implementación Evaluada

Se ha empleado como base de la implementación realizada, el código del algoritmo disponible en la página web oficial de sus creadores Lindstrom y Pascucci [117]. Esta implementación se ha adaptado para almacenar en un vertex array la tira de triángulos producida por este algoritmo, aprovechando los beneficios asociados al uso de esta estructura en las nuevas tarjetas gráficas.

3.1.3 – Geometry Clipmap

El algoritmo Geometry Clipmap [84][3] fue diseñado para utilizar, de forma óptima, las prestaciones de las nuevas tarjetas gráficas (como los vertex buffers y vertex shaders), y elaborar una triangulación regular y eficiente con la que llevar a cabo la visualización de la superficie del terreno en tiempo real. Este algoritmo trata de realizar el mínimo número de operaciones posibles en la CPU del ordenador, transfiriendo la mayor parte de éstas a los procesadores gráficos de las tarjetas gráficas (GPU: Graphics Processing Unit).

El algoritmo Geometry Clipmap construye una pirámide de niveles de detalle donde la base se corresponde con la malla de alturas a máxima resolución, y donde cada nivel superior está formado por una malla simplificada con la mitad de valores de altura que su nivel anterior. En la Figura 3.7 se puede observar dicha pirámide de niveles de detalle. La selección del nivel de detalle adecuado para cada región depende exclusivamente de la distancia de esa región al punto de vista del observador, sin emplear ninguna métrica del error dependiente de la orografía del terreno.

Para la representación del terreno se define, para cada nivel de detalle de la pirámide, una región rectangular centrada en la posición del punto de vista. Esta región se conoce como región activa. El centro de cada región activa, menos el del nivel de máxima resolución, se encuentra vacío, ya que este hueco se rellena con la información de los niveles de mayor resolución. (Figura 3.7 - Derecha).

Una restricción importante de este algoritmo es que el tamaño de la región activa ha de ser potencia de dos, lo que limita la cantidad de niveles de detalle que se pueden combinar para construir el terreno virtual. Por lo tanto, el tamaño de cada una de las regiones activas disminuye a la mitad a medida que aumenta el nivel de detalle de la pirámide.

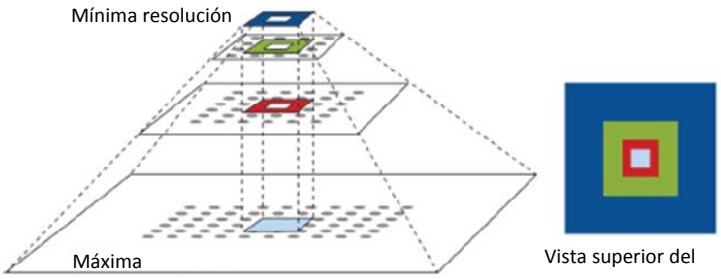


Figura 3.7 – Niveles de Geometry Clipmap

Estas regiones activas se triangulan de forma independiente, generando una tira de triángulos para cada nivel de resolución (Figura 3.8 y Figura 3.9). Al realizarse esta triangulación de forma independiente, pueden aparecer huecos o grietas entre los distintos niveles de resolución. Para evitar la aparición de estos huecos, Losasso and Hoppe realizan un morphing de los vértices próximos al borde, para lo que emplean otra de las prestaciones de las nuevas tarjetas gráficas, los vertex shaders, que son pequeños programas que se almacenan en la memoria de las tarjetas gráficas que permiten realizar operaciones sobre cada vértice dibujado de forma muy rápida. En este caso, los vertex shaders llevan a cabo un proceso de morphing combinando las alturas de los dos niveles de resolución adyacentes dentro del área que se denomina región de transición.

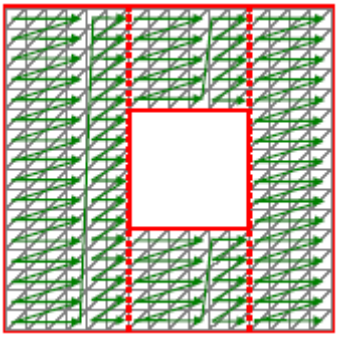


Figura 3.8 – Triangulación de un nivel de detalle de Geometry Clipmap

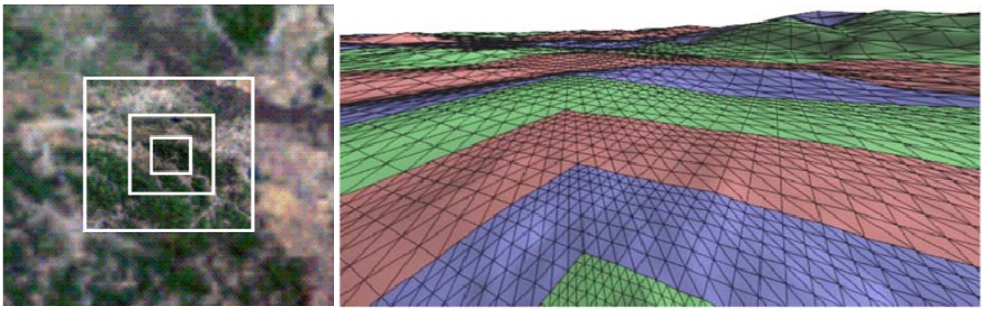


Figura 3.9 – Niveles de detalle y triangulación Geometry Clipmap

Como se ha comentado, este algoritmo hace uso de las nuevas prestaciones de las tarjetas gráficas actuales, de forma que emplea vertex buffers para almacenar los vértices de las regiones activas en la memoria de la propia tarjeta gráfica. Como el algoritmo emplea mallas regulares, se pueden generar fácilmente tiras de triángulos que son representadas de forma muy eficiente.

Para modificar la información contenida en la memoria de la tarjeta gráfica, resulta muy costoso sobreescribir por completo estos vertex buffers. Por lo tanto, los autores implementan un mecanismo que permite modificar sólo los datos que es necesario actualizar cuando el punto de vista del observador cambia. Para realizar esta operación de forma eficiente, los vértices se almacenan en un array toroidal que permite actualizaciones progresivas, añadiendo a la estructura sólo los nuevos vértices que aparecen en escena, sobreescribiendo los valores anteriores que estaban almacenados en esa posición (Figura 3.10).

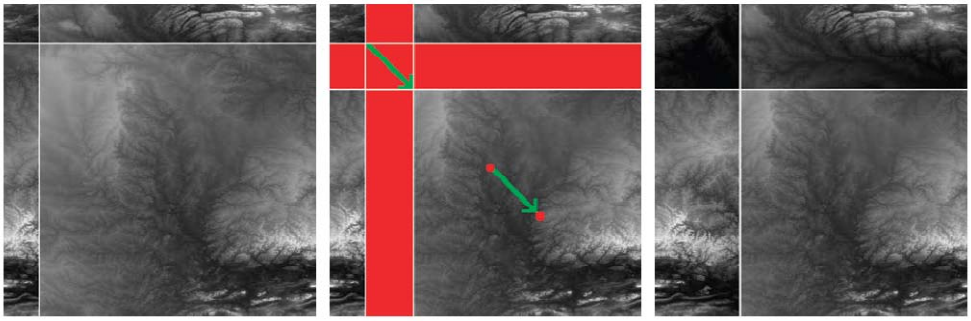


Figura 3.10 – Actualización del array toroidal ante un desplazamiento del punto de vista.

En principio, este algoritmo requiere que toda la malla de alturas de máxima resolución permanezca en la memoria del ordenador. Como este tamaño puede ser muy grande, se puede emplear un esquema de compresión con pérdidas para reducir su tamaño; concretamente en este algoritmo se especifica el uso del codificador PTC [87].

Para llevar a cabo la compresión se realiza el proceso que se describe a continuación: partiendo del nivel de detalle de menor resolución de la pirámide, se predice un nuevo valor de altura interpolado entre cada par de valores de altura de ese nivel de resolución, formando un nuevo nivel de detalle de mayor resolución. La diferencia entre ese valor de altura interpolado y el valor real de altura que se corresponde con ese nivel de detalle de mayor resolución se almacena en lo que se denomina un residuo, el cual se codifica empleando PTC. Este proceso se repite para cada nivel de resolución de la pirámide. Siguiendo este proceso, sólo es necesario almacenar en memoria la malla de alturas del nivel de detalle de menor resolución del terreno y el conjunto de residuos comprimidos.

En el proceso de visualización del terreno, para aumentar el nivel de resolución de una región, se interpolan nuevos valores de alturas de la misma forma que en el proceso de compresión, obteniendo una primera aproximación de mayor resolución, que se puede refinar incorporando la información almacenada en los residuos. Se puede mantener un control de la velocidad de refresco que proporciona este algoritmo limitando el número de residuos que se procesan en cada iteración.

En este algoritmo, la resolución empleada en cada región del terreno sólo depende de la distancia de la región al punto de vista, y no de la rugosidad del terreno, por lo que el mejor funcionamiento de este algoritmo se producirá sólo cuando la superficie del terreno tenga una rugosidad uniforme. Sin embargo, generalmente la superficie del terreno suele constar de regiones con un variado grado de rugosidad. En estas situaciones, este algoritmo se encuentra con la dificultad de alcanzar un compromiso entre la calidad de las

aproximaciones y la velocidad de refresco. Si se opta por favorecer la calidad de las aproximaciones, se dibujarán regiones poco rugosas con un gran número de triángulos que no aportarán mayor calidad visual a la aproximación, y en cambio, sí que penalizarán la velocidad de refresco en el proceso de visualización de la escena. Si en cambio se opta por favorecer la velocidad de refresco, se dibujarán regiones que tienen un alto grado de rugosidad con un número reducido de triángulos, proporcionando una calidad visual insuficiente.

Este algoritmo está diseñado para funcionar correctamente con terrenos poco extensos donde toda la información está almacenada en la memoria principal. Para adaptarlo a aplicaciones que utilizan bases de datos remotas de mayor tamaño serían necesarias varias modificaciones, como, por ejemplo, incorporar al algoritmo una estructura que permita conocer qué información tiene disponible en la memoria y cuál se debe descargar de la base de datos remota. Además, se debe decidir cómo se empaquetará esta información que se recibe de forma remota, por ejemplo, se pueden empaquetar los residuos codificados con PTC y almacenarlos en una base de datos remota para ser descargados progresivamente, y finalmente, sería necesario implementar los mecanismos necesarios para procesar estos residuos e incorporarlos a la nueva estructura. Este conjunto de tareas es previsible que aumenten considerablemente la complejidad y el coste computacional del algoritmo.

Implementación Realizada

De este algoritmo no hay disponible en la red ninguna implementación oficial, por ello se ha tenido que elaborar una completamente nueva siguiendo las directrices descritas por Lossaso y Hoppe en [84]. Se han incorporado a esta implementación todas las características descritas en el algoritmo original excepto la compresión, entre otras cosas porque la compresión empleada por este algoritmo es propietaria y cerrada, y no existe ninguna implementación disponible de la misma. Además, la definición original del resto de algoritmos evaluados no emplea ningún tipo compresión, y se desea comparar todos ellos bajo las mismas condiciones.

Los triángulos en este algoritmo, a diferencia de las implementaciones hechas para ROAM y SOAR, se almacenan en vertex buffers. La diferencia principal con los vertex arrays es que los vertex buffers se almacenan en la memoria de la tarjeta gráfica, mientras que los vertex arrays lo hacen en la propia memoria del ordenador. Sin embargo, en la actualidad, debido al aumento de la velocidad de los buses de comunicación de las tarjetas gráficas con la placa base, y al uso cada vez más habitual de tarjetas gráficas integradas en la placa base que emplean de forma compartida la memoria del ordenador, la diferencia en eficiencia al usar una estructura u otra es prácticamente nula.

3.1.4 – NRQT

En este apartado se va a describir el algoritmo NRQT (None Restricted Quadtree Triangulation) [93]. Este algoritmo es un modelo de triangulación basado en quadtrees no restrictivos, que obtiene versiones simplificadas de las mallas del terreno de la misma o mayor calidad que otros modelos basados en quadtrees restrictivos (RQT: Restricted Quadtree Triangulation) pero utilizando un menor número de triángulos.

El descenso en el número de triángulos se consigue gracias a la supresión de las restricciones que imponen los RQT, los cuales añaden una importante cantidad de triángulos para evitar la aparición de grietas entre regiones vecinas sin que ello mejore la calidad visual del terreno (Figura 3.11).

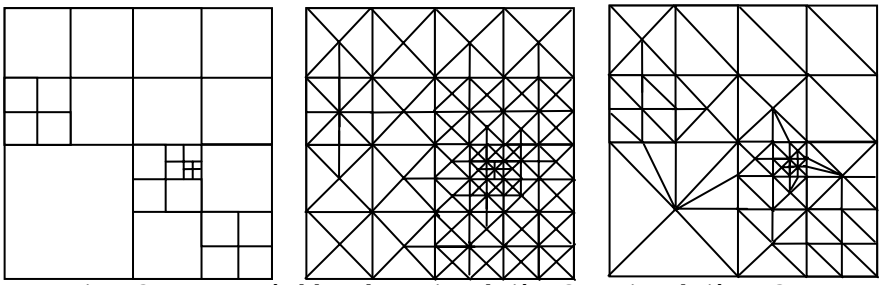


Figura 3.11 – Jerarquía del quadtree, triangulación RQT y triangulación NRQT.

El algoritmo NRQT se basa en una división jerárquica, cuaternaria y no restrictiva del espacio en regiones, guiada por una métrica dependiente del punto de vista y de la rugosidad de la superficie.

Generación del Quadtree

La división espacial se genera siguiendo una estrategia descendente. El proceso comienza en el nodo raíz y añade nuevos nodos de forma recursiva. El proceso está dirigido por una métrica del error y un algoritmo de recorte de la vista. Los nodos hoja del quadtree resultante representan la superficie del terreno simplificada.

Triangulación

Una vez que se ha construido el quadtree, las regiones asociadas a los nodos hoja que representan la superficie simplificada del terreno se dividen en triángulos. Es posible realizar las tareas de generación del quadtree y de elaboración de la triangulación de forma óptima y simultánea. Para ello, se crea un conjunto de listas enlazadas de vértices en las que los vértices se insertan ordenados espacialmente, reproduciendo sus relaciones de vecindad

(Figura 3.12). En principio, el coste de insertar un nodo en una lista ordenada es lineal O(N), sin embargo, se puede reducir este coste a constante O(1) si se recorre el quadtree en un orden determinado, con el que los vértices se insertan siempre al inicio de la lista de forma ordenada.

Grietas

NRQT emplea una técnica diferente a la de los algoritmos RQT para evitar la aparición de grietas entre regiones vecinas de diferente resolución. Esta técnica permite dividir, de forma independiente, cada región en triángulos, sin provocar la división de otras regiones vecinas, tal como suele ocurrir en los algoritmos RQT.

Se pueden diferencias dos tipos de regiones:

1. Regiones sin vecinos de mayor resolución.
2. Regiones con vecinos de mayor resolución.

La división en triángulos para cada tipo de región es diferente. Las regiones del primer tipo no presentan problemas de grietas y por ello se dividen tan sólo en dos triángulos (Figura 3.12). En cambio las regiones del segundo tipo sí que pueden presentar grietas, por lo tanto estas regiones se dividen en varios triángulos, conectando todos los vértices situados sobre sus fronteras con un nuevo vértice añadido en el centro de la región. El número de triángulos en los que se dividen este tipo de regiones es siempre mayor al de los del primer tipo, pero gracias a esta división se asegura que la superficie final está libre de grietas (Figura 3.12).

Al final, se obtiene un conjunto de triángulos que se agrupan formando tiras y abanicos, para posteriormente, con la ayuda de nuevos triángulos degenerados, construir una sola tira de triángulos que se almacena en un vertex array. El contenido de este vertex array se envía a la tarjeta gráfica de una sola vez, limitando al máximo el tiempo de transferencia y procesamiento necesarios.

El uso de triángulos degenerados no supone ningún problema a la hora de realizar la representación y visualización de la escena, debido a que el hardware gráfico se encarga de eliminarlos automáticamente y de forma muy rápida. De hecho, la agrupación de múltiples tiras de triángulos en una sola de tamaño superior es un mecanismo de optimización recomendado habitualmente para aumentar el rendimiento gráfico [13][121] .

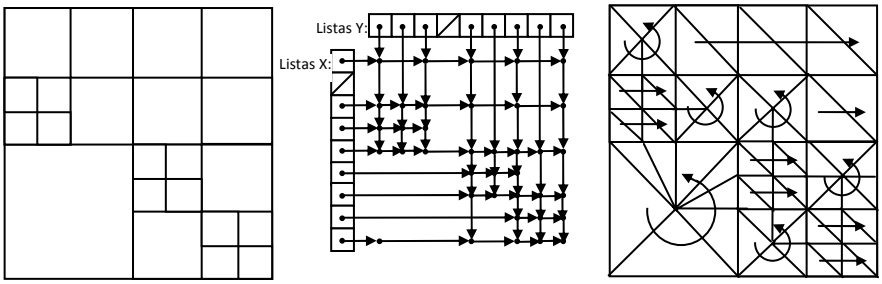


Figura 3.12 – Jerarquía del quadtree, listas enlazadas y triangulación en tiras y abanicos.

Métrica del Error

Como se ha comentado previamente, para realizar el refinamiento descendente de la malla del terreno, se define una métrica del error en el espacio del objeto independiente del punto de vista y, posteriormente, se adapta dicha métrica al espacio de la pantalla. El cómputo de la primera métrica puede hacerse en tiempo de preproceso o una vez cargados los datos del cuadrante, mientras que la adaptación al punto de vista se realiza en tiempo de ejecución para cada frame.

Esta métrica del error (Ecuación 3.1) es similar a la definida en ROAM[36], y se basa en el tamaño de una cuña rectangular que engloba a la superficie de una región. El error ε asociado a cada región del quadtree se define: en función del error de los cuatro hijos en los que se divide en el siguiente nivel de detalle (ε_{q1}, ε_{q2}, ε_{q3}, ε_{q4}), en función de la diferencia de altura entre los nuevos vértices que dividen los lados de la región y los lados sin dividir (h_{l1}, h_{l2}, h_{l3}, h_{l4}), y en función de la diferencia de alturas entre el nuevo vértice aparecido en el centro de la región y las dos diagonales del mismo (h_{d1}, h_{d2}).

$$\varepsilon = \max\{\varepsilon q1, \varepsilon q2, \varepsilon q3, \varepsilon q4\} + \max\{hl1, hl2, hl3, hl4, hd1, hd2\}$$

Ecuación 3.1 – Métrica del error empleada en NRQT

El cómputo de estos valores se realiza siguiendo una estrategia ascendente, comenzando por el nivel de detalle de mayor resolución.

Una vez realizada la adaptación de la métrica del error asociada al objeto, se llevará a cabo la adaptación del error asociada al punto de vista. Existen dos posibles adaptaciones de la métrica al espacio de la pantalla [83]: a través de una proyección del error isotrópica, donde sólo se tiene en cuenta la distancia del polígono al punto de vista; o a través de una proyección del error anisotrópica, donde se tiene en cuenta, además, el ángulo relativo que forman dicho polígono con la línea de la cámara virtual. La expresión de la proyección isotrópica de la métrica es la siguiente:

$$\rho(\epsilon) = \lambda \frac{\epsilon}{d}$$

Ecuación 3.2 – Proyección isotrópica

Siendo "λ" un valor constante y siendo "*d*" la distancia del polígono al punto de vista.

La ecuación de la proyección isotrópica es bastante más simple y eficiente que la de la proyección anisotrópica. Además, la reducción de la complejidad de la malla conseguida con esta última es poco significativa frente a la primera [59][81][83].

Grietas entre cuadrantes

La división de las bases de datos de terrenos en cuadrantes, hace que puedan aparecer nuevas grietas entre las fronteras de esos cuadrantes. Dado que la carga de los cuadrantes se realiza de forma independiente, la relación entre los datos de los diferentes cuadrantes debe ser lo mínima posible. Esto dificulta la creación de una métrica monótona a lo largo de varios cuadrantes [6][83], introduciendo una complejidad extra a la hora de evaluar las dependencias entre los vértices vecinos que pertenecen a cuadrantes diferentes [36][97]. Generalmente, en otros algoritmos de triangulación, la determinación de esas dependencias es clave para evitar la aparición de dichas grietas (Figura 3.13). Sin embargo, NRQT evita la aparición de grietas de forma sencilla, sin la necesidad de dependencias adicionales, simplemente fusionando las listas enlazadas de vértices correspondientes a las fronteras comunes entre cuadrantes vecinos.

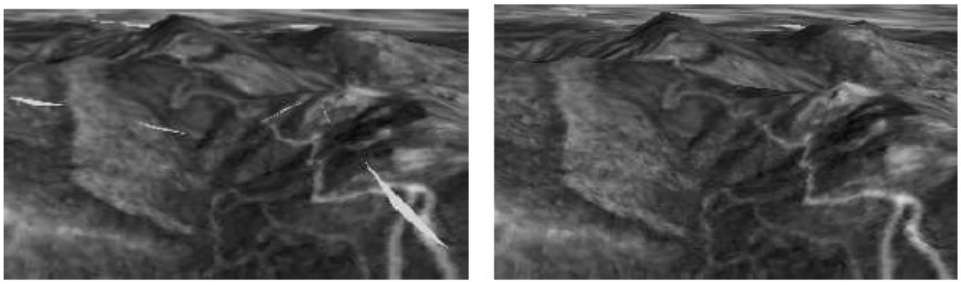


Figura 3.13 – Izquierda: grietas entre cuadrantes vecinos de distinta resolución. Derecha: grietas corregidas empleando la triangulación NRQT.

Coherencia de frame

El algoritmo NRQT permite emplear coherencia de frame. La coherencia de frame es un mecanismo que permite reutilizar la información calculada en frames previos en el frame actual. En el caso de la visualización de terrenos, permite evitar el cálculo de la triangulación de toda la escena en cada frame, generando de nuevo únicamente la parte de

la escena que ha cambiado. Durante la fase de creación del quadtree se almacena la información necesaria para poder llevar a cabo la coherencia de frame de forma eficiente.

Bases de datos remotas

Como se ha comentado previamente, el conjunto de algoritmos de triangulación estudiados se basa en el uso de bases de datos locales. El algoritmo NRQT es independiente de la ubicación de estas bases de datos. Se puede emplear tanto en aplicaciones que hacen uso de bases de datos locales, como en aquellas que emplean bases de datos remotas. Las bases de datos sólo tienen que ser capaces de enviar información relativa a un nodo del quadtree de forma independiente.

Representación planetaria

En [95] se extiende este algoritmo para permitir la representación de superficies del terreno planetarias de forma eficiente. Para ello se emplea una estructura cúbica que permite representar toda la superficie del planeta (Figura 3.14).

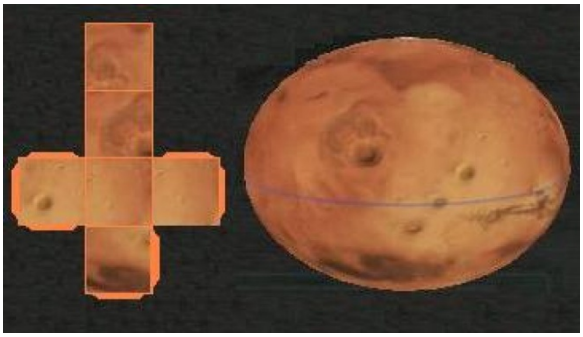


Figura 3.14 – Representación planetaria con NRQT

Implementación Realizada

Respecto a la implementación, se han seguido las directrices e incorporado todos los elementos expuestos en [93] y [95] . Además, se han almacenado los triángulos de cada cuadrante producidos por este algoritmo en una sola tira de triángulos, usando triángulos degenerados para agrupar el conjunto de tiras y abanicos que se obtienen durante el proceso de la triangulación, en una sola tira de triángulos que se almacena en un vertex array, de forma similar a como se hace en los algoritmos vistos previamente.

Se ha realizado la implementación manteniendo la independencia de la ubicación de las bases de datos que incorpora inherentemente este algoritmo, de forma que se podrán

emplear tanto bases de datos locales como remotas, sin necesidad de modificar el código del mismo.

3.2 – Metodología de Evaluación

Para realizar la comparativa de los diferentes algoritmos, se ha implementado una aplicación de prueba desarrollada utilizando el lenguaje de programación C++ y la librería gráfica OpenGL. Esta aplicación emplea la arquitectura típica de una aplicación de visualización de terrenos en tiempo real aunque simplificada. La simplificación que se ha realizado es la eliminación del módulo de transmisión y carga progresiva de la información, debido a que en este apartado nos vamos a centrar en los algoritmos de representación de los datos del terreno, independientemente de la ubicación y de la transmisión de esos datos. De esta forma, se evitará que los resultados obtenidos se vean afectados por problemas en la carga o en la transmisión de la información. Por ello, se va a cargar toda la información del terreno en la memoria.

En la Figura 3.15 se muestra el esquema del sistema de prueba:

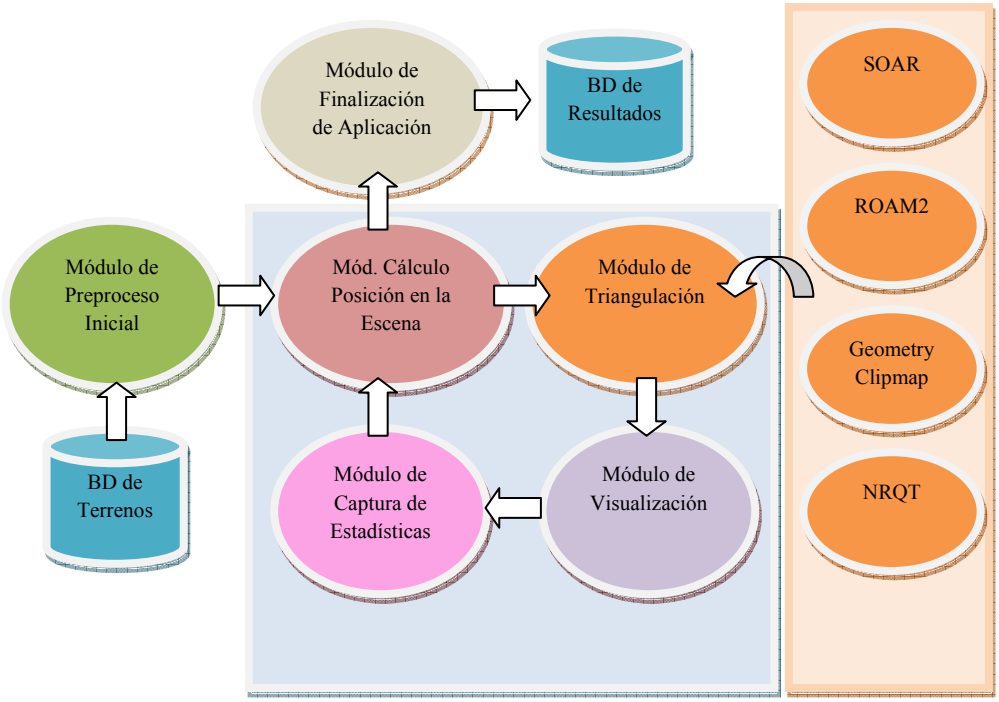


Figura 3.15 – Módulos y esquema de funcionamiento del Sistema de Prueba

Fuera del ciclo de dibujo, antes de iniciarse éste, se ejecuta el siguiente módulo:

Módulo de preproceso inicial: este módulo se encarga de realizar todo los cálculos de preproceso necesarios para el funcionamiento de los diferentes algoritmos de triangulación. También realiza la carga de los datos del terreno en la memoria del sistema para eliminar la influencia que pudiera tener el acceso a disco en la ejecución de los algoritmos mientras se obtienen los resultados de las pruebas.

Dentro del "Ciclo de dibujado y captura de estadísticas", se ejecutan los módulos que van a llevar a cabo la generación y la visualización de la superficie del terreno, así como la obtención de las estadísticas que servirán para evaluar estos algoritmos. Este ciclo se repetirá a lo largo de todo el proceso de visualización del terreno.

Este ciclo se compone de cuatro módulos:

- Módulo de cálculo de la posición en la escena: realiza el cálculo de la posición actual del punto de vista del usuario en la escena, siguiendo un conjunto de rutas precalculadas que serán empleadas para los cuatro algoritmos de triangulación. Cuando se llegue al final de cada ruta, concluirá el ciclo de dibujado y captura de estadísticas y se ejecutará el módulo de finalización de la aplicación.

- Módulo de Triangulación: a partir de la posición actual del usuario en la escena se construye una triangulación de la superficie del terreno adaptada al punto de vista. Para ello, se ejecuta el módulo correspondiente al algoritmo de triangulación que se desea evaluar (ROAM2, SOAR, Geometry Clipmap o NRQT). Estos módulos se han construido de forma independiente al resto del sistema de prueba, de forma que se puede garantizar que las condiciones de funcionamiento para todos los algoritmos son las mismas.

- Módulo de Visualización: lleva a cabo la visualización de la superficie del terreno empleando las funciones típicas de dibujo de la librería gráfica OpenGL.

- Módulo de Captura de Estadísticas: se encarga de calcular y almacenar en memoria los datos con los que se generan las estadísticas que nos servirán para comparar los algoritmos.

Una vez finalizada la trayectoria programada, se ejecuta el siguiente módulo:

Módulo de Finalización de la Aplicación: es el último módulo que se ejecuta dentro del sistema, y se encuentra fuera del ciclo de visualización y captura de estadísticas. Este módulo guarda en la base de datos los resultados obtenidos por el módulo de captura de estadísticas.

Datos Empleados

Para las pruebas se han empleado los datos de altura y las texturas del modelo de terreno virtual "Puget Sound" [105]. Estos datos han sido utilizados en múltiples trabajos para evaluar los algoritmos de triangulación implementados. Así Lindstrom y Pascucci en [82] los emplean para probar el algoritmo de SOAR, Hoppe et al. en [3][84] los emplean para probar el algoritmo de Geometry Clipmap, y Olanda et al. en [93] los emplean para probar el algoritmo NRQT. Estos datos proporcionan una superficie del terreno con zonas de variada rugosidad, donde se entremezclan zonas planas con regiones montañosas, así como cauces de ríos.

El modelo del terreno está disponible en varias resoluciones. Se ha elegido para las pruebas los ficheros de alturas de tamaño 4097 x 4097 y los ficheros de texturas de tamaño 4096 x 4096, ya que proporciona una extensión real de terreno de 163840 x 163840 metros, que es suficiente extensa y variada para llevar a cabo las pruebas. En estos datos, la resolución de cada valor de altura es de 16 bits (de 0 a 65535), lo que se corresponde con una precisión de 0.1 metros, siendo la separación de cada una de las muestras de 40 metros. La resolución de las texturas empleadas es de 40 metros / pixel.

Métrica Empleada

El objetivo de la prueba es realizar una comparativa entre los diferentes algoritmos de triangulación, para lo cual, se va a medir la calidad visual de la simplificación del terreno y la velocidad de refresco obtenida con cada algoritmo.

Para evaluar la calidad visual se va a capturar cada uno de los fotogramas generados en los recorridos precalculados empleando cada uno de los algoritmos de triangulación. Posteriormente, se calcularán las diferencias entre estos fotogramas y los obtenidos al realizar el recorrido sin emplear ningún tipo de simplificación, es decir, empleando la malla de alturas a máxima resolución. Las diferencias entre las imágenes de los fotogramas se medirán empleando el error cuadrático medio (RMSE).

Prueba

Se han precalculado un conjunto de recorridos generados de forma semi-aleatoria sobre la representación virtual del terreno, que transcurren a lo largo de regiones con diferente grado de rugosidad. Cada uno de los recorridos se repetirá para todos los algoritmos de triangulación.

Se van a realizar tres pruebas donde varía el tipo de rugosidad de la superficie del terreno por la que transcurren las rutas precalculadas (Figura 3.16), con el objetivo de ver cómo se comporta cada algoritmo en los diferentes tipos de superficies existentes habitualmente en las bases de datos de terrenos:

- Prueba 1: Rugosidad baja. El recorrido precalculado transcurrirá sobre superficies de terreno generalmente planas como mesetas o praderas. En esta prueba se desea evaluar el comportamiento de los diferentes algoritmos cuando tienen que elaborar simplificaciones de las superficies del terreno que no necesitan de una gran cantidad de triángulos para ser representadas fielmente.

- Prueba 2: Rugosidad alta. El recorrido precalculado transcurrirá por superficies del terreno muy irregulares, que se corresponden con regiones montañosas. En esta prueba se desea evaluar el comportamiento de los diferentes algoritmos cuando tienen que elaborar simplificaciones de las superficies del terreno que necesitan de un gran número de triángulos para ser representadas fielmente.

- Prueba 3: Rugosidad variada. El recorrido precalculado pasará por superficies del terreno en los que habrá una rugosidad baja, como en las regiones de la prueba 1, pero también transcurrirá por superficies en los que la rugosidad será elevada, como en las regiones montañosas. En esta prueba se desea evaluar los diferentes algoritmos en el caso habitual de la representación de terrenos, en los que los usuarios atraviesan un conjunto heterogéneo de superficies del terreno. Las simplificaciones que deben generar los algoritmos a lo largo del recorrido tendrán un diferente número de triángulos según la rugosidad de la superficie.

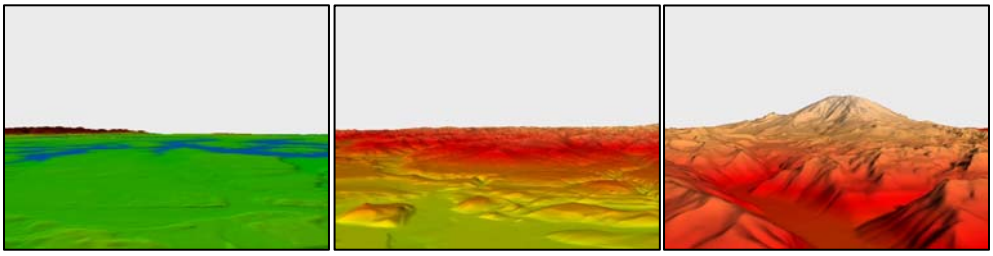


Figura 3.16 – Diferentes recorridos: izquierda: rugosidad baja, centro: rugosidad variada, derecha: rugosidad alta

Para evaluar los algoritmos de triangulación se han medido y almacenado los parámetros habitualmente usados en este tipo de pruebas [82][84][93] :

- La velocidad de refresco que es capaz de obtener cada algoritmo, que permite conocer la rapidez con la que se generan y visualizan las simplificaciones de las superficies del terreno en pantalla.

- La calidad de las simplificaciones de las superficies del terreno (a través de la medición del RMSE en cada fotograma), que permite conocer la diferencia de calidad de las simplificaciones respecto a la superficie de terreno original a máxima resolución.

- El número de triángulos empleado en cada fotograma, que permite conocer la cantidad de datos que emplea cada algoritmo para llevar a cabo las simplificaciones de la superficie del terreno.

Los algoritmos utilizados han sido: NRQT, SOAR, ROAM2 y Geometry Clipmaps.

Las pruebas han sido realizadas en un Pentium 4 a 3 Ghz, con 1GB de memoria RAM, y una tarjeta gráfica Nvidia 7900 de 256 MB de memoria.

3.3 – Resultados

A continuación se van a estudiar los resultados obtenidos en las pruebas, evaluando los algoritmos NRQT, SOAR, ROAM2 y Geometry Clipmap.

3.3.1 – RMSE (Root Mean Square Error)

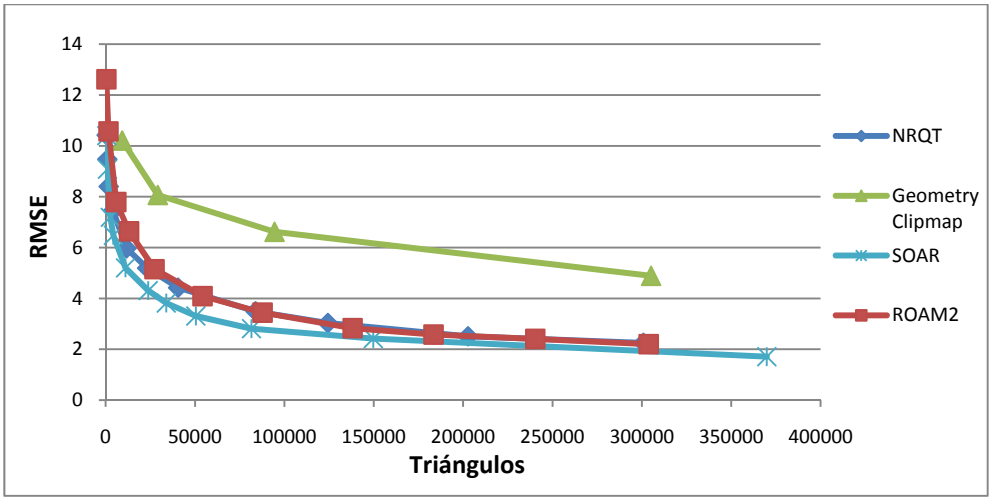
Primero se va a comparar la calidad visual de las simplificaciones del terreno generadas por cada uno de los algoritmos a lo largo de los recorridos precalculados del mundo virtual.

Como se ha comentado en el apartado anterior, la calidad visual se evalúa midiendo las diferencias visuales entre los fotogramas generados empleando la malla a máxima resolución, y los fotogramas generados tras activar las simplificaciones obtenidas con cada uno de los algoritmos de triangulación. Estas diferencias se han medido empleando el error cuadrático medio.

A la hora de comparar los algoritmos de triangulación, hay que tener en cuenta que cada algoritmo genera, a priori, cada uno de estos fotogramas con un número de triángulos diferente. Para poder realizar una comparativa de estos algoritmos en condiciones similares, y como no es posible fijar el número de triángulos empleados por cada algoritmo en cada fotograma, se ha procedido a calcular el número medio de triángulos empleado a lo largo de todo el recorrido precalculado. Cada recorrido se repite en varias ocasiones, ajustando los parámetros que afectan a la calidad visual de las simplificaciones generadas por cada algoritmo, obteniendo un conjunto diverso de simplificaciones con diferentes valores de RMSE y medias de triángulos empleados en cada una de esas repeticiones.

Esta información se emplea en el conjunto de las gráficas que se presentan a continuación, para mostrar la relación entre el valor del RMSE medio para cada fotograma del recorrido y el número de triángulos medio empleado en la generación de cada fotograma.

Prueba 1 – Rugosidad Baja



Gráfica 3.1 – RMSE – Prueba 1 – Rugosidad Baja

	NRQT		Geometry Clipmap		SOAR		ROAM2	
Prueba	**Triángulos**	**RMSE**	**Triángulos**	**RMSE**	**Triángulos**	**RMSE**	**Triángulos**	**RMSE**
1	562	10,42	9217	10,21	654	10,4	424	12,62
2	964	9,47	29167	8,07	997	9,08	1590	10,57
3	1704	8,40	94513	6,62	2594	7,19	5916	7,79
4	4138	7,45	305157	4,89	4318	6,48	13096	6,64
5	11675	5,97			11089	5,20	27355	5,15
6	23256	5,19			23834	4,31	54203	4,09
7	40544	4,42			33933	3,82	87803	3,44
8	83841	3,49			50466	3,31	138165	2,83
9	124319	3,03			81565	2,81	183472	2,58
10	202705	2,51			149769	2,42	240348	2,41
11	300835	2,25			369907	1,71	303808	2,20

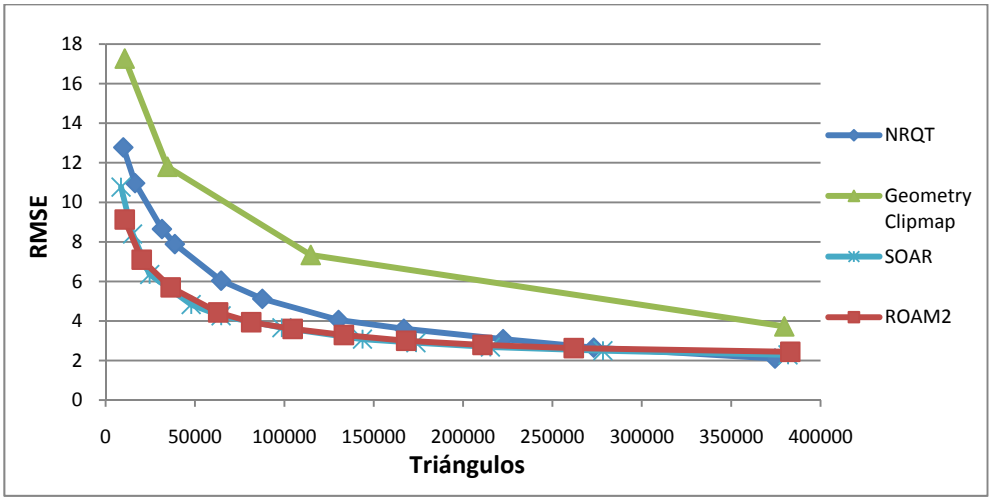
Tabla 3.1 – RMSE – Prueba 1 – Rugosidad Baja

En la Gráfica 3.1 y en la Tabla 3.1 se puede observar como el algoritmo Geometry Clipmap necesita emplear un número superior de triángulos que el resto de algoritmos para obtener la misma calidad visual en sus simplificaciones de las superficies del terreno. Además, no es capaz de generar simplificaciones que alcancen los mismos niveles de calidad que el resto de algoritmos.

También se observa como el algoritmo SOAR es el que necesita un menor número de triángulos para obtener simplificaciones de la misma calidad que los otros algoritmos.

Por último, se observa también como los algoritmos NRQT y ROAM2 consiguen realizar simplificaciones que alcanza la misma calidad que las elaboradas por el algoritmo SOAR, pero empleando un número superior de triángulos, siendo este número de triángulos similar en ambos algoritmos.

Prueba 2 – Rugosidad Alta



Gráfica 3.2 – RMSE – Prueba 2 – Rugosidad Alta

	NRQT		Geometry Clipmap		SOAR		ROAM2	
Prueba	**Triángulos**	**RMSE**	**Triángulos**	**RMSE**	**Triángulos**	**RMSE**	**Triángulos**	**RMSE**
1	9897	12,78	10684	17,27	8610	10,77	10710,1	9,13
2	16510	10,97	34599	11,80	15096	8,40	20165	7,10
3	31529	8,65	114797	7,34	24661	6,35	36369	5,70
4	38756	7,89	379719	3,73	47866	4,83	62968	4,43
5	64642	6,04			64635	4,26	81481	3,94
6	87710	5,12			98648	3,66	104678	3,60
7	130328	4,05			143735	3,08	133326	3,29
8	166873	3,61			173635	2,90	168182	3,00
9	222442	3,08			215259	2,68	210830	2,79
10	273111	2,65			278262	2,50	261954	2,63
11	374573	2,10			381797	2,29	383097	2,44

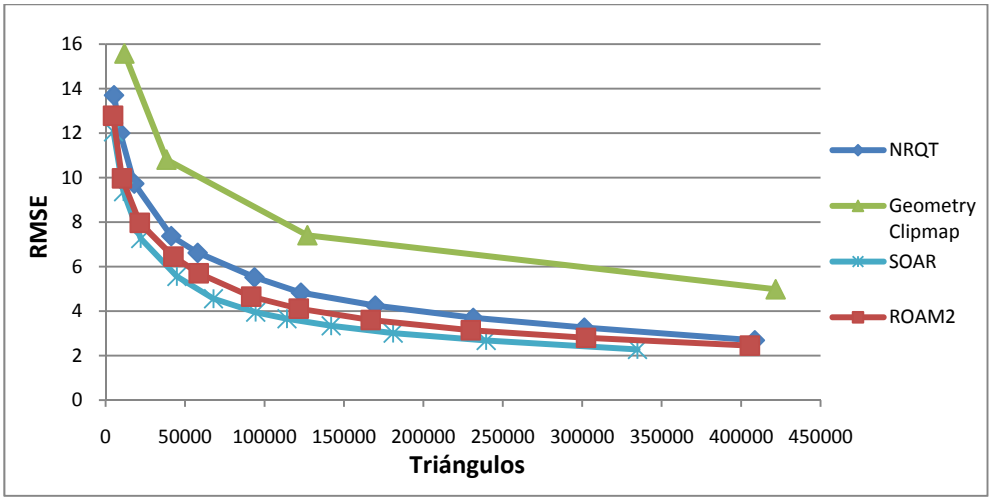
Tabla 3.2 – RMSE – Prueba 2 – Rugosidad Alta

En la Gráfica 3.2 y en la Tabla 3.2 se puede observar como el algoritmo Geometry Clipmap necesita emplear un número superior de triángulos que el resto de algoritmos para obtener la misma calidad visual en sus simplificaciones de las superficies del terreno. Además, no es capaz de generar simplificaciones que alcancen los mismos niveles de calidad que el resto de algoritmos.

También se observa como los algoritmos SOAR y ROAM2 generan simplificaciones de las superficies del terreno de la misma calidad que el algoritmo NRQT, pero empleando un menor número de triángulos, siendo similar el número de triángulos empleando por ambos algoritmos.

Por último, se observa como el algoritmo NRQT se mantiene en un nivel intermedio, si bien conforme disminuye el error producido en las simplificaciones de las superficies del terreno, el número de triángulos que emplea va disminuyendo hasta equipararse, e incluso, llegar a ser menor que el empleado por los algoritmos SOAR y ROAM2.

Prueba 3 – Rugosidad Variada



Gráfica 3.3 – RMSE – Prueba 3 – Rugosidad Variada

	NRQT		Geometry Clipmap		SOAR		ROAM2	
Prueba	**Triángulos**	**RMSE**	**Triángulos**	**RMSE**	**Triángulos**	**RMSE**	**Triángulos**	**RMSE**
1	5267	13,70	11855	15,58	4967	12,06	4666	12,78
2	9211	12,00	38361	10,81	11270	9,37	10277	9,97
3	18038	9,73	127024	7,41	22087	7,26	21497	7,97
4	41336	7,37	421754	4,98	44854	5,56	42662	6,45
5	57939	6,62			67915	4,56	58556	5,70
6	93719	5,52			94454	3,95	91663	4,65
7	122943	4,83			114154	3,66	121552	4,11
8	169764	4,25			141957	3,34	166869	3,60
9	231444	3,70			180993	3,01	229843	3,14
10	301356	3,26			239553	2,68	302391	2,80
11	408707	2,69			334783	2,28	405532	2,45

Tabla 3.3 – RMSE – Prueba 3 – Rugosidad Variada

En la Gráfica 3.3 y en la Tabla 3.3 se puede observar como el algoritmo Geometry Clipmap necesita emplear un número superior de triángulos que el resto de algoritmos para obtener la misma calidad visual en sus simplificaciones de las superficies del terreno. Además, no es capaz de generar simplificaciones que alcancen los mismos niveles de calidad que el resto de algoritmos.

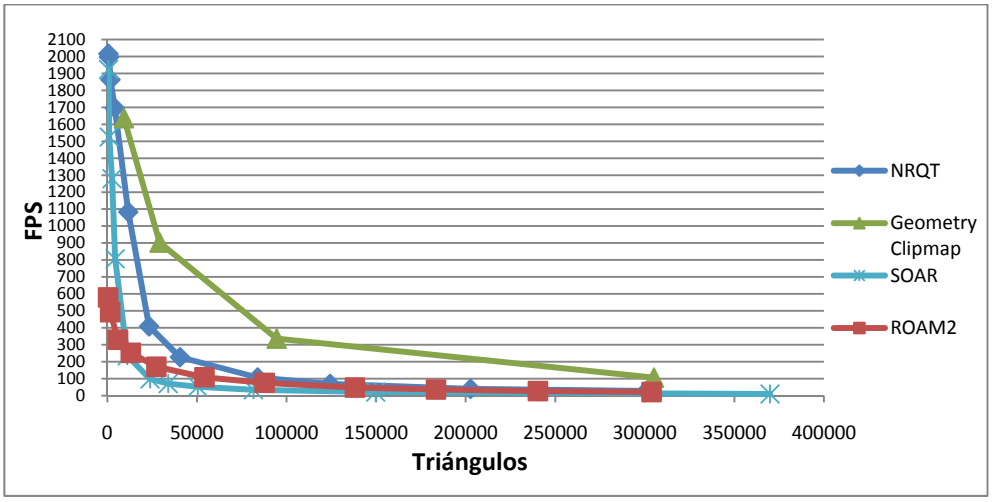
También se observa que el resto de algoritmos consiguen alcanzar simplificaciones de una calidad similar, si bien, cada uno emplea un número de triángulos algo distinto, siendo el que menos utiliza el algoritmo SOAR, seguido de ROAM2 y de NRQT.

3.3.2– Velocidad de refresco

En este apartado, se va a evaluar la velocidad de cada uno de los algoritmos para generar y visualizar simplificaciones de la superficie del terreno, calculando y comparando la velocidad de refresco media que cada algoritmo es capaz de obtener. Esta velocidad de refresco se obtiene contando, a lo largo de todo el recorrido precalculado, las veces por segundo que cada algoritmo es capaz de enviar a la tarjeta gráfica la información a visualizar.

Las gráficas siguientes muestran, para cada algoritmo, la velocidad media de refresco obtenida por ellos durante el recorrido precalculado, en función del número de triángulos empleados en la generación y visualización de las simplificaciones del terreno.

Prueba 1 – Rugosidad Baja



Gráfica 3.4 – FPS – Prueba 1 – Rugosidad Baja

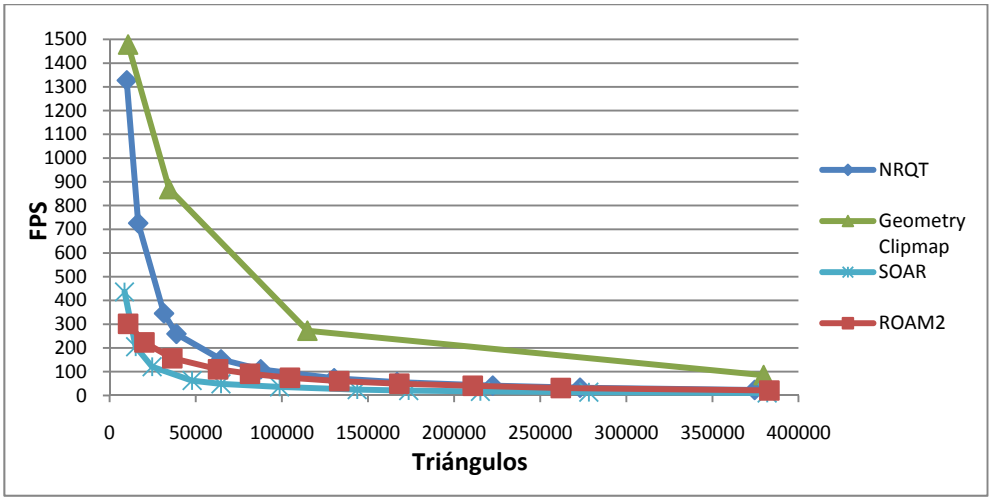
	NRQT		Geometry Clipmap		SOAR		ROAM2	
Prueba	**Triángulos**	**FPS**	**Triángulos**	**FPS**	**Triángulos**	**FPS**	**Triángulos**	**FPS**
1	562	2016,26	9217	1636,52	654	1922,95	424	578,91
2	964	1997,06	29167	902,99	997	1525,24	1590	491,82
3	1704	1863,51	94513	335,93	2594	1279,39	5916	329,41
4	4138	1696,00	305157	105,85	4318	806,96	13096	252,35
5	11675	1083,73			11089	235,12	27355	170,09
6	23256	408,60			23834	97,66	54203	108,03
7	40544	226,92			33933	71,09	87803	75,09
8	83841	107,18			50466	51,91	138165	47,51
9	124319	68,88			81565	34,24	183472	35,50
10	202705	39,77			149769	20,57	240348	26,53
11	300835	27,52			369907	9,69	303808	20,81

Tabla 3.4 – FPS – Prueba 1 – Rugosidad Baja

Como se observa en la Gráfica 3.4 y en la Tabla 3.4, el algoritmo Geometry Clipmap es el que es capaz de generar y dibujar simplificaciones de las superficies del terreno con un mayor número de triángulos para una velocidad de refresco determinada.

Los algoritmos SOAR y ROAM2 son los que obtienen unas velocidades de refresco más bajas, mientras que el algoritmo NRQT se encuentra en un nivel intermedio. Sin embargo, cabe destacar que a medida que aumenta el número de triángulos contenidos en las simplificaciones que se generan y visualizan, la diferencia entre la velocidad de refresco del algoritmo NRQT y la de SOAR y ROAM2 se reduce, aunque ésta sigue siendo considerable.

Prueba 2 – Rugosidad Alta



Gráfica 3.5 – FPS – Prueba 2 – Rugosidad Alta

	NRQT		Geometry Clipmap		SOAR		ROAM2	
Prueba	Triángulos	FPS	Triángulos	FPS	Triángulos	FPS	Triángulos	FPS
1	9897	1327,36	10684	1478,28	8610	436,13	10710,1	301,38
2	16510	725,62	34599	868,99	15096	205,07	20165	223,06
3	31529	345,43	114797	272,32	24661	120,74	36369	156,23
4	38756	259,34	379719	85,26	47866	62,41	62968	110,07
5	64642	151,99			64635	48,54	81481	90,89
6	87710	110,08			98648	35,74	104678	74,12
7	130328	72,09			143735	24,01	133326	60,06
8	166873	55,79			173635	20,35	168182	49,51
9	222442	41,33			215259	16,95	210830	40,68
10	273111	32,67			278262	13,53	261954	31,15
11	374573	22,85			381797	10,29	383097	20,72

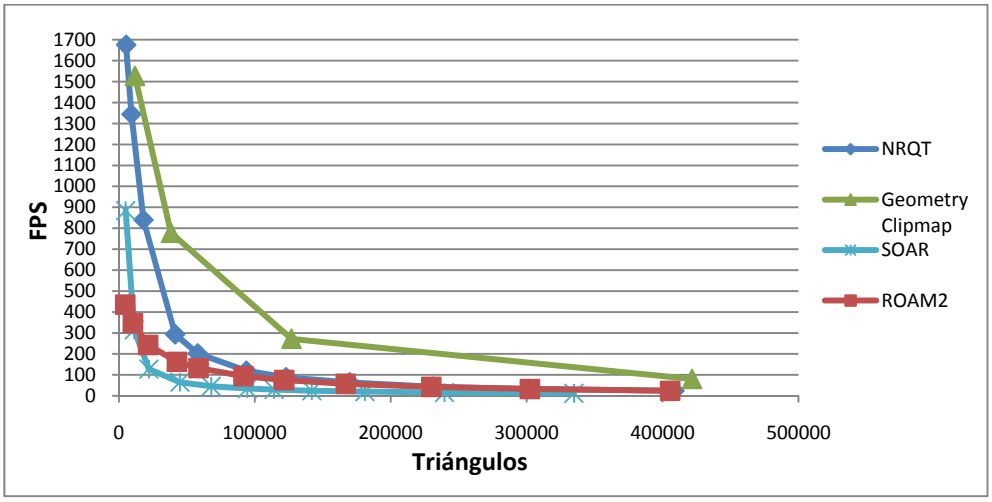
Tabla 3.5 – FPS – Prueba 2 – Rugosidad Alta

Como se observa en la Gráfica 3.5 y en la Tabla 3.5 el algoritmo Geometry Clipmap es el que es capaz de generar y dibujar simplificaciones con un mayor número de triángulos para una velocidad de refresco determinada.

El algoritmo SOAR es el que menor velocidad de refresco obtiene en la generación y visualización de simplificaciones para un mismo número de triángulos empleados, llegando a ser la velocidad de refresco conseguida por los otros algoritmos de más del doble que la de éste.

Los algoritmos NRQT y ROAM2 se encuentran en un nivel intermedio, sin embargo cabe destacar que los valores de velocidad de refresco del algoritmo NRQT son siempre superiores a los conseguidos por el algoritmo ROAM2, disminuyéndose esa diferencia conforme aumenta el número de triángulos empleados en la generación y visualización de las simplificaciones de las superficies del terreno.

Prueba 3 – Rugosidad Variada



Gráfica 3.6 – FPS – Prueba 3 – Rugosidad Variada

	NRQT		Geometry Clipmap		SOAR		ROAM2	
Prueba	**Triángulos**	**FPS**	**Triángulos**	**FPS**	**Triángulos**	**FPS**	**Triángulos**	**FPS**
1	5267	1675,39	11855	1527,14	4967	884,15	4666	435,15
2	9211	1344,15	38361	778,20	11270	314,88	10277	346,24
3	18038	839,68	127024	272,10	22087	127,70	21497	242,66
4	41336	293,67	421754	80,29	44854	65,00	42662	162,15
5	57939	201,98			67915	45,77	58556	132,37
6	93719	119,67			94454	34,93	91663	93,85
7	122943	88,23			114154	29,87	121552	75,20
8	169764	64,14			141957	24,26	166869	57,11
9	231444	43,61			180993	20,06	229843	42,35
10	301356	33,41			239553	15,45	302391	32,28
11	408707	23,95			334783	11,93	405532	23,87

Tabla 3.6 – FPS – Prueba 3 – Rugosidad Variada

Como se observa en la Gráfica 3.6 y en Tabla 3.6 el algoritmo Geometry Clipmap es el que es capaz de generar y dibujar simplificaciones con un mayor número de triángulos para una velocidad de refresco determinada.

El algoritmo SOAR es el que menor velocidad de refresco obtiene en la generación y visualización de simplificaciones para un mismo número de triángulos empleados, llegando a ser la velocidad de refresco conseguida por los otros algoritmos de más del doble que la de éste.

Los algoritmos NRQT y ROAM2 se encuentran en un nivel intermedio, sin embargo cabe destacar que los valores de velocidad de refresco del algoritmo NRQT son siempre superiores a los conseguidos por el algoritmo ROAM2, disminuyéndose esa diferencia conforme aumenta el número de triángulos empleados en la generación y visualización de las simplificaciones de las superficies del terreno.

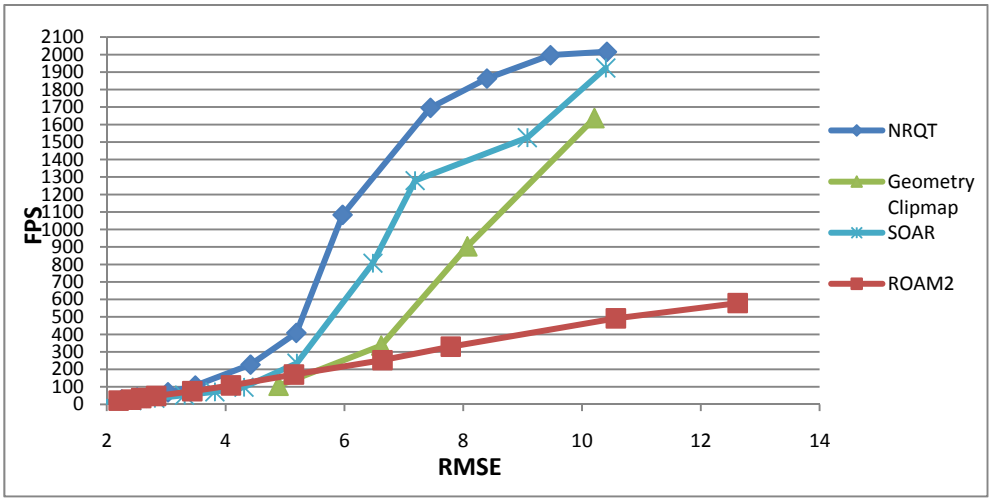
3.3.3 – RMSE / Framerate

Los resultados anteriores nos muestran por separado, por un lado, cómo de buenas son las simplificaciones que se pueden obtener con cada algoritmo, y por otro lado, lo veloz que es cada algoritmo en la generación y visualización de esas simplificaciones.

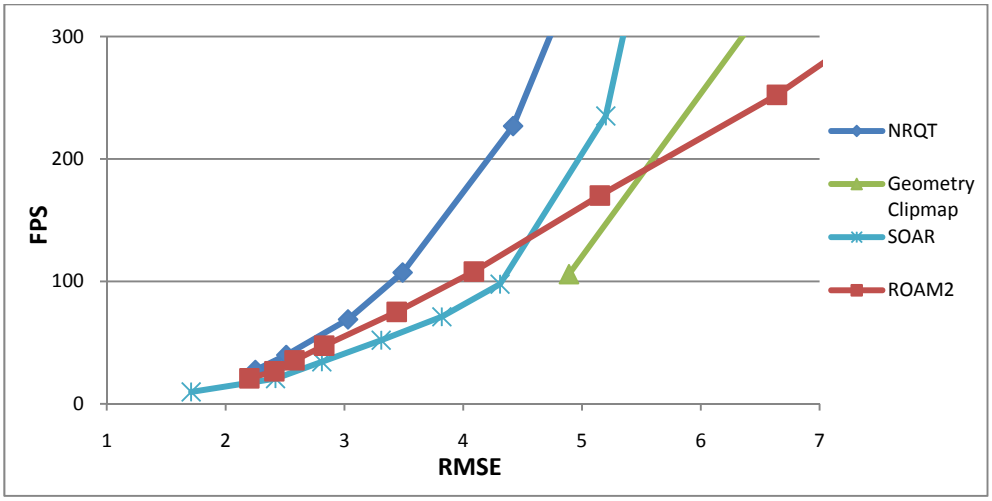
En una aplicación típica de visualización de terrenos en tiempo real, está limitado el tiempo máximo que la aplicación puede invertir en cada iteración para la construcción de la simplificación de la superficie del terreno. La aplicación deber mostrar la información de forma fluida para permitir una interacción adecuada con el usuario. Por lo tanto, realmente lo que interesa conocer es qué calidad puede llegar a ofrecer cada algoritmo de triangulación en función del tiempo disponible en cada iteración. Se puede deducir qué algoritmo realiza simplificaciones de mayor calidad de forma más rápida combinando la información empleada en las gráficas anteriores.

Las gráficas siguientes muestran la relación entre la calidad visual de las simplificaciones del terreno generadas por cada algoritmo y la velocidad con la que se generan y dibujan esas simplificaciones.

Prueba 1 – Rugosidad Baja



Gráfica 3.7 – RMSE/FPS – Prueba 1 – Rugosidad Baja



Gráfica 3.8 – RMSE/FPS – Prueba 1 – Rugosidad Baja (ampliación RMSE bajo)

	NRQT		Geometry Clipmap		SOAR		ROAM2	
Prueba	**RMSE**	**FPS**	**RMSE**	**FPS**	**RMSE**	**FPS**	**RMSE**	**FPS**
1	10,42	2016,26	10,21	1636,52	10,4	1922,95	12,62	578,91
2	9,47	1997,06	8,07	902,99	9,08	1525,24	10,57	491,82
3	8,40	1863,51	6,62	335,93	7,19	1279,39	7,79	329,41
4	7,45	1696,00	4,89	105,85	6,48	806,96	6,64	252,35
5	5,97	1083,73			5,20	235,12	5,15	170,09
6	5,19	408,60			4,31	97,66	4,09	108,03
7	4,42	226,92			3,82	71,09	3,44	75,09
8	3,49	107,18			3,31	51,91	2,83	47,51
9	3,03	68,88			2,81	34,24	2,58	35,50
10	2,51	39,77			2,42	20,57	2,41	26,53
11	2,25	27,52			1,71	9,69	2,20	20,81

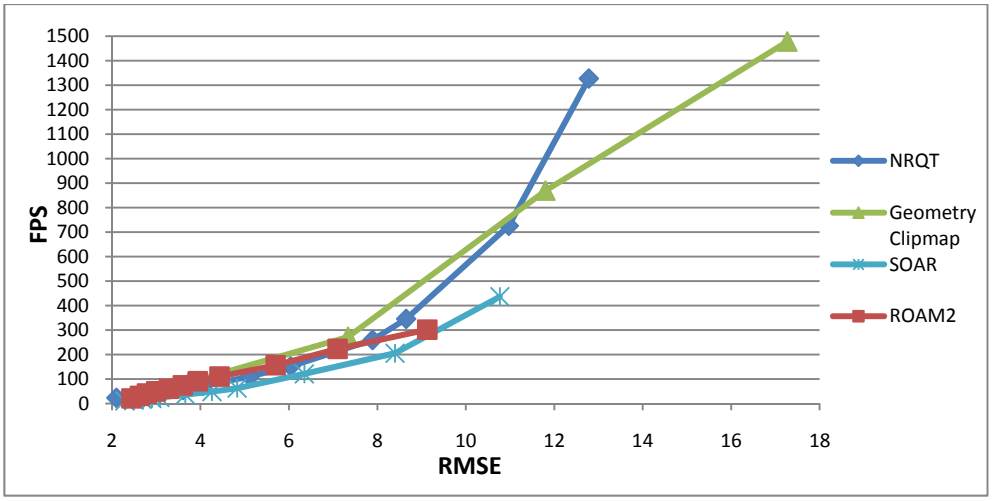
Tabla 3.7 – RMSE/FPS – Prueba 1 – Rugosidad Baja

En la Gráfica 3.7, en la Gráfica 3.8 y en la Tabla 3.7 se observa como todos los algoritmos son capaces de obtener y visualizar simplificaciones de las superficies del terreno con errores bajos a velocidades de refresco similares.

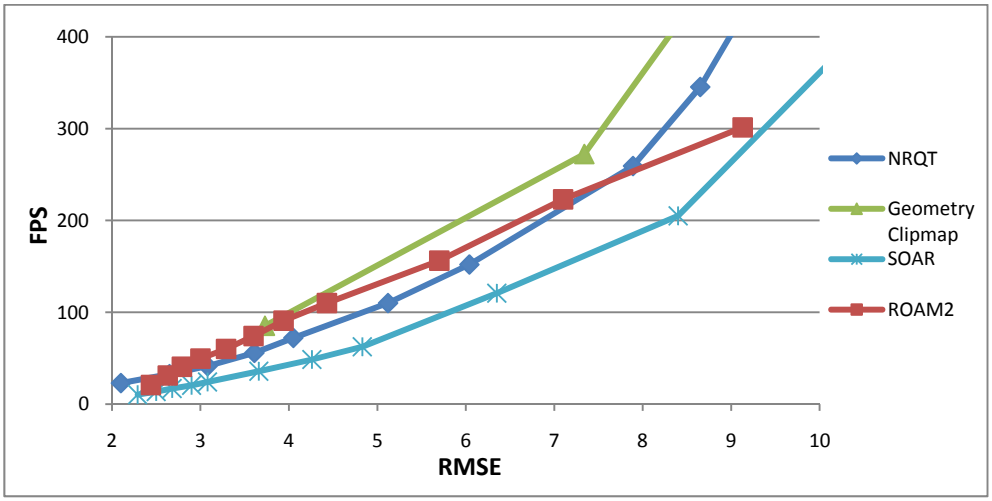
Sin embargo, la triangulación NRQT siempre obtiene los mejores resultados en la relación calidad/velocidad, incrementándose la diferencia con el resto de algoritmos conforme las simplificaciones generadas y visualizadas contienen un error superior.

Para el resto de algoritmos se observa que, en general, ROAM2 es el que tendría una relación calidad/velocidad peor, seguido por el algoritmo Geometry Clipmap y por SOAR. Si bien, hay que hacer notar que para niveles de error bajos, ROAM2 tendría una relación calidad/velocidad mejor que SOAR.

Prueba 2 – Rugosidad Alta



Gráfica 3.9 – RMSE/FPS – Prueba 2 – Rugosidad Alta



Gráfica 3.10 – RMSE/FPS – Prueba 2 – Rugosidad Alta (ampliación RMSE bajo)

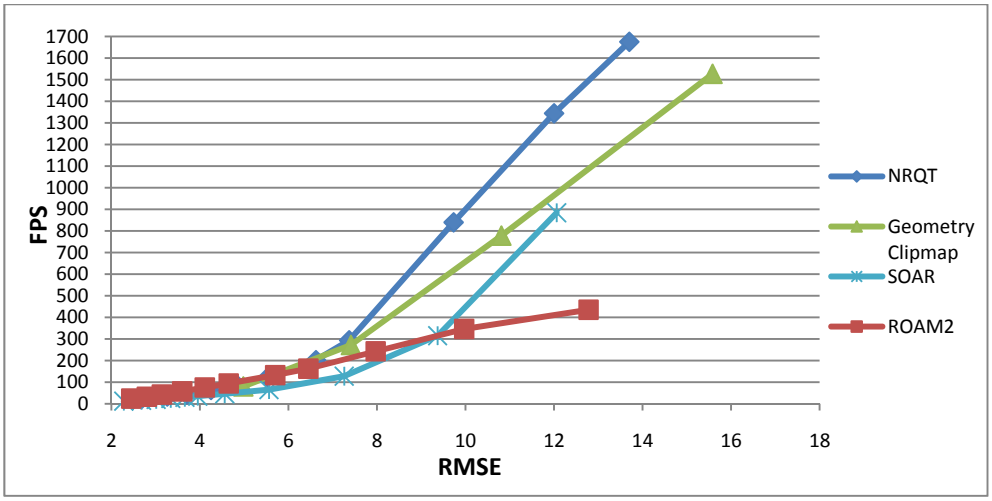
	NRQT		Geometry Clipmap		SOAR		ROAM2	
Prueba	**RMSE**	**FPS**	**RMSE**	**FPS**	**RMSE**	**FPS**	**RMSE**	**FPS**
1	12,78	1327,36	17,27	1478,28	10,77	436,13	9,13	301,38
2	10,97	725,62	11,80	868,99	8,40	205,07	7,10	223,06
3	8,65	345,43	7,34	272,32	6,35	120,74	5,70	156,23
4	7,89	259,34	3,73	85,26	4,83	62,41	4,43	110,07
5	6,04	151,99			4,26	48,54	3,94	90,89
6	5,12	110,08			3,66	35,74	3,60	74,12
7	4,05	72,09			3,08	24,01	3,29	60,06
8	3,61	55,79			2,90	20,35	3,00	49,51
9	3,08	41,33			2,68	16,95	2,79	40,68
10	2,65	32,67			2,50	13,53	2,63	31,15
11	2,10	22,85			2,29	10,29	2,44	20,72

Tabla 3.8 – RMSE/FPS – Prueba 2 – Rugosidad Alta

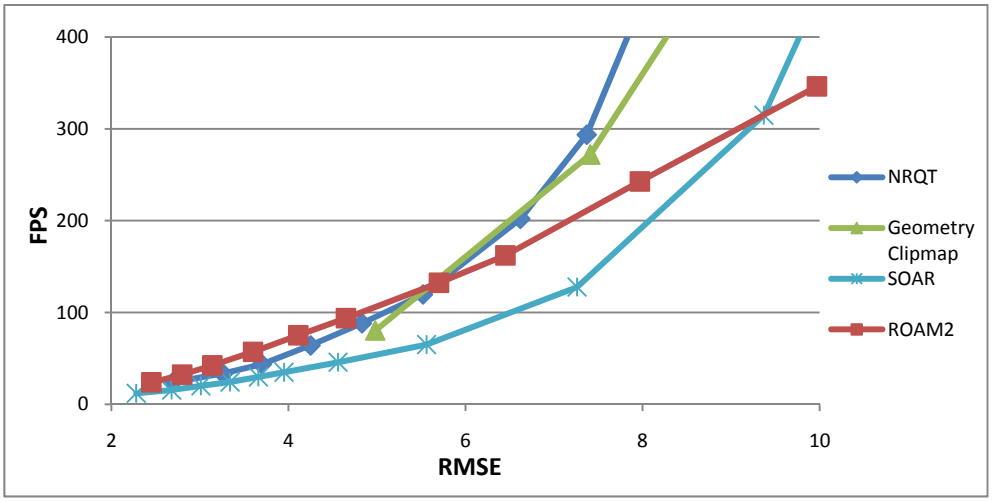
En la Gráfica 3.9, en la Gráfica 3.10 y en la Tabla 3.8 se observa como para errores bajos en las simplificaciones de las superficies del terreno, los algoritmos Geometry Clipmap, NRQT y ROAM2 obtienen una relación calidad/velocidad similar y algo superior a SOAR.

Conforme aumenta el error de las simplificaciones, el algoritmo Geometry Clipmap obtiene los mejores resultados en esta relación, seguido por los algoritmos NRQT, ROAM2 y SOAR. Hay que destacar que cuando se emplean simplificaciones con un error superior a 11, el algoritmo NRQT obtiene mejores resultados que el algoritmo de Geometry Clipmap.

Prueba 3 – Rugosidad Variada



Gráfica 3.11 – RMSE/FPS – Prueba 3 – Rugosidad Variada



Gráfica 3.12 – RMSE/FPS – Prueba 3 – Rugosidad Variada (ampliación RMSE bajo)

	NRQT		Geometry Clipmap		SOAR		ROAM2	
Prueba	**RMSE**	**FPS**	**RMSE**	**FPS**	**RMSE**	**FPS**	**RMSE**	**FPS**
1	13,70	1675,39	15,58	1527,14	12,06	884,15	12,78	435,15
2	12,00	1344,15	10,81	778,20	9,37	314,88	9,97	346,24
3	9,73	839,68	7,41	272,10	7,26	127,70	7,97	242,66
4	7,37	293,67	4,98	80,29	5,56	65,00	6,45	162,15
5	6,62	201,98			4,56	45,77	5,70	132,37
6	5,52	119,67			3,95	34,93	4,65	93,85
7	4,83	88,23			3,66	29,87	4,11	75,20
8	4,25	64,14			3,34	24,26	3,60	57,11
9	3,70	43,61			3,01	20,06	3,14	42,35
10	3,26	33,41			2,68	15,45	2,80	32,28
11	2,69	23,95			2,28	11,93	2,45	23,87

Tabla 3.9 – RMSE/FPS – Prueba 3 – Rugosidad Variada

En la Gráfica 3.11, en la Gráfica 3.12 y en Tabla 3.9 se observa como para errores bajos en las simplificaciones de las superficies del terreno, los algoritmos NRQT y ROAM2 obtienen una mejor relación calidad/velocidad que el resto de algoritmos.

Conforme aumenta el error permitido en las simplificaciones, el algoritmo NRQT obtiene los mejores resultados en esa relación, seguido por los algoritmos Geometry Clipmap y SOAR, quedando el algoritmo ROAM2 relegado a la última posición.

3.3.4 – Conclusiones

Se ha realizado una comparativa de los algoritmos de triangulación más empleados habitualmente en la visualización de terrenos en tiempo real, adaptados para aprovechar las nuevas características de las GPU de las tarjetas gráficas modernas.

Se puede extraer como conclusión de los resultados, que el algoritmo SOAR es el que obtiene, independientemente de la rugosidad del terreno, unas simplificaciones de las superficies del terreno de mayor calidad visual que el resto de algoritmos, debido a que su proceso de generación es más complejo. Sin embargo, esta complejidad hace que este proceso de generación sea bastante costoso computacionalmente, lo que provoca que este algoritmo no sea capaz de alcanzar velocidades de refresco tan altas como los otros algoritmos.

Por otro lado, los resultados indican que el algoritmo de Geometry Clipmap es el que, independientemente de la rugosidad del terreno, permite generar y visualizar simplificaciones de las superficies del terreno con una velocidad de refresco mayor que el resto de algoritmos, pero a cambio de que estas simplificaciones tengan una calidad visual más baja que las generadas por los otros algoritmos. Esto es debido a que el algoritmo de Geometry Clipmap no realiza ningún tipo de adaptación de sus simplificaciones a la rugosidad de la superficie del terreno, por lo que para que sus simplificaciones alcancen la misma calidad visual que las de los otros algoritmos, necesita emplear un número más elevado de triángulos, e incluso en ocasiones, como se ha visto en las pruebas, aún empleando más triángulos no es capaz de alcanzar los mismos valores de calidad visual.

También podemos extraer de las pruebas que en el punto medio de los dos algoritmos anteriores nos encontramos con los algoritmos de NRQT y ROAM2, que realizan un proceso de generación no tan complejo como el del algoritmo de SOAR, pero que adaptan sus simplificaciones a la rugosidad de la superficie del terreno, lo que permite obtener simplificaciones con una calidad aceptable manteniendo una velocidad de refresco también aceptable.

Si nos centramos en la relación entre la calidad de las simplificaciones y la velocidad de refresco con que esas simplificaciones se pueden generar y visualizar, se observa que, generalmente, el algoritmo que obtiene unos mejores resultados es el algoritmo NRQT, debido a que emplea un proceso de generación de simplificaciones en las que éstas se adaptan a la rugosidad del terreno sin requerir de un elevado coste computacional. Este hecho unido a que los mecanismos que emplea este algoritmo para llevar a cabo una visualización rápida y eficiente de las simplificaciones del terreno, le permiten generar unas simplificaciones de una calidad aceptable y alcanzar una velocidad de refresco adecuada para una aplicación en tiempo real.

Si nos fijamos, además, en como varían estos resultados para los diferentes grados de rugosidad de las superficies del terreno por las que tienen lugar las pruebas, se observa que el algoritmo de Geometry Clipmap obtiene unos mejores resultados conforme el nivel de rugosidad de la superficie del terreno se incrementa. Esto se debe a que conforme aumenta la rugosidad de la superficie, se hace necesario emplear un mayor número de triángulos para generar una simplificación que represente fielmente la superficie original, y esto es justamente lo que realiza este algoritmo de forma óptima: generar y visualizar simplificaciones formadas por una gran cantidad de triángulos. Sin embargo, se puede observar como en los casos en los que no es necesario emplear tantos triángulos de forma continua, como sería el caso de uso más habitual, la visualización de superficies de terrenos con rugosidad variada, sus resultados son peores. El resto de algoritmos mantienen los valores de su relación calidad/velocidad bastante parecidos para los diferentes tipos de rugosidad.

Los resultados obtenidos en estas pruebas son válidos para las aplicaciones de visualización de terrenos que emplean bases de datos locales. En estas condiciones, los diferentes algoritmos funcionan de forma más o menos correcta, si bien, como hemos visto en los resultados, NRQT tiene un mejor rendimiento que los otros algoritmos, proporcionando versiones simplificadas de las superficies del terreno de una calidad adecuada en un tiempo compatible para un sistema de visualización en tiempo real, independientemente de la rugosidad de la superficie del terreno.

Si se plantea el problema para bases de datos remotas, la elección es clara. Sólo para el algoritmo NRQT no es necesario modificar el código, puesto que la organización de los datos del terreno en una estructura jerárquica con forma de quadtree, le permite encapsular dentro de un nodo de esa jerarquía, la información obtenida directamente de las bases de datos, independientemente de si esa información proviene de una base de datos local como de una base de datos remota, sin necesidad de reorganizar ni manipular esa información.

En el caso del algoritmo Geometry Clipmap, éste tendría que sufrir un proceso de adaptación para poder hacer uso de estas bases de datos remotas. Pero como se ha comentado en el apartado 3.1.3, las modificaciones necesarias para ello aumentarían la complejidad y el coste computacional del algoritmo, penalizando su rendimiento.

En el caso de los algoritmos ROAM2 y SOAR, la adaptación resulta aún más complicada, debido principalmente a que, tal como se ha mencionado en el apartado 3.1.1 y 3.1.2, las operaciones de refinamiento y simplificación se aplican de forma individual sobre los vértices. Este hecho complica enormemente la posibilidad de predecir las necesidades futuras de datos a descargar, con lo que la descarga de nuevos vértices se tiene que realizar también de forma individualizada, y por lo tanto, de forma poco eficiente.

Atendiendo a todo lo anterior, como conclusión final, podemos decir que el algoritmo de triangulación que mejor se adapta a un sistema de visualización de terrenos en tiempo real con bases de datos remotas es NRQT. Por lo tanto, este algoritmo será el elegido para ser empleado dentro del sistema desarrollado en esta tesis. Sin embargo, cabe mencionar que se intentará implementar este algoritmo de forma independiente al resto del sistema para permitir, en caso necesario, un cambio del algoritmo de triangulación.

Capítulo 4 – Compresión de la Información

Una aplicación de visualización de terrenos distribuidos en tiempo real normalmente necesita almacenar y transmitir una gran cantidad de información. Para reducir el tamaño de esa información se debe llevar a cabo algún tipo de compresión de la información.

Como se ha comentado en el capítulo 2, los esquemas de compresión basados en la transformada wavelet son, a priori, los más adecuados para realizar la compresión y transmisión de la información, gracias a sus propiedades innatas de multirresolución. En la actualidad, existe un estándar de compresión, sucesor del jpeg, que emplea la transformada wavelet para comprimir la información, el estándar Jpeg2000 [68]. Este estándar también dispone de un protocolo para la transmisión de la información en entornos remotos llamado Jpip.

Por otro lado, además de emplear un esquema de compresión, también es necesario organizar la información de forma eficiente. En el capítulo 2 se vio que la estructura de organización más empleada y que mejor se ajusta al sistema que se va a desarrollar en la tesis es la pirámide de cuadrantes, por lo que se va a emplear esta estructura para organizar la información del terreno.

Un esquema de compresión que podría adaptarse bien a la pirámide de cuadrantes sería el esquema de compresión estándar Jpeg2000, si bien, se prevé que surjan algunos problemas en su utilización, dado que no es posible reconstruir regiones aisladas del terreno de forma independiente a sus regiones vecinas sin que se produzca la aparición de artefactos visuales en las fronteras de esas regiones.

Para evitar esos problemas, el esquema Jpeg2000 incorpora un mecanismo que consiste en dividir la imagen original en trozos, tratando cada uno de ellos como una imagen independiente, lo que permite llevar a cabo su transmisión y reconstrucción de forma aislada. Sin embargo, este mecanismo tiene el inconveniente de hacer disminuir la tasa global de compresión, además de provocar un efecto de bloque en la imagen una vez descomprimida.

A lo largo de este capítulo se va a definir un nuevo esquema de compresión para solucionar los problemas que presenta Jpeg2000 a la hora de comprimir de forma eficiente la información de la pirámide de cuadrantes, realizando varias modificaciones al esquema original de Jpeg2000.

Después, se va a realizar una implementación de estos dos esquemas de compresión que emplean la transformada wavelet, el nuevo y el del estándar Jpeg2000 clásico, realizando un estudio de cuál se puede adaptar mejor a nuestro sistema de visualización de terrenos en entornos distribuidos que empleará una pirámide de cuadrantes para organizar la información.

El capítulo se organiza de la siguiente manera: primero se va a explicar el funcionamiento de estos dos esquemas; después se va a especificar la metodología que se va a llevar a cabo para evaluar y comparar estos esquemas, y por último, se va a realizar la comparativa de estos esquemas.

4.1 – JPEG 2000

Jpeg2000 es un estándar desarrollado por el "Joint Photographic Expert Group" (Jpeg) y basa su funcionamiento en la transformada wavelet discreta (DWT) y en la codificación EBCOT [125]. En este apartado se va a realizar una revisión de los aspectos más destacables del estándar Jpeg2000. Para encontrar información más detallada se recomienda consultar [68], [115] y [126].

4.1.1 – Motor de compresión y reconstrucción

El esquema general del motor de compresión de JPEG2000 se muestra en Figura 4.1. Éste se puede dividir en los siguientes pasos:

1.- La imagen se descompone en sus componentes de color.

2.- La imagen se divide opcionalmente en cuadrantes (Figura 4.2 izquierda).

3.- Se aplica la transformada wavelet a cada cuadrante (Figura 4.2 derecha).

4.- Las subbandas de los coeficientes que resultan de la transformada son cuantizadas y almacenadas en code-blocks.

5.- Los planos de bits de los coeficientes de un code-block son codificados entrópicamente.

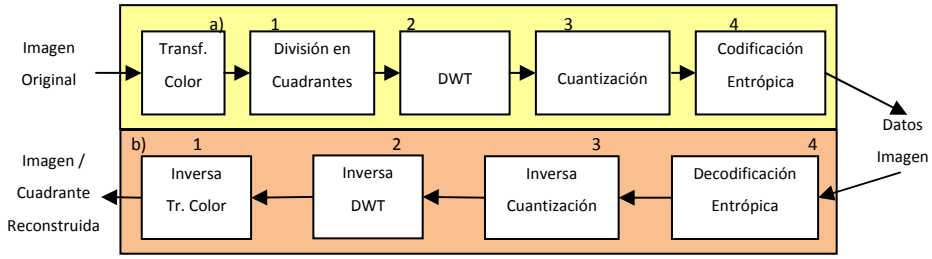


Figura 4.1 – a) Motor de compresión de Jpeg2000, b) Motor de reconstrucción de Jpeg2000

Del esquema general cabe destacar lo siguiente:

4.1.1.1 – Transformación de Color

Este proceso se encarga de convertir las coordenadas de color de la imagen expresadas en componentes de color RGB, en otras componentes de color YCbCr, para evitar la redundancia que existe entre los canales R, G y B, y poder así aumentar las tasas de compresión.

4.1.1.2 – División en Cuadrantes de Imágenes (Image Tiling)

Este proceso es opcional, y permite dividir la imagen original en un conjunto de trozos rectangulares no solapados que forman una rejilla (Figura 2.4 - Izquierda). Con esta división, se pueden reconstruir de forma independiente estos trozos de imagen, reduciendo los requerimientos de memoria necesarios para procesar imágenes extensas.

Cada uno de estos trozos se trata como si fuera una imagen diferente, aplicando a cada uno, de forma independiente, la transformada wavelet. En cada paso del proceso de análisis de la transformada se obtiene una nueva versión de menor resolución de la imagen (Figura 4.2 - Derecha).

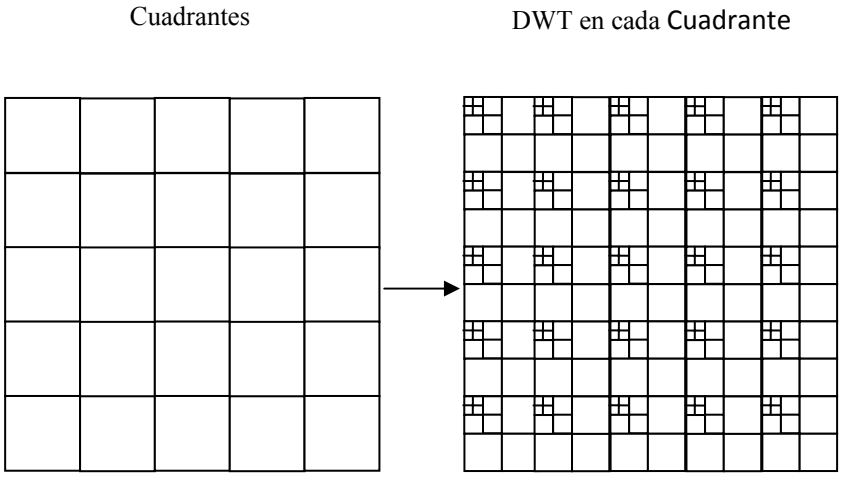


Figura 4.2 – Izquierda: Imagen dividida en cuadrantes. Derecha: Transformada wavelet aplicada a cada cuadrante de forma independiente.

La principal desventaja de esta división en cuadrantes es que afecta a la calidad de la imagen reconstruida alrededor de las fronteras de los cuadrantes, tanto objetiva como subjetivamente, debido a que no se comparte información entre los cuadrantes vecinos (Figura 4.3). Este problema se conoce como efecto bloque y se hace más evidente cuando se emplean tasas de compresión altas. Además, empleando esta división se obtiene una tasa de compresión más baja de la imagen que si no se emplea. Debido a estos problemas, la división en cuadrantes de jpeg2000 no se suele utilizar.

Figura 4.3 – Textura de tamaño 512x512 pixels reconstruida empleando Jpeg2000 con una tasa de compresión de 0.2bpp. Izquierda: Resultado obtenido sin dividir la imagen en cuadrantes. Derecha: Resultado obtenido dividiendo la imagen con cuadrantes de 64x64 pixels.

4.1.1.3 – Transformada Wavelet Discreta Directa (DWT)

La transformada wavelet discreta directa de cada cuadrante, se calcula de forma recursiva aplicando un filtro de análisis wavelet 2D a la subbanda pasa-baja obtenida en cada nivel. Cada aplicación de este filtro de análisis da lugar a cuatro subbandas: pasa-baja horizontal y vertical (LL), pasa-baja horizontal y pasa-alta vertical (LH), pasa-alta horizontal y pasa-baja vertical (HL) y pasa-alta horizontal y vertical (HH) (Figura 4.4). Esto significa que la transformada wavelet directa realiza una descomposición diádica del cuadrante, obteniendo, en cada paso, una subbanda pasa-baja horizontal y vertical (LL) que representa una versión de menor resolución de la imagen original, y otras tres subbandas donde se codifican los detalles perdidos en este paso (Figura 4.5).

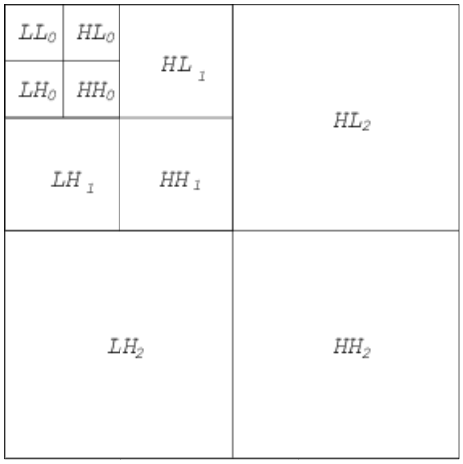


Figura 4.4 – Estructura de subbandas para tres niveles de descomposición wavelet (análisis).

Figura 4.5 – Dos niveles de descomposición wavelet de una textura.

Por lo tanto, para cada nivel de resolución, siempre se van a generar el mismo número de versiones simplificadas, sin embargo, el tamaño de esas versiones será cada vez menor. Esto supone un problema a la hora de adaptar este esquema de compresión a la pirámide de cuadrantes, puesto que en una pirámide de cuadrantes todas las versiones simplificadas de los diferentes niveles de resolución siempre tienen el mismo tamaño, y lo que va disminuyendo es el número de esas versiones para cada nivel.

4.1.1.4 – Cuantización

En esta fase se emplea una cuantización uniforme con zona muerta que se aplica para cada coeficiente wavelet mediante la siguiente expresión:

$$q = sign(c)\left\lfloor \frac{|c|}{\Delta_b} \right\rfloor$$

Ecuación 4.1 – Función de Cuantización

Donde c es el coeficiente wavelet de entrada al cuantizador, Δ_b es el tamaño de paso del cuantizador, q es el valor cuantizado resultante y sign(c) denota el signo de c.

4.1.1.5 – Code-blocks y codificación entrópica

Después de llevarse a cabo la transformada wavelet y de que se aplique la cuantización, cada subbanda de coeficientes se divide, a su vez, en grupos rectangulares no solapados llamados code-blocks. Cada uno de estos code-blocks se codifica entrópicamente de forma independiente sin hacer referencia a otros code-blocks de la misma o de otra subbanda. Esta codificación independiente proporciona una mayor capacidad de acceso espacial aleatorio a los datos.

4.1.1.6 – Reconstrucción

A la hora de reconstruir la imagen original, se realiza el proceso inverso al de la fase de compresión. Los code-blocks de cada subbanda se envían en orden inverso al que se generaron con la transformada wavelet directa. Se comienza desde el nivel de menor

resolución hasta llegar al de máxima resolución. Estos code-blocks son decodificados entrópicamente y se les aplica la inversa de la cuantización. A continuación, se les aplica la transformada wavelet inversa tan pronto como los code-blocks son recibidos para, progresivamente, ir generando versiones simplificadas de la imagen (LL0, LL1, LL2, etc.) hasta la reconstrucción de la imagen original (Figura 4.6). A las componentes de color de estas imágenes se les aplica la transformada de color inversa para que vuelvan a estar expresadas con el modelo de color RGB.

Los filtros empleados en la fase de síntesis para la reconstrucción de un nivel de detalle de la imagen comprimida, son los inversos a los utilizados en la fase de análisis. En la fase de análisis, se aplicaban estos filtros a todo el nivel de detalle de la imagen que se iba a comprimir, obteniendo un conjunto de coeficientes wavelet. Por lo tanto, para obtener una reconstrucción de ese nivel de detalle de la imagen libre de artefactos visuales, es necesario disponer de todo ese conjunto de coeficientes wavelet.

Si se desea reconstruir únicamente una región de ese nivel de detalle de forma independiente, se debe disponer de los coeficientes que pertenecen a esa región, los cuales se encuentran dentro del conjunto total de coeficientes wavelet de ese nivel de detalle. A la hora de reconstruir las fronteras de esa región, el filtro empleado necesita coeficientes wavelet de las regiones vecinas. Si no se dispone de ellos, se puede suplir la falta de esos coeficientes wavelet extendiendo de forma simétrica los coeficientes en las regiones vecinas para poder llevar a cabo la reconstrucción, sin embargo, esto producirá como resultado una imagen diferente de la imagen original, que provocará la aparición de artefactos visualmente notables en las fronteras de la región con sus regiones vecinas, una vez éstas sean también reconstruidas.

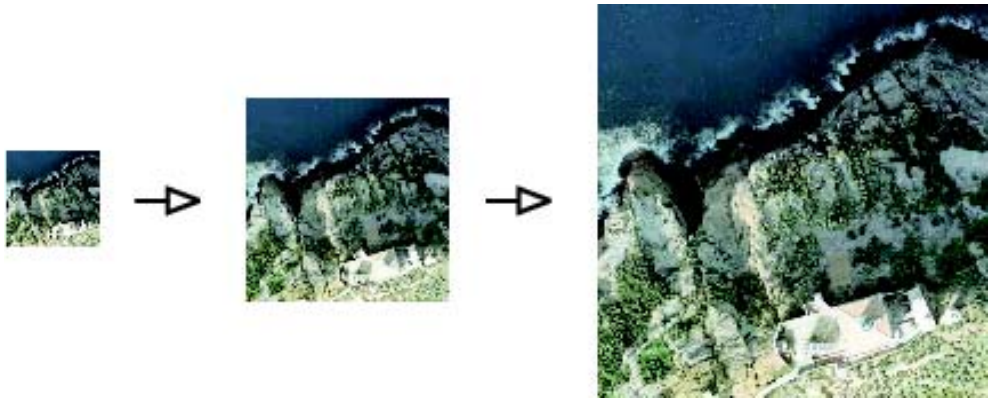


Figura 4.6 – Dos niveles de reconstrucción wavelet de una imagen

4.1.2 – Jpip

Jpip es la especificación del protocolo de comunicaciones empleado para la transmisión de contenidos de Jpeg2000 en aplicaciones distribuidas por la red. El objetivo de Jpip es establecer un protocolo que permita la transmisión progresiva de datos codificados en una arquitectura cliente-servidor. Se puede encontrar información más detallada de este protocolo en [69] y [127].

El estándar Jpeg2000 ofrece muchas características idóneas para el acceso interactivo de imágenes extensas, como escalabilidad por nivel de resolución, refinamiento progresivo o acceso espacial aleatorio.

Empleando Jpip, los clientes son capaces de realizar peticiones de las regiones de la imagen que necesitan, obteniendo además de la información de esas regiones, cualquier otra información asociada a las mismas, como por ejemplo los metadatos almacenados.

En el protocolo Jpip, el cliente lleva a cabo las peticiones a través de una ventana de visión. Esta ventana define la resolución, el tamaño, la localización, los componentes, las capas y el resto de parámetros relacionados con los datos requeridos. El servidor responde la imagen o partes de la imagen adecuados, además de otra información relacionada con la petición del cliente (Figura 4.7).

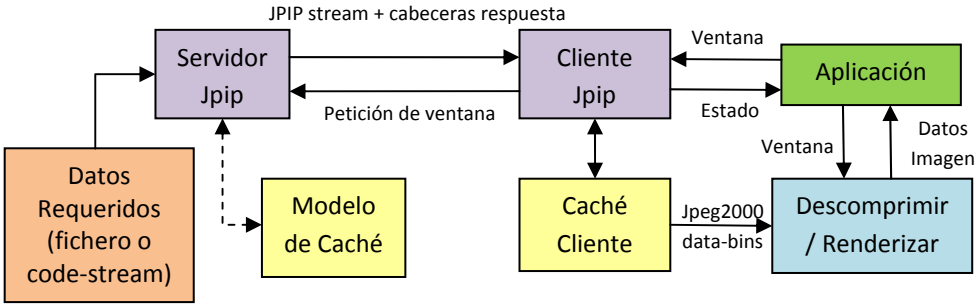


Figura 4.7 – Interacción Cliente-Servidor en Jpip

Jpip también establece un mecanismo de caché que permite reutilizar la información de las peticiones previas, sin embargo, las peticiones han de ser idénticas para poder obtener esta información, de forma que, si dos clientes distintos realizan dos peticiones diferentes a una misma imagen resultando la información requerida para cada petición la misma o muy similar, esta información no se extraería de la caché, lo que hace que éste sea un mecanismo poco eficiente.

Implementación Evaluada

Para realizar la evaluación de este esquema de compresión se ha empleado la librería de compresión Kakadu [71]. Esta librería ofrece, en la actualidad, la implementación más completa del estándar Jpeg2000, además de proporcionar un mejor rendimiento y unos tiempos de compresión y descompresión menores que el resto de librerías actuales del mercado que implementan el estándar jpeg2000.

4.2 – Nuevo esquema

A partir de Jpeg2000 se ha diseñado un nuevo esquema de compresión que evita algunos de los problemas del esquema estándar, como son la aparición de artefactos visuales en las fronteras cuando se trata de decodificar de forma independiente una región de la imagen, y que las reconstrucciones obtenidas no encajen con la pirámide de cuadrantes que se emplea para organizar la información del terreno.

4.2.1 – Funcionamiento

Este esquema se basa en el estándar Jpeg2000, pero emplea un proceso de análisis y síntesis diferente. Como se ha mencionado previamente, las pirámides de cuadrantes son especialmente útiles en las aplicaciones interactivas de visualización de terrenos extensos, por lo que uno de los objetivos de este nuevo esquema de compresión es obtener reconstrucciones del terreno que encajen dentro de una pirámide de cuadrantes.

Los cuadrantes generados por este esquema se ajustan con los que emplean las pirámides de cuadrantes, ya que en cada paso de síntesis de la transformada wavelet, se generan cuadrantes de tamaño constante, cuyo tamaño se especifica inicialmente, y el número de cuadrantes que se generan disminuye en cada nivel de resolución. Estos cuadrantes son diferentes de los generados por el esquema Jpeg2000 clásico, donde en cada paso de síntesis se generan el mismo número de cuadrantes, pero el tamaño de los mismos va decreciendo para cada nivel de resolución.

El esquema general del motor de compresión se muestra en la Figura 4.8. Este esquema es similar al que emplea el Jpeg2000 clásico, las principales diferencias se encuentras en la fase de división en cuadrantes y en la aplicación de la transformada wavelet.

Antes de cada paso de descomposición de la transformada wavelet, la textura se divide en cuadrantes de tamaño fijo (en el esquema JPEG2000 clásico se divide la textura original en cuadrantes una sola una vez al inicio del proceso de compresión). Entonces, se aplica un paso del filtro de análisis de extensión simétrica 2D a cada cuadrante, generando cuatro bandas de coeficientes: LL, HL, LL y HH.

Los coeficientes de las bandas pasa alta (HL, LH y HH) se agrupan en code-blocks, cada uno de ellos es cuantizado y codificado entrópicamente de forma independiente de los de otros cuadrantes, de forma similar a la del esquema estándar JPEG2000.

Antes de que un nuevo paso de descomposición tenga lugar, todas las subbandas pasa-baja (LL) de los cuadrantes se combinan formando una versión de menor resolución de la textura completa.

El proceso se repite hasta que el tamaño de la textura pasa-baja sea más pequeño que el tamaño de cuadrante fijado. En la Figura 4.9 se describe esquemáticamente el proceso en cada paso, y en la Figura 4.10 se muestra el proceso para una imagen real.

Este proceso de compresión se puede resumir en los siguientes pasos (Figura 4.8 - a):

1.- Divide la textura en cuadrantes.

2.- Para cada cuadrante i:

a) Se realiza un paso de la transformada wavelet directa discreta para obtener las subbandas LL_i, LH_i, HL_i y HH_i.

b) Las subbandas LH_i, HL_i y HH_i generadas se agrupan en tres code-blocks, donde cada uno de ellos se cuantiza y se codifica entrópicamente de forma separada.

3.- Se recombinan las subbandas pasabaja LL_i de los cuadrantes formando una textura de menor nivel de resolución (el siguiente nivel de la pirámide de cuadrantes).

4- Si la textura es mayor que el tamaño de cuadrante fijado, se vuelve al paso 1.

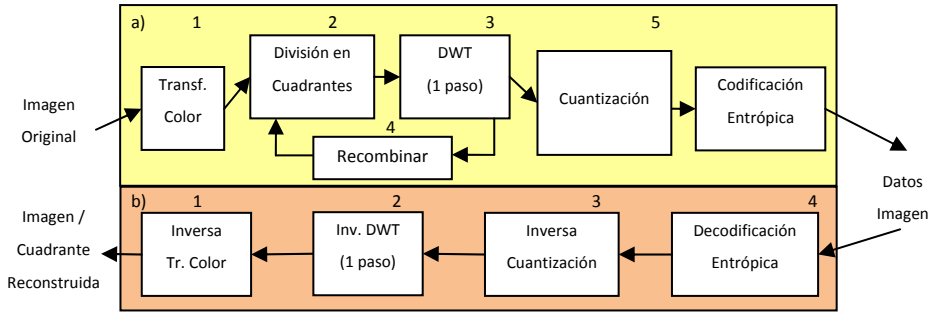


Figura 4.8 – a) Esquema de compresión, b) Esquema de reconstrucción

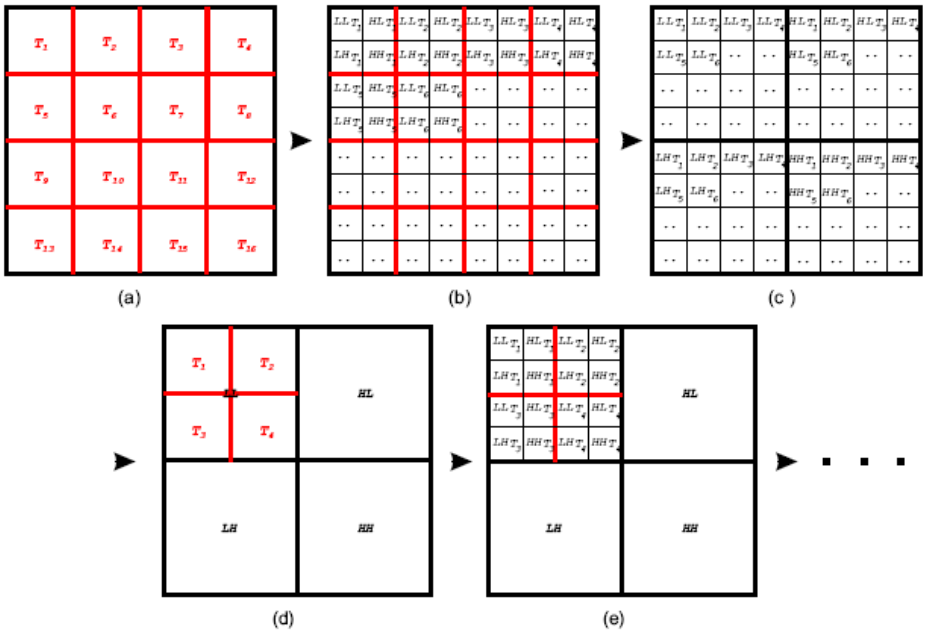


Figura 4.9 – Proceso de descomposición wavelet empleado en el nuevo esquema. (a) División inicial en cuadrantes, (b) Un paso de la DWT en cada cuadrante, (c) Reordenación de subbandas, (d) Segunda división en cuadrantes, (d) Un paso de la DWT en cada cuadrante, etc.

Para reconstruir estos cuadrantes se realiza el proceso inverso de forma progresiva, aplicando los mismos pasos que en el esquema JPEG2000 clásico (Figura 4.8 - b). Una vez que un cuadrante ha sido reconstruido, cualquiera de los cuadrantes hijos puede ser reconstruido simplemente descomprimiendo y aplicando un paso de la transformada wavelet a los coeficientes de las subbandas HL, LH y HH del cuadrante hijo, y usando parte de la información del cuadrante padre como los coeficientes de la subbanda LL.

Esta reconstrucción de cuadrantes es perfecta, sin artefactos visuales en las fronteras, debido a que se dispone de los mismos coeficientes para realizar el proceso de síntesis de la transformada wavelet que los que fueron usados en el proceso de análisis, evitando el uso de coeficientes que pertenecen a cuadrantes vecinos en ambas operaciones. Este hecho, junto al de que los coeficientes de las subbandas LL, HL, LH y HH de cada cuadrante son codificados entrópicamente en code-blocks sin usar información correspondiente a otros cuadrantes o subbandas, hace posible la reconstrucción de los cuadrantes sin errores.

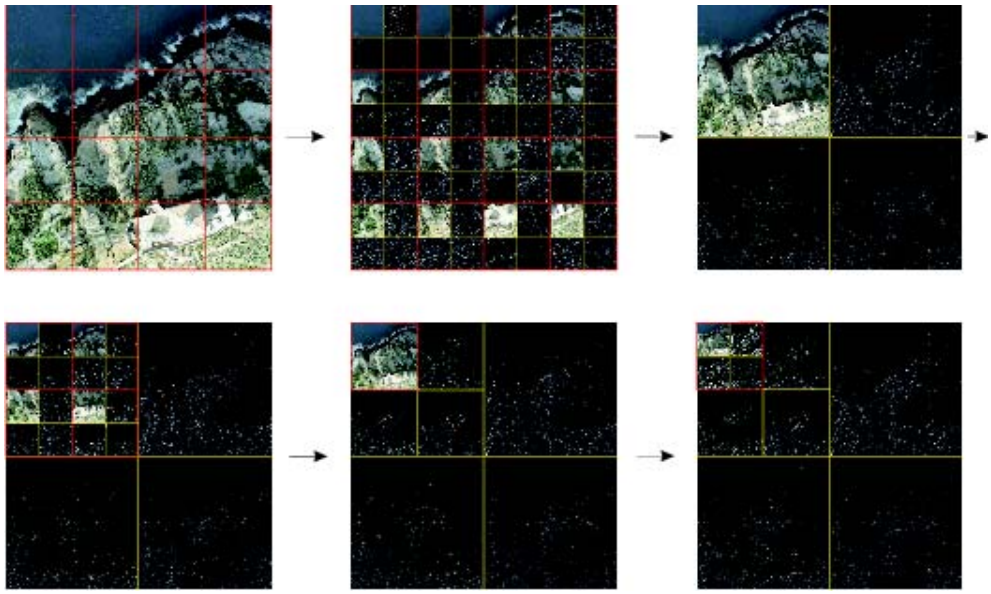


Figura 4.10 – Proceso de descomposición wavelet del nuevo esquema en una imagen.

Implementación Realizada

Para realizar la implementación de este esquema de compresión se ha empleado como base la librería estándar kakadu [71], que realiza una implementación del estándar Jpeg2000. Partiendo de esta librería, se han modificado varios de sus elementos, entre ellos, los procesos de análisis y síntesis de la transformada wavelet estándar empleada por Jpeg2000, para que realicen las tareas especificadas por el nuevo motor de compresión y reconstrucción especificado en la Figura 4.8.

4.3 – Metodología de Evaluación

Se ha implementado una aplicación de prueba, usando el lenguaje de programación C++, sobre la cual se integran los dos esquemas de compresión que se van a evaluar: Jpeg2000 clásico y el nuevo esquema.

Para realizar la evaluación, inicialmente se procederá a la compresión de un conjunto heterogéneo de imágenes de texturas y de información de las alturas del terreno codificadas como imágenes empleando los dos esquemas de compresión analizados. Además, dentro del esquema Jpeg2000 clásico, se realizarán dos tipos de compresión, una empleando la división en cuadrantes que incorpora este esquema y otra sin hacer uso de esa división.

Una vez obtenidas las imágenes comprimidas, se procederá a la reconstrucción de las mismas empleando diferentes estrategias. Para el nuevo esquema de compresión se empleará una estrategia en la que cada una de las imágenes se divide en cuadrantes de un tamaño predeterminado, que simboliza la unidad mínima de intercambio de información entre el cliente y el servidor de un sistema de visualización remota de información de terrenos. Se simulará el envío de estos cuadrantes de forma independiente para evaluar la calidad visual de las reconstrucciones. Esta estrategia se empleará también para el esquema clásico Jpeg2000, con el que además se empleará otra estrategia que consiste en reconstruir la imagen por niveles de resolución de forma progresiva hasta alcanzar la imagen de máxima resolución. Esta estrategia no sirve para un sistema de visualización remoto como el desarrollado en esta tesis, ya que supondría descargar toda la información del terreno en el cliente sin realizar ningún tipo de selección de los datos en función de lo que el usuario está visualizando en cada momento, pero servirá para estudiar el comportamiento del nuevo esquema frente al caso de uso habitual del esquema de Jpeg2000 clásico.

El esquema de la arquitectura de la aplicación de prueba es el siguiente:

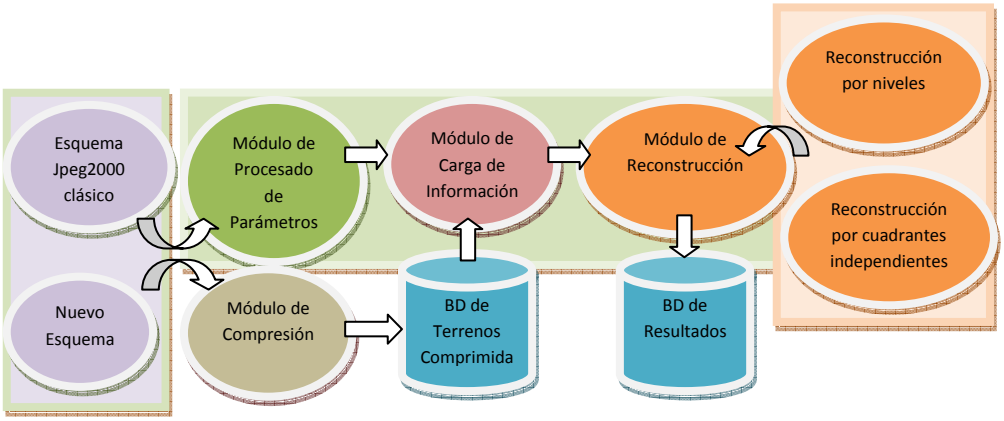


Figura 4.11 – Módulos y esquema de funcionamiento del sistema de prueba

- Módulo de Compresión: este módulo se encarga de realizar la compresión de todo el conjunto de imágenes que se van a emplear en las pruebas. Comprimirá las imágenes empleando los dos esquemas de compresión, Jpeg2000 y el nuevo esquema que se ha desarrollado. Se llevarán a cabo compresiones con pérdidas y sin pérdidas, y para el caso de Jpeg2000 clásico, empleando la división en cuadrantes que incorpora el algoritmo y sin emplearla. Todas las imágenes comprimidas resultantes se almacenan en una base de datos.

- Módulo de procesado de parámetros iniciales: la aplicación recibe tres parámetros iniciales: el primero será la imagen comprimida con la que va a trabajar la aplicación. El segundo identifica que tipo de compresión se le ha aplicado a esa imagen: es decir, si es una imagen comprimida con el esquema de compresión clásico Jpeg2000 o con el nuevo esquema de compresión. El tercer parámetro indicará si se realiza una reconstrucción por niveles de la imagen, o se realiza una reconstrucción de la misma por cuadrantes. Dependiendo de estos parámetros, este módulo configura de forma adecuada el entorno de ejecución de la aplicación.

- Módulo de Carga de Información: este módulo carga en memoria la imagen comprimida siguiendo las directrices especificadas en los parámetros iniciales.

- Módulo de Reconstrucción: este módulo se encarga de realizar la reconstrucción de la imagen comprimida. Como se ha comentado antes, las imágenes comprimidas con el nuevo esquema se reconstruirán a partir de la reconstrucción por cuadrantes independientes, y para el esquema de Jpeg2000 clásico, además de esta estrategia, se empleará otra en la que la reconstrucción de realizara por niveles, donde para la reconstrucción se emplean todos los coeficientes wavelet de cada nivel de resolución. Empleando estas estrategias se reconstruirá el conjunto de imágenes, que se almacenarán en una base de datos para, posteriormente, obtener la calidad visual que proporcionan a través del cálculo del error cuadrático medio (RMSE) de las reconstrucciones.

Datos Empleados

Para probar la eficiencia de este nuevo esquema se han comprimido un conjunto heterogéneo de imágenes que representan texturas y alturas del terreno empleando el esquema jpeg2000 tradicional y el nuevo esquema. Las texturas han sido extraídas a partir de ortofotos de terrenos extensos, mientras que las alturas se han obtenido a partir de los puntos de rejillas regulares de alturas generadas a partir de datos extraídos por satélite.

Se emplearán imágenes de distinto tamaño y con diferentes tamaños de cuadrantes y de code-blocks. Las imágenes que se muestran como ejemplo en las figuras del apartado de resultados no son excesivamente grandes (entre 2048x2048 y 16384x16384 pixels), con el objetivo de que se aprecien más fácilmente los artefactos visuales que se producen en las fronteras de los cuadrantes cuando se emplea el estándar clásico de Jpeg2000.

Métrica empleada

Para evaluar qué esquema proporciona unas reconstrucciones de mejor calidad visual dentro de un sistema de visualización de terrenos distribuido, se comprimirán y reconstruirán un conjunto de imágenes, calculando las diferencias entre las imágenes reconstruidas y las imágenes originales. Las diferencias entre estas imágenes se medirán empleando el error cuadrático medio (RMSE).

Para evaluar la capacidad de compresión de cada esquema de compresión, se comprimirá un conjunto de imágenes calculando la tasa de compresión alcanzada para cada una de ellas.

Pruebas

En los esquemas de compresión que emplean la transformada wavelet, como en el caso de Jpeg2000 y el nuevo esquema, el empleo de una compresión con pérdidas va introduciendo errores en cada paso del proceso de análisis de la transformada wavelet, provocando que la imagen final reconstruida, la de máxima resolución y del mismo tamaño que la imagen de partida, pueda ser bastante diferente de la original. Es por ello que, generalmente, para el caso de terrenos, se emplea una compresión sin pérdidas para evitar la degradación progresiva de los niveles de detalle del terreno.

Por otro lado, en un sistema de visualización de terrenos, la reconstrucción de las imágenes se realiza a partir de la reconstrucción de regiones (cuadrantes) del terreno de forma independiente. Esto provoca, como se comentó en el apartado 4.1, que aún empleando una compresión sin pérdidas, si se emplea el esquema de compresión Jpeg2000 clásico, aparezcan artefactos visuales en las fronteras de esas regiones, lo que afecta a la calidad visual de las reconstrucciones.

Por lo tanto, se va a diseñar una primera prueba para evaluar el alcance de estos artefactos visuales en las reconstrucciones. Para ello, se procederá a comprimir sin pérdidas un conjunto de imágenes empleando los dos esquemas de compresión para, posteriormente, obtener las imágenes finales a partir de la reconstrucción independiente de las regiones de la imagen. La calidad de las imágenes reconstruidas se medirá cualitativa y cuantitativamente, comparando el efecto visual producido por la aparición de los artefactos visuales y el error cuadrático medio de las reconstrucciones.

Dentro de un sistema de visualización de terrenos remoto, otro punto importante a tener en cuenta es el tamaño de los datos cuando se almacenan en disco y el tiempo necesario para la transmisión de los mismos, el cual estará directamente relacionado con su tamaño. Dado que la cantidad de información de las bases de datos de terrenos es considerable, se hace necesario comprimir esta información de la forma más eficiente posible, es decir, consiguiendo la mayor tasa de compresión posible. Por ello, otra de las pruebas que se llevará a cabo, es la comparación de las tasas de compresión obtenidas en la compresión sin pérdidas de las imágenes de prueba empleando los dos esquemas de compresión evaluados.

En ocasiones, en un sistema de visualización de terrenos remoto, un cliente puede disponer de una línea de conexión con un ancho de bando bajo, o el dispositivo que se está utilizando para visualizar la información puede tener una capacidad de almacenamiento reducida (como puede ser una pda o un móvil). En estos casos, puede ser conveniente reducir aún más el tamaño de la información de terrenos empleando una compresión con pérdidas, sacrificando un poco la calidad visual y precisión de los datos del terreno. Para comprobar qué esquema de compresión proporciona reconstrucciones de mejor calidad visual cuando se emplea compresión con pérdidas, se va a realizar otra prueba donde se comprimirá y

reconstruirá con ambos esquemas, un conjunto de imágenes utilizando diferentes tasas de compresión, empleando de nuevo el error cuadrático medio de las reconstrucciones como medida de la calidad visual obtenida.

Fuera ya del ámbito de un sistema de visualización de terrenos remoto, el nuevo esquema de compresión que hemos desarrollado también podría ser de utilidad en los campos en los que se utiliza habitualmente el esquema de Jpeg2000; por ejemplo, en la transmisión y visualización progresiva de una imagen por niveles de resolución, donde se adelanta la visualización final de una imagen enviando y mostrando progresivamente la imagen desde el nivel de menor resolución hasta el de mayor resolución, que corresponde a la imagen final reconstruida. En este ámbito, el esquema de Jpeg2000 clásico emplea una reconstrucción por niveles de las imágenes comprimidas para reconstruir la imagen original (en vez de por regiones independientes, evitando la aparición de los artefactos visuales que se producían en las fronteras de las regiones). Para comprobar la utilidad del nuevo esquema en este ámbito, se va a realizar también una prueba para comparar la calidad visual de los niveles de detalle reconstruidos utilizando los dos esquemas. La reconstrucción de los niveles de detalle intermedios de cada imagen original, proporciona imágenes de tamaño inferior al de la imagen original, por lo que, para poder calcular el error cuadrático, se amplían aplicándoles un filtro de interpolación bilineal hasta alcanzar el mismo tamaño que la imagen de referencia (la imagen original).

4.4 – Resultados

A continuación se van a estudiar los resultados obtenidos en las pruebas enunciadas en el apartado anterior.

4.4.1 – Artefactos Visuales

Como se comentó en el apartado 4.1.1.6, en un sistema de visualización de terrenos con bases de datos remotas, la reconstrucción de las regiones del terreno se realiza de forma independiente. Esto provoca que, si se emplea el esquema de compresión Jpeg2000 clásico, aparezcan artefactos visuales en las fronteras de esas regiones, afectando a la calidad visual.

En esta prueba se va a comprobar cómo este problema afecta cualitativa y cuantitativamente a la calidad visual de las imágenes reconstruidas. Para ello, se han comprimido un conjunto de imágenes sin pérdidas empleando los dos esquemas de compresión para, posteriormente, reconstruir esas imágenes por regiones de forma independiente.

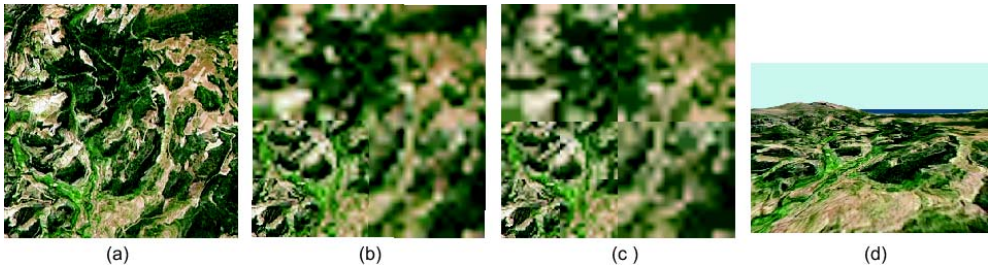


Figura 4.12 – a) Textura original, b) Reconstrucción empleando el nuevo esquema, c) Reconstrucción empleando el esquema clásico de Jpeg2000, d) Representación 3D en tiempo real empleando el nuevo esquema.

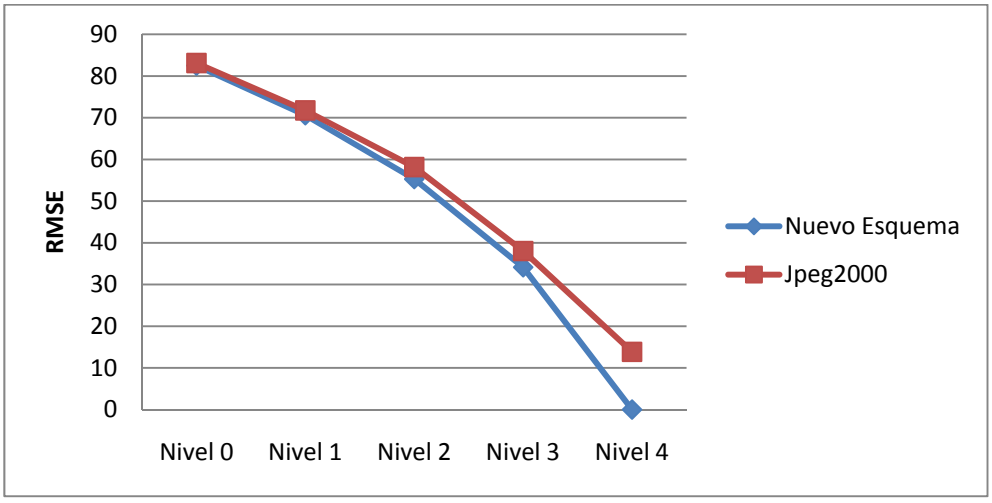
En la Figura 4.12 se muestra una reconstrucción de una superficie del terreno (a) de tamaño 2048x2048 pixels, en la que la máxima resolución se encontraría en la esquina inferior izquierda, que es donde se encuentra el usuario visualizando la escena (d). El tamaño de los cuadrantes empleado es de 128x128 pixels, con un tamaño de precintos de 64x64 pixels. Tal como se define en el estándar Jpeg2000 [68], un precinto está formado por un triplete de información espacial relacionada que permite el acceso aleatorio y la reconstrucción de una región del terreno.

Se puede observar que, empleando el esquema de compresión de Jpeg2000 clásico (c), aparecen artefactos visuales entre las fronteras de las regiones del mismo nivel de resolución, cosa que no ocurre si el emplea el nuevo método de compresión elaborado (b).

Para evaluar la calidad visual de forma cuantitativa se ha procedido a reconstruir la imagen de la superficie del terreno mostrada en la Figura 4.12 por niveles de la pirámide de cuadrantes (ver apartado 2.2.2) que la almacena. Se reconstruirá cada región que forma parte de cada nivel de forma independiente, calculando el error que se produce en la reconstrucción de la imagen que representa a todo el nivel respecto a la imagen original. Las imágenes se han comprimido empleando cuatro niveles de resolución, donde el nivel 0 se corresponde con el nivel de menor resolución, y el nivel 4 se corresponde con la reconstrucción completa de la imagen.

	Nuevo Esquema RMSE	Jpeg2000 RMSE
Nivel 0	82,51	83,10
Nivel 1	70,61	71,76
Nivel 2	55,29	58,17
Nivel 3	34,14	38,03
Nivel 4	0,00	13,84

Tabla 4.1 – RMSE – Figura 4.12



Gráfica 4.1 – RMSE – Figura 4.12

En la Tabla 4.1 y en la Gráfica 4.1 se observa que el error que se produce empleando el esquema de Jpeg2000 clásico es superior al obtenido utilizando el nuevo esquema. Además, cuando se emplea el nuevo esquema se obtiene una reconstrucción completa de la imagen perfecta, cosa que no ocurre con el esquema clásico.

Esto se debe a que, como se comentó en el apartado 4.1.1.6, cuando se emplea el esquema de compresión Jpeg2000 clásico para reconstruir regiones del terreno de forma independiente, aparecen artefactos visuales en las fronteras de las regiones, debido a que sólo se dispone de los coeficientes de la región que se va a reconstruir para realizar el proceso de síntesis, cuando serían necesarios también los coeficientes de las regiones vecinas. Estos artefactos visuales se van multiplicando en cada proceso de síntesis de la transformada wavelet, provocando que la reconstrucción completa de la imagen no sea perfecta.

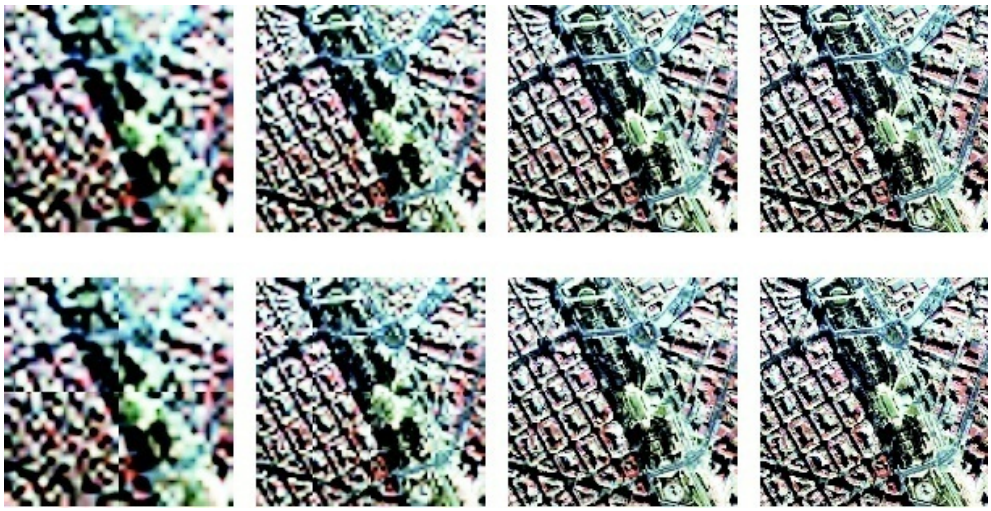


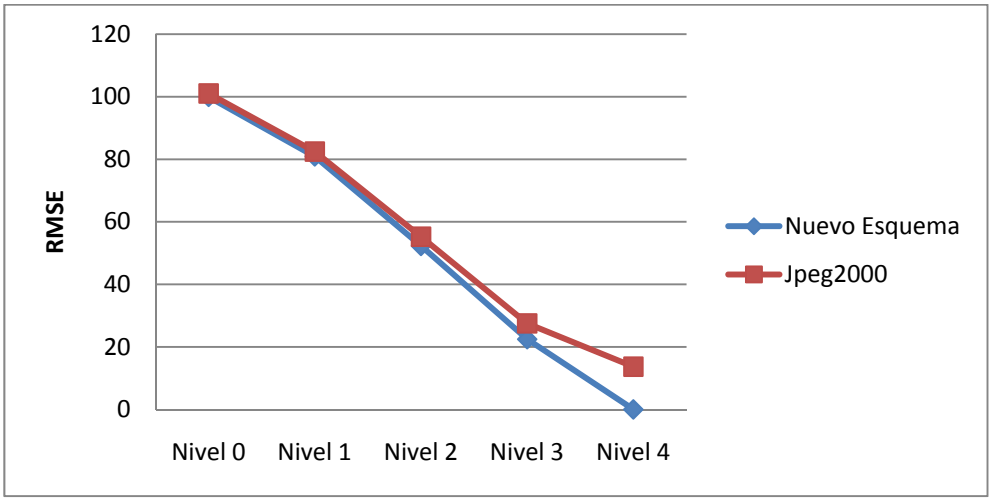
Figura 4.13 – Reconstrucción de los niveles 0, 1, 2 y 3 de una pirámide de cuadrantes empleando el nuevo esquema (arriba) y usando el esquema clásico de Jpeg2000 (abajo)

Para la Figura 4.13 se ha realizado el mismo proceso de reconstrucción por niveles aplicado a la Figura 4.12. El tamaño de la imagen original es de 2048x2048 pixels, con un tamaño de cuadrantes de 128x128 pixels, y unos precintos de 64x64 pixels.

En las imágenes se pueden observar los artefactos visuales que aparecen en las fronteras de las regiones vecinas con el mismo nivel de resolución, cuando se emplea el esquema clásico de Jpeg2000 (Figura 4.13 - abajo). Estos artefactos no aparecen empleando el nuevo esquema de compresión (Figura 4.13 - arriba).

	Nuevo Esquema RMSE	Jpeg2000 RMSE
Nivel 0	99,97	101,04
Nivel 1	80,92	82,46
Nivel 2	52,29	55,23
Nivel 3	22,46	27,51
Nivel 4	0,00	13,67

Tabla 4.2 – RMSE – Figura 4.13



Gráfica 4.2 – RMSE – Figura 4.13

En la Tabla 4.2 y en la Gráfica 4.2 se puede observar cuantitativamente, que el error que se produce en cada uno de los niveles de reconstrucción de la pirámide de cuadrantes es menor empleando el nuevo esquema de compresión, el cual realiza una reconstrucción completa de la imagen perfecta, cosa que no ocurre empleando el esquema clásico de Jpeg2000.

El motivo de que se produzcan estos resultados es el mismo que el que se ha comentado para la Tabla 4.1 y la Gráfica 4.1, la aparición de artefactos visuales en las fronteras de las regiones cuando se emplea el esquema clásico de jpeg2000 y se reconstruyen estas regiones de forma independiente, debido a que al realizar el proceso de síntesis sólo se dispone de los coeficientes de la región reconstruida, faltando los coeficientes de las regiones vecinas.

4.4.2 – Tasa de Compresión

Como se ha comentado previamente, en un sistema de visualización de terrenos remoto, el tamaño de los datos es un factor importante a tener en cuenta porque afecta tanto a la cantidad de almacenamiento necesaria, como al ancho de banda y el tiempo requerido para transmitir la información. Por eso es importante alcanzar la mayor tasa de compresión posible.

Con esta prueba se va a estudiar la tasa de compresión que puede obtener cada esquema de compresión. Para ello, se ha comprimido con cada esquema un conjunto de imágenes de varios tamaños empleando diferentes tamaños de cuadrantes (Figura 4.14 y Figura 4.15). Para el esquema clásico de Jpeg2000 se comprimirán las imágenes utilizando la división por cuadrantes que implementa y sin utilizarla.

Para la compresión de Jpeg2000 sin utilizar cuadrantes, obviamente no se empleará ningún tamaño de cuadrante, lo que si variará será el tamaño del precinto, que siguiendo las recomendaciones del estándar Jpeg2000, se ha mantenido entre 16x16 y 64x64.

Las imágenes empleadas para realizar la compresión son las siguientes:

Figura 4.14 – Imagen A (izquierda) e Imagen B (derecha)

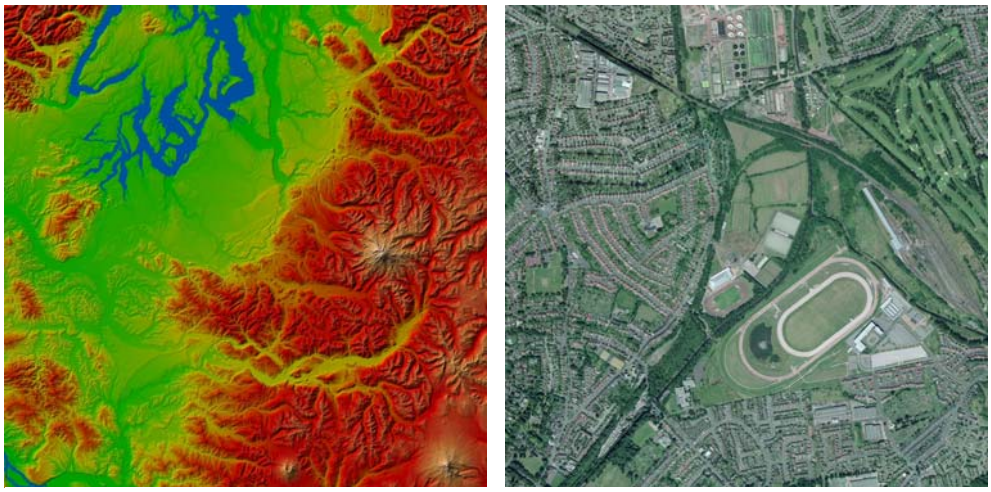
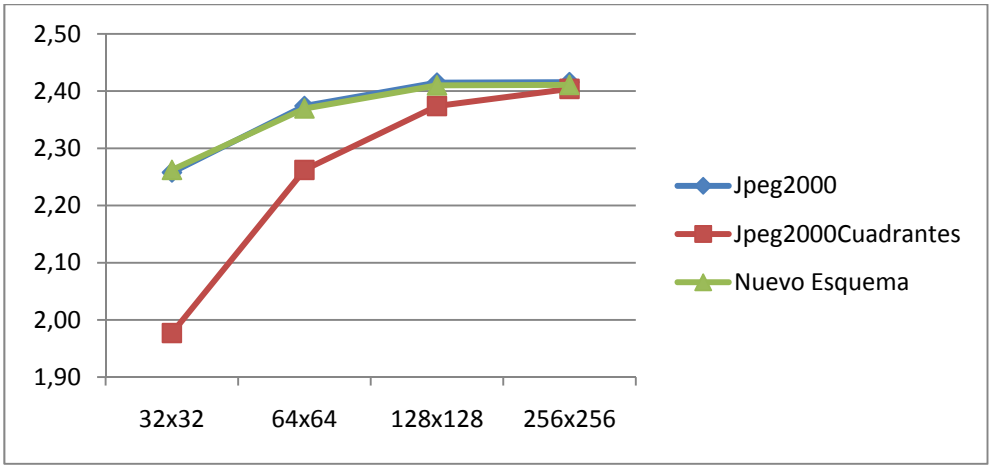


Figura 4.15 – Imagen C (izquierda) e Imagen D (derecha)

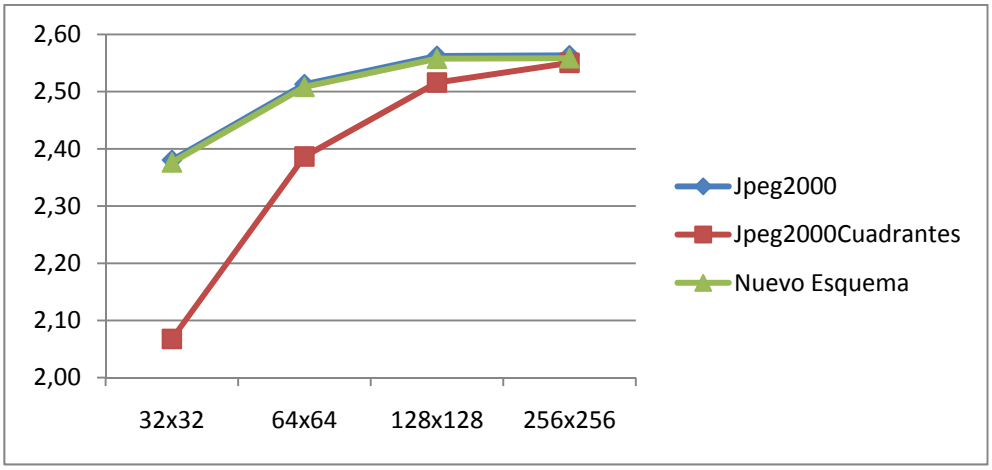
Tamaño Cuadrante	Tam. Prec.	Esquema Compresión	Imagen A (2048x2048)	Imagen B (4096x4096)	Imagen C (8192x8192)	Imagen D (16384x16384)
32 x 32	16 x 16	Jpeg2000	2,26	2,38	2,24	2,55
		Jpeg2000 Cuadrantes	1,98	2,07		
		Nuevo Esquema	2,26	2,38	2,23	2,55
64 x 64	32 x 32	Jpeg2000	2,37	2,51	2,34	2,68
		Jpeg2000 Cuadrantes	2,26	2,39	2,24	
		Nuevo Esquema	2,37	2,51	2,34	2,68
128 x 128	64 x 64	Jpeg2000	2,41	2,56	2,37	2,73
		Jpeg2000 Cuadrantes	2,37	2,52	2,34	2,68
		Nuevo Esquema	2,41	2,56	2,37	2,73
256 x 256	64 x 64	Jpeg2000	2,42	2,56	2,37	2,73
		Jpeg2000 Cuadrantes	2,40	2,55	2,36	2,72
		Nuevo Esquema	2,41	2,56	2,37	2,73

Tabla 4.3 – Tasa de compresión de las imágenes comprimidas con los diferentes esquemas de compresión

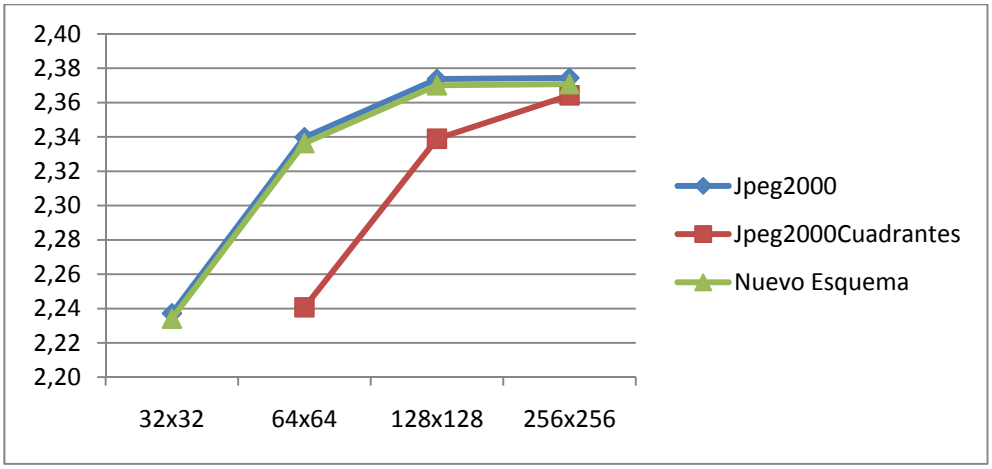
Los resultados de esta tabla se pueden observar mejor en las siguientes gráficas:



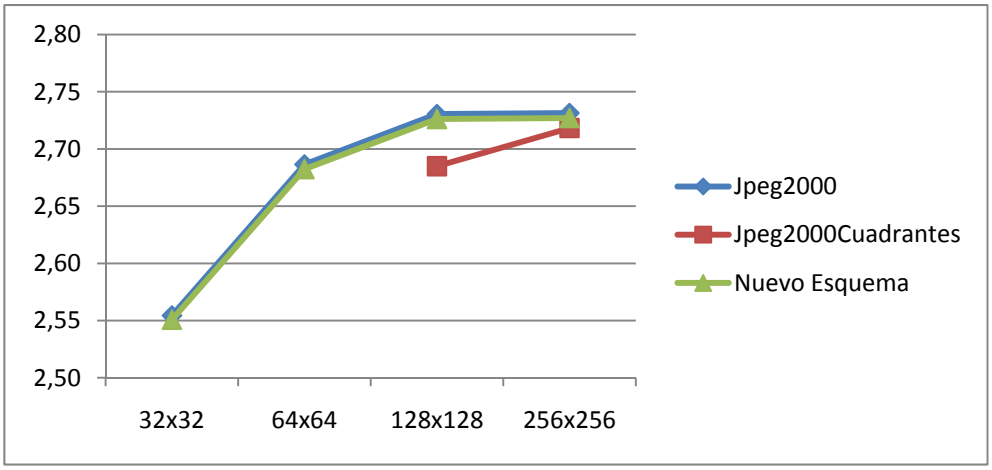
Gráfica 4.3 – Tasa de compresión de la Imagen A comprimida con los diferentes esquemas de compresión



Gráfica 4.4 – Tasa de compresión de la Imagen B comprimida con los diferentes esquemas de compresión



Gráfica 4.5 – Tasa de compresión de la Imagen C comprimida con los diferentes esquemas de compresión



Gráfica 4.6 – Tasa de compresión de la Imagen D comprimida con los diferentes esquemas de compresión

Se puede observar en la Tabla 4.3 y en las gráficas anteriores (Gráfica 4.3, Gráfica 4.4, Gráfica 4.5 y Gráfica 4.6), que las tasas de compresión obtenidas empleando el nuevo esquema son superiores a las obtenidas empleando el esquema clásico Jpeg2000 cuando emplea la división por cuadrantes. Esto se debe a que el esquema Jpeg2000 trata cada cuadrante como si fuera una imagen independiente, lo que limita la cantidad de información disponible a la hora de ser comprimida.

Además, también se observa que el nuevo esquema obtiene unas tasas de compresión similares al esquema Jpeg2000 clásico cuando no emplea división en cuadrantes, debido a

que en ambos casos, el proceso de compresión utilizado no difiere en exceso, aplicándose el mismo número de veces la transformada wavelet de forma recursiva sobre los datos.

En la Tabla 4.3, también se observa que a medida que el tamaño de los cuadrantes aumenta, la tasa de compresión también aumenta. Sin embargo, este incremento es cada vez menor, siendo la diferencia muy pequeña entre los cuadrantes de tamaño 128x128 y 256x256.

Cabe mencionar que el número máximo de cuadrantes en los que el esquema de Jpeg2000 clásico permite dividir un imagen es de 65535, por eso no ha sido posible comprimir las texturas C Y D empleando cuadrantes de 32x32 pixels, ni la textura D empleando cuadrantes de 64x64, ya que se superaba este número máximo.

4.4.3 – Compresión con pérdidas

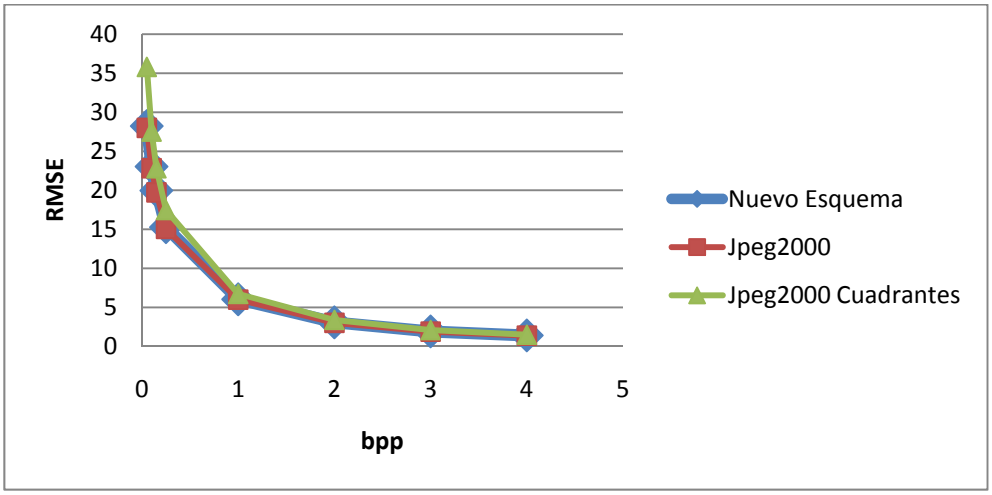
El uso de compresión con pérdidas en un esquema de compresión que emplea la transformada wavelet, introduce errores en cada paso del proceso de análisis de la transformada, provocando que la imagen final reconstruida pueda diferir bastante de la original. Por eso, generalmente no se emplea este tipo de compresión para el caso de la visualización de terrenos, donde queremos conseguir la máxima calidad visual posible. Sin embargo, en ocasiones podemos encontrarnos con sistemas de visualización como pdas o móviles, donde la capacidad de almacenamiento es muy limitada, y puede ser adecuado emplear compresión con pérdidas para reducir al máximo el tamaño de la información a visualizar, a pesar de perder cierta calidad visual en el proceso.

Para estudiar qué esquema de compresión proporciona una mayor calidad visual cuando se emplea compresión con pérdidas, se van a comprimir un conjunto de imágenes utilizando diferentes tasas de compresión con pérdidas medidas en bits por pixel (bpp). Se emplearán para ello, los esquemas de compresión Jpeg2000 clásico utilizando cuadrantes y sin utilizarlos, y el nuevo esquema. Después, se medirá el error cuadrático medio (RMSE) que se comete en la reconstrucción de las imágenes respecto a la imagen original.

Figura 4.16 - Imágenes comprimidas con compresión con pérdidas y reconstruidas por niveles. Jpeg2000 empleando cuadrantes (arriba), el nuevo esquema (medio), Jpeg2000 sin emplear cuadrantes (abajo). Ratio de compresión: 0.05 (izquierda) y 0.10 (derecha).

	Nuevo Esquema RMSE	Jpeg2000 RMSE	Jpeg2000 Cuadrantes RMSE
0,05	28,22	27,99	35,80
0,10	23,03	22,83	27,52
0,15	19,95	19,73	22,82
0,25	15,26	15,05	17,47
1,00	6,03	5,98	6,72
2,00	3,07	3,02	3,39
3,00	1,91	1,88	2,09
4,00	1,40	1,38	1,49

Tabla 4.4 – RMSE – Compresión con pérdidas



Gráfica 4.7 – RMSE – Compresión con pérdidas

En la Figura 4.16 se muestra una imagen (Figura 4.14 – Imagen A) comprimida empleando compresión con pérdidas y posteriormente reconstruida por niveles. Cada fila corresponde a un algoritmo de compresión: la primera fila se corresponde con el algoritmo Jpeg2000 clásico empleando cuadrantes, la segunda fila se corresponde con el nuevo esquema de compresión, y la última con el esquema de Jpeg2000 clásico sin emplear cuadrantes. Se han empleado diferentes ratios de compresión, en la columna de la izquierda 0.05 y en la columna de la derecha 0.10. El tamaño de la imagen empleada es de 2048x2048 pixels, con un tamaño de cuadrante de 128x128 pixels y un tamaño de precintos de 64x64 pixels.

En esta figura se puede observar visualmente que para valores bajos de bpp, el empleo de jpeg2000 con cuadrantes ofrece reconstrucciones de las imágenes de menor calidad visual que cuando se emplean los otros dos esquemas. Si bien, esta diferencia va disminuyendo conforme se incrementa el valor de bpp. Los otros dos esquemas obtienen valores similares.

En la Tabla 4.4 y en la Gráfica 4.7 se puede observar cuantitativamente lo que se ha observado cualitativamente en la Figura 4.16 que existe una diferencia en la calidad de las imágenes reconstruidas al emplear el esquema jpeg2000 clásico con cuadrantes respecto a las otras dos opciones, y que conforme se incrementa el valor de los bpp empleado, esta diferencia disminuye.

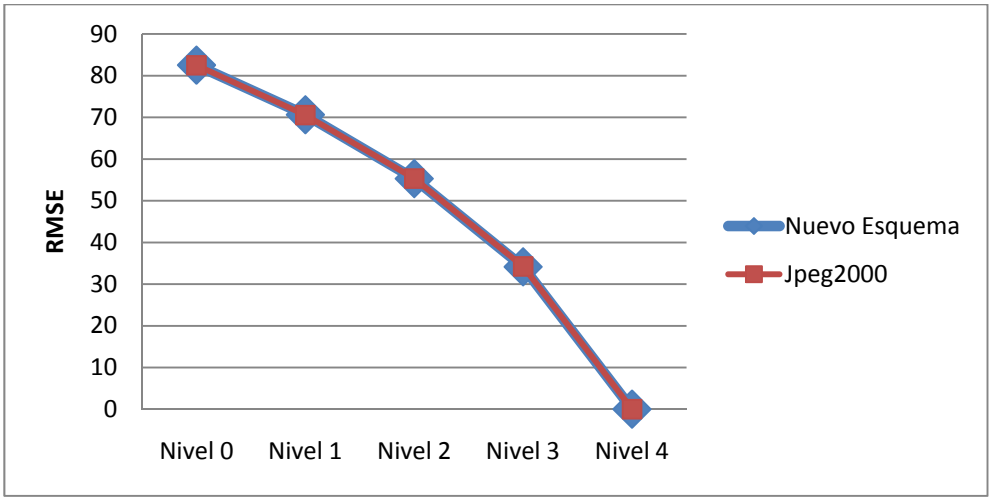
4.4.4 – Reconstrucción por niveles

Esta prueba se plantea fuera del ámbito de la visualización de terrenos. Con ella, se desea observar si el nuevo esquema de compresión obtiene la misma calidad visual que el esquema de Jpeg2000 clásico cuando se usa en otro conjunto de aplicaciones en los que no es necesario una reconstrucción por regiones independientes, sino que se emplea una reconstrucción por niveles de detalle que no provoca la aparición de artefactos visuales cuando se utiliza el esquema de Jpeg2000 clásico.

Para ello, se ha procedido a reconstruir las imágenes utilizadas en el apartado 4.4.1 (Figura 4.12 y Figura 4.13), pero esta vez, para el caso de Jpeg2000 clásico, se ha realizado la reconstrucción de la imagen por niveles completos, sin realizar la reconstrucción de las regiones que forman parte de cada nivel de forma independiente, de esa forma, no deben aparecer artefactos visuales.

	Nuevo Esquema RMSE	Jpeg2000 RMSE
Nivel 0	82,51	82,48
Nivel 1	70,61	70,54
Nivel 2	55,29	55,32
Nivel 3	34,14	34,25
Nivel 4	0,00	0,00

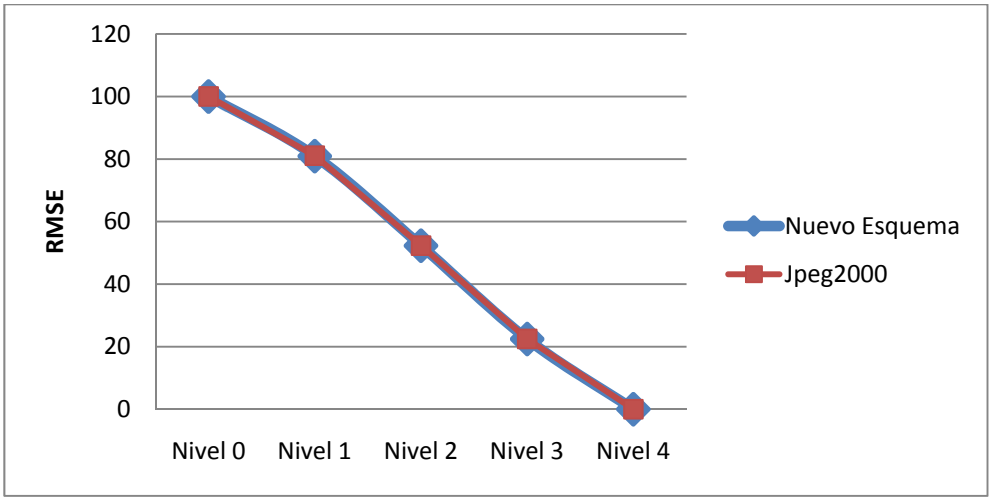
Tabla 4.5 – RMSE – Figura 4.12



Gráfica 4.8 – RMSE – Figura 4.12

	Nuevo Esquema RMSE	Jpeg2000 RMSE
Nivel 0	99,97	100,02
Nivel 1	80,92	81,06
Nivel 2	52,29	52,3
Nivel 3	22,46	22,48
Nivel 4	0,00	0,00

Tabla 4.6 – RMSE – Figura 4.13



Gráfica 4.9 – RMSE – Figura 4.13

Como se observa en la Tabla 4.5 y la Tabla 4.6 y en la Gráfica 4.8 y la Gráfica 4.9, el error que se comete al realizar la reconstrucción de las imágenes por niveles es muy parecido empleando ambos esquemas. También se observa que ambos esquemas son capaces de obtener una reconstrucción total de la imagen perfecta.

4.5 – Conclusiones

Se ha realizado una comparativa del nuevo esquema de compresión que se ha desarrollado y del esquema de Jpeg2000 clásico, empleando para este último la funcionalidad de división por cuadrantes que tiene definida dentro del algoritmo y sin utilizarla, con el objetivo de ver cuál de estas tres posibles soluciones se adapta mejor a una aplicación de visualización de terrenos remotos.

De los resultados se puede extraer que, cuando se realiza una reconstrucción del terreno a partir de regiones reconstruidas de forma independiente, tal cual se realiza en las aplicaciones de visualización de terrenos remotos, si se emplea el esquema clásico de Jpeg2000 aparecen artefactos visuales claramente visibles en las fronteras de esas regiones, que se van multiplicando en cada paso del proceso de síntesis de la transformada wavelet. Esto hace que, no sólo no se pueda obtener una reconstrucción perfecta de la imagen a máxima resolución, sino que además provoca que el resultado final sea de una calidad visual pobre.

Como se comentó en el apartado 4.1.1.6, estos artefactos visuales se producen porque al realizar la reconstrucción de las regiones del terreno de forma independiente, no se dispone de los coeficientes wavelets de las fronteras de las regiones vecinas. Este hecho, ya sería de por sí, un motivo para descartar el uso del esquema Jpeg2000 clásico en un sistema de visualización de terrenos en tiempo real.

En el caso del nuevo esquema, este problema no se reproduce en las fronteras entre cuadrantes, ya que se ha modificado el mecanismo de descomposición wavelet de forma que en el proceso de síntesis se emplea el mismo conjunto de coeficientes wavelet que se generaron en el proceso de análisis, por lo que puede reconstruirse la región del cuadrante de forma independiente al resto de cuadrantes sin pérdida de información.

Por otro lado, en los resultados se puede observar también que, cuando se usa compresión sin pérdidas, la tasa de compresión alcanzada por el nuevo esquema es superior a la que es capaz de obtener el esquema de Jpeg2000 empleando división por cuadrantes. Esto se debe principalmente a que Jpeg2000 trata cada cuadrante como una imagen independiente. Dado que estos cuadrantes son de un tamaño más pequeño que la imagen original, el esquema de compresión dispone de una menor cantidad de información de la cual extraer semejanzas para llevar a cabo la compresión. Además, usando la división por cuadrantes definida en Jpeg2000, el número de veces que se podrá aplicar la transformada wavelet de forma recursiva sobre el cuadrante será menor que el número de veces que se puede aplicar a la imagen completa, lo que también influye en la tasa de compresión. Para que el esquema Jpeg2000 pueda alcanzar unas tasas de compresión similares al nuevo esquema, tendría que

evitar el uso de la división por cuadrantes, lo que provocaría, como hemos visto en el apartado de resultados, la aparición de artefactos visuales.

En los resultados mostrados en el apartado anterior, también se puede observar como cuando se emplea compresión con pérdidas, para un mismo factor de compresión, el esquema Jpeg2000 empleando división por cuadrantes obtiene reconstrucciones de las imágenes de una calidad visual inferior a las del nuevo esquema. Ello es debido, al igual que en el caso de la compresión sin pérdidas, al hecho de que el esquema Jpeg2000 trate cada cuadrante como una imagen independiente, de forma que se limita la capacidad de compresión que puede alcanzar. De nuevo, para que el esquema Jpeg2000 logre una reconstrucción de las imágenes de la misma calidad que el nuevo esquema, no ha de emplear la división en cuadrantes, lo que provoca, dentro del contexto de una aplicación de visualización remota, la aparición de artefactos visuales sobre las fronteras de las regiones que se reconstruyen de forma independiente.

Si se realiza una reconstrucción por niveles de detalle de las imágenes, situación que como hemos comentado antes, no suele plantearse dentro de una aplicación de visualización de terrenos con bases de datos remotas, pero sí dentro de otros tipos de aplicación multimedia, se observa que tanto empleando compresión con pérdidas como sin pérdidas, ambos esquemas proporcionan una calidad visual similar. Este hecho permitiría poder emplear el nuevo esquema en otros ámbitos en los que sea necesario utilizar la compresión de imágenes, ya que ofrece similares prestaciones que el esquema jpeg2000 clásico.

Por lo tanto, a la vista de los resultados extraídos en las pruebas realizadas, donde se observa que el nuevo esquema de compresión proporciona una tasa de compresión mayor y una calidad visual mejor respecto al esquema Jpeg2000 clásico empleando división en cuadrantes, y dado que el uso del esquema Jpeg2000 clásico sin emplear división en cuadrantes provoca la aparición de artefactos visuales en las fronteras de las regiones cuando se realiza su reconstrucción de forma independiente, nos vamos a decantar por el nuevo esquema de compresión para que sea implementado en el sistema de visualización a desarrollar en esta tesis.

Capítulo 5 – Arquitectura del Sistema

Generalmente, el tamaño de las bases de datos de terrenos empleadas en un sistema de visualización de terrenos remoto es muy elevado, sin embargo, los usuarios de estos sistemas suelen necesitar una parte relativamente pequeña de esa información. Por ello, no suele resultar eficiente, desde el punto de vista del almacenamiento, que cada usuario disponga de toda esa información almacenada de forma local. En vez de ello, parece más conveniente que sólo tenga almacenada la información que realmente va a necesitar.

La estrategia más idónea es almacenar la información completa en algún lugar al que cualquier usuario pueda tener acceso. A priori, el lugar más adecuado es un servidor remoto. Este servidor podría recibir peticiones de información por parte de los usuarios y elaborar mensajes de respuesta con esa información que el usuario almacenaría de forma local para posteriormente visualizarla.

Otra ventaja de almacenar la información del terreno en un servidor remoto sería que permitiría tener un mayor control sobre esa información, de forma que se podrían incorporar mecanismos de acceso restringido a la información o realizar un seguimiento del uso que se hace de la misma. Además, si se añade nueva información, o se corrige o se actualiza la existente, los clientes obtendrían esas modificaciones de forma automática al realizar las peticiones al servidor.

Como se estudió en el capítulo 2.3, la arquitectura típica dentro de un sistema de visualización de terrenos en tiempo real en el que se emplean servidores remotos es la arquitectura cliente-servidor. En esta arquitectura, toda la información se encuentra almacenada en el servidor y es transmitida al cliente para su visualización.

En la actualidad, la facilidad de acceso a una línea de conexión a Internet y el aumento del ancho de banda de estas líneas de conexión, ha provocado que el número de potenciales usuarios que pueden desear acceder a estos sistemas de visualización se incremente de forma considerable. La arquitectura cliente-servidor no está diseñada para adaptarse de forma flexible a un incremento considerable del número de clientes, ya que conforme aumenta éste, la carga de trabajo del servidor aumenta, y el tiempo de respuesta del servidor se alarga, pudiendo superar el límite de tiempo considerado como aceptable, y por consiguiente, no alcanzándose una calidad de servicio aceptable para la visualización interactiva del terreno.

A priori, empleando una arquitectura cliente-servidor, la única forma de evitar este aumento en la carga del servidor es añadiendo nuevos servidores que la hagan frente, lo que supone un aumento en el coste de la adquisición de estos equipos y de su mantenimiento. Además, este aumento del número de servidores puede provocar que la comprobación de la integridad de las bases datos de los diferentes servidores sea más compleja y costosa, teniendo que replicarse en todas ellas cada cambio o información nueva introducida.

Para solucionar este problema de escalabilidad con el número de clientes, una opción que se está empleando en la actualidad en otras áreas como la de los DVE [116] o la de los sistemas de información [1], es utilizar una arquitectura P2P. Esta arquitectura tiene como característica innata la escalabilidad en función del número de nodos o usuarios del sistema, debido a que cada cliente actúa a su vez como servidor. De esta forma, si aumenta el número de peticiones debido al incremento del número de usuarios, como también aumenta el número de nodos que pueden hacerse cargo de esas peticiones, éstas se reparten entre todos ellos manteniendo la carga general del sistema estable. Por lo tanto, una arquitectura P2P es un sistema escalable con el número de clientes y de bajo coste que no necesita el empleo de servidores adicionales para evitar el aumento de la carga del sistema.

Sin embargo, una arquitectura P2P pura presenta algunos problemas cuando se emplea dentro de un sistema de visualización de terrenos en tiempo real. Debido a la extensión de las bases de datos, no se puede asegurar ni que toda la información se encuentre disponible en todo momento, porque para ello habría que garantizar que siempre se encuentren conectados un conjunto de clientes mínimo y que entre todos dispusieran de toda la información de la base de datos, ni que el nodo o nodos que dispongan de esa información puedan enviarla dentro de un límite de tiempo adecuado. Para solventar en parte el problema, se tendría que repartir inicialmente toda la información de la base de datos entre los clientes conectados, cosa que es inviable en la práctica.

Para encontrar una solución válida sea cual sea el número de usuarios conectados al sistema, en esta tesis se ha decidido diseñar una arquitectura mixta cliente-servidor / P2P que herede los puntos fuertes de cada una de estas arquitecturas evitando sus problemas. En esta arquitectura mixta habrá un conjunto de clientes que se intercambiarán la información del terreno de la misma forma que en una arquitectura P2P pura, pero además, existirá un conjunto de servidores tradicionales a los que los clientes podrán acudir cuando los datos no se encuentren en ninguno de estos clientes, o estos no puedan responderlos dentro de un tiempo aceptable. Estos servidores tendrán almacenada toda la información de la base de datos del terreno, garantizando la disponibilidad de toda la información en todo momento y la transmisión de la misma dentro de unos límites de tiempo aceptables. El número de servidores tradicionales que será necesario utilizar, pensamos que será mucho menor que en una arquitectura cliente-servidor pura, ya que la mayor parte de las necesidades de

información de los clientes se resolverán entre ellos mismos, sin la necesidad de acceder a estos servidores.

Las características de esta arquitectura serán las siguientes:

- Ser escalable en función del número de usuarios. Obtener y mantener esta escalabilidad ha de tener un coste computacional y económico bajo.

- Permitir una transmisión rápida y eficiente de la información del terreno.

- Disponibilidad continua de la información de las bases de datos de terrenos.

- Ser compatible con un esquema de compresión que permita el envío de información comprimida de forma progresiva sin redundancias, como por ejemplo, el nuevo esquema de compresión que se ha elaborado en el Capítulo 4.

- Controlar la información, permitiendo emplear mecanismos para restringir el acceso a la información y facilitando las tareas de comprobación de la integridad de los datos y de la actualización de los mismos.

- Facilidad de mantenimiento, haciendo posible reemplazar, reparar, actualizar o trasladar un servidor sin que los clientes vean afectada su calidad de servicio.

-Robustez ante fallos en los servidores o en los clientes, permitiendo obtener la información requerida de ubicaciones alternativas.

- Evitar la sobrecarga de cualquier elemento del sistema, repartiendo la carga entre todos los nodos.

A continuación se va a explicar el esquema general de esta nueva arquitectura mixta cliente-servidor / P2P, detallando cada uno de los elementos que la integran. Después se va a especificar la metodología que se va a utilizar para evaluar y probar esa arquitectura, y por último, se realizará la prueba de la misma comparándola con la arquitectura clásica cliente-servidor.

5.1 – Esquema General

Atendiendo a estas características se ha diseñado una arquitectura con cuatro tipos de nodos diferentes:

- El nodo servidor principal, que será el que tenga acceso a toda la información de la base de datos del terreno permitiendo tener una disponibilidad continua de toda la información. Además, será el único nodo que tenga acceso directo a la base de datos íntegra, lo que le permite llevar a cabo un control del acceso a la información y facilitar las tareas de comprobación de la integridad de los datos y de la actualización de los mismos.

- El nodo servidor secundario, que será el que realice de intermediario entre los clientes y el servidor principal. Este nodo permitirá facilitar el mantenimiento del sistema y proporcionar robustez ante fallos de algún servidor, debido a que si se emplean varios de estos nodos, el trabajo del servidor que haya que reparar o que haya fallado se puede repartir entre los otros servidores.

- El cliente, que será el nodo encargado de mostrar la visualización de la superficie del terreno en la pantalla del usuario. Este nodo actuará como un nodo típico de una arquitectura P2P, proporcionando al sistema una escalabilidad en función del número de usuarios que tendrá un coste bajo, ya que serán los propios clientes los que, actuando a la vez como servidores, harán frente al aumento de la carga producida por el incremento del número de usuarios conectados al sistema.

- El servidor de comunicaciones, que será el nodo encargado de gestionar toda la topología del sistema. Permitirá evitar la sobrecarga de cualquier elemento del mismo, repartiendo la carga entre todos los nodos del sistema y obteniendo, además, una organización que proporcione la transmisión más rápida y eficiente posible de la información del terreno.

El esquema básico de conexión de esta arquitectura es el siguiente:

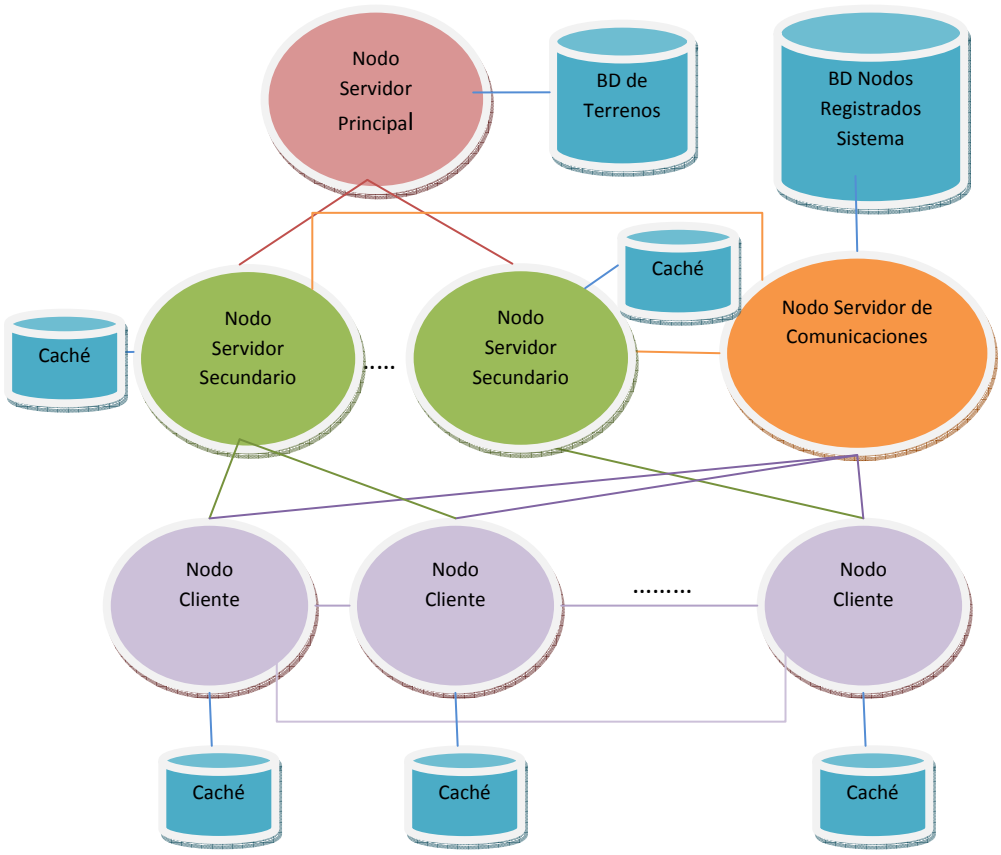


Figura 5.1 – Esquema de conexiones de la arquitectura

En la Figura 5.1 se pueden observar las conexiones que existen entre los diferentes nodos del sistema, así como las unidades de almacenamiento principales de cada uno de ellos. Para llevar a cabo la comunicación entre los diferentes elementos de la arquitectura se diseñará un protocolo de comunicaciones basado en mensajes.

El nodo servidor principal tendrá acceso a la base de datos donde se almacena toda la información del terreno. Este nodo sólo tendrá conexiones con los servidores secundarios, de forma que éstos puedan asegurar que las peticiones que se le realizan sean correctas.

Uno o varios servidores secundarios estarán conectados con el servidor principal. Estos servidores secundarios dispondrán de una caché con un tamaño preestablecido, donde almacenarán una parte de la base de datos que corresponderá a la información de las últimas peticiones realizadas al servidor principal.

Los clientes conectados al sistema dispondrán de una caché con un tamaño limitado donde almacenarán la última información del terreno que se han descargado. Esta información almacenada en su caché será la que se pueda intercambiar con el conjunto de clientes vecinos con los que se encuentran conectados. Además, cada cliente estará conectado con un servidor secundario de forma que, en el caso de que ningún cliente vecino disponga de la información que necesita, o si la tiene pero no puede enviarla dentro de un tiempo determinado, pueda acudir a ese servidor para obtenerla.

El servidor de comunicaciones planificará el conjunto de clientes vecinos y el servidor secundario con el que se conectará cada cliente. Para realizar dicha planificación, los servidores secundarios y los clientes establecen conexiones periódicas con el servidor de comunicaciones transfiriéndole toda la información que necesite para realizar esta tarea.

Entrando un poco más en detalle sobre cada uno de estos nodos:

5.1.1 – Nodo Servidor Principal

Este tipo de nodo es el único que puede acceder directamente a la base de datos que almacena toda la información del terreno, por lo que podría llevar a cabo, si se considera necesario, un control del acceso a esa información, el registro de su uso y la comprobación de la integridad de la misma.

La información almacenada en esta base de datos se ha comprimido empleando el nuevo esquema de compresión que se ha desarrollado en el Capítulo 4, que está basado en el esquema de compresión estándar Jpeg2000.

Este nodo puede estar conectado con uno o varios nodos servidores secundarios empleando el protocolo de transmisión Jpip incluido dentro del estándar Jpeg2000. El uso de este protocolo estándar podría permitir además, que este servidor principal pueda ser utilizado aparte de por los servidores secundarios definidos dentro de nuestro sistema, por otros sistemas o aplicaciones elaboradas de forma externa.

La conexión entre el servidor principal y el servidor secundario se realiza a través de conexiones TCP, lo que permite mantener una comunicación continua entre ellos. El

número de servidores secundarios conectados a un servidor principal suele ser reducido, en comparación con el número de servidores necesarios en una arquitectura cliente-servidor pura, donde los clientes siempre se conectan con el servidor para obtener la información. En el nuevo sistema mixto cliente-servidor / P2P, la mayor parte de las peticiones de información de los clientes se espera que se resuelvan entre ellos mismos, reduciéndose el número de accesos necesarios a los servidores secundarios. Por todo ello, el número de estas conexiones permanentes es bajo.

La tarea principal de este nodo consiste en recoger las peticiones de información llevadas a cabo por los servidores secundarios y devolver la información requerida en forma de un conjunto de precintos. El precinto será la unidad mínima de información con la que trabajará el sistema, y hace referencia a los elementos de información generados por el nuevo algoritmo de compresión desarrollado en el Capítulo 4 al comprimir una región del terreno.

En las peticiones que realiza el servidor secundario al servidor principal, se especifican regiones del terreno con un nivel de resolución adecuado en vez de un conjunto concreto de precintos, siguiendo el formato especificado por el protocolo Jpip. Esto le permite a los servidores secundarios anticiparse a las futuras necesidades de los clientes y recibir, por parte del servidor principal, junto con los precintos requeridos por los clientes, aquellos que probablemente se vayan a necesitar en breve. Además, el servidor principal mantiene un control de los datos que ya se han enviado y se encuentran disponibles en cada servidor secundario, de forma que sólo se envían los precintos estrictamente necesarios. El uso de estas regiones del terreno en lugar de precintos individuales es necesario para poder utilizar el protocolo estándar Jpip, de esta forma, este servidor principal podrá ser utilizado aparte de por los servidores secundarios definidos dentro de nuestro sistema, por otros sistemas o aplicaciones elaboradas de forma externa.

5.1.2 – Nodo Servidor Secundario

Es el único tipo de nodo que tiene acceso al servidor principal, actuando como intermediario entre el servidor principal y los clientes del sistema en el intercambio de la información del terreno. Recibe las peticiones del cliente comprobando que sean correctas y bloqueando cualquier intento de acceso malintencionado al servidor. Además, evita la sobrecarga del servidor principal, controlando su disponibilidad y enviando las peticiones de forma progresiva.

El esquema básico de funcionamiento es el siguiente:

- Recibe peticiones de precintos por parte de los clientes.

- Comprueba si esos precintos se encuentran almacenados ya en su caché, en cuyo caso los devolverá de manera inmediata.

- En caso contrario, pide al servidor principal la región del terreno que contiene estos precintos, retrasando su envío al cliente hasta que se reciban.

Estos servidores van a actuar como servidores de respaldo a los que pueden acudir los clientes cuando sea necesario. El acceso a estos servidores secundarios por parte de los clientes se realizará sólo en el caso de que los clientes no puedan obtener la información que necesitan de otros clientes, bien porque los clientes no dispongan de esa información, o bien porque se haya producido un fallo en la conexión, o porque el tiempo esperado para obtener la información de cualquier cliente sea demasiado elevado. Por ello, generalmente sólo será necesario disponer de un número reducido de estos servidores.

Este nodo se ha diseñado bajo el supuesto de que va a haber más de uno de ellos conectado al sistema. En caso contrario, el hecho de mantener un servidor principal y un único servidor secundario no parece adecuado, debido a que se introduce un retraso motivado por la comunicación entre esos dos servidores. Por lo tanto, en ese caso, puede que sea conveniente integrar las funcionalidades del servidor secundario dentro del servidor principal, y dirigir las peticiones de los clientes directamente a este servidor.

Además, si se emplean varios servidores secundarios en vez de uno sólo, se puede realizar una distribución de las peticiones entre ellos, atendiendo tanto a criterios de carga computacional (balanceando la carga entre los diferentes servidores), como a criterios espaciales (distribuyendo las peticiones entre los servidores según la posición dentro de la escena de los datos requeridos). De esta forma, tenemos unos servidores secundarios especializados en determinadas regiones del terreno.

Para ello, cada servidor secundario monitoriza su carga computacional y calcula la posición media de los datos que tiene almacenados en su caché en cada momento. Esta información se envía al servidor de comunicaciones de forma periódica, que será el encargado de asignar a cada cliente, el servidor secundario más adecuado en función de estos parámetros.

Como se ha comentado en el apartado 5.1.1, la conversión de los precintos requeridos por los clientes en regiones del terreno que contengan esos precintos, permite al servidor secundario anticiparse a las necesidades futuras de los clientes, de forma que obtenga del servidor principal los precintos que probablemente se vayan a necesitar en breve. Además, de esa forma se puede utilizar el protocolo estándar Jpip para la transmisión de la

información, de forma que el servidor secundario podría conectarse a otros servidores principales elaborados de forma externa que soportasen ese protocolo.

Cuando un servidor secundario se desconecta del sistema, envía un mensaje al servidor de comunicaciones para que éste pueda llevar a cabo la reasignación de los clientes que tenía asignado ese servidor secundario. Si la desconexión del servidor secundario se produce por un fallo del mismo o de la línea de conexión, y por lo tanto, no puede llevar a cabo el envío del mensaje de desconexión al servidor de comunicaciones, como el servidor secundario dejará de enviar información de forma periódica al servidor de comunicaciones, éste podrá detectar la desconexión del servidor secundario, y podrá llevar a cabo la reasignación de los clientes que lo tenían asignado.

5.1.3 – Nodo Servidor de Comunicaciones

Este nodo es el encargado de gestionar la topología de la red P2P. Inicialmente cada cliente debe conectarse obligatoriamente con este servidor, de forma que quede registrado en el sistema. Este hecho permite establecer un control de acceso de los usuarios al sistema.

En esta conexión, el servidor de comunicaciones asigna al cliente una lista de los clientes vecinos con más posibilidades de disponer de la información que va a necesitar, y además, le proporciona un servidor secundario que podrá emplear cuando ninguno de los clientes de la lista pueda suministrarle la información que necesita.

La asignación a cada cliente de su lista de clientes vecinos a los que podrán solicitar datos se realiza siguiendo un criterio de proximidad dentro de la escena. Se puede suponer que si dos clientes se encuentran visualizando la misma región del terreno, a priori, van a necesitar descargar la misma información, la cual quedará almacenada en su caché. Por lo tanto, parece adecuado que estos clientes se encuentren conectados entre sí de forma que puedan intercambiarse esa información que tienen almacenada en su caché, porque probablemente sea la que van a necesitar descargar en un intervalo corto de tiempo.

La asignación del servidor secundario sigue un criterio parecido. Al cliente se le debe asignar como servidor secundario, aquel que disponga de más información en su caché de la región del mundo virtual que el cliente se encuentra visualizando actualmente. Siguiendo esta estrategia, se disminuye la probabilidad de que el servidor secundario reciba peticiones de precintos que no se encuentren en su caché y de que deba, por tanto, acceder al servidor principal para descargárselos, evitando así el retraso en el tiempo de respuesta que ello conlleva.

Además de la situación de los datos de la caché del servidor secundario dentro de la escena, la asignación del servidor secundario se realiza también atendiendo a criterios de calidad de servicio, donde se estima el tiempo que tiene que esperar un cliente para obtener la información requerida, no debiendo ese tiempo superar un cierto valor umbral. Este tiempo se calcula en función de la carga del servidor secundario y de la probabilidad de que el servidor tenga almacenada en su caché esa información. Por lo tanto, a cada cliente se le asigna el servidor secundario que cumpla con estos criterios de calidad de servicio, y que a su vez, almacene en su caché una mayor cantidad de información de la región del terreno que el cliente está visualizando.

5.1.4 – Nodo Cliente (CL)

Este nodo realiza todos los procesos necesarios para permitir al usuario llevar a cabo la visualización en tiempo real de los datos del terreno.

Como se ha comentado, inicialmente este nodo se conecta con el servidor de comunicaciones para registrarse en el sistema y obtener la lista de clientes de los que puede descargar información, además del servidor secundario al que puede acudir en caso necesario. La comunicación se repite periódicamente para mantener actualizada esta información.

Cuando este nodo necesita descargar nueva información del terreno, primeramente envía a cada cliente de su lista un mensaje preguntando qué información de la que necesita tiene disponible cada uno. Además, cada cliente envía un conjunto de parámetros, como por ejemplo, su carga de trabajo, el tiempo de transmisión esperado, etc. que le sirven para estimar la calidad de servicio que cada uno de esos clientes puede ofrecer en el momento actual.

Atendiendo a esta información, se selecciona uno o varios clientes de la lista para realizar la descarga de la información de forma simultánea. Si ninguno de los clientes de la lista dispusiera de la información requerida, o la calidad de servicio que pueden proporcionar fuera inferior a la aceptable, el cliente haría uso del servidor secundario para obtener esa información.

Una vez obtenida la información, ésta se almacena en la caché del cliente, de forma que otros clientes puedan acceder a ella en caso necesario. Esta caché puede tener establecida una capacidad máxima, de forma que cuando se alcance esa capacidad, se reemplazará la información más antigua almacenada por la nueva.

Cuando un cliente se desconecta del sistema, envía un mensaje al servidor de comunicaciones para que éste deje de asignar ese cliente a otros clientes vecinos que se encuentren conectados. Si la desconexión del cliente se produce por un fallo del mismo o de la línea de conexión, y por lo tanto, no puede llevar a cabo el envío del mensaje de desconexión al servidor de comunicaciones, como el cliente dejará de enviar peticiones de forma periódica al servidor de comunicaciones para actualizar su lista de clientes vecinos, éste podrá detectar la desconexión del cliente. Además, los clientes también detectarán la desconexión de un cliente vecino debido a que éste dejará de responder a los mensajes de petición de información que realizan, de forma que se eliminará de su lista de clientes vecinos.

En el caso de que un cliente no disponga de un número mínimo de clientes vecinos dentro de su lista de clientes vecinos, o no contenga un número mínimo de ellos que le puedan ofrecer una calidad de servicio adecuada (situación que ocurrirá cuando el cliente cambie de región que está visualizando súbitamente y de forma drástica), se adelantará la petición al servidor de comunicaciones de una nueva lista de clientes vecinos, sin esperar a que se cumpla el tiempo necesario para llevar a cabo la petición de forma periódica. También se adelantará esta petición cuando el cliente detecte que el servidor secundario que tiene asignado se ha desconectado o no está disponible.

5.2 – Elementos Software del Sistema

A continuación se va a definir más detalladamente el conjunto de los módulos software en los que se descompone cada uno de los cuatro tipos de nodos.

5.2.1 – Servidor Principal

Como hemos comentado, este nodo tendrá acceso a toda la información de la base de datos del terreno, la cual ha sido comprimida empleando nuestro esquema de compresión, y la mantendrá disponible en todo momento. El acceso a esta información se realiza a través de los servidores secundarios, que establecerán la comunicación con este nodo utilizando el protocolo de transmisión Jpip definido en el estándar Jpeg2000.

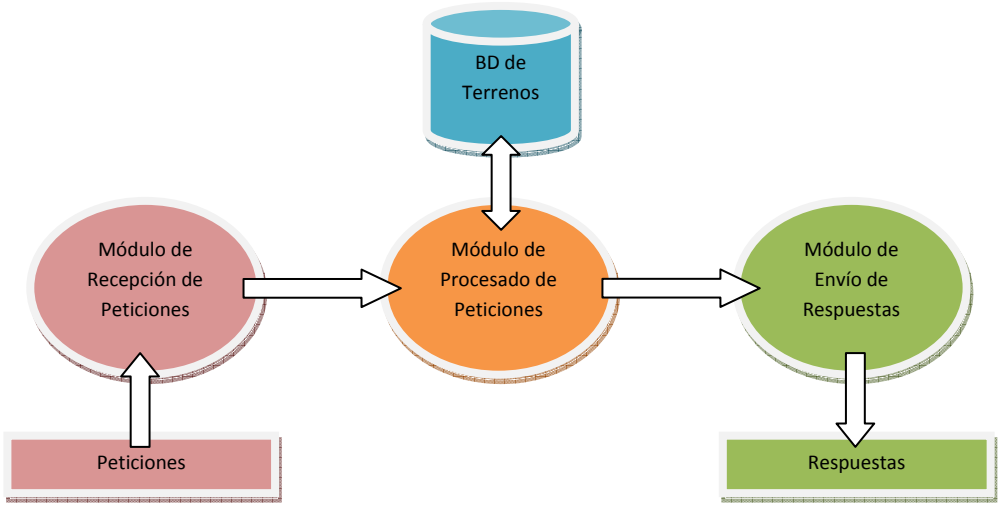


Figura 5.2 – Módulos y esquema de funcionamiento del Servidor Principal

En la Figura 5.2 se observa el funcionamiento del servidor principal. Existe un módulo que se encarga de recibir las peticiones y enviarlas al módulo de procesado de peticiones, que será el encargado de procesarlas, extrayendo la información requerida de la base de datos. Esta información se envía al módulo de envío de respuestas que se encarga de elaborar los mensajes en los que se incluirá esa información y de enviarlos a su destinatario.

Una explicación más detallada de estos módulos es la siguiente:

5.2.1.1 – Módulo de Recepción de Peticiones

Este módulo se encarga de recibir las peticiones de datos del terreno efectuadas por los servidores secundarios, de comprobar que estén correctamente construidas y de colocarlas en una cola para que sean procesadas por el módulo de procesado de peticiones. Si se considera oportuno, este módulo puede implementar algún mecanismo de control de acceso a los datos.

5.2.1.2 – Módulo de Procesado de Peticiones

Este módulo se encarga de procesar las peticiones de datos del terreno. Las peticiones recibidas contienen una región del terreno con un nivel de resolución determinado, a partir de los cuales se obtiene el conjunto de precintos que abarca esa región, para ese nivel de detalle.

El hecho de recibir peticiones de regiones del terreno en lugar de recibir directamente peticiones de precintos, permite que el servidor principal pueda predecir los precintos que el servidor secundario va a necesitar en un futuro, y enviárselos junto con los estrictamente requeridos. Esta predicción es adecuada debido a que si el servidor secundario necesita de un conjunto de precintos concreto es porque se los ha pedido algún cliente, el cual se encontrará visualizando la región a la que pertenecen esos precintos, de forma que la probabilidad de que el cliente necesite otros precintos de esa misma región es alta. Al realizar peticiones de regiones, el servidor secundario recibe anticipadamente esos precintos, evitando tener que acceder al servidor principal cuando el cliente los solicite, y por lo tanto, enviándoselos en un tiempo menor que si tuviera que acceder al servidor principal.

De este conjunto de precintos requeridos sólo se envían como respuesta al servidor secundario aquellos que no se le hayan enviado previamente. Para ello, el servidor principal almacena para cada servidor secundario una lista con estos precintos enviados.

5.2.1.3 – Módulo de Envío de Respuestas:

Este módulo se encarga de generar y enviar al servidor secundario los mensajes de respuesta que contienen el conjunto de precintos requeridos por la petición, empleando para ello el protocolo de transmisión Jpip.

5.2.2 – Servidor Secundario (SS)

Este nodo es el encargado de realizar las peticiones de datos del terreno al servidor principal actuando de intermediario entre los clientes y el servidor principal.

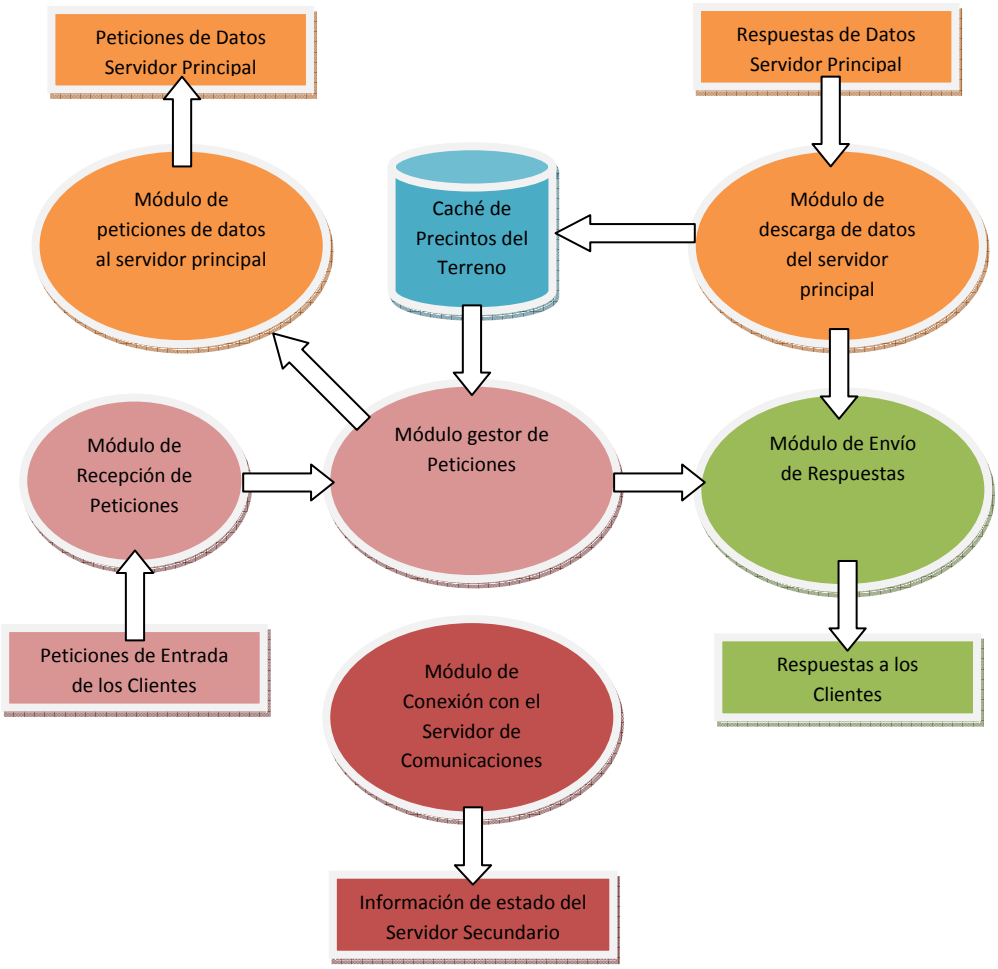


Figura 5.3 – Módulo y esquema de funcionamiento del Servidor Secundario

En la Figura 5.3 se puede observar el funcionamiento de este nodo. En dicha figura se pueden diferenciar dos partes. La primera se corresponde con la gestión de las peticiones de información del terreno y la segunda con el envío de la información del estado del servidor secundario al servidor de comunicaciones.

En la parte que se encarga de la gestión de las peticiones de información del terreno, primeramente tenemos un módulo que se encarga de recibir las peticiones del cliente y enviárselas al módulo gestor de peticiones, que comprobará si los precintos que se requieren en la petición se encuentran almacenados en su caché, en cuyo caso los enviará de forma inmediata al módulo encargado de enviar los mensajes de respuesta a los clientes, o por el contrario, es necesario pedírselos al servidor principal, de forma que la petición se le envía al módulo encargado de realizar las peticiones al servidor principal.

Las respuestas a las peticiones realizadas al servidor principal se encarga de recibirlas el módulo de descarga de datos del servidor principal, que las almacenará en la caché y se las enviará al módulo que se encarga del envío de la respuestas. El módulo de envío de respuestas construirá los mensajes que contendrán los precintos pedidos por los clientes y los enviará a su destinatario.

En la segunda parte, la que se está relacionada con el envío de la información del estado del servidor secundario al servidor de comunicaciones, existe un módulo que se encarga de monitorizar la carga actual del servidor secundario y de enviar periódicamente esa información al servidor de comunicaciones, de forma que éste pueda asignar a cada cliente el servidor secundario más adecuado en cada momento.

Una explicación más detallada de estos módulos es la siguiente:

5.2.2.1– Módulo de recepción de peticiones

Este módulo se encarga de recibir las peticiones de datos del terreno de los clientes, de comprobar que estén correctamente construidas y de introducirlas en una cola de peticiones pendientes a la espera de que el módulo gestor de peticiones de los clientes pueda procesarlas.

5.2.2.2 – Módulo gestor de peticiones

Este módulo es el encargado de procesar las peticiones de datos del terreno realizadas por los clientes, las cuales contienen una lista de precintos requeridos.

El servidor secundario comprueba cuáles de estos precintos se encuentran almacenados en su caché y por lo tanto pueden ser enviados inmediatamente al cliente, y cuáles de ellos no están disponibles y por lo tanto tendrán que ser pedidos al servidor principal.

Las peticiones de datos del terreno recibidas se separan en dos listas de peticiones, las peticiones que contienen precintos que se encuentran disponibles en la caché del servidor secundario y por lo tanto pueden enviarse inmediatamente al cliente, y las peticiones que contienen precintos que requieren hacer uso del servidor principal. Estas dos listas de peticiones se procesan de forma simultánea. De esta forma se evita que las peticiones que pueden enviarse inmediatamente queden bloqueadas por otras peticiones que esperan la respuesta del servidor principal.

Los precintos que se encuentran en la caché se devuelven inmediatamente al cliente que los ha requerido, para lo cual este conjunto de precintos se remiten al módulo de envío de respuestas. Los precintos que no se encuentran en la caché se introducen en una cola de peticiones pendientes, a la espera de que el módulo de peticiones de datos al servidor principal pueda procesarlas.

5.2.2.3 – Módulo de peticiones de datos al servidor principal

Este módulo es el encargado de enviar al servidor principal las peticiones de los precintos que no se encuentran almacenados en la caché del servidor secundario.

Las peticiones que realizan los clientes al servidor secundario contienen un conjunto de precintos, sin embargo, el protocolo de comunicación con el servidor principal, Jpip, establece que en las peticiones se ha de especificar una región del terreno con un nivel de resolución. Por lo tanto, este módulo se encarga de traducir las peticiones de los clientes al formato adecuado.

Se establecen dos conexiones independientes para el envío de peticiones al servidor principal, una para la información de alturas del terreno y otra para la información de texturas. Estas conexiones son de tipo TCP, y permiten mantener conectados en todo momento al servidor secundario con el servidor principal.

5.2.2.4 – Módulo de descarga de datos del servidor principal

Este módulo se encarga de recibir los precintos enviados por el servidor principal y de almacenarlos en la caché del servidor secundario, de forma que, además de poder responder al cliente que ha realizado la petición, pueden también servir para responder las futuras peticiones de los clientes que requieran de alguno de esos precintos sin tener que volver a acceder al servidor principal para obtenerlos.

La caché puede tener un tamaño máximo predefinido, de forma que cuando se llene, se eliminen de ella los precintos que lleven más tiempo sin ser requeridos por los clientes, para no sobrepasar este tamaño máximo.

5.2.2.5 – Módulo de envío de respuestas

Este módulo se encarga de elaborar y enviar los mensajes de respuesta que contienen los precintos requeridos por los clientes. Los precintos que se incluirán dentro de estos mensajes pueden llegar a este módulo de dos formas. A través del módulo gestor de peticiones, en el caso de que los precintos se encuentren en la caché del servidor secundario, o a través del módulo de descarga de datos del servidor principal, si ha sido necesario pedírselos al servidor principal.

5.2.2.6 – Módulo de conexión con el servidor de comunicaciones

Periódicamente, cada servidor secundario se conecta con el servidor de comunicaciones para transmitirle sus parámetros actuales de funcionamiento con el objetivo de que el servidor de comunicaciones pueda decidir adecuadamente que servidor secundario asignar a cada cliente.

Entre la información que este módulo envía al servidor de comunicaciones se encuentra: la posición media de los datos almacenados en su caché, los parámetros que definen la función cuadrática que determina la probabilidad de que un precinto se encuentre en su caché, el tiempo de procesado medio de los precintos cuando éstos se encuentran en la caché y cuando no se encuentran en ella, el número de precintos pendientes de procesar y el tiempo medio que tienen que esperar en la cola para ser procesados.

5.2.2.6.1 – Posición media de los datos

Indica la posición media ponderada de los datos almacenados en el instante actual en la caché del servidor secundario.

Cuando se emplean varios servidores secundarios, es interesante dividir toda la información del terreno entre esos servidores para obtener servidores especializados en regiones del

terreno. Los clientes que se encuentren visualizando una misma región del terreno deberían realizar las peticiones de información al servidor secundario especializado en esa región, de forma que exista una mayor probabilidad de encontrar los datos que necesitan en la caché de ese servidor secundario sin tener que acudir al servidor principal, obteniendo con ello una respuesta más rápida. Con el valor de la posición media de los datos, el servidor de comunicaciones puede llevar a cabo la asignación a cada cliente del servidor secundario especializado adecuado.

El cálculo de esta posición media se realiza de forma ponderada dando mayor peso a los precintos más recientemente servidos, ya que como la caché puede tener fijada una capacidad máxima que en caso de alcanzarse supone la eliminación de los elementos almacenados en la caché a los que no se ha accedido durante más tiempo, las precintos más recientemente pedidos persistirán por más tiempo en la caché, y por lo tanto, el valor de la posición media de los datos se desplazará más rápidamente hacia la posición de estos últimos datos.

Además, el hecho de que un cliente haya tenido que realizar peticiones al servidor secundario en busca de información, significa a priori, que ningún otro cliente dispone de esa información, por lo tanto, probablemente el cliente tenga que realizar nuevas peticiones de otros precintos situados en esa zona, de forma que empleando esta asignación ponderada se está prediciendo el comportamiento de la información que va a estar almacenada en la caché en un futuro próximo, proporcionando una información más valiosa al servidor de comunicaciones a la hora de decidir qué servidor secundario asignar a cada cliente.

Este parámetro se calcula de la siguiente manera:

- Cuando el número de precintos previos pedidos es menor que el valor NPrecCaché, siendo NPrecCaché el número máximo de precintos que se puede almacenar en la caché:

$$Posición\ media = \frac{\sum posición\ precintos\ previos}{número\ de\ precintos\ previos}$$

Ecuación 5.1 – Cálculo de la Posición media de los datos en la caché del servidor secundario cuando el número de precintos previos es < NPrecCaché

- Cuando el número de precintos previos pedidos es mayor o igual al valor de NPrecCaché:

$$Posición\ media = \frac{(N * \sum posición\ precintos\ previos) + posición\ último\ precinto}{N + 1}$$

Ecuación 5.2 – Cálculo de la Posición media de los datos en la caché del servidor secundario cuando el número de precintos previos es >= NPrecCaché

De esta forma, inicialmente la posición de todos los precintos tendrá la misma importancia, pero a partir del precinto NPrecCaché, la posición de cada precinto servido más recientemente tendrá un mayor peso, debido a que, como se ha comentado previamente, tiene una mayor probabilidad de persistencia en la caché y nos proporciona una predicción de la información que va a estar almacenada en el futuro en la caché.

5.2.2.6.2 – Los parámetros que definen la función cuadrática que determina la probabilidad de que un precinto se encuentre en la caché del servidor secundario (K1, K2 y K3).

Como se ha comentado, es lógico suponer que cuantos más datos de la región que está visualizando el cliente estén almacenados en la caché del servidor secundario, menos veces el servidor secundario tendrá la necesidad de acceder al servidor principal para obtenerlos, siendo así la respuesta de la información más rápida.

Se han realizado diversas pruebas para evaluar la relación entre la probabilidad de que el servidor secundario tenga almacenada cierta información en su caché, y la distancia entre la posición de esa información en la escena y la posición media de los datos almacenados en la caché. Con esta pruebas, se ha observado de forma empírica, que está relación puede ajustarse a la inversa de una función cuadrática. Los resultados de estas pruebas se muestran en el apartado 5.4.9.

$$Probabilidad = \frac{1}{K1 * distancia^2 + K2 * distancia + K3}$$

Ecuación 5.3 – Cálculo de la probabilidad de que un precinto se encuentre en la caché de un servidor secundario.

Asumiendo que esto es así, el servidor secundario ajusta dinámicamente el cálculo de esta función cuadrática de acuerdo a los fallos y aciertos que se producen en las peticiones de precintos de los clientes, y envía los parámetros que definen la curva al servidor de comunicaciones para que éste pueda decidir con un mayor grado de precisión cuál es el servidor secundario más adecuado para cada cliente.

5.2.2.6.3 – Tiempo de procesado medio de los precintos que se encuentran en la caché

Es el tiempo que el servidor secundario tarda en procesar la petición de un precinto que se encuentra almacenado en su caché, contando el tiempo que le cuesta obtener ese precinto de

la caché y elaborar el mensaje donde se incluirán esos precintos para su envío al cliente. Este tiempo se empleará por parte del servidor de comunicaciones como referencia del tiempo que tardaría en responder el servidor secundario un precinto que se encuentre almacenado en su caché, sirviendo como un elemento más a la hora de decidir qué servidor secundario asignar a cada cliente.

5.2.2.6.4 – Tiempo de procesado medio de los precintos que no se encuentran en la caché

Es el tiempo que el servidor secundario tarda en procesar la petición de un precinto que no se encuentra almacenado en su caché, contando el tiempo que le cuesta obtener ese precinto del servidor principal y elaborar el mensaje donde se incluirán estos precintos para su envío al cliente. Este tiempo se empleará por parte del servidor de comunicaciones como referencia del tiempo que tardaría en responder un servidor secundario un precinto que no se encuentre almacenado en su caché, sirviendo como un elemento más a la hora de decidir qué servidor secundario asignar a cada cliente.

5.2.2.6.5 – Número de precintos pendientes

Es el número medio de precintos pendientes de ser procesados por parte del servidor secundario. Cuando una petición de precintos llega al servidor secundario, ésta se introduce en una cola de peticiones de precintos pendientes. El módulo gestor de peticiones se encargará de procesar estas peticiones.

Este parámetro se emplea por parte del servidor de comunicaciones como un indicador de la carga del servidor secundario. Si el valor de este parámetro aumenta continuamente sin estabilizarse significa que el servidor secundario se encuentra sobrecargado, ya que recibe más peticiones de las que es capaz de responder, lo que indicará al servidor de comunicaciones que no debe asignar este servidor a más clientes, y que en aquellos clientes que ya lo tengan asignado debe reemplazarlo por otro con una carga menor.

El servidor de comunicaciones utiliza este parámetro para impedir la asignación del servidor secundario a los clientes cuando este parámetro supera un cierto umbral, intentando evitar que se llegue a la situación de que el servidor secundario se sobrecargue.

5.2.2.6.6 – Tiempo Medio de Espera en la Cola

Es el tiempo medio que permanece un precinto en la cola de peticiones pendientes de ser procesadas. Al igual que el parámetro anterior, éste es un indicador de lo cargado que se encuentra el servidor secundario. Conforme el número de peticiones aumente, el tiempo que tiene que esperar un precinto para ser procesado será mayor, al tener que esperar a que se procesen un mayor número de precintos previos. El servidor secundario emplea este tiempo, junto con el número de precintos pendientes de ser procesados, para decidir si un servidor secundario ya está demasiado cargado.

5.2.3 – Cliente (CL)

Este nodo se encarga de la visualización de los terrenos en pantalla, y de realizar las peticiones de información del terreno que sean necesarias a los clientes vecinos y al servidor secundario en función de los movimientos del usuario por la escena.

Los clientes se conectarán inicialmente con el servidor de comunicaciones para registrarse dentro en el sistema y obtener la lista de clientes vecinos y el servidor secundario de los que va a poder descargar la información del terreno. Las peticiones de información siempre irán dirigidas hacia los clientes vecinos, y sólo en el caso de que éstos no dispongan de esa información o no puedan responder dentro de un tiempo aceptable, las peticiones se dirigirán hacia el servidor secundario.

El intercambio de información entre los clientes se realizará usando el protocolo udp. Se ha elegido este protocolo por su sencillez, el pequeño tamaño de sus mensajes y por su velocidad de transmisión, que permite obtener la información de los precintos en un tiempo adecuado para realizar una visualización en tiempo real.

El principal problema que presenta este protocolo es la posible pérdida de mensajes. Una aplicación en tiempo real necesita obtener la información dentro de un determinado intervalo de tiempo, ya que fuera de ese intervalo la información puede ser que ya no sea necesario visualizarla. En nuestro sistema, el efecto ocasionado por la pérdida de información será una visualización de la escena con una calidad visual algo inferior a la deseada, pero manteniendo la velocidad de refresco mientras el usuario se mueve por la escena, proporcionando de esa forma una sensación de inmersión y una calidad de servicio aceptable.

Si se requiriera asegurar la descarga de toda la información, generalmente el tiempo para obtenerla sería mayor y la velocidad de refresco sería inferior, proporcionando una menor sensación de inmersión en la escena, y por lo tanto, una calidad de servicio peor, a pesar de que la calidad visual de la escena fuera la adecuada.

Por ello, se ha decidido que es preferible obtener la información lo más rápidamente posible, aún a riesgo de perder algo de calidad visual en la representación, asegurando de esta forma una velocidad de refresco adecuada para la visualización en tiempo real de los terrenos.

De todas formas, los clientes van a gestionar las posibles pérdidas de los mensajes de información, evaluando si todavía es necesaria la información que debían obtener, en cuyo caso se repetirían las peticiones pero dirigidas esta vez a otros clientes vecinos o al servidor secundario, evitando el uso del cliente vecino con el que se ha tenido el problema de transmisión.

Si se prevé que el sistema va a funcionar en un entorno en el que las líneas de conexión tienen un alto porcentaje de fallos de transmisión que pudieran provocar demasiadas pérdidas de mensajes, se podría enviar la misma petición de información a un conjunto de clientes de forma simultánea, aumentando la probabilidad de obtener esa información a pesar de los errores de transmisión.

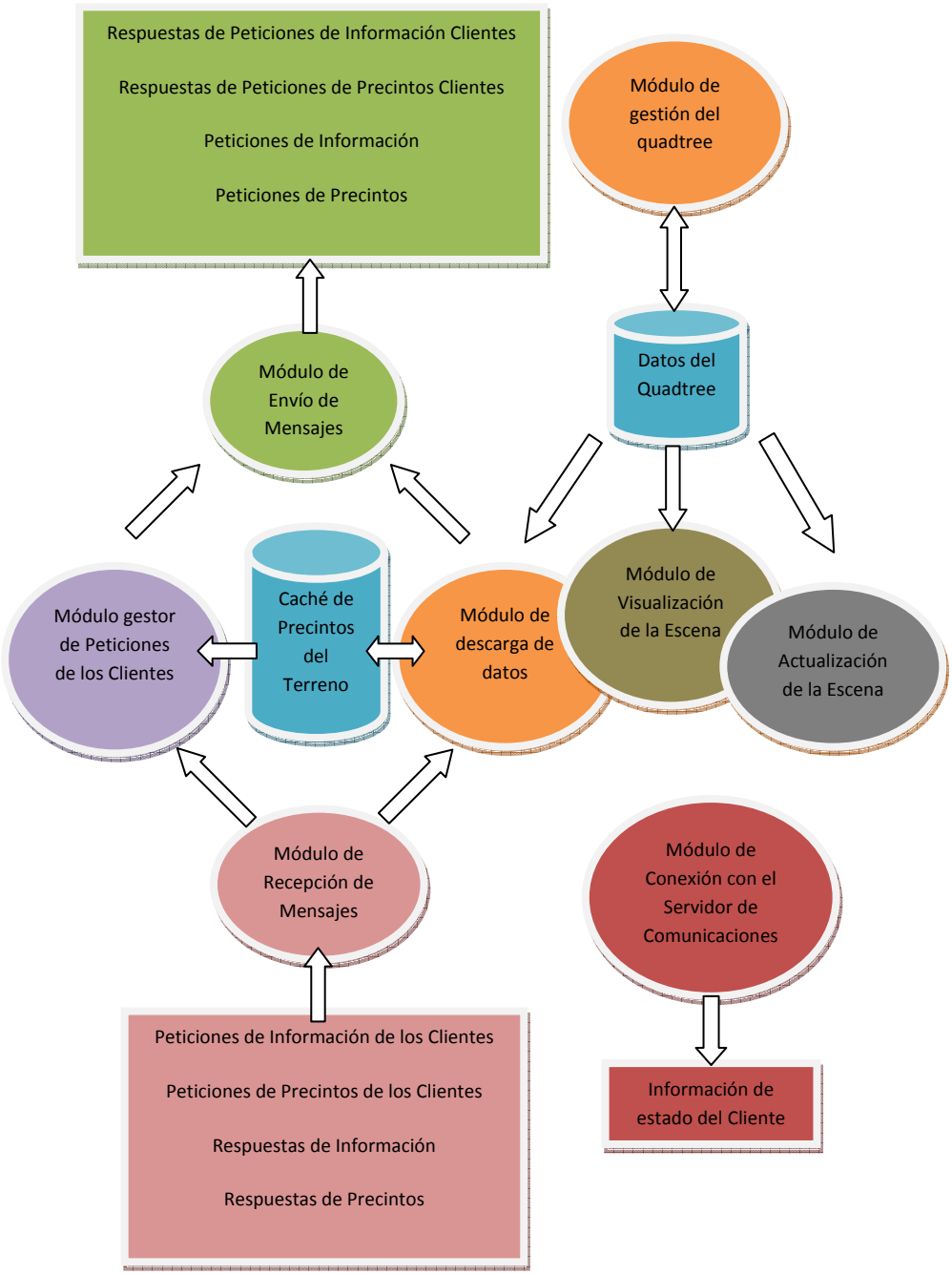


Figura 5.4 – Módulos y esquema de funcionamiento del Cliente

En la Figura 5.4 se muestra el funcionamiento de los clientes. Se pueden diferenciar dos partes. La primera se corresponde con la gestión de la información del terreno y la segunda con el envío de la información del estado del cliente al servidor de comunicaciones.

En la parte que se corresponde con la información del terreno, el cliente puede recibir diferentes mensajes que el módulo de recepción de mensajes se encargará de procesar. Los mensajes que hacen referencia a peticiones de precintos por parte de otros clientes vecinos se envían al módulo gestor de peticiones de los clientes, quien obtendrá los precintos requeridos en las peticiones, de los precintos que tenga almacenados en su caché y se los enviará al cliente. En el caso de que se trate de mensajes que contienen los precintos de información del terreno requerida por el propio cliente, será el módulo de descarga de datos el que se encargue de procesarlos y de almacenarlos en su caché.

La decisión de qué precintos es necesario descargar la lleva a cabo el módulo de actualización, que implementa la parte del algoritmo de triangulación encargado de esa tarea. El proceso de visualización de la escena implementa la parte del algoritmo de triangulación encargado de visualizar la superficie del terreno. Como ambos procesos pueden acceder al mismo tiempo a la misma estructura de quadtree que organiza la información de la escena, se emplea el módulo de gestión del quadtree para realizar de forma segura las operaciones de añadir y eliminar elementos del quadtree, evitando la modificación del quadtree cuando alguno de los módulos anteriores lo esté utilizando, lo que podría dar lugar a errores en la ejecución o en la visualización de la escena.

En la segunda parte, la que se está relacionada con el envío de la información del estado del cliente al servidor de comunicaciones, existe un módulo que se encarga de monitorizar la posición actual y un conjunto de parámetros sobre la carga actual del cliente, para enviarlos periódicamente al servidor de comunicaciones, de forma que éste pueda asignar a cada cliente el conjunto de clientes vecinos más adecuados.

Una explicación más detallada de estos módulos es la siguiente:

5.2.3.1 – Módulo de Recepción de Mensajes

Este módulo se encarga de recibir todos los mensajes del protocolo de comunicaciones que le llegan al cliente, de comprobar que estén correctamente construidos y de introducirlos en la cola de mensajes adecuada según el tipo de mensaje de que se trate.

Cada cliente almacena la información del terreno en una caché de un tamaño limitado, por lo tanto, cuando esta caché se llene, será necesario eliminar algunos elementos de la caché

para incorporar otros nuevos. Los datos que se van a eliminar son los precintos que hace más tiempo que han sido descargados, porque son los que tienen una menor probabilidad de volver a ser necesitados por parte del cliente.

Cuando un cliente necesita nueva información del terreno, elabora una lista de precintos que debe descargar de sus clientes vecinos, sin embargo, desconoce qué clientes vecinos disponen de esos precintos almacenados en su caché. Si se sigue la estrategia de pedir los precintos a cada cliente vecino de manera consecutiva hasta encontrar a uno que los tenga almacenados en su caché y los devuelva, no se puede asegurar cuándo se obtendrá esa información, e incluso puede darse el caso de que ningún cliente vecino dispusiera de esos precintos, y entonces habría que hacer la petición al servidor secundario asignado, pudiéndose superar el tiempo considerado como aceptable para la recepción de esos precintos.

Si se sigue la estrategia de pedir a todos los clientes vecinos los precintos, puede que éstos se encuentren en varios de ellos y se reciban los mismos precintos varias veces, desperdiciando ancho de banda de la conexión y tiempo de proceso tanto de los clientes vecinos como del propio cliente.

Por ello, se ha decidido realizar, previamente a la petición de los precintos, el envío de una petición de información acerca de los precintos requeridos que se encuentran en la caché de cada cliente vecino. Cada cliente vecino responderá esa información, y por lo tanto, ahora el cliente podrá realizar de forma simultánea un conjunto de peticiones de precintos a diferentes clientes vecinos, teniendo la certeza de que esos clientes disponen de esos precintos en su caché y van a poder responder dentro de un tiempo aceptable.

Además, este conjunto de mensajes permite conocer de manera inmediata cuáles de estos precintos no se encuentran en ninguno de los clientes, pudiendo realizar las peticiones directamente al servidor secundario que tiene asignado, evitando un tiempo de espera que podría ser elevado si se tienen que ir pidiendo los precintos a cada cliente de manera consecutiva.

Por eso, se van a emplear dos tipos de mensajes:

- **Mensajes de petición de información**, en los que un cliente pedirá información a los clientes vecinos acerca de la disponibilidad de un conjunto de precintos almacenados en su caché.

- **Mensajes de petición de precintos**, en los que una vez que se conoce que un cliente dispone de esos precintos almacenados en su caché, se le piden los precintos en sí.

Por lo tanto, como el intercambio de información entre los clientes es bidireccional, los tipos de mensajes que pueden llegar al cliente son cuatro:

a) Peticiones de Información de los Clientes Vecinos: son mensajes enviados por los clientes vecinos para averiguar si un conjunto de precintos se encuentran almacenados en la caché del cliente que recibe el mensaje.

b) Peticiones de Precintos de los Clientes Vecinos: son mensajes enviados por los clientes vecinos para obtener un conjunto de precintos que, a priori, se encuentran almacenados en la caché del cliente que recibe el mensaje.

c) Respuestas de Información: son mensajes de respuesta enviados por los clientes vecinos con información de la disponibilidad en su caché de un conjunto de precintos requeridos por el cliente que recibe el mensaje.

d) Respuestas de Precintos: son mensajes de respuesta que contienen un conjunto de precintos requeridos por el cliente que recibe el mensaje. Estos mensajes de respuesta pueden haber sido enviados tanto por los clientes vecinos como por los servidores secundarios.

Los tipos de mensaje **a)** y **b)** se colocan en unas colas de mensajes que se encarga de gestionar el módulo gestor de peticiones de los clientes.

Los tipos de mensaje **c)** y **d)** se colocan en unas colas de mensajes que se encarga de gestionar el módulo de descarga de datos.

5.2.3.2 – Módulo gestor de peticiones de los clientes

Este módulo se encarga de procesar los dos tipos de mensajes relacionados con las peticiones de información del terreno realizadas por los clientes vecinos.

- **Peticiones de información de precintos**: para este tipo de mensajes, se comprueba si el conjunto de precintos especificado por el mensaje se encuentra o no en la caché del cliente, elaborando un mensaje de respuesta con los precintos que tiene almacenados y con los que no. Este mensaje se coloca en una lista de mensajes pendientes de enviar que será procesada por el módulo de envío de mensajes.

- **Peticiones de precintos**: para este tipo de mensajes, se comprueba si el conjunto de precintos especificado en el mensaje se encuentra disponible o no en la caché del cliente, elaborando un mensaje de respuesta que contiene los precintos que se encuentren

disponibles. Este mensaje se coloca en una lista de mensajes pendientes de enviar que será procesada por el módulo de envío de mensajes.

5.2.3.3 – Módulo de descarga de datos

Este módulo decide qué información del terreno se ha de descargar, ya que es necesaria para la visualización de la escena, y de realizar los pasos necesarios para descargarla.

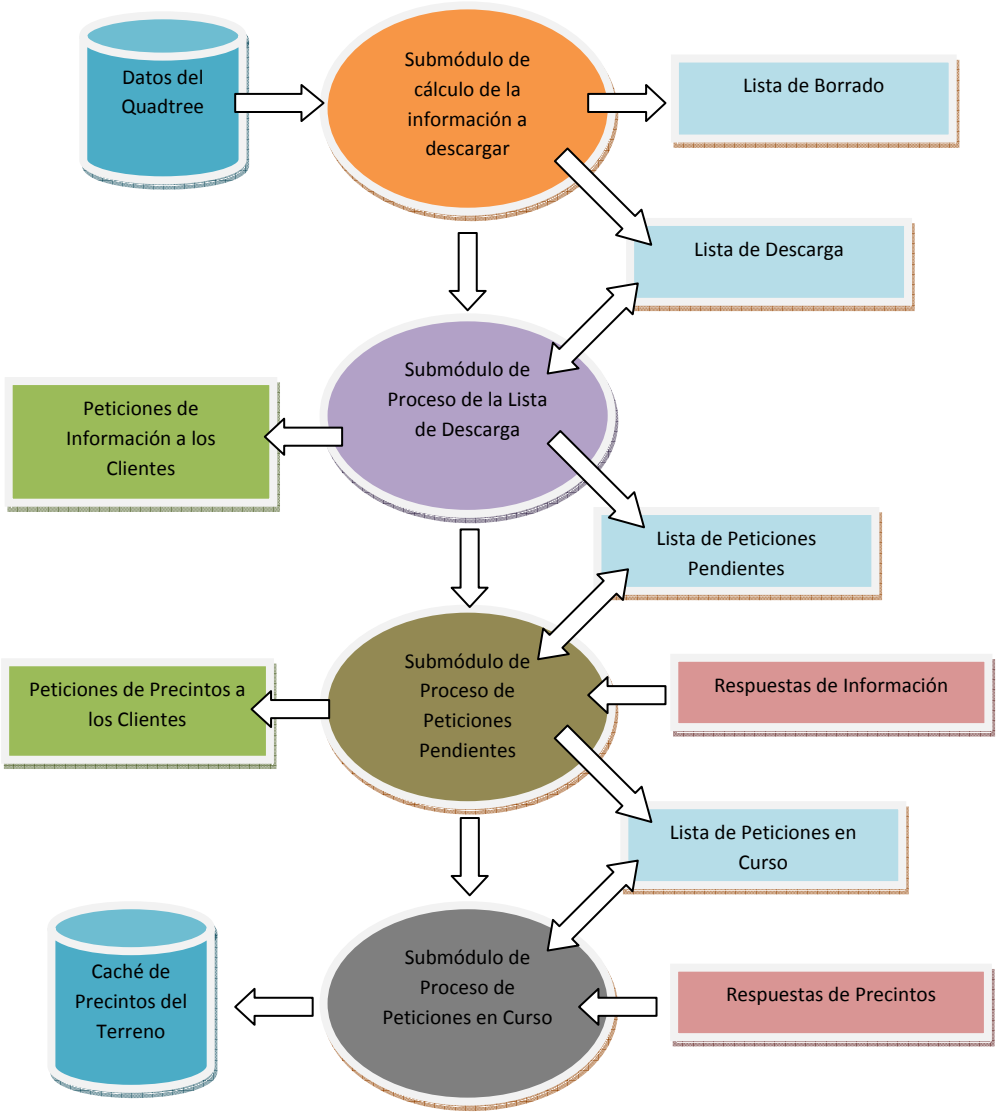


Figura 5.5 – Submódulos y esquema de funcionamiento del módulo de Descarga de Datos

En la Figura 5.5 se muestra el funcionamiento de este módulo, el cual se puede desglosar en un conjunto de submódulos que se ejecutan de forma secuencial.

Primero se calcula la nueva información del terreno que se ha de descargar, para ello se recorre la estructura del quadtree que almacena la información cargada en el sistema.

A continuación se procede a realizar esa descarga de información. Como se ha comentado, primeramente es necesario conocer qué clientes vecinos disponen de esa información almacenada en su caché, por lo que el submódulo de proceso de la lista de descarga envía los mensajes necesarios para obtener esa información.

Una vez que se han recibido las respuestas a esos mensajes de información, el submódulo de proceso de peticiones pendientes realiza un conjunto de peticiones de precintos a los diferentes clientes vecinos de forma simultánea. Si alguno de los precintos no se encuentra disponible en la caché de ningún cliente vecino, se realiza la petición de esos precintos al servidor secundario que tiene asignado el cliente.

Por último, el submódulo de proceso de peticiones en curso se encarga de recoger las respuestas a esas peticiones de precintos y de almacenarlas en la caché del cliente.

Una explicación más detallada de estos submódulos es la siguiente:

5.2.3.3.1 – Submódulo de cálculo de la información a descargar

Tal como se explicaba en el apartado 2.1.4, la información de la superficie del terreno que se encuentra disponible para el sistema se organiza en un quadtree. Un quadtree es una estructura jerárquica en forma de árbol que permite dividir el terreno en regiones más pequeñas. Cada nivel del quadtree representa una versión del terreno de diferente resolución. Se obtendrá una mayor resolución a medida que se profundice en el quadtree.

El submódulo de cálculo de la información a descargar decide qué información se ha de descargar en cada momento. Para ello lleva a cabo el proceso de generación del quadtree del algoritmo de triangulación, obteniendo el conjunto de nodos del quadtree a descargar. Los detalles de este proceso y la métrica empleada se comentaron en el apartado 3.1.4.

Una vez obtenido el conjunto de nodos a descargar, se crea un nodo de petición para cada uno de ellos, introduciéndolos dentro de una lista de descarga. A su vez, aquellos nodos del quadtree ya descargados que se encuentren alejados y fuera del campo de visión del observador, se introducen en una lista de borrado, para que en el caso de que sea necesario eliminar algún nodo de la memoria del sistema, primeramente se borren estos nodos, ya que éstos serán los que tengan menos probabilidades de volver a ser necesitados por el cliente para su visualización.

5.2.3.3.2 – Submódulo de Proceso de la Lista de Descarga

Este submódulo se encarga de ordenar los nodos de la lista de descarga siguiendo la misma métrica empleada por el algoritmo de triangulación. Una vez ordenada, se realiza la descarga de los NDescSim primeros elementos de esa lista, donde NDescSim hace referencia al número máximo de descargas simultáneas que puede llevar a cabo un cliente. Este parámetro se fija inicialmente en función del ancho de banda y de la potencia computacional del equipo del cliente, ya que dependiendo del ancho del banda podrá enviar y recibir una mayor o menor cantidad de información, y dependiendo de la potencia computacional será capaz de procesar una mayor o menor cantidad de esa información recibida.

Los nodos de la lista de descarga contienen un conjunto de precintos requeridos de los cuáles algunos pueden encontrarse ya descargados en el cliente y otros no. Por lo tanto, el cliente comprueba qué precintos no tiene disponibles y realiza, para esos precintos, peticiones de información acerca de su disponibilidad a los clientes de su lista de clientes.

Los nodos de petición procesados se eliminan de la lista de descarga y se introducen en la lista de peticiones pendientes.

5.2.3.3.3 – Submódulo de Proceso de Peticiones Pendientes

Este submódulo se encarga de procesar las respuestas a las peticiones de información sobre la disponibilidad de los precintos requeridos enviadas a los clientes vecinos.

Para cada nodo introducido dentro de la lista de peticiones pendientes, se espera un tiempo T para recibir los mensajes de respuesta del conjunto de clientes a los que se ha pedido información. Este tiempo T depende del estado de la red y del ancho de banda de las conexiones entre los clientes. Este valor se calcula de forma empírica y se establece como un parámetro inicial del sistema.

Una vez transcurrido este tiempo T, se evalúan todas las respuestas y se decide a qué cliente se le pide cada precinto. Para ello se crea una lista ordenada de clientes por el tiempo esperado de descarga de un precinto de información.

El cálculo de este tiempo esperado de descarga se realiza de la siguiente forma:

$$T_{esperado} = T_{transmisión} + \left(n_{precintos\ cola} + 1\right) * T_{proceso}$$

Ecuación 5.4 – Cálculo del tiempo esperado de descarga de un precinto

Donde:

$T_{transmisión}$ es el tiempo medio que se tarda en transmitir un precinto entre los dos clientes que van a intercambiárselo.

$N_{precintos\ cola}$ es el número de precintos que el cliente vecino tiene en espera para ser procesados y enviados.

$T_{proceso}$ es el tiempo medio que el cliente vecino tarda en extraer un precinto de su caché y enviarlo al cliente que lo ha solicitado.

Se le suma uno al número de peticiones en cola pendientes de ser procesadas porque se tiene en cuenta la petición actual que se le va a realizar.

Si alguno de los precintos no puede obtenerse de ningún cliente, bien porque no se encuentra disponible en sus cachés, o bien porque el valor de tiempo esperado de respuesta es superior al permitido, se llevará a cabo una petición de ese precinto al servidor secundario.

Una vez realizadas todas las peticiones, se elimina el nodo de petición de la lista de peticiones pendientes y se almacena en la lista de peticiones en curso.

5.2.3.3.4 – Submódulo de Proceso de Peticiones en Curso

Este proceso se encarga de gestionar la llegada de los precintos pedidos y de almacenarlos en la caché de precintos para su posterior utilización.

Este proceso realiza un control de los errores en el envío de precintos, de forma que si una petición de un precinto tarda más tiempo en recibirse que el valor de tiempo T_E (tiempo esperado) fijado inicialmente, se evalúa si todavía es necesario descargar esos precintos, y en caso afirmativo se procede a repetir la petición de ese precinto a otro cliente o al servidor secundario, según se considere adecuado atendiendo al valor del tiempo esperado de respuesta.

Este valor T_E se calcula de forma empírica en función del ancho de banda y la velocidad de transmisión de la red utilizada.

Una vez se hayan recibido todos los precintos de la petición, ésta se elimina de la lista de peticiones en curso.

5.2.3.4 – Módulo de gestión del quadtree

Este módulo se encarga de introducir y eliminar nodos dentro de la estructura del quadtree.

En función de las capacidades de memoria y de procesamiento del cliente, se establece una cantidad máxima de información del terreno que se puede mantener almacenada en la memoria del cliente, y por lo tanto, organizada dentro del quadtree.

Cuando la información de un nuevo nodo ha sido completamente descargada y se encuentra disponible en la caché del cliente, este módulo se encarga de introducir el nodo en la estructura del quadtree en su posición adecuada.

Cuando se haya superado la capacidad máxima de almacenamiento en la memoria del cliente, este módulo procesa la lista de borrado que ha sido rellenada por el módulo de descarga de datos**,** eliminando del quadtree aquellos nodos que se encuentran más alejados y fuera del campo de visión del observador, y que, por lo tanto, tienen una menor probabilidad de volver a ser necesitados por el cliente para su visualización. De esta forma, se pueden incorporar sin problemas nuevos nodos al quadtree manteniendo el uso de la memoria del cliente controlado.

5.2.3.5 – Módulo de actualización de la escena

Este módulo se encarga de decidir el conjunto de regiones del terreno que se han de dibujar en pantalla empleando para ello el mecanismo especificado en el algoritmo de triangulación, en el cuál se recorre el quadtree que almacena la información de la escena introduciendo en una lista de dibujo aquellos nodos que han de visualizarse. Los detalles del proceso de actualización de la escena se comentaron en el apartado 3.1.4.

5.2.3.6 – Módulo de visualización de la escena

Se encarga de realizar el proceso de visualización del conjunto de nodos almacenados en la lista de dibujo elaborada por el módulo de actualización de la escena. Los detalles del proceso de visualización se comentaron en el apartado 3.1.4.

5.2.3.7 – Módulo de envío de mensajes

Se encarga de elaborar y enviar los diferentes tipos de mensajes empleados en la comunicación entre los clientes.

Como se comentó en el módulo de recepción de mensajes (apartado 5.2.3.1), estos mensajes pueden ser de cuatro tipos diferentes:

a) Peticiones de Información: son mensajes que envía el cliente a cada cliente de su lista de clientes para averiguar en cuáles de ellos se encuentran disponibles en su caché los precintos requeridos.

b) Peticiones de precintos: son mensajes enviados por el cliente a un cliente o a un servidor secundario para obtener un conjunto de precintos requeridos.

c) Respuestas de Información: son mensajes de respuesta enviados por el cliente en respuesta a las peticiones de información acerca de la disponibilidad de un conjunto de precintos en su caché realizados por otros clientes.

d) Respuestas de Precintos: son mensajes de respuesta enviados por el cliente que contienen un conjunto de precintos requeridos por algún otro cliente.

5.2.3.8 – Módulo de conexión con el servidor de comunicaciones

Este módulo se conecta periódicamente con el servidor de comunicaciones para obtener la lista de clientes vecinos y el servidor secundario de los que se puede descargar la información del terreno.

El cliente envía al servidor de comunicaciones su posición, el tiempo de procesado medio de un precinto, el número de precintos en la cola y el servidor secundario actual al que se encuentra conectado. El servidor de comunicaciones le proporciona la posición, el tiempo

de procesado medio de un precinto y el número de precintos en la cola de cada uno de los clientes vecinos de su lista. Esta información permite al cliente ordenar inicialmente esa lista de clientes vecinos empleando el valor de tiempo esperado de respuesta, cuyo cálculo se comentó en el submódulo de proceso de peticiones pendientes (apartado 5.2.3.3.3).

Cuando el cliente recibe una nueva lista de clientes de los que puede descargar, no se substituye directamente la antigua lista, sino que de la antigua lista se eliminan aquellos clientes que peor calidad de servicio le han proporcionado, sustituyéndolos por los nuevos clientes recibidos, manteniendo el número máximo de clientes en NCliVec. NCliVec es un parámetro que hace referencia al número máximo de clientes vecinos que cada cliente puede tener almacenados en su lista. Este parámetro es especificado inicialmente y se selecciona en función del ancho de banda y de la potencia computacional del equipo del cliente. Debido a que según el ancho de banda de que disponga el cliente, éste podrá intercambiar un mayor o menor número de mensajes, y dependiendo de la potencia computacional, podrá procesar un mayor o menor número de estos mensajes.

La eliminación de un cliente de la lista antigua se produce cuando el porcentaje entre las peticiones servidas por ese cliente y las peticiones totales realizadas no supera un cierto valor umbral. Este valor umbral se especifica inicialmente como un parámetro del sistema.

5.2.3.9 – Módulo de gestión de eventos de teclado y ratón

Es el módulo encargado de capturar los eventos del teclado y del ratón que permiten al usuario navegar de forma interactiva por la superficie del terreno. También se encarga de realizar el control del vuelo de un conjunto de rutas predefinidas que permiten al usuario desplazarse de forma automática a lo largo de la escena.

5.2.4 – Servidor de Comunicaciones

Este nodo es el encargado de gestionar la topología de la red mixta cliente-servidor / P2P. Proporciona a los clientes una lista de clientes vecinos de los cuáles pueden descargar información, además del servidor secundario más adecuado para el cliente en cada momento.

Es una pieza fundamental del sistema debido a que la eficiencia del sistema depende en gran medida de una buena selección de los clientes vecinos y de los servidores secundarios que se asignan a cada cliente. Para realizar esa selección emplea la información que recibe periódicamente sobre el estado actual de los clientes y de los servidores secundarios.

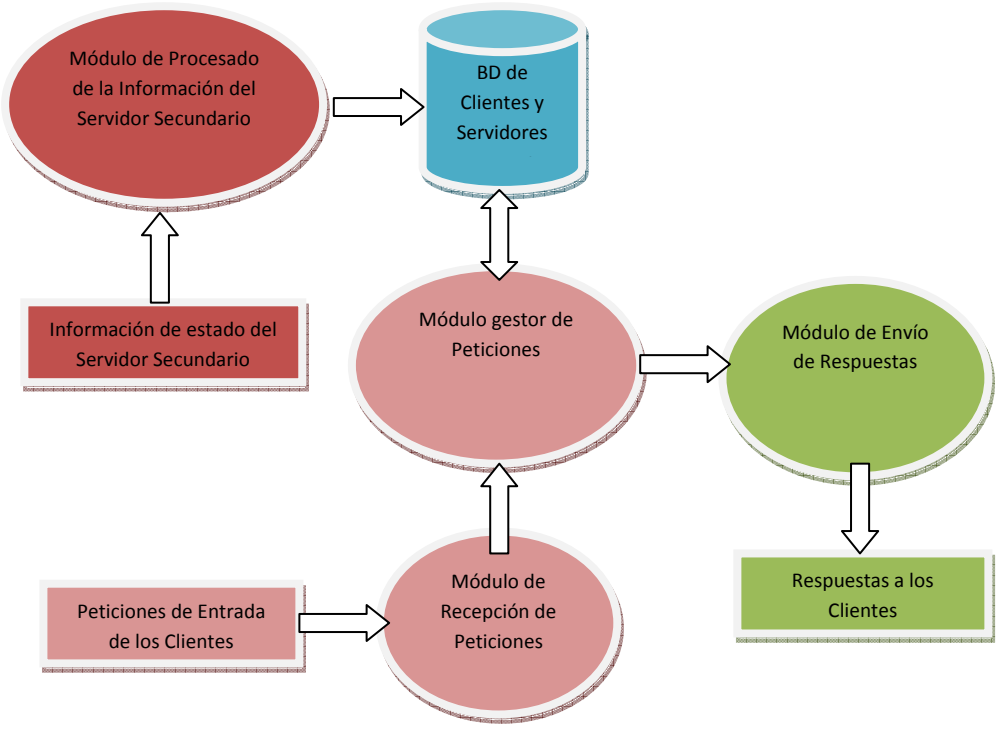


Figura 5.6 – Módulos y esquema de funcionamiento del Servidor de Comunicaciones

En la Figura 5.6 se muestra el funcionamiento del servidor de comunicaciones. Se pueden diferenciar dos partes. La primera se corresponde con la gestión de las peticiones realizadas por los clientes y la segunda con la recepción de la información del estado de los servidores de secundarios.

En la parte relacionada con las peticiones de los clientes, éstas son recogidas por el módulo de recepción de peticiones que se las enviará al módulo gestor de peticiones para su procesamiento. Este módulo registrará al cliente en el sistema en el caso de que sea un cliente nuevo, y obtendrá el conjunto de clientes vecinos y el servidor secundario más adecuado para él, pasando esa información al módulo de envío de respuestas que se encargará de elaborar y enviar el mensaje con esta información.

En la otra parte, relacionada con los servidores secundarios, el módulo de procesado de la información del servidor secundario se encargará de recibir la información del estado actual de los servidores secundarios y de actualizar esa información dentro de la base de datos de los elementos registrados en el sistema.

Una explicación más detallada de estos módulos es la siguiente:

5.2.4.1 – Módulo de recepción de peticiones

Este módulo se encarga de recibir las peticiones de los clientes, de comprobar que estén correctamente construidas, y de colocarlas en una cola de peticiones pendientes a la espera de que el módulo gestor de peticiones las procese.

5.2.4.2 – Módulo Gestor de peticiones

Este módulo se encarga de procesar las peticiones de los clientes de la siguiente forma:

Si es la primera vez que se conecta el cliente al sistema, se registra en el mismo. Para este registro se puede introducir algún mecanismo de control de acceso mediante login y password en el caso de que se considere apropiado, o permitir un acceso libre a todos los clientes.

Una vez el cliente ha sido registrado, o si ya se encontraba registrado previamente en el sistema, se procede a acceder a la base de datos de clientes y servidores secundarios registrados en el sistema para elaborar una lista de los NCliVec clientes con más

probabilidades de disponer de la información que va a necesitar el cliente en el momento actual, así como el servidor secundario que, además de no encontrarse saturado, tenga más posibilidades de tener almacenada en su caché la información que va a necesitar. Como se ha comentado en el apartado 5.2.3.8, el valor del parámetro NCliVec se especifica inicialmente y se selecciona en función del ancho de banda y de la potencia computacional del equipo del cliente.

Selección de los clientes más adecuados: para la selección de los clientes más adecuados, se sigue el criterio comentado anteriormente de que los clientes que se encuentren visualizando la misma región del terreno en el entorno virtual, van a necesitar un conjunto de datos similares. De forma que se van a seleccionar para su envío los NCliVec clientes vecinos cuya posición dentro de la escena sea más cercana a la del cliente.

Como se ha comentado en el módulo de conexión con el servidor de comunicaciones del cliente (apartado 5.2.3.8), junto con la información necesaria para conectar con los clientes (su dirección IP), el servidor de comunicaciones envía de cada cliente vecino su posición, el tiempo de procesado medio de un precinto y el número de peticiones en la cola. Con esta información el cliente ordena la lista de clientes vecinos en función del valor de tiempo esperado de respuesta, tal como se comentó en el apartado 5.2.3.3.3.

Selección del servidor secundario más adecuado: la selección del servidor secundario más adecuado sigue el mismo criterio de proximidad empleado para los clientes, pero además se tiene en cuenta el nivel de carga de los servidores, algo habitual cuando se emplean múltiples servidores en un entorno distribuido [8][10]. Se selecciona, entre los servidores secundarios que sean capaces de proporcionar al cliente unas condiciones de calidad de servicio adecuadas, aquel que posea en su caché una mayor cantidad de datos de la región que está visualizando actualmente el cliente.

Con ello, el servidor de comunicaciones busca dos objetivos:

- Distribuir la carga entre los diferentes servidores secundarios evitando la saturación de los mismos y de esa forma garantizar un nivel de calidad de servicio al cliente mínimo.

- Distribuir a los clientes entre los diferentes servidores secundarios de forma que se agrupen espacialmente según la posición de la escena en la que se encuentren.

Cumpliendo estos objetivos, obtendremos unos servidores escalables que evitan la saturación y que se especializan por regiones del terreno, de forma que las peticiones de los clientes que se encuentren en una misma zona se realizan al mismo servidor secundario, aumentando con ello la probabilidad de que los datos ya se encuentren en su caché, y por lo tanto, no se tenga que acudir al servidor principal para obtenerlos.

El proceso para seleccionar el servidor secundario es el siguiente:

1 – Primero, se ordena el conjunto se servidores secundarios respecto a la distancia entre la posición media de los datos almacenados en la caché del servidor secundario y la posición del cliente en la escena.

2 - Una vez ordenados, se seleccionará el servidor secundario más cercano, considerando sólo aquellos que cumplan con las condiciones de calidad de servicio establecidas. La calidad de servicio se relaciona con el tiempo esperado de respuesta de un precinto por parte del servidor secundario, y para que cumpla con los requisitos mínimos, este tiempo no debe superar un umbral U_r específico.

El servidor secundario envía los siguientes parámetros al servidor de comunicaciones de forma periódica, para que éste pueda seleccionar para cada cliente el servidor secundario más adecuado (estos parámetros se han definido en el apartado 5.2.2.6):

- Los parámetros que definen la función cuadrática que determina la probabilidad de que un precinto se encuentre en la caché del servidor secundario (K1, K2 y K3).

- La posición media de los datos almacenados en la caché del servidor secundario (P_{datos}).

- El tiempo de procesado medio de los precintos que se encuentran en la caché (TP_{cache}).

- El tiempo de procesado medio de los precintos que no se encuentran en la caché ($TP_{nocache}$).

- El número de precintos pendientes ($NPre_{cola}$).

- El tiempo de espera en cola medio (T_{esp}).

Con estos parámetros se realiza el cálculo del tiempo esperado de respuesta de un precinto por parte de cada servidor secundario. Si este tiempo es menor que un cierto umbral, se considerará que es un servidor válido para ser asignado a los clientes, ya que se encuentra dentro de los límites de carga permitidos.

El cálculo de este tiempo se realiza de la siguiente forma:

$$distancia = calculodistancia(posicióncliente, P_{datos})$$

Ecuación 5.5 – Cálculo de la distancia entre la posición del cliente en la escena y la posición media de los datos de la caché del servidor secundario

$$Probabilidad = \frac{1}{K1 * distancia^2 + K2 * distancia + K3}$$

Ecuación 5.6 – Cálculo de la probabilidad de que un precinto se encuentre en la caché de un servidor secundario

$$Tiempo_{esperado} = (T_{esp} * NPre_{cola}) + (Probabilidad * TP_{cache}) + ((1 - Probabilidad) * TP_{nocache})$$

Ecuación 5.7 – Cálculo del tiempo esperado de respuesta de un precinto por un servidor secundario

Donde:

calculodistancia es una función que calcula la distancia entre la posición del cliente en la escena y la posición media de los datos almacenados en la caché del servidor secundario.

Probabilidad es la probabilidad (valor entre 0 y 1) de que un precinto se encuentre en la caché del servidor secundario. Para el cálculo de la probabilidad se emplean los parámetros que describen la función cuadrática de probabilidad calculada por el servidor secundario (descrita en el módulo de conexión con el servidor de comunicaciones en el apartado 5.2.2.6.2).

Para el cálculo del tiempo esperado, se tiene en cuenta, tanto el tiempo que va a tener que esperar la petición de un precinto a ser procesada por el servidor secundario ($T_{esp} * NPre_{cola}$), como el tiempo que va a tardar ese servidor secundario en procesar la petición y enviar el precinto al cliente. Este tiempo de procesado y envío dependerá de si el precinto pedido se encuentra en la caché del servidor secundario, y por lo tanto puede ser enviado inmediatamente al cliente ($Probabilidad * TP_{cache}$), o por el contrario, el precinto no se encuentra en la caché, y hay que esperar a obtenerlo del servidor principal para poder enviarlo ($(1 - Probabilidad) * TP_{nocache}$).

Además, este módulo es capaz de detectar cuando un cliente o un servidor secundario se han desconectado del sistema. Puesto que ambos tipos de nodo deben enviar mensajes de forma periódica al servidor de comunicaciones, a la hora de seleccionar el conjunto de clientes y de servidores secundarios para un cliente, se comprueba que estos se hayan comunicado previamente con el servidor de comunicaciones dentro de un periodo de tiempo T_{vida}, en caso contrario se eliminará al nodo del sistema.

El T_{vida} se establece como un parámetro inicial del sistema, y tiene como valor el triple del tiempo con que periódicamente se conecta un cliente o servidor secundario con el servidor de comunicaciones. Con ello se permite la pérdida de dos paquetes consecutivos por parte de cada elemento del sistema, que se considerarán motivados por fallos temporales de las líneas de conexión, no por la desconexión del elemento del sistema.

5.2.4.3 – Módulo de envío de respuestas

Este módulo se encarga de elaborar y enviar a los clientes los mensajes de respuesta que contienen el conjunto de clientes vecinos y el servidor secundario más adecuado en el momento actual.

En estos mensajes, se enviará junto a la IP de cada cliente vecino y del servidor secundario asignado, la posición, el tiempo de procesado medio de un precinto y el número de peticiones en la cola de cada cliente vecino, de forma que el cliente pueda ordenar inicialmente esa lista de clientes vecinos empleando el valor de tiempo esperado de respuesta, cuyo cálculo se comentó en el submódulo de procesado de peticiones pendientes (apartado 5.2.3.3.3).

5.2.4.4 – Módulo de procesado de la información del servidor secundario

Este módulo recibe los mensajes con los parámetros de la información del estado actual del servidor secundario, y los almacena en la base de datos para su posterior uso a la hora de seleccionar el servidor secundario más adecuado para cada cliente.

Tal como se ha comentado previamente en el apartado 5.2.2.6, estos parámetros son:

- Los parámetros que definen la función cuadrática que determina la probabilidad de que un precinto se encuentre en la caché del servidor secundario.

- La posición media de los datos almacenados en la caché del servidor secundario.

- El tiempo de procesado medio de los precintos que se encuentran en la caché.

- El tiempo de procesado medio de los precintos que no se encuentran en la caché.

- El número de precintos pendientes.

- El tiempo de espera en cola medio.

5.3 – Metodología de Evaluación

En este apartado se describe la metodología que se va a emplear para evaluar la arquitectura mixta cliente-servidor / P2P desarrollada en esta tesis.

Como se comentó en el apartado 2.3.3, el uso de una arquitectura P2P es común en múltiples campos, sin embargo, en el área de la visualización de terrenos en tiempo real es una arquitectura que apenas se ha utilizado hasta el momento. Esto supone que actualmente no exista una metodología de evaluación definida y estandarizada para llevar a cabo la validación de esta arquitectura.

Sin embargo, existe un campo relacionado con los gráficos, donde las arquitecturas P2P son ampliamente utilizadas, que es en el de los entornos virtuales distribuidos (DVE). En este campo, se elaboran aplicaciones donde un conjunto de clientes interactúa e intercambia información entre ellos, por lo que guarda cierta semejanza con nuestro sistema.

Por ello, a pesar de que se va a definir un esquema propio de validación, se va a prestar atención a la metodología empleada para validar los sistemas DVE [90][91], con el objetivo de que la validación sea lo más completa y precisa posible.

Al igual que en los sistemas DVE, la metodología que se va a emplear para validar nuestra arquitectura se basa en la evaluación de prestaciones mediante la prueba y la posterior revisión de los diferentes parámetros de la arquitectura, teniendo en cuenta tanto la influencia del número de servidores y usuarios que participan en el sistema, como el movimiento de estos últimos a lo largo de la escena, ya que éstos son parámetros que caracterizan habitualmente tanto a una arquitectura P2P, como a un sistema de visualización de terrenos con bases de datos remotas.

Para realizar la evaluación, se llevarán a cabo un conjunto de pruebas para validar el correcto funcionamiento de los diferentes módulos que componen esta arquitectura mixta. Además, también se van a realizar pruebas para comparar esta arquitectura con la arquitectura clásica cliente-servidor, con el objetivo de ver cuál de las dos arquitecturas se adapta mejor a un sistema de visualización de terrenos en tiempo real con bases de datos remotas.

Como se ha comentado previamente, una arquitectura P2P pura carece de sentido en un sistema de visualización de terrenos con bases de datos extensas, ya que con ella resulta muy complicado tener disponible en todo momento toda la información del terreno necesaria, al no poderse asegurar que siempre va a haber un conjunto mínimo de nodos conectados que almacenen esa información. Para ello, sería necesario repartir toda la información inicialmente entre los primeros clientes conectados, y asegurarse de que

conforme se desconecten siempre haya un cliente que cubra sus datos, cosa que parece imposible de llevar a cabo en la práctica. Por todo ello se descarta, desde el inicio, como una posible solución la arquitectura P2P pura, y no se va a realizar ninguna prueba con dicha arquitectura.

Para poder realizar las pruebas con las otras dos arquitecturas, se han implementado todos los módulos que componen el sistema descrito en el apartado anterior, empleando como lenguaje de programación C++. Dentro del sistema se han desarrollado cuatro aplicaciones diferentes que pueden funcionar de forma independiente en distintos equipos: el servidor principal, el servidor secundario, el servidor de comunicaciones y el cliente.

En las aplicaciones del servidor principal, del servidor secundario y del cliente, se emplea el esquema de compresión que se ha elaborado en el apartado 4.2 para la compresión, descompresión y transmisión de la información del terreno, ya que como se comenta en el apartado 4.5, es el esquema de compresión que permite llevar a cabo una transmisión y descompresión de regiones de la superficie del terreno de forma independiente sin la aparición de artefactos visuales en las fronteras, y manteniendo unas tasas de compresión altas.

Para ello, al inicio de las pruebas, toda la información del terreno se comprime empleando este esquema de compresión y se coloca en la base de datos a la que tiene acceso el servidor principal.

Para realizar la visualización de la información de los terrenos almacenados en esa base de datos, la aplicación del cliente empleará el algoritmo de triangulación NRQT descrito en el apartado 3.1.4, debido a que, como se comenta en el apartado 3.3.4, es un algoritmo de triangulación que se adapta perfectamente a un sistema de visualización de terrenos con bases de datos remotas.

Para probar la arquitectura mixta cliente-servidor / P2P se pondrán en funcionamiento las diferentes aplicaciones que forman parte del sistema: el servidor principal, el servidor de comunicaciones, el servidor secundario y los clientes, empleando máquinas diferentes para cada elemento del sistema, que funcionarán de forma simultánea.

Para probar el sistema clásico cliente-servidor, sólo es necesario poner en funcionamiento uno o varios servidores con acceso a toda la información de la base de datos del terreno, y un conjunto de clientes que accedan directamente a ese servidor o servidores, sin utilizar servidores de comunicaciones.

5.3.1 – Parámetros de las pruebas

Para evaluar la arquitectura mixta cliente-servidor / P2P, se ha planteado un modelo de evaluación de prestaciones atendiendo a la variación de diversos parámetros fundamentales en el sistema. Esta variación de los parámetros va a introducir sistemáticamente diferentes niveles de carga en el sistema. De entre todas las opciones configurables, se consideran los siguientes como los parámetros fundamentales:

- **Número de clientes conectados**. Indica el número de clientes que se van a conectar al sistema.

El número de clientes conectados es un factor que influye de forma importante en las prestaciones de la arquitectura mixta cliente-servidor / P2P, y servirá para determinar la capacidad de escalabilidad de la misma, que es uno de los principales objetivos a conseguir por el sistema diseñado en esta tesis.

Variando el número de clientes conectados se podrá estudiar cómo afecta esta variación a la carga de los clientes y de los servidores secundarios. Es previsible que a medida que haya un mayor número de clientes conectados, las necesidades de descarga de información aumenten y que ésta deba repartirse entre el conjunto de clientes y servidores secundarios conectados al sistema. También permitirá observar si la variación en el número de clientes conectados al sistema afectará a la necesidad de éstos de acudir a los servidores secundarios a por información (debido a que esta variación influirá previsiblemente en la cantidad de clientes vecinos a los que se podrá pedir información y por tanto en la cantidad de información que se podrá extraer de ellos). La cantidad de accesos a los servidores secundarios por parte del cliente también podría afectar a la información que se almacena en la caché de los servidores secundarios, afectando a su vez a la cantidad de accesos necesarios al servidor principal por parte de éstos para conseguir la información que le es requerida por los clientes, y por consiguiente, influyendo en el tiempo necesario para enviar esta información al cliente.

Para todas las pruebas se va a variar el número de clientes entre 10 y 35 clientes. El número máximo es 35 porque es la cantidad de equipos que tenemos disponibles para ejercer como clientes en las pruebas. El momento de la conexión de los clientes al sistema se realiza de forma aleatoria aunque dentro de un intervalo de tiempo máximo.

Debido a que el número de clientes empleado es pequeño comparado con el número de clientes habitual en este tipo de redes, se ha reducido el tamaño de la base de datos del terreno empleada, como se explicará en el apartado 5.3.2, de forma que las pruebas sean equivalentes a las condiciones reales con extensas bases de datos y un elevado número de clientes.

Además, este bajo número de clientes conectados es uno de los motivos por los que se ha decidido la elaboración de un simulador de este sistema real, de forma que se pueda probar el mismo con un número de clientes conectados más elevado. Este simulador se explicará en el Capítulo 6.

- **Número de servidores secundarios conectados.** Indica el número de servidores secundarios conectados al sistema.

Variando el número de servidores secundarios se puede evaluar cómo se distribuye la carga general entre ellos, y si el empleo de un diferente número de servidores afecta de alguna manera a la necesidad de acceso de éstos al servidor principal cuando no disponen en su caché de la información requerida por los clientes. Esta prueba también permitirá evaluar cómo afecta la variación del número de servidores secundarios a su especialización espacial por regiones del terreno.

El número de servidores secundarios utilizados en las pruebas variará entre 1 y 3. El número máximo elegido es 3 debido a que se prevé que esta cantidad será suficiente para soportar la carga a la que puedan someter al sistema el número máximo de clientes conectados al mismo (que será de 35). Inicialmente todos los servidores secundarios se encontrarán en funcionamiento y disponibles para los clientes.

- **Ruta.** Establece la trayectoria que va a recorrer un cliente dentro de la escena durante las pruebas.

El movimiento de los clientes a lo largo de la escena puede tener una influencia importante en el rendimiento de todo el sistema. La ruta seguida por el cliente va a determinar la información que éste va a necesitar descargar para su visualización, y por tanto, la que posteriormente almacenará en su caché. Además, también condicionará la información almacenada en la caché de los servidores secundarios, ya que toda esa información ha sido servida o tendrá que ser servida por ellos a los clientes en algún momento. Así mismo, el movimiento de los clientes también influirá en la frecuencia con la que los servidores secundarios deberán acudir al servidor principal para obtener esa información requerida, que dependerá de si ya se encuentra o no almacenada en su cache.

Puesto que se van a emplear un conjunto relativamente elevado de clientes y resulta complicado disponer de suficientes agentes humanos para llevar a cabo las pruebas, se ha decidido automatizar el movimiento de los clientes, siguiendo un conjunto de rutas generadas a partir de un conjunto de puntos aleatorios de la escena obtenidos empleando un tipo de distribución predeterminada.

Se ha realizado la distribución espacial de los puntos aleatorios teniendo en cuenta las distribuciones empleadas en los sistemas DVE [86][89][92], con las modificaciones

adecuadas para que se ajusten a los comportamientos más habituales de los usuarios dentro de los sistemas de visualización de terrenos en tiempo real. Estas distribuciones se comentan con más detalle en el apartado 5.3.3, y serán de los siguientes tipos: uniforme, normal empleando un punto caliente y normal empleando varios puntos calientes.

Cada par de puntos generados siguiendo una de estas distribuciones formará un segmento de recta que marcará el recorrido que los clientes van a seguir entre esos dos puntos. Los puntos intermedios de cada segmento incluidos dentro de la ruta son generados a partir de interpolaciones lineales de los extremos de esos segmentos de recta, estableciéndose un número fijo de estos puntos intermedios. Las trayectorias se generan en tiempo de preproceso y se almacenan de forma que se puedan emplear en todas las otras pruebas, manteniéndose de esta forma las mismas condiciones de funcionamiento.

- Tamaño de la Caché: Indica la cantidad de información medida en bytes que se puede almacenar en una cache.

El tamaño de la información del terreno suele ser muy elevado, del orden de varios terabytes, por lo tanto, los clientes generalmente no van a contar con los recursos de ancho de banda y almacenamiento necesarios para poder descargar toda esa información de forma local para posteriormente visualizarla. Lo que ocurrirá normalmente será que los clientes sólo se descargarán la información que van a visualizar. Aún así, en ocasiones, esta información también puede llegar a ser demasiado grande y puede pasar que el cliente no tenga suficiente capacidad de almacenamiento local disponible, o simplemente que el usuario no desee reservar en su disco local espacio para tal cantidad de datos.

Por ello, otro parámetro que se va a variar en las pruebas es el tamaño de la caché local que va a utilizar cada cliente para almacenar la información, el cual influirá en la cantidad de información que va a poder intercambiar con otros clientes vecinos, lo que a su vez afectará a la necesidad del cliente a la hora de acudir al servidor secundario a por información que no pueda encontrar en sus clientes vecinos.

También se harán pruebas variando el tamaño de la caché de los servidores secundarios. Puesto que se desea obtener unos servidores secundarios especializados por regiones del terreno, a priori, no se considera necesario replicar en cada servidor secundario toda la información de la base de datos principal, que como ya se ha comentado, puede tener un tamaño muy elevado. La limitación del tamaño de esa caché, permitirá estudiar cómo afecta ésta a la necesidad del servidor secundario a la hora de acudir al servidor principal para obtener la información que no se encuentra almacenada en su caché.

A la hora de fijar el tamaño de las cachés, tanto del cliente como de los servidores secundarios, se ha decidido limitar por separado el tamaño de la caché dedicada a la información de las alturas de la superficie del terreno y la dedicada a la información de las

texturas del terreno, porque habitualmente el tamaño de cada tipo de información es diferente. Ello es debido principalmente a que necesitan almacenar diferente tipo de información (tres componentes de color para las texturas por un sólo valor de altura para las alturas). Además, las aplicaciones de visualización de terrenos generalmente suelen manejar datos de las texturas que tienen más resolución y un mayor tamaño que los datos de las alturas.

Por lo tanto, vamos a considerar cuatro parámetros de tamaño de caché que se van a poder variar en las pruebas; dos para especificar los tamaños de la caché de los clientes y dos para fijar los de los servidores secundarios. Éstos son el tamaño de la caché dedicada a la información de las alturas del terreno y el tamaño de la caché dedicada a la información de las texturas.

Los tamaños de caché para los servidores secundarios van a variar entre el 10% y el 50% del tamaño total de la base de datos empleada en las pruebas. Para los clientes, el tamaño de caché va a variar entre el 2.5% y el 40%. Estos porcentajes se consideran, a priori, unos valores razonables de almacenamiento para los servidores secundarios y para los clientes.

- Tamaño de la lista de clientes vecinos. Indica el número máximo de clientes vecinos a los cuales el cliente puede solicitar información del terreno.

La lista de vecinos de cada cliente determina de quien se puede descargar la información que necesita en cada momento. Es importante que sea correcta la elección asignada a cada cliente porque ello influirá en una mayor o menor necesidad de acceder al servidor secundario para obtener la información que esos clientes vecinos no pueden ofrecerle.

El tamaño de la lista de clientes vecinos afectará a la cantidad de información que se intercambia y se procesa entre los diferentes elementos del sistema. Variando el número de clientes vecinos incluidos en la lista se podrá estudiar cómo afecta el tamaño de la lista a la carga de la red y a los tiempos de latencia inducidos en el intercambio de la información entre los diferentes elementos del sistema.

El número máximo de clientes vecinos de la lista variará entre 2 y 10, que a priori, se consideran unos valores adecuados atendiendo al número máximo de clientes que pueden estar conectados al sistema (que como se ha comentado será de 35).

Por último, comentar que la variación de los parámetros descritos anteriormente (número de clientes conectados, número de servidores secundarios conectados, etc.) puede hacerse de forma totalmente libre e independiente, ya que no existen dependencias estrictas entre ellos que restrinjan los valores que se puedan emplear.

5.3.2 – Definición del escenario

Como se ha comentado en el apartado 5.3.1, las pruebas que se van a realizar del sistema están limitadas físicamente por el número de clientes disponibles, que como máximo será 35.

En un caso habitual de uso, se suele disponer de una base de datos extensa por la que se desplazan un número elevado de clientes, dando lugar a una determinada densidad de usuarios en la superficie del terreno. Esta densidad estará relacionada con la probabilidad de que los clientes puedan compartir información del terreno. Al tener limitado el número de clientes, si se emplea una base de datos extensa, la probabilidad de que los clientes compartan información es reducida, debido a que la densidad de usuarios en la superficie del terreno es muy pequeña. Por eso, se ha decidido limitar el tamaño de la base de datos empleada de forma que en las pruebas que se van a realizar, la densidad de usuarios en la superficie del terreno sea equivalente al caso habitual de uso, obteniendo de esa forma una similar probabilidad de que los clientes compartan información del terreno.

Por ello, para llevar a cabo las pruebas se han empleado los datos de altura y texturas del modelo "Puget Sound" [105]; concretamente, los ficheros de alturas de tamaño 8193 x 8193 y la textura de tamaño de 8192 x 8192. Como se comentó en el apartado 3.2, estos datos se han empleado anteriormente en múltiples trabajos aparecidos en la bibliografía científica, sobre todo para evaluar algoritmos de triangulación, debido a que abarcan superficies del terreno de diferentes niveles de rugosidad, por lo que van a ser adecuados para las pruebas a realizar.

Además, el escenario definido en "Puget Sound" encaja con el tipo de escenario que suele emplearse dentro de los sistemas DVE a la hora de evaluar sus prestaciones: es una escena virtual cuadrada, libre de obstáculos y con un tamaño fijo [4][72][86]. La única diferencia sustancial entre éstos y el escenario definido en "Puget Sound" radica en el tamaño de la región; en el caso de los DVE se suele restringir a menos de un kilómetro cuadrado mientras que en el caso de la visualización de terrenos se suelen utilizar datos de una extensión muy superior. En el caso de "Puget Sound", la región empleada tiene una superficie de 327680 x 327680 metros.

En este escenario, la resolución de cada valor de altura del terreno es de 16 bits (de 0 a 65535), y cada uno de estos valores tiene una correspondencia real de 0.1 metros. La separación de cada una de las muestras es de 40 metros.

Cada texel de la textura abarca la región del terreno comprendido entre cuatro vértices de altura vecinos. Por eso el tamaño de la malla de alturas es una fila y una columna mayor que el tamaño de la textura.

El tamaño de los ficheros que almacenan la información del terreno es de 192MB para las texturas y de 128MB para las alturas. Una vez comprimido con nuestro esquema de compresión sin pérdidas, el tamaño se reduce a 81 MB para las texturas y 37 MB para las alturas.

El tamaño del terreno empleado en las pruebas, su resolución y la velocidad de crucero equivaldría, por ejemplo, a un vuelo sobre la superficie del terreno en una avioneta que se desplaza a una altura superior a 1000 metros con una velocidad superior a 100 km/h. También podría ser equivalente al recorrido en un vehículo en un entorno urbano de alrededor de 10 kilómetros cuadrados, a una velocidad superior a 10km/h.

Para la realización de las pruebas se han utilizado un total de 40 equipos distintos distribuidos físicamente en 3 salas diferentes.

Cada uno de estos equipos asume un único rol dentro de la arquitectura del sistema. Así un conjunto de equipos hacen de clientes y el resto toma el papel de los diferentes tipos de servidores del sistema.

Para llevar a cabo las pruebas, los equipos con el rol de clientes van a seguir una de las rutas precalculadas de forma automática. Esta ruta se genera de forma semi-aleatoria siguiendo alguna de las distribuciones comentadas en el apartado 5.3.3

Asociados a cada uno de los equipos empleados para la realización de las pruebas, se almacenarán un conjunto de estadísticas que se emplearán para la posterior evaluación del funcionamiento del sistema.

5.3.3 – Modelos de Distribuciones

En una aplicación de visualización de terrenos, el comportamiento de los usuarios a la hora de visualizar los datos de la escena puede diferir mucho de unos a otros usuarios. Uno de los motivos puede estar relacionado con la naturaleza de los datos. Si los datos de terreno muestran una zona geográfica conocida por el usuario, éste generalmente tiende a desplazarse por ella (su zona de residencia, parajes que ha visitado, etc.). En cambio, si el usuario desconoce por completo la región, o dentro de la misma no hay ningún elemento de mayor interés para él que el resto, su movimiento por la escena suele ser uniforme, explorando cada región del terreno por igual.

Por ello, se va a evaluar el funcionamiento del sistema simulando los diferentes comportamientos de un usuario dentro de la escena (Figura 5.7). Para ello, los puntos

incluidos en las rutas se distribuirán espacialmente siguiendo los tipos de distribución que se suelen emplear a la hora de evaluar los sistemas DVE [86][89][92], los cuales resultan lo suficientemente generalistas y representativos para cubrir la mayor parte de los comportamientos habituales de los usuarios por la escena.

5.3.3.1 – Distribución Uniforme

En este tipo de distribución, los clientes se reparten inicialmente de forma uniforme por toda la escena y realizan desplazamientos dentro de la misma de forma aleatoria, desplazándose hacia puntos del mundo virtual distribuidos de manera uniforme. Este tipo de distribución simula el caso de un mundo virtual en el que no hay ningún punto de interés destacado para el usuario, sino que toda la superficie del terreno tiene el mismo nivel de interés para él. En los sistemas DVE se puede encontrar un ejemplo de su uso en [72].

5.3.3.2 – Distribución normal con un Punto Caliente (HotPoint)

En este tipo de distribución, los clientes se reparten inicialmente siguiendo una distribución normal alrededor de una misma región del terreno, y el movimiento dentro de la escena se centra en esa región. Esta distribución simula un mundo virtual con un punto de interés o punto caliente, el cual llama la atención de todos los usuarios. Este tipo de distribución podría corresponder, por ejemplo, a una superficie del terreno dentro de la cual se sitúe una ciudad o un accidente geográfico importante. En los sistemas DVE se puede encontrar un ejemplo de su uso en [88].

5.3.3.3 – Distribución normal con varios Puntos Calientes (Multiple Hotpoint)

En este tipo de distribución, los clientes se reparten inicialmente siguiendo distribuciones normales alrededor de un conjunto de regiones o puntos calientes del terreno, y el movimiento dentro de la escena se centra en cada una de esas regiones. Esta distribución simula un mundo virtual con varios puntos de interés de pequeño o mediano tamaño que pueden o no solaparse. Un ejemplo práctico podría ser una superficie de terreno en la que existieran varias ciudades de interés. En los sistemas DVE se puede encontrar un ejemplo de su uso en [88].

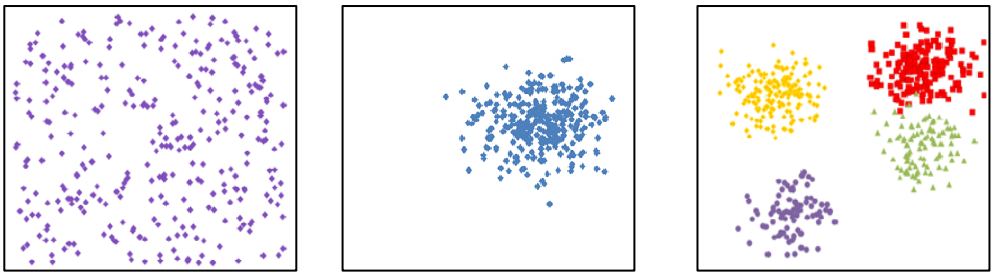


Figura 5.7 – Distribuciones empleadas: izquierda: uniforme; centro: normal con un punto caliente; derecha: normal con cuatro puntos calientes.

5.3.4 – Definición de Métricas

En este apartado se va a explicar qué valores se van a medir para evaluar las diferentes configuraciones de la arquitectura del sistema.

El objetivo principal de las pruebas es evaluar el buen funcionamiento del sistema, centrándonos en el problema de la transmisión de la información por la red. Por lo tanto, uno de los factores principales a tener en cuenta es la latencia, puesto que es la medida de este parámetro lo que se emplea principalmente para evaluar los sistemas basados en redes [35][42][76][120].

Otro factor importante a tener en cuenta es que se pueda obtener la información requerida antes de un tiempo límite predeterminado. Considerando en nuestro sistema la latencia como el tiempo transcurrido desde que un cliente decide que necesita una determinada información hasta que esa información se encuentra almacenada en su caché, se considerará que el sistema ofrece una calidad de servicio adecuada cuando esta latencia no supere el umbral del tiempo límite predeterminado.

Asumiendo que los relojes de los diferentes equipos pueden tener pequeños desajustes, para el cálculo de la latencia se van a emplear los propios mensajes que se intercambian los diferentes elementos de la arquitectura. Para ello, el nodo que inicia la comunicación introduce una marca de tiempo en los mensajes que envía, que será devuelta por el nodo destino dentro del mensaje de respuesta, permitiendo de esa forma que el nodo que inició la comunicación conozca el tiempo real transcurrido, ya que siempre emplea su reloj interno como referencia.

En el caso de que se produzca un error o un problema en las líneas de comunicación y los precintos se pierdan o tarden más tiempo del permitido en llegar al destino, el cliente, en el caso de que continúe necesitando esos precintos, procederá a realizar una nueva petición.

En estos casos, en vez de resetear la marca de tiempo del nuevo mensaje de petición, se mantendrá la que se empleaba en el mensaje original que inició la petición de los precintos. De esta forma, se obtendrá una latencia en la que se reflejará el efecto de la necesidad de repetir la petición de los mensajes de precintos.

Se van a considerar tres diferentes tipos de latencia para evaluar cómo transcurre el intercambio de la información entre los diferentes nodos de la arquitectura, así como el intercambio global de la información.

Los tres tipos son los siguientes:

- **latPreClientes**: es el valor de la latencia media para un cliente cuando los precintos del terreno han sido obtenidos de sus clientes vecinos. Este parámetro nos permitirá conocer cómo de rápido es el intercambio de información entre los clientes.

- **latPreServidores**: es el valor de la latencia media para un cliente cuando los precintos del terreno han sido obtenidos de los servidores secundarios. Este parámetro nos permitirá conocer cómo de rápido es el intercambio de información entre los servidores secundarios y los clientes.

- **latPreGlobal**: es el valor de la latencia media global para un cliente, teniendo en cuenta tanto los precintos que el cliente obtiene de sus clientes vecinos como de los servidores secundarios. Este valor nos permitirá evaluar la velocidad global del intercambio de información en el sistema.

El cálculo de las medias de los diferentes tipos de latencias se iniciará de nuevo tras cada comunicación con el servidor de comunicaciones, de forma que a la hora de realizar una nueva comunicación con el servidor de comunicaciones, tanto por parte de los clientes como por parte de los servidores secundarios, éstos le enviarán el valor actualizado de esa latencia para que éste pueda realizar la asignación a cada cliente de los clientes vecinos y del servidor secundario más adecuada.

Además de estos valores, se van a medir otros que ayudarán a validar los diferentes elementos de la arquitectura:

- **prSS (precintos respondidos por el Servidor Secundario)**: los clientes pueden realizar peticiones a otros clientes o a los servidores secundarios. Este parámetro mide el porcentaje de precintos del terreno pedidos por los clientes que al final han sido respondidos por los servidores secundarios. Con él se podrá comprobar cómo de buena es la selección de los clientes vecinos realizada por el servidor de comunicaciones, lo que influirá en una mayor o menor necesidad de acceso a los servidores secundarios. También permitirá analizar cómo

afecta la variación del tamaño de la caché del cliente a esta necesidad de acudir al servidor secundario, puesto que la cantidad de información disponible en sus clientes vecinos será directamente proporcional a ese tamaño.

El cálculo de este parámetro se realiza para cada cliente, almacenando el total de solicitudes de precintos que han realizado junto con el número de precintos que han sido respondidos por los clientes y por los servidores secundarios. Para que este parámetro se corresponda con la situación actualizada del sistema, se tomará como intervalo de tiempo para obtener el valor de este parámetro, el tiempo que transcurre entre dos comunicaciones consecutivas con el servidor de comunicaciones.

- **pvp (precintos vueltos a pedir)**: este parámetro mide el porcentaje de precintos pedidos por los clientes que, bien por errores en la transmisión, o bien porque han tardado más tiempo del permitido en ser recibidos por parte del cliente, han tenido que ser pedidos de nuevo. Es decir, este parámetro nos informará de los problemas ocurridos en la transmisión de la información.

El cálculo de este parámetro se lleva a cabo para cada cliente, considerando las veces en las que ha tenido que pedir de nuevo un precinto respecto al número total de precintos solicitados. Al igual que en el parámetro anterior, para que este parámetro se corresponda con la situación actualizada del sistema, se tomará como intervalo de tiempo el mismo que se emplea en el caso anterior.

- **pESS (precintos encontrados con éxito en la caché del servidor secundario)**: cuando los servidores secundarios reciben peticiones de precintos por parte de los clientes, los precintos pueden o no encontrarse en sus caché. Este parámetro mide el porcentaje de éxito a la hora de buscarlos en la caché, es decir, qué porcentaje de los precintos pedidos se encuentran en la caché y por la tanto no es necesario que sean pedidos al servidor principal. Este parámetro nos permitirá conocer cómo de buena es la especialización espacial por regiones del terreno de los servidores secundarios, y por consiguiente, permitirá estudiar si la asignación de los servidores secundarios a los clientes llevada a cabo por el servidor de comunicaciones es adecuada o no. Además, también permitirá conocer cómo van a afectar los cambios en el tamaño de la caché de los servidores secundarios a la presencia de los precintos solicitados por los clientes asignados a ese servidor en la caché.

El cálculo de este parámetro se lleva a cabo en los servidores secundarios, los cuales almacenarán además del número total de precintos que han respondido, el número de ellos que se encontraban en su caché y el número de ellos que han tenido que ser pedidos al servidor principal. Para que este parámetro se corresponda con la situación actualizada del sistema, se tomará como intervalo de tiempo para obtener el valor de este parámetro, el tiempo que transcurre entre dos comunicaciones consecutivas con el servidor de comunicaciones.

- **nColaCL (número de precintos medio en la cola del cliente)**: los clientes disponen de una cola en la que se almacenan las peticiones de precintos realizadas por otros clientes hasta que éstas pueden ser procesadas. Este parámetro almacena el número medio de precintos que se encuentran esperando en esa cola, es decir, nos informa de la carga del proceso del cliente encargado de responder a las peticiones de precintos de otros clientes.

El cálculo de este parámetro se realiza en los clientes, donde se almacena en cada instante el número de precintos que se encuentran esperando en la cola a ser procesados. Para que este parámetro se corresponda con la situación actualizada del sistema, se tomará como intervalo de tiempo para obtener el valor de este parámetro, el tiempo que transcurre entre dos comunicaciones consecutivas con el servidor se comunicaciones.

- **nColaSS (número de precintos medio en la cola del servidor secundario)**: Los servidores secundarios, al igual que los clientes, disponen de una cola en la que se almacenan las peticiones de precintos de los clientes a la espera de ser procesadas. Este parámetro almacena el número medio de precintos que se encuentran esperando en esa cola. Por lo tanto, nos permitirá conocer la carga del proceso del servidor secundario encargado de responder a las peticiones de precintos de los clientes.

El cálculo de este parámetro se realiza en los servidores secundarios, donde se almacena en cada instante el número de precintos que se encuentran esperando en la cola a ser procesados. Para que este parámetro se corresponda con la situación actualizada del sistema, se tomará como intervalo de tiempo para obtener el valor de este parámetro, el tiempo entre dos comunicaciones consecutivas con el servidor se comunicaciones.

5.3.5 – Características técnicas de los equipos y de la red local

Para realizar las pruebas se han empleado ordenadores con dos tipos de procesadores de similares características, Core2Duo 4400 a 2.00Ghz y Core2Duo 4600 2.40Ghz.

La cantidad de memoria RAM con la que cuenta cada ordenador es de 2 GB, y el espacio de almacenamiento libre en su disco duro oscila entre los 20 GB y los 50 GB.

Todos los ordenadores disponen de un chipset gráfico integrado en la placa base. Este chipset puede ser el Q965/Q963 Express y el Q35 Express. Ambos modelos han sido desarrollados por Intel, tienen características similares y la cantidad de memoria que pueden utilizar se encuentra compartida con la del ordenador y oscila entre 8 y 256 MB.

Las tarjetas de red empleadas van integradas en la placa base del ordenador y son del modelo Intel 82566DM Gigabit Network Connection. Todos los ordenadores se encuentran conectados en una red local de 100 Mbps perteneciente a la Universidad de Valencia.

Nuestro sistema compartirá la red local junto con el resto de usuarios habituales de la Universidad de Valencia, por lo que no se tiene un control preciso del tráfico de la red durante el tiempo que se prueba el sistema. Este es uno de los motivos por los que se ha decidido diseñar un simulador del sistema real que permita realizar diferentes pruebas, probando distintos tipos de red con distintas limitaciones en su ancho de banda, emulando las diferentes líneas de conexión habitualmente empleadas en Internet. Este simulador se explicará en el Capítulo 6.

5.4 – Resultados

En este apartado se van a mostrar los resultados de las pruebas realizadas para evaluar los diferentes elementos que componen la arquitectura mixta cliente-servidor / P2P.

5.4.1 – Variación en el tamaño de la caché de los clientes

Como se ha comentado previamente, generalmente un usuario no va a disponer de los suficientes recursos de almacenamiento para albergar en su equipo toda la base de datos del terreno. Además, de esa base de datos cada cliente normalmente sólo necesitará emplear una pequeña cantidad. Por ello, se ha decidido diseñar la arquitectura del sistema limitando la cantidad de información que cada cliente puede almacenar localmente. A priori, esto debería afectar a la cantidad de información que los clientes pueden intercambiar entre ellos, y por lo tanto, a la necesidad de los mismos de acudir al servidor secundario a por la información que no puedan obtener de sus clientes vecinos.

El objetivo de esta prueba es conocer cómo afecta esa limitación al funcionamiento general de sistema. Se han llevado a cabo diversas pruebas en las que se ha variado el tamaño de la caché empleada por los clientes con el objetivo de estudiar qué influencia tiene en el sistema esa variación. Los parámetros de configuración de las pruebas son los siguientes:

Número de clientes conectados = 35 (el número máximo de clientes disponible).

Número de servidores secundarios conectados = 2 (al menos son necesarios dos para que haya una distribución de la carga del sistema entre ellos).

Rutas: Distribución uniforme (toda la superficie del terreno tiene el mismo nivel de interés).

Tamaño de la caché del servidor secundario = 50% (se reparte la base de datos entre los dos servidores secundarios).

Tamaño de la lista de clientes vecinos = 10 (consideremos este valor como aceptable dado el número máximo de clientes utilizados en la prueba).

Las pruebas realizadas han sido:

Prueba_A: Tamaño de la caché de los clientes para las texturas = 2.5%.

Tamaño de la caché de los clientes para las alturas = 2.5%.

Prueba_B: Tamaño de la caché de los clientes para las texturas = 5%.

Tamaño de la caché de los clientes para las alturas = 5%.

Prueba_C: Tamaño de la caché de los clientes para las texturas = 10%.

Tamaño de la caché de los clientes para las alturas = 10%.

Prueba_D: Tamaño de la caché de los clientes para las texturas = 20%.

Tamaño de la caché de los clientes para las alturas = 20%.

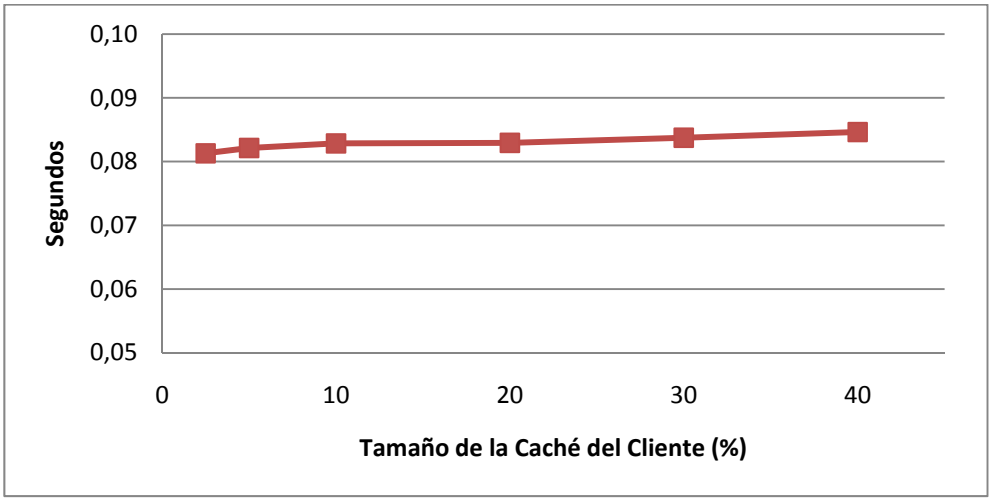
Prueba_E: Tamaño de la caché de los clientes para las texturas = 30%.

Tamaño de la caché de los clientes para las alturas = 30%.

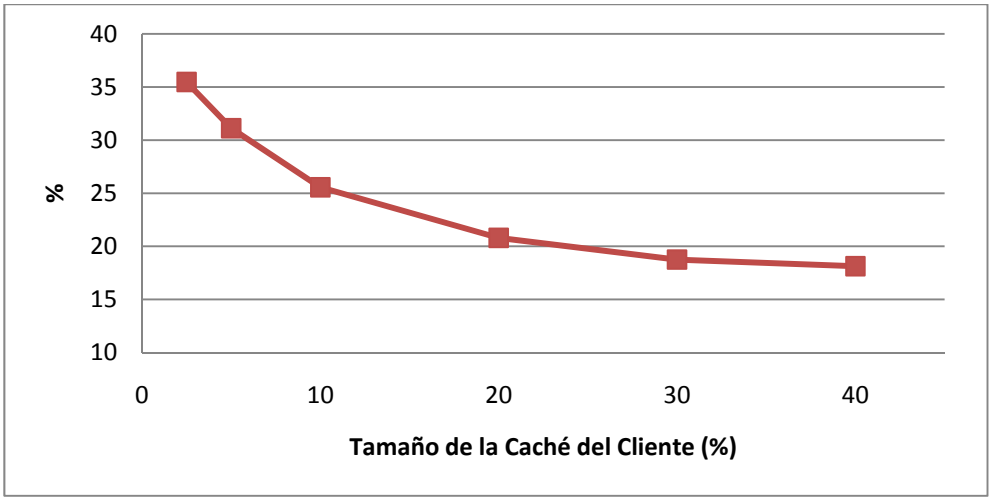
Prueba_F: Tamaño de la caché de los clientes para las texturas = 40%.

Tamaño de la caché de los clientes para las alturas = 40%.

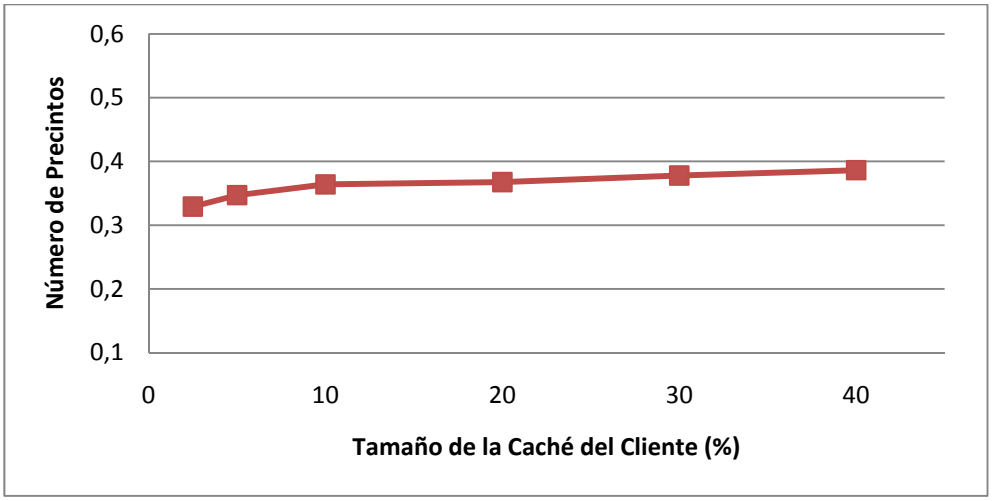
5.4.1.1 – Clientes



Gráfica 5.1 – Latencia media de los precintos intercambiados entre los Clientes



Gráfica 5.2 – Porcentaje medio de precintos respondidos por los Servidores Secundarios



Gráfica 5.3 – Número de precintos medio en las colas de los Clientes

En las gráficas anteriores se muestra la influencia que tiene en los clientes el tamaño de la caché que utilizan.

En la Gráfica 5.1 se muestra la latencia media de los precintos respondidos por los clientes. En ella se observa como el incremento del tamaño de la caché de los clientes no hace variar demasiado esta latencia, la cual crece ligeramente conforme aumenta el tamaño de la caché empleada.

En la Gráfica 5.2 se muestra el porcentaje medio de peticiones que realizan los clientes a los servidores secundarios. En ella se observa como a medida que se incrementa el tamaño de la caché de los clientes disminuye este porcentaje, aunque el ritmo con el que decrece es menor conforme el tamaño de la caché es mayor.

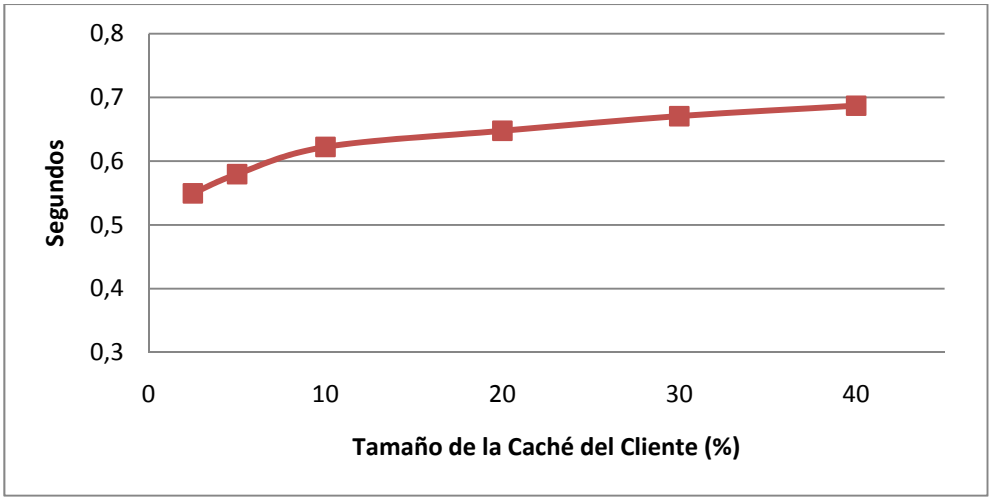
En la Gráfica 5.3 se muestra el número medio de precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por los clientes vecinos. En ella se observa como el incremento del tamaño de la caché de los clientes provoca un ligero incremento del tamaño de la cola, con lo que aumenta su carga.

La disminución del porcentaje de peticiones que se realizan a los servidores secundarios que se observa en la Gráfica 5.2 es debido a nuestro entender a que, a medida que los clientes emplean una caché mayor, disponen de una mayor cantidad de información almacenada que pueden intercambiar con otros clientes vecinos, reduciéndose de esta forma la necesidad de acceder a los servidores secundarios a por ella.

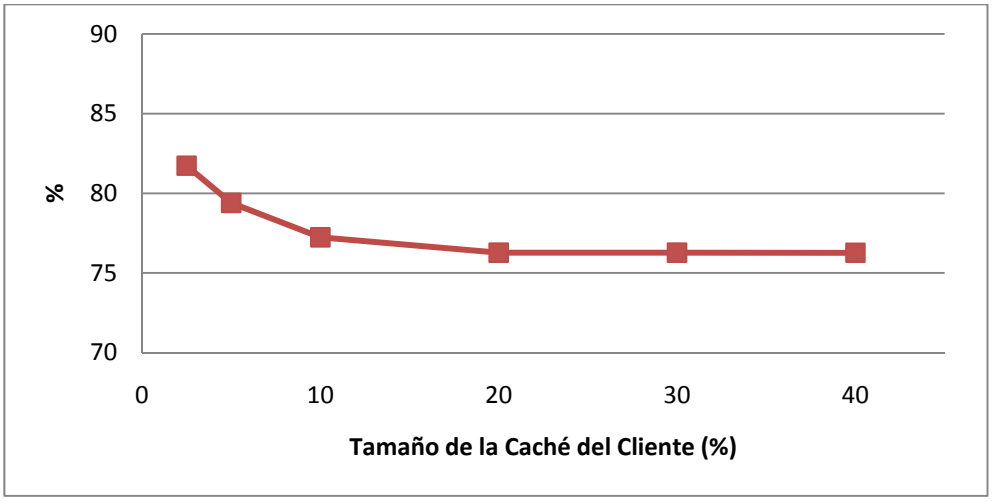
A su vez, esta reducción del número de peticiones a los servidores secundarios debe suponer necesariamente un aumento del número de peticiones entre clientes vecinos, lo que provoca un ligero incremento de la carga de los mismos, tal como se puede observar en la Gráfica 5.3.

Este aumento de la carga de los clientes pensamos que es la causa que provoca el ligero incremento de la latencia media de los precintos de información intercambiados entre los clientes conforme aumenta el tamaño de caché empleada por ellos, tal como se observa en la Gráfica 5.1

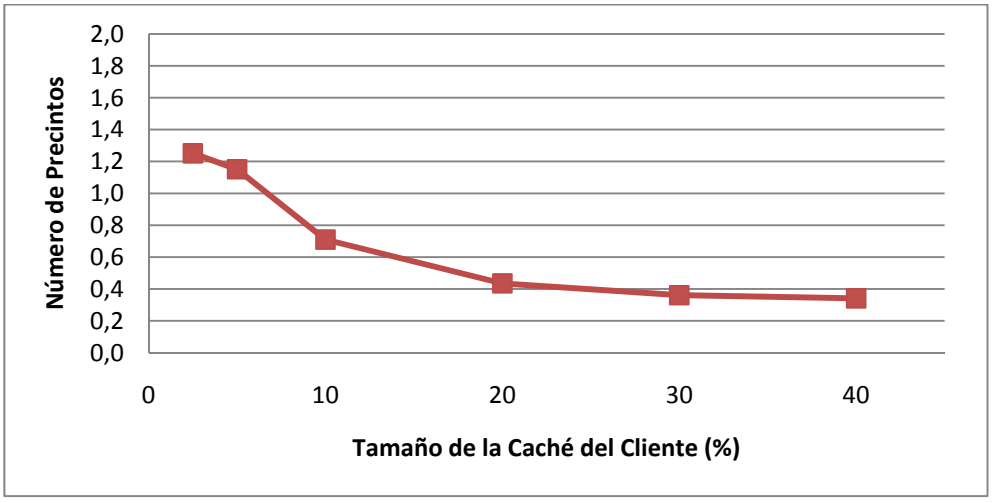
5.4.1.2 – Servidores Secundarios



Gráfica 5.4 – Latencia media de los precintos respondidos por los Servidores Secundarios



Gráfica 5.5 – Porcentaje medio de los precintos encontrados con éxito en la caché de los Servidores Secundarios



Gráfica 5.6 – Número de precintos medio en las colas de los Servidores Secundarios

En las gráficas anteriores se muestra la influencia que tiene en los servidores secundarios el tamaño de la caché que emplean los clientes.

En la Gráfica 5.4 se muestra la latencia media de los precintos respondidos por los servidores secundarios a los clientes. En ella se observa como a medida que se incrementa el tamaño de la caché de los clientes, aumenta esta latencia.

En la Gráfica 5.5 se muestra el porcentaje medio de los precintos pedidos por los clientes a los servidores secundarios que se han encontrado con éxito en la caché de los servidores secundarios. En ella se observa que conforme aumenta el tamaño de la caché empleada por los clientes, el porcentaje de éxito disminuye, si bien a partir de un tamaño de caché del 10%, este porcentaje se mantiene más o menos estable.

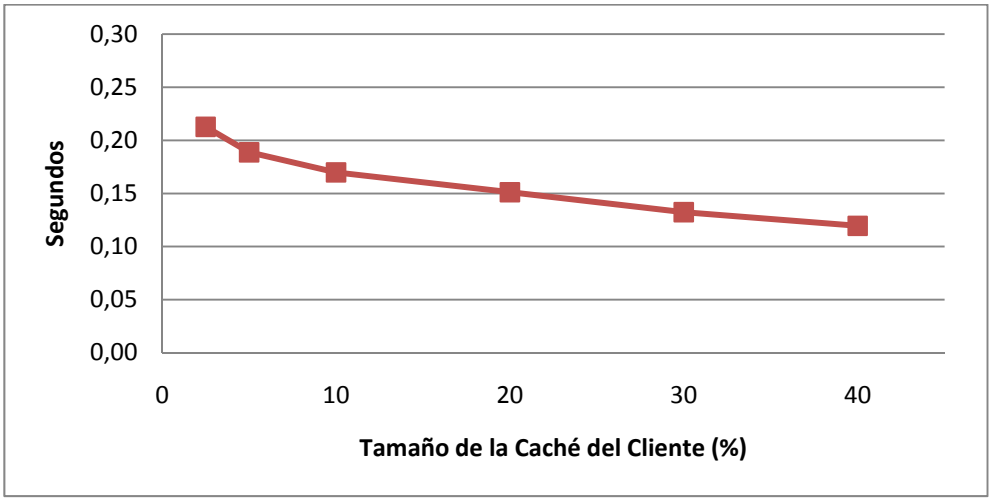
En la Gráfica 5.6 se muestra el número medio de precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por los servidores secundarios. En ella se observa que conforme se incrementa el tamaño de la caché empleada por los clientes disminuye el tamaño de la cola, con lo que disminuye su carga.

La reducción de la carga observada en la Gráfica 5.6 se explica con los resultados observados en el apartado anterior (5.4.1.1), donde se veía que el incremento del tamaño de la caché de los clientes provocaba que disminuyera el número de peticiones que se realizaban a los servidores secundarios.

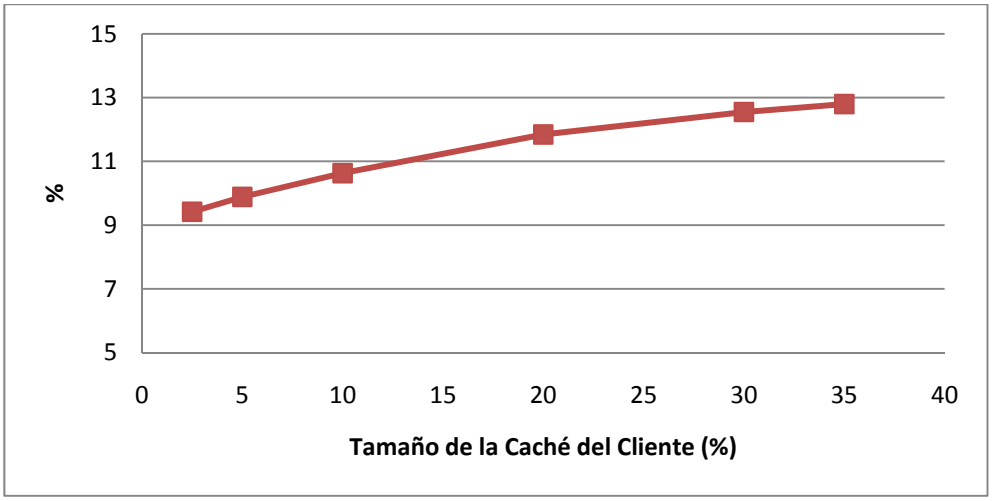
A pesar de la disminución de la carga de los servidores secundarios, en la Gráfica 5.4 se observa como a medida que se incrementa el tamaño de la caché de los clientes, aumenta el tiempo requerido por los servidores secundarios para devolver la información al cliente. Esto es debido a nuestro entender a que hay un mayor intercambio de precintos entre los clientes, y en consecuencia, un menor intercambio entre clientes y servidores secundarios, lo que provocará que los precintos que éstos van a pedir a los servidores secundarios sean más “singulares”, de forma que la probabilidad de que éstos se encuentren en su caché será menor, tal como se observa en la Gráfica 5.5. En consecuencia, los servidores secundarios tendrán que acceder al servidor principal con una mayor frecuencia, lo que provocará un incremento en la latencia media de los precintos de información respondidos a los clientes por los servidores secundarios.

Hay que hacer notar que los valores de latencia media de los precintos enviados por los servidores secundarios a los clientes son superiores a los que se podría esperar, porque en caso de que se pierda un mensaje de petición o de respuesta de precintos, como se comentó en el apartado 5.3.4, para calcular la latencia se considera el tiempo desde el que el cliente realizó la primera petición hasta que finalmente obtiene la respuesta de la misma, lo que va a aumentar el valor de esa media. De esta forma se puede observar cómo afecta al sistema la necesidad de volver a pedir los precintos cuando éstos se hayan perdido o no lleguen a su destino dentro del tiempo permitido.

5.4.1.3 – Sistema



Gráfica 5.7 – Latencia media global del intercambio de precintos en el Sistema



Gráfica 5.8 – Porcentaje medio de los precintos vueltos a pedir

En las gráficas anteriores se muestra la influencia que tiene en el sistema en su conjunto el tamaño de la caché que emplean los clientes.

En la Gráfica 5.7 se muestra la latencia media global del intercambio de los precintos en el sistema. En ella se puede observar como la latencia media global del sistema disminuye conforme el tamaño de la caché de los clientes se incrementa.

En la Gráfica 5.8 se muestra el número medio de precintos que los clientes han necesitado volver a pedir bien porque se han perdido, o bien porque han tardado más tiempo del permitido en obtenerse. En la gráfica se observa como ese número se incrementa conforme aumenta el tamaño de la caché de los clientes.

La disminución de la latencia media global del sistema pensamos que se debe a que al aumentar la caché de los clientes, se favorece una distribución más equilibrada de la carga entre todos los elementos del sistema. A pesar de que la latencia media en el intercambio de precintos por parte de los clientes crece ligeramente y a que la de los servidores secundarios también se incrementa, como aumenta el número de precintos servidos por los clientes (cuya latencia es menor que la de los servidores secundarios) y disminuye el de los servidores secundarios, esto hace que se produzca un decremento de la latencia global.

El hecho de que haya un incremento del número de precintos que se vuelven a pedir conforme aumenta el tamaño de la caché, pensamos que se debe, a priori, a que ello conlleva un incremento en el número de mensajes intercambiados entre los clientes que afecta a las colisiones que tienen lugar en las líneas de conexión. Este aumento del número de colisiones se produce porque todos los elementos del sistema se encuentran conectados en una misma red local. Si estos elementos se encontraran distribuidos a través de la red, a priori, no debería producirse este problema de aumento del número de colisiones.

5.4.1.4 – Conclusiones de la prueba

Como conclusión de esta prueba se puede decir que conforme mayor sea el tamaño de la caché que se emplee en los clientes, menor será la latencia global que obtendremos. Sin embargo, habrá que llegar a un compromiso entre almacenamiento y prestaciones, ya que las bases de datos de terrenos suelen tener un gran tamaño (varios terabytes) y por lo tanto el cliente no va a poder almacenarlas de forma completa.

Para el resto de las pruebas que se van a realizar para la evaluación del sistema, en vista de los resultados, pensamos que fijar el tamaño de la caché alrededor del 10% del tamaño de la base de datos original podría ser adecuado, ya que no supone un tamaño excesivamente elevado de información a almacenar en cada cliente y proporciona unas prestaciones aceptables.

5.4.2 – Variación en la distribución de la ruta.

Como se ha comentado en el apartado 2.1.3, el comportamiento de los usuarios que se conectan a un sistema de visualización de terrenos puede ser muy variado. Este comportamiento, a priori, va a determinar la información que cada cliente necesitará descargar y visualizar, y por lo tanto, determinará la información que intercambiará con sus clientes vecinos, lo que también podría afectar a la necesidad de los clientes de acceder o no a los servidores secundarios.

El objetivo de esta prueba es conocer cómo afectan los diferentes comportamientos de los usuarios al funcionamiento general del sistema. Para ello se van a emplear un conjunto de diferentes tipos de distribuciones de los puntos de las rutas que seguirán los clientes, que simulan los diferentes comportamientos de los usuarios por la escena.

Se han llevado a cabo varias pruebas cuyos parámetros de configuración son los siguientes:

Número de clientes conectados = 35 (el número máximo de clientes disponibles).

Número de servidores secundarios conectados = 2 (al menos son necesarios dos para que haya una distribución de la carga del sistema entre ellos).

Tamaño de la caché del servidor secundario = 50% (se reparte la base de datos entre los servidores secundarios).

Tamaño de la caché del cliente = 10% (consideramos este valor como aceptable dado el tamaño de la base de datos utilizada en la prueba).

Tamaño de la lista de clientes vecinos = 10 (consideramos este valor como aceptable dado el número máximo de clientes utilizados en la prueba).

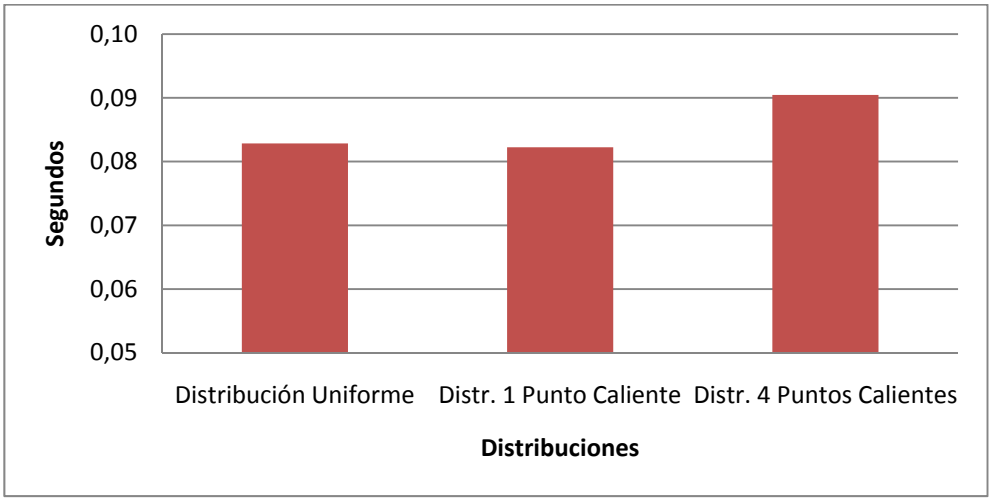
Las pruebas realizadas han sido:

Prueba_A: Ruta: Distribución uniforme.

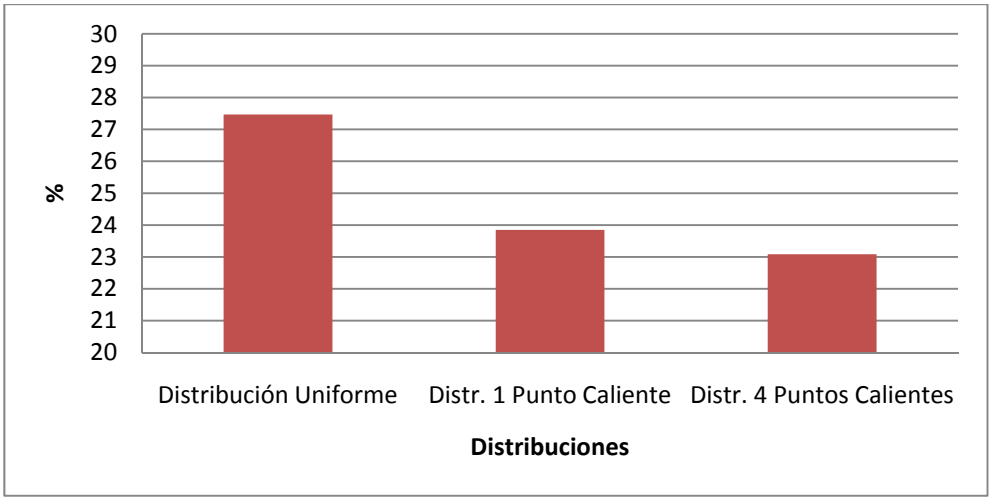
Prueba_B: Ruta: Distribución normal con un Punto Caliente (dispersión 25000).

Prueba_C: Ruta: Distribución normal con cuatro Puntos Calientes (dispersión 20000).

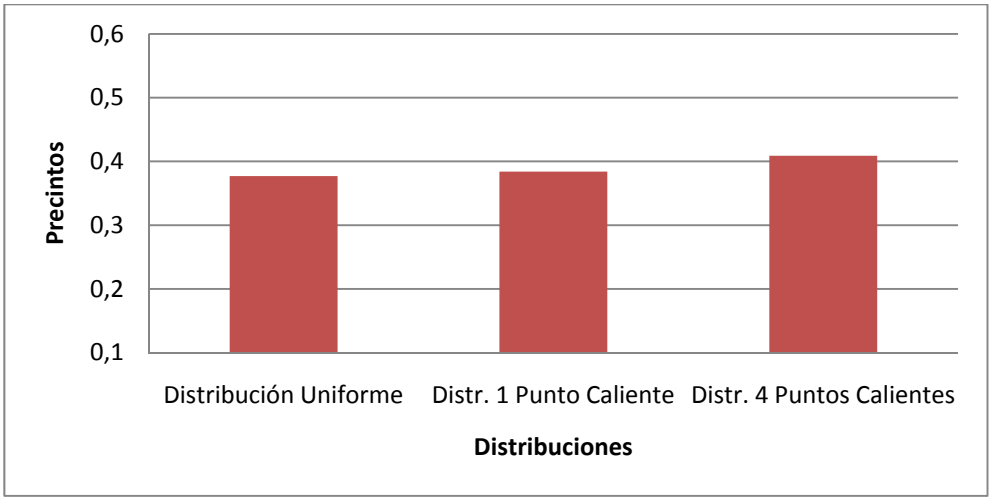
5.4.2.1 – Clientes



Gráfica 5.9 – Latencia media de los precintos intercambiados entre los Clientes



Gráfica 5.10 – Porcentaje medio de precintos respondidos por los Servidores Secundarios



Gráfica 5.11 – Número de precintos medio en las colas de los Clientes

En las gráficas anteriores se muestra la influencia que tiene en los clientes los diferentes tipos de distribuciones que simulan el comportamiento de los usuarios.

En la Gráfica 5.9 se muestra la latencia media de los precintos respondidos por los clientes. En ella se observa como el uso de diferentes distribuciones no afecta demasiado a la latencia, si bien ésta es ligeramente superior para la distribución normal en la que se emplean cuatro puntos calientes.

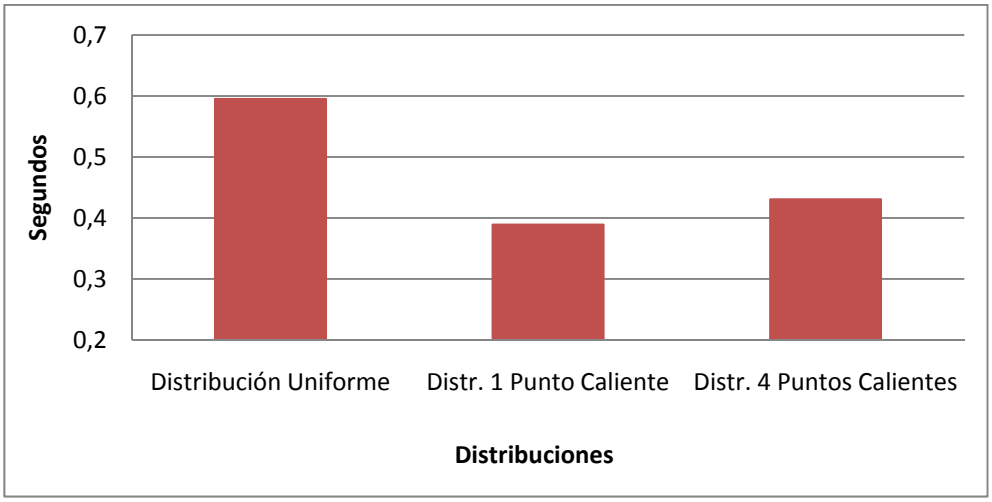
En la Gráfica 5.10 se muestra el porcentaje medio de peticiones que realizan los clientes a los servidores secundarios. En ella se observa como este porcentaje es menor cuando se emplean distribuciones normales, tanto de uno como de cuatro puntos calientes.

En la Gráfica 5.11 se muestra el número medio de precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por los clientes vecinos. En ella se observa como el tamaño de la cola es similar para todas las distribuciones, pudiendo destacar que ese tamaño es ligeramente superior para la distribución normal con cuatro puntos calientes, por lo que la carga para ese caso será algo mayor.

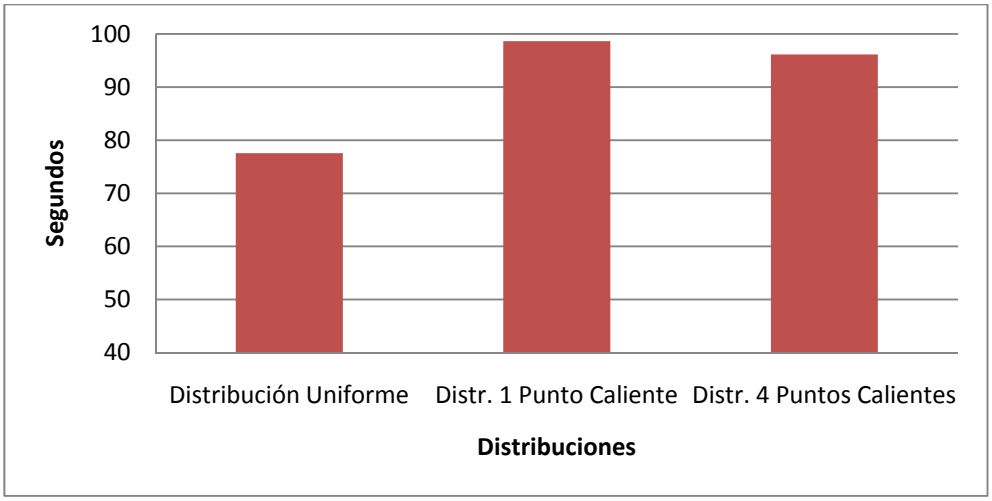
El hecho de que el porcentaje de peticiones que se realizan a los servidores secundarios sea menor cuando se emplean distribuciones normales (Gráfica 5.10) es debido a nuestro entender, a que como en esos casos los clientes se desplazan alrededor de las mismas regiones del terreno y no alrededor de toda la superficie del terreno, es más probable que éstos contengan en su caché los precintos que van a ser necesitados por otros clientes vecinos.

A su vez, cuando se emplean distribuciones normales, esta reducción del número de peticiones a los servidores secundarios va a suponer un aumento del número de peticiones entre los clientes vecinos, aumentando ligeramente su carga, como se observa en la Gráfica 5.11. Esto conlleva que la latencia para intercambiar los precintos de información entre los clientes sea ligeramente superior cuando se emplean distribuciones normales frente a la distribución uniforme (Gráfica 5.9).

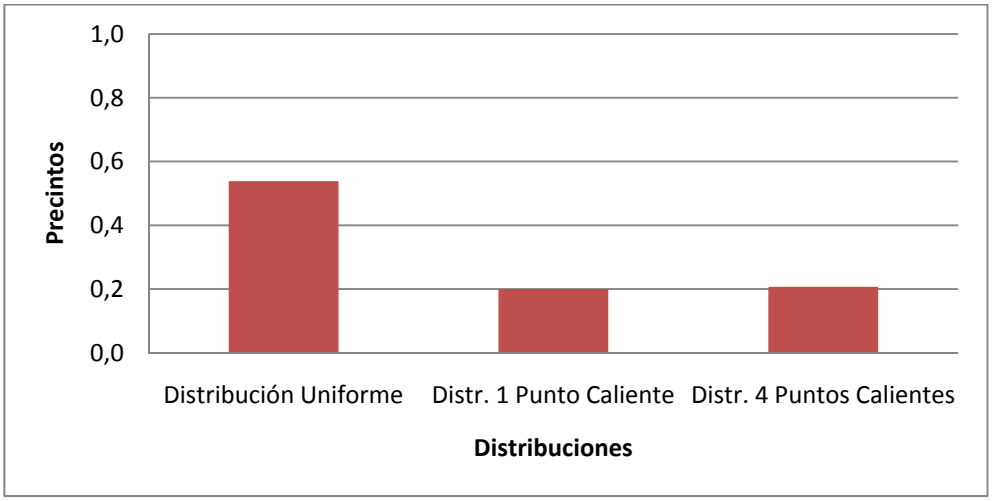
5.4.2.2 – Servidores Secundarios



Gráfica 5.12 – Latencia media de los precintos respondidos por los Servidores Secundarios



Gráfica 5.13 – Porcentaje medio de los precintos encontrados con éxito en la caché de los Servidores Secundarios



Gráfica 5.14 – Número de precintos medio en las colas de los Servidores Secundarios

En las gráficas anteriores se muestra la influencia que tiene en los servidores secundarios los diferentes tipos de distribuciones que simulan el comportamiento de los usuarios.

En la Gráfica 5.12 se muestra la latencia media de los precintos respondidos por los servidores secundarios a los clientes. En ella se observa como esta latencia es inferior cuando se emplea una distribución normal con puntos calientes que cuando se emplea una distribución uniforme.

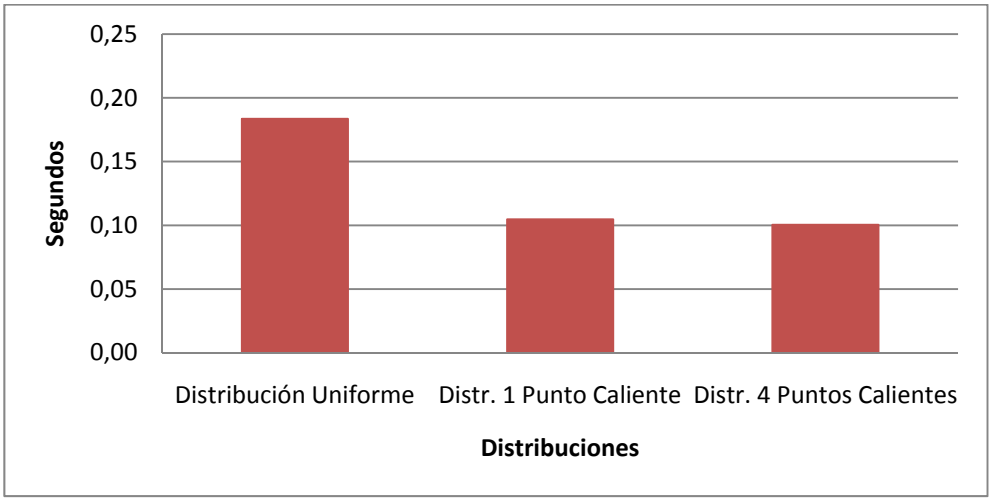
En la Gráfica 5.13 se muestra el porcentaje medio de los precintos pedidos por los clientes a los servidores secundarios que se han encontrado con éxito en la caché de los servidores secundarios. En ella se observa que cuando se emplea una distribución con puntos calientes, el porcentaje de precintos que se encuentran en la caché es muy alto, cercano al 100%. En cambio, cuando se emplea una distribución uniforme ese porcentaje es alrededor del 80%.

En la Gráfica 5.14 se muestra el número medio de precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por los servidores secundarios. En ella se observa que el tamaño de la cola es menor cuando se emplea una distribución normal con puntos calientes que cuando se emplea una distribución uniforme, lo que se traduce en una carga menor.

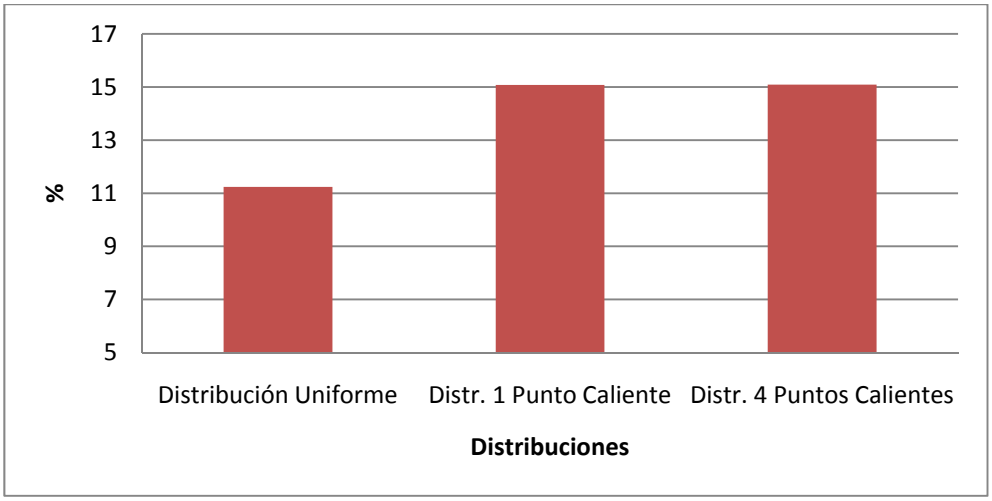
El elevado porcentaje de éxito de encontrar los precintos pedidos por los clientes en las cachés de los servidores secundarios cuando se emplean distribuciones normales con puntos calientes (Gráfica 5.13), pensamos que se debe a que las peticiones de precintos de los clientes se concentran sobre las mismas regiones del terreno, por lo que es más probable que los precintos ya se hayan pedido previamente a los servidores secundarios y estén almacenados en su caché. Esto supone que los servidores secundarios tienen que acceder en menos ocasiones al servidor principal, lo que permite que puedan responder antes a los clientes. Este hecho provocará que la latencia de respuesta de los precintos a los clientes sea inferior cuando se empleen distribuciones normales con puntos calientes, tal como se observa en la Gráfica 5.12.

La reducción de la carga observada en la Gráfica 5.14 cuando se emplea una distribución con puntos calientes frente al uso de una distribución uniforme, se explica con los resultados obtenidos en el apartado anterior (5.4.2.1), donde se veía que en ese caso el número de peticiones que se realizan a los servidores secundarios era inferior dado que el número de peticiones resuelto por los clientes vecinos era mayor. Además, esta reducción de la carga en ese caso también está relacionada con el descenso de la latencia de respuesta de precintos de los servidores secundarios que se observa en la Gráfica 5.12.

5.4.2.3 – Sistema



Gráfica 5.15 – Latencia media global del intercambio de precintos en el Sistema



Gráfica 5.16 – Porcentaje medio de los precintos vueltos a pedir

En las gráficas anteriores se muestra la influencia que tienen en el sistema en su conjunto los diferentes tipos de distribuciones que simulan el comportamiento de los usuarios.

En la Gráfica 5.15 se muestra la latencia media global del intercambio de los precintos en el sistema. En ella se puede observar como la latencia global es menor cuando se emplea una distribución con puntos calientes respecto al uso de una distribución uniforme.

En la Gráfica 5.16 se muestra el número medio de precintos que los clientes han necesitado volver a pedir bien porque se han perdido, o bien porque han tardado más tiempo del permitido en obtenerse. En esa gráfica se observa como ese número es mayor cuando se emplea una distribución con puntos calientes respecto al uso de una distribución uniforme.

La menor latencia global obtenida para las distribuciones normales con uno o cuatro puntos calientes pensamos que se debe a que aumenta el número de peticiones que se resuelven entre los clientes (que tienen una latencia menor que las resueltas por los servidores secundarios) y además porque para estos casos las peticiones que se realizan a los servidores secundarios tienen una menor latencia de respuesta, debido a que la mayor parte de precintos pedidos ya se encuentran disponibles en sus cachés y no es necesario acudir al servidor principal.

El hecho de que haya un mayor número de precintos que se vuelven a pedir cuando se emplea una distribución normal con puntos calientes pensamos que se debe, a priori, a un mayor número de mensajes que se intercambian entre los clientes que afecta a las colisiones en las líneas de conexión. Este aumento del número de colisiones se produce porque todos los elementos del sistema se encuentran conectados en una misma red local. Si estos elementos se encontraran distribuidos a través de la red, a priori, no debería producirse este problema de aumento del número de colisiones.

5.4.2.4 – Conclusiones de la prueba

Como conclusión de esta prueba podemos decir que el sistema de visualización funciona adecuadamente con diferentes supuestos de comportamiento de los usuarios, y que la especialización por regiones del terreno llevada a cabo por los servidores secundarios funciona adecuadamente, obteniendo unos mejores resultados cuando se emplean varias zonas de interés que concentran a los usuarios, ya que en este caso la especialización por regiones es más evidente.

Debido a que si se emplea una distribución con puntos calientes, el porcentaje de éxito de que se encuentren los precintos en la caché de los servidores secundarios es casi del 100%, limitando la posibilidad de mejora de ese porcentaje, y debido también a que la superficie del terreno empleada en las pruebas, a priori, no tiene ningún punto de interés que sobresalga del resto, se ha decidido emplear la distribución uniforme para el resto de pruebas que se van a realizar para la evaluación del sistema.

5.4.3 – Variación en el número de clientes conectados

Otro elemento importante a estudiar es cómo afecta a las prestaciones del sistema el número de clientes conectados al mismo, lo que nos permitirá comprobar la escalabilidad del sistema. La variación del número de clientes conectados puede afectar a todo el sistema, puesto que influirá en la cantidad de información disponible entre ellos, lo que podría afectar a la cantidad de información transferida en la red, en la cantidad de conexiones que atienden los servidores, etc.

El objetivo de esta prueba es conocer la capacidad de escalabilidad de nuestro sistema en función del número de clientes conectados al mismo.

Se han llevado a cabo para ello varias pruebas cuyos parámetros de configuración son los siguientes:

Número de servidores secundarios conectados = 2 (al menos son necesarios dos para que haya una distribución de la carga del sistema entre ellos).

Ruta: Distribución uniforme (toda la superficie del terreno tiene el mismo nivel de interés).

Tamaño de la caché del servidor secundario = 50% (se reparte la base de datos entre los dos servidores secundarios).

Tamaño de la caché del cliente = 10% (consideramos este valor como aceptable dato el tamaño de la base de datos utilizada en la prueba).

Tamaño de la lista de clientes vecinos = 10 (consideramos este valor como aceptable dado el número máximo de clientes utilizados en la prueba).

Las pruebas realizadas han sido:

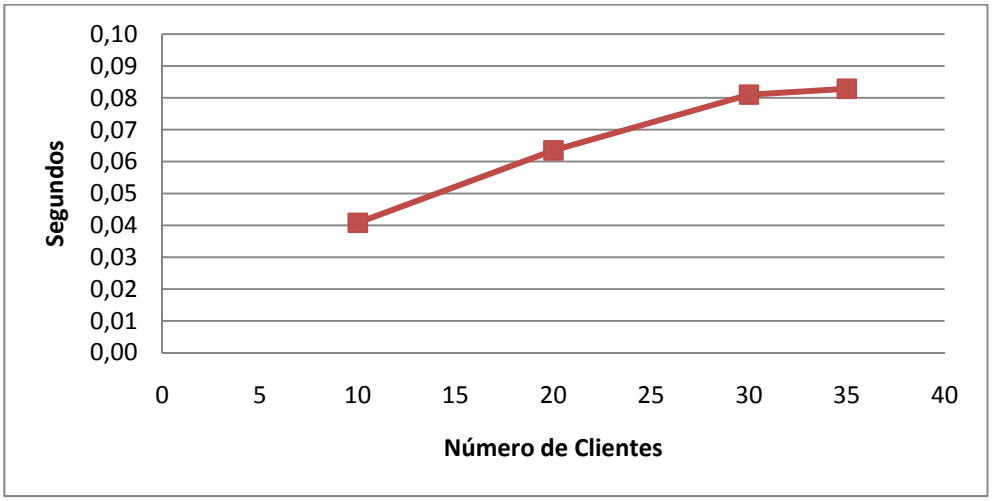
Prueba_A: Número de clientes conectados = 10.

Prueba_B: Número de clientes conectados = 20.

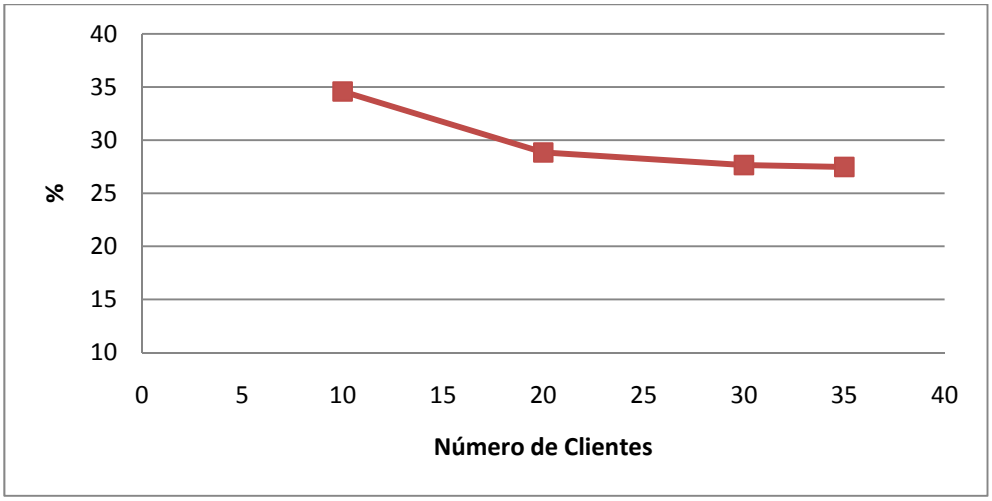
Prueba_C: Número de clientes conectados = 30.

Prueba_D: Número de clientes conectados = 35.

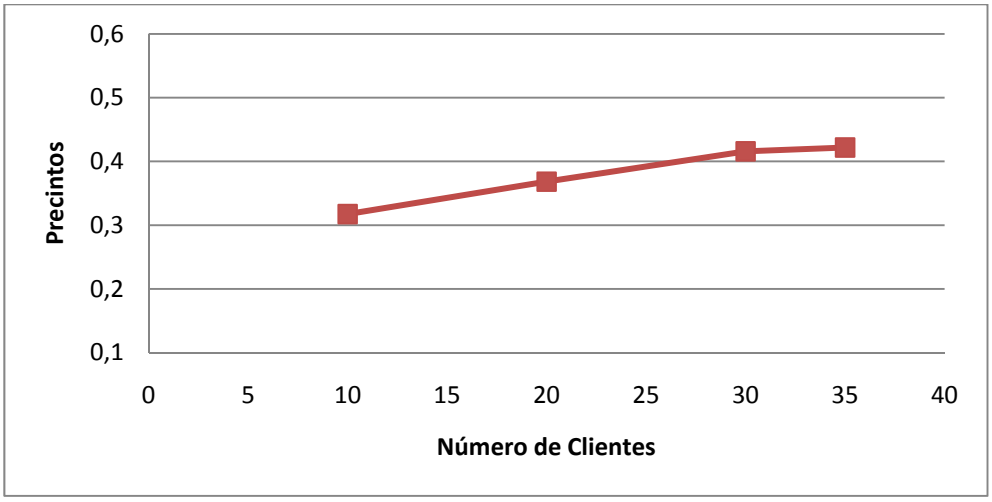
5.4.3.1 – Clientes



Gráfica 5.17 – Latencia media de los precintos intercambiados entre los Clientes



Gráfica 5.18 – Porcentaje medio de precintos respondidos por los Servidores Secundarios



Gráfica 5.19 – Número de precintos medio en las colas de los Clientes

En las gráficas anteriores se muestra la influencia que tiene en los propios clientes la cantidad de clientes conectados al sistema.

En la Gráfica 5.17 se muestra la latencia media de los precintos respondidos por los clientes. En ella se observa como el incremento del número de los clientes conectados al sistema provoca un aumento de esa latencia. Pero a partir de 30 clientes parece ser que la latencia se estabiliza.

En la Gráfica 5.18 se muestra el porcentaje medio de peticiones que realizan los clientes a los servidores secundarios. En ella se observa como este porcentaje disminuye conforme aumenta el número de clientes conectado al sistema, si bien el porcentaje parece estabilizarse a partir de 30 clientes.

En la Gráfica 5.19 se muestra el número medio de precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por los clientes vecinos. En ella se observa como el incremento del número de clientes conectados al sistema provoca un aumento del tamaño de esta cola, por lo que aumenta su carga, sin embargo parece que el tamaño de la cola se estabiliza a partir de 30 clientes.

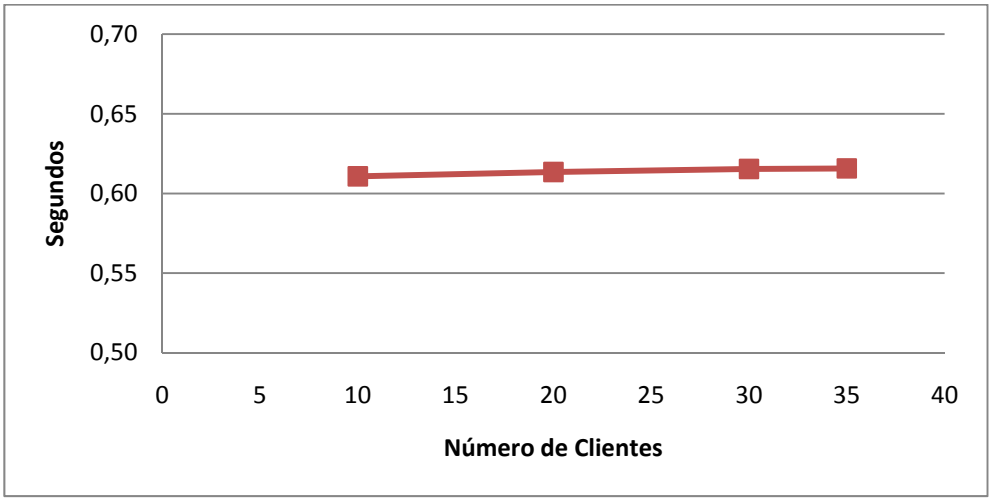
Conforme aumenta el número de clientes conectados al sistema, la información almacenada en la base de datos se distribuirá en mayor medida entre los clientes. De esta forma los clientes podrán obtener una mayor cantidad de información de sus clientes vecinos, evitando tener que acudir a los servidores secundarios. Este hecho provoca que el porcentaje de peticiones realizadas a los servidores secundarios por parte de los clientes

disminuya con un mayor número de clientes conectados al sistema, tal como se muestra en la Gráfica 5.18.

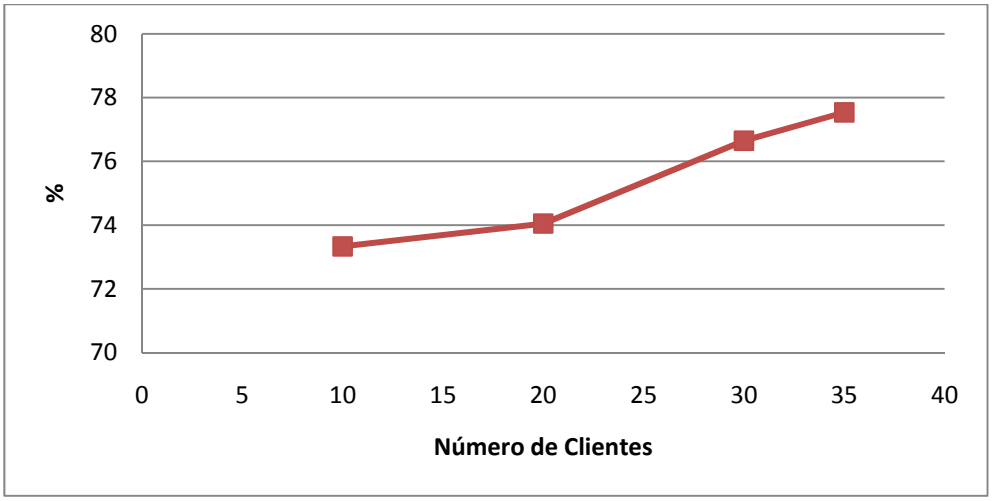
A su vez, como consecuencia de este aumento del porcentaje de peticiones resueltas entre los clientes a medida que se incrementa el número de ellos conectados, aumenta la carga de los mismos, tal como se muestra en la Gráfica 5.19. Además, este aumento en la carga va a suponer a su vez un aumento de la latencia de respuesta de precintos de los clientes (Gráfica 5.17).

En las gráficas anteriores se observa que a partir de un número de 30 clientes conectados al sistema, los valores mostrados en ellas parece que se estabilizan. Este comportamiento es el que suele darse en una arquitectura P2P, donde la carga del sistema se reparte entre los clientes conectados manteniéndose estable. Si bien, en las arquitecturas P2P, hace falta un número suficiente de clientes para que esta carga sea estable. Esto es lo que pensamos que ocurre en las gráficas anteriores; como empleamos una arquitectura mixta cliente-servidor / P2P, cuando no hay conectados un número suficiente de clientes en el sistema, éste se comporta como una arquitectura cliente-servidor, donde la carga del sistema recaerá en los servidores secundarios. Conforme aumenta el número de clientes conectados, parte de la carga soportada con los servidores secundarios se transfiere a los clientes, provocando que su carga y su latencia aumenten, tal como se muestra en la Gráfica 5.17 y en la Gráfica 5.19 desde 10 a 30 clientes conectados. A partir de 30 clientes el sistema pasa a comportarse como una arquitectura P2P, debido a que parece que existe un número de clientes conectados suficiente para realizar una distribución adecuada de la carga entre todos ellos, por lo que la carga de los clientes y su latencia parece que se estabilizan.

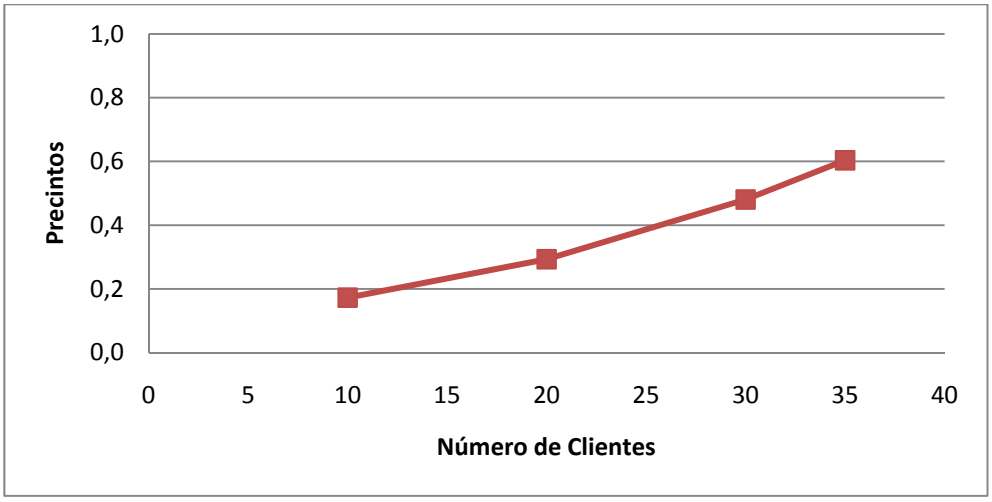
5.4.3.2 – Servidores Secundarios



Gráfica 5.20 – Latencia media de los precintos respondidos por los Servidores Secundarios



Gráfica 5.21 – Porcentaje medio de los precintos encontrados con éxito en la caché de los Servidores Secundarios



Gráfica 5.22 – Número de precintos medio en las colas de los Servidores Secundarios

En las gráficas anteriores se muestra la influencia que tiene en los servidores secundarios el aumento del número de clientes conectados al sistema.

En la Gráfica 5.20 se muestra la latencia media de los precintos respondidos por los servidores secundarios a los clientes. En ella se observa como esta latencia se mantiene más o menos en los mismos niveles, incrementándose ligeramente con el número de clientes conectados al sistema.

En la Gráfica 5.21 se muestra el porcentaje medio de los precintos pedidos por los clientes a los servidores secundarios que se han encontrado con éxito en la caché de los servidores secundarios. En ella se observa como el porcentaje de éxito aumenta conforme se incrementa el número de clientes conectados al sistema.

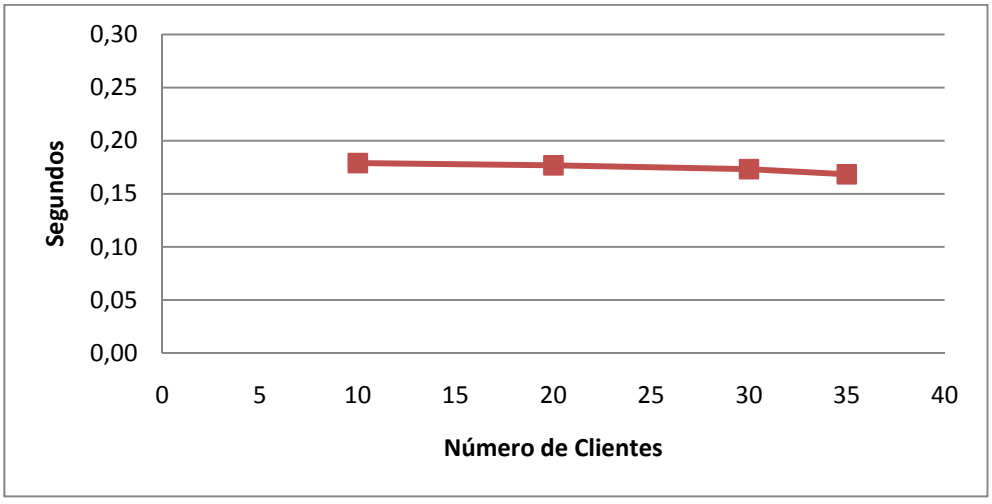
En la Gráfica 5.22 se muestra el número medio de precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por los servidores secundarios. En ella se observa que conforme el número de clientes conectado al sistema aumenta, el tamaño de esta cola aumenta, con lo que aumenta su carga.

Como se observó en la Gráfica 5.18 del apartado anterior (5.4.3.1), el porcentaje de peticiones que realizan los clientes a los servidores secundarios no decrece linealmente con el número de clientes conectados al sistema, sino que parece que tiende a estabilizarse alrededor del 27%. Por lo tanto, a medida que aumente el número de clientes conectados de forma lineal, aumentará la carga de los servidores secundarios, tal como puede observarse en la Gráfica 5.22.

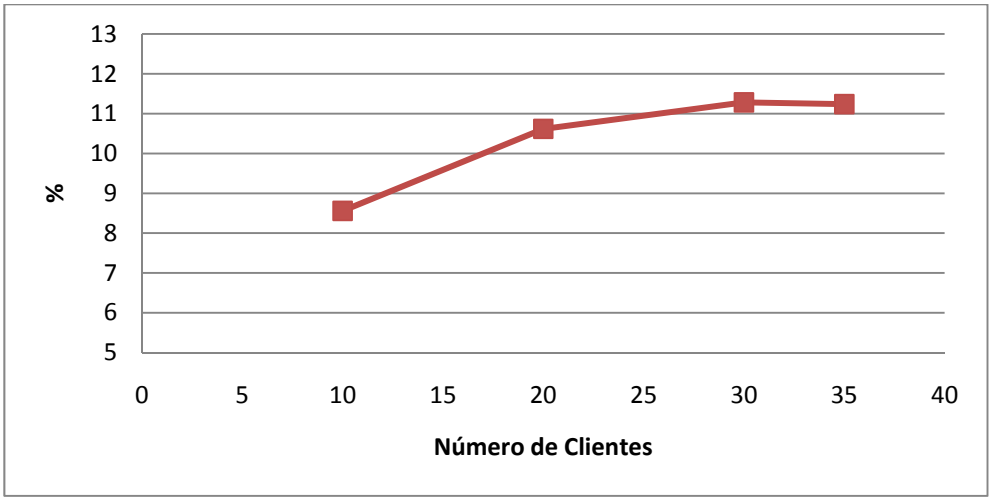
El objetivo del servidor de comunicaciones en el sistema es el de repartir la carga global del sistema entre los clientes y los servidores secundarios tanto de forma espacial como de forma computacional. Para el caso de los servidores secundarios, el aumento en el número de clientes conectados al sistema permite al servidor de comunicaciones realizar una mejor distribución de la carga de los servidores secundarios siguiendo estos dos criterios, ya que podrá asignar más fácilmente a los clientes el servidor secundario más especializado en la región del terreno en la que se encuentren, manteniendo a su vez la carga computacional de los servidores secundarios estable. Este hecho explica, a nuestro entender, el aumento en el porcentaje de éxito de precintos pedidos por los clientes que se encuentran en la caché de los servidores secundarios conforme hay un mayor número de clientes conectados al sistema, tal como se muestra en la Gráfica 5.21.

Por lo tanto, a pesar del aumento de la carga de los servidores secundarios, el incremento del porcentaje de precintos pedidos que se encuentran en la caché de los servidores secundarios (que evita tener que acceder al servidor principal y por lo tanto permite responder los precintos de manera más rápida a los clientes), compensa ese aumento de carga y permite mantener una latencia de respuesta de precintos por parte de los servidores secundarios más o menos estable ante el incremento del número de clientes conectados al sistema.

5.4.3.3 – Sistema



Gráfica 5.23 – Latencia media global del intercambio de precintos en el Sistema



Gráfica 5.24 – Porcentaje medio de los precintos vueltos a pedir

En las gráficas anteriores se muestra la influencia que tiene en el sistema en su conjunto el aumento del número de clientes conectados al mismo.

En la Gráfica 5.23 se muestra la latencia media global del intercambio de los precintos en el sistema. En ella se puede observar como esta latencia parece que se mantiene estable a pesar del aumento del número de clientes conectados al sistema.

En la Gráfica 5.24 se muestra el número medio de precintos que los clientes han necesitado volver a pedir bien porque se han perdido, o bien porque han tardado más tiempo del permitido en obtenerse. En esa gráfica se observa como ese número se incrementa conforme aumenta el número de clientes conectados al sistema, si bien a partir de 20 clientes ese incremento se suaviza.

La estabilidad de la latencia global de intercambio de precintos pensamos que se debe a que, a pesar de que la latencia de respuesta de los precintos por parte de los clientes aumenta, la de los servidores secundarios se mantiene estable, y como aumenta el número de precintos servidos por los clientes (cuya latencia de respuesta es menor que la de los servidores secundarios) y disminuye el de los servidores secundarios, esto hace que se la latencia global se mantenga más o menos estable.

El incremento del número de precintos que se vuelven a pedir que se produce cuando aumenta el número de clientes, pensamos que se debe, a priori, al mayor número de mensajes intercambiados entre los clientes que afectará al número de colisiones en las líneas de conexión, provocando un mayor número de precintos perdidos. Este aumento del número de colisiones se produce porque todos los elementos del sistema se encuentran conectados en una misma red local. Si estos elementos se encontraran distribuidos a través de la red, a priori, no debería producirse este problema de aumento del número de colisiones.

5.4.3.4 – Conclusiones de la prueba

Como conclusión de esta prueba se puede decir que el sistema parece ser escalable con el número de clientes conectados al mismo. Como se ha podido observar en los resultados, el sistema se comporta de la forma en la que fue diseñado, es decir, ante un número bajo de clientes conectados al mismo, funciona como una arquitectura cliente-servidor, mientras que con un número suficiente de clientes conectados, el sistema se comporta como una arquitectura P2P. En ambas situaciones el sistema es capaz de ofrecer una calidad se servicio adecuada.

Otra conclusión que se puede extraer de los resultados es la mejora en el funcionamiento del servidor de comunicaciones conforme aumenta el número de clientes conectados, ya que dispone de una mayor versatilidad a la hora de asignar a los clientes el servidor secundario más adecuado consiguiendo servidores secundarios más especializados y manteniendo a su vez la carga de los mismos estable.

Debido a que el sistema ha sido diseñado para soportar una gran carga de trabajo, para el resto de las pruebas que se van a realizar para la evaluación del mismo, se empleará el mayor número de clientes disponible, que va a ser de 35, para proporcionar la mayor carga posible al sistema de forma que se puedan evaluar mejor sus funcionalidades.

5.4.4 – Variación en el tamaño de la caché de los servidores secundarios

Al igual que con los clientes, se realizarán pruebas para observar cómo influye el tamaño de caché de los servidores secundarios en el funcionamiento del sistema. Como se quiere realizar una especialización de los servidores secundarios por regiones del terreno, no debería ser necesario que cada servidor tuviera una copia completa de la base de datos general. Además, generalmente habrá regiones del terreno que, por carecer de interés para los usuarios no se suelan necesitar nunca. Por ello, parece tener sentido diseñar la arquitectura del sistema limitando la cantidad de información que se pueda almacenar en cada servidor secundario. La variación del tamaño de esa caché afectará previsiblemente a la necesidad del servidor secundario de acceder al servidor principal para obtener la información requerida por los clientes que no se encuentra en su caché, lo que aumentaría, en el caso de producirse, el tiempo de respuesta a los clientes.

El objetivo de esta prueba es conocer cómo afecta el tamaño de la caché de los servidores secundarios al funcionamiento general de sistema, centrándonos principalmente en cómo afecta a la proporción de accesos al servidor principal que realizan los servidores secundarios y a la latencia de respuesta de los precintos de los servidores secundarios a los clientes.

Se han llevado a cabo diversas pruebas cuyos parámetros de configuración son los siguientes:

Número de clientes conectados = 35 (el número máximo de clientes disponibles).

Número de servidores secundarios conectados = 2 (al menos son necesarios dos para que haya una distribución de la carga del sistema entre ellos).

Ruta: Distribución uniforme (toda la superficie del terreno tiene el mismo nivel de interés).

Tamaño de la caché del cliente = 10% (consideramos este valor como aceptable dado el tamaño de la base de datos utilizada en la prueba).

Tamaño de la lista de clientes vecinos = 10 (consideramos este valor como aceptable dado el número máximo de clientes utilizados en la prueba).

Las pruebas realizadas han sido:

Prueba_A: Tamaño de la caché del servidor secundario para las texturas = 10%

Tamaño de la caché del servidor secundario para las alturas = 10%

Prueba_B: Tamaño de la caché del servidor secundario para las texturas = 20%

Tamaño de la caché del servidor secundario para las alturas = 20%

Prueba_C: Tamaño de la caché del servidor secundario para las texturas = 30%

Tamaño de la caché del servidor secundario para las alturas = 30%

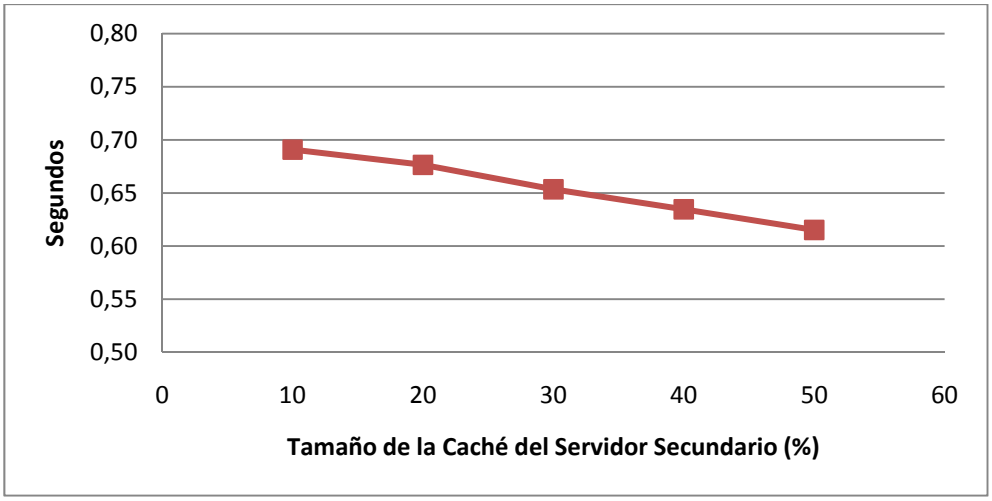
Prueba_D: Tamaño de la caché del servidor secundario para las texturas = 40%

Tamaño de la caché del servidor secundario para las alturas = 40%

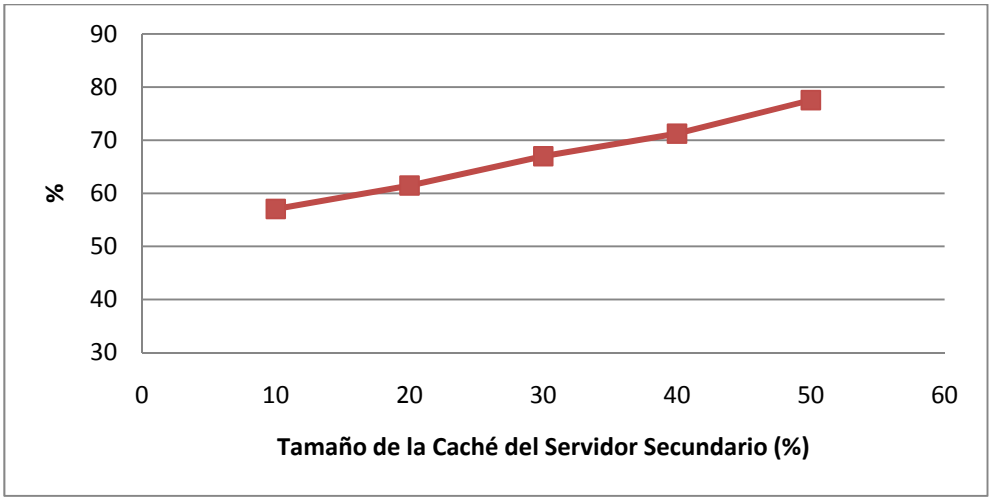
Prueba_E: Tamaño de la caché del servidor secundario para las texturas = 50%

Tamaño de la caché del servidor secundario para las alturas = 50%

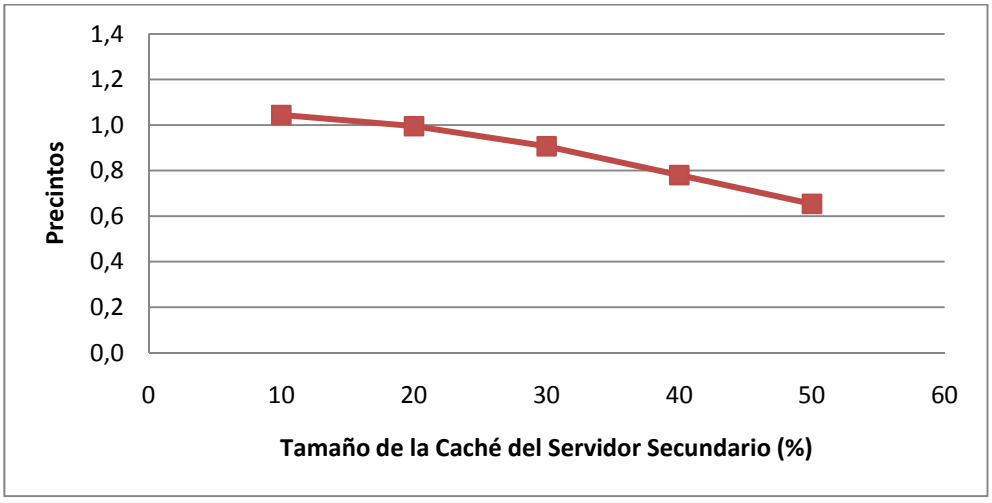
5.4.4.1 – Servidores Secundarios



Gráfica 5.25 – Latencia media de los precintos respondidos por los Servidores Secundarios



Gráfica 5.26 – Porcentaje medio de los precintos encontrados con éxito en la caché de los Servidores Secundarios



Gráfica 5.27 – Número de precintos medio en las colas de los Servidores Secundarios

En las gráficas anteriores se muestra la influencia que tiene en los servidores secundarios el tamaño de la caché que utilizan.

En la Gráfica 5.25 se muestra la latencia media de los precintos respondidos por los servidores secundarios a los clientes. En ella se observa como a medida que se incrementa el tamaño de la caché de los servidores secundarios, disminuye el valor de esta latencia.

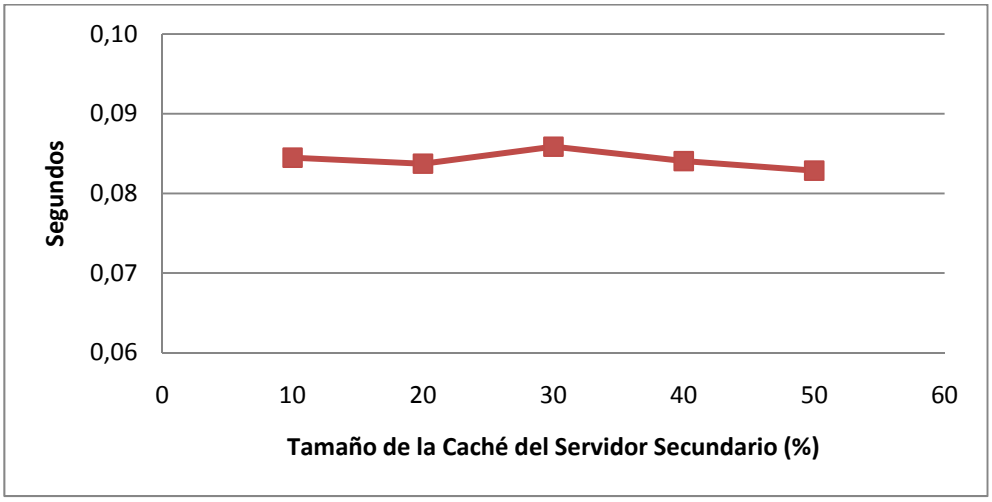
En la Gráfica 5.26 se muestra el porcentaje medio de los precintos pedidos por los clientes a los servidores secundarios que se han encontrado con éxito en la caché de los servidores secundarios. En ella se observa que conforme aumenta el tamaño de la caché empleada, el porcentaje de éxito aumenta.

En la Gráfica 5.27 se muestra el número medio de precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por los servidores secundarios. En ella se observa como el tamaño de la cola disminuye conforme aumenta el tamaño de la caché empleada por los servidores secundarios, con lo que disminuye su carga.

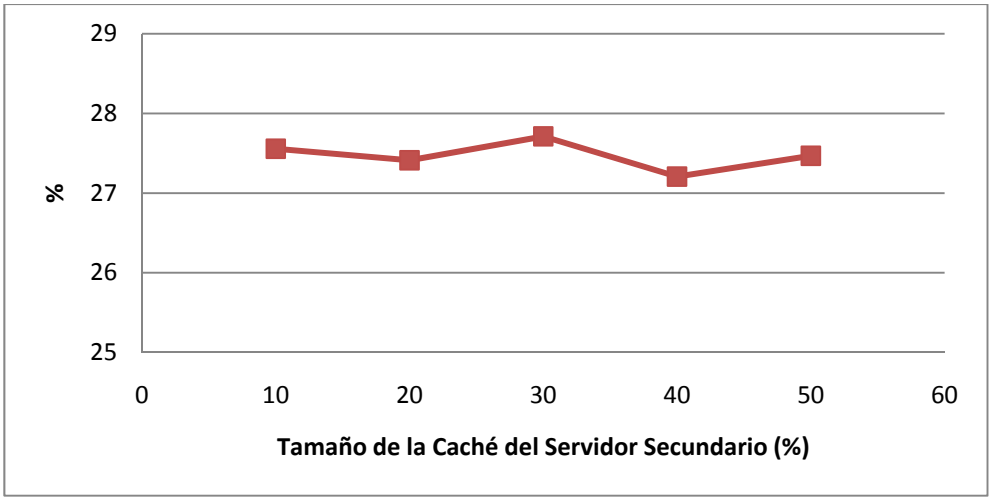
El incremento del tamaño de la caché de los servidores secundarios les permite almacenar una mayor cantidad de precintos. Esto provoca que sea más probable que los precintos pedidos por los clientes se encuentren en la caché, aumentándose el porcentaje de éxito, tal como se observa en la Gráfica 5.26.

Este aumento del número de precintos que se encuentran en la caché permite que los servidores secundarios respondan más rápido a los clientes, ya que no tendrán que acudir al servidor principal a por la información. Este hecho provoca un descenso de la latencia de respuesta de precintos a los clientes (Gráfica 5.25). A su vez, al responder más rápido los precintos a los clientes, los servidores secundarios podrán procesar un mayor número de peticiones en el mismo intervalo de tiempo, disminuyendo su carga, tal como se observa en la Gráfica 5.27.

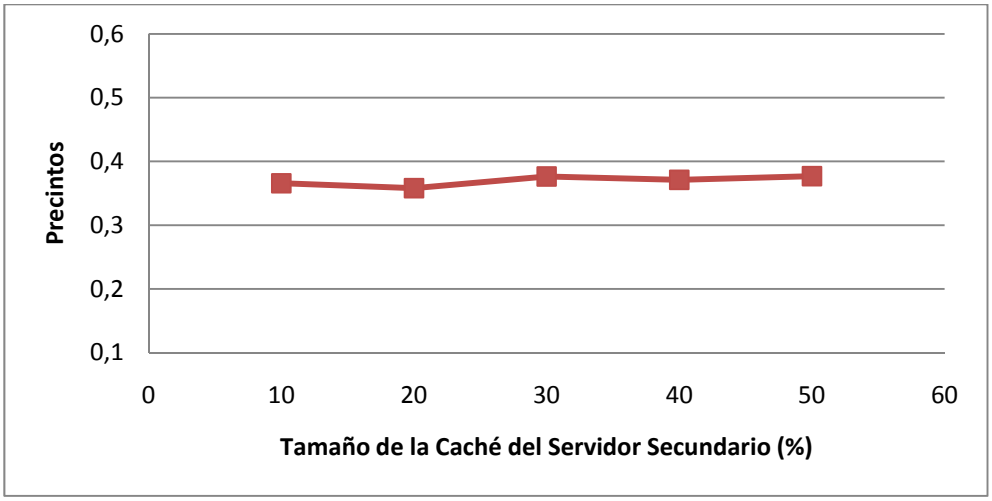
5.4.4.2 – Clientes



Gráfica 5.28 – Latencia media de los precintos intercambiados entre los Clientes



Gráfica 5.29 – Porcentaje medio de precintos respondidos por los Servidores Secundarios



Gráfica 5.30 – Número de precintos medio en las colas de los Clientes

En las gráficas anteriores se muestra la influencia que tiene en los clientes el tamaño de la caché que emplean los servidores secundarios.

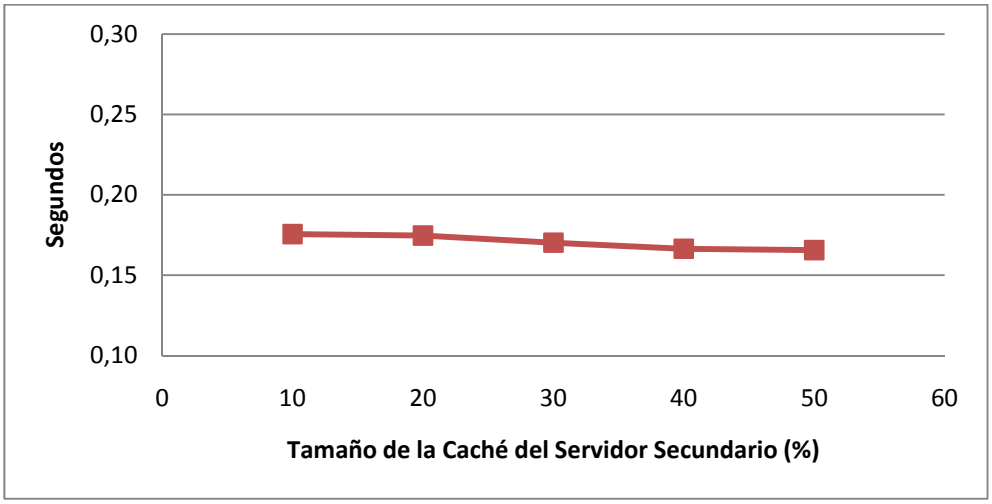
En la Gráfica 5.28 se muestra la latencia media de los precintos respondidos por los clientes. En ella se observa como esta latencia se mantiene estable para los diferentes tamaños de caché empleados en los servidores secundarios.

En la Gráfica 5.29 se muestra el porcentaje medio de peticiones que realizan los clientes a los servidores secundarios. En ella se observa como este porcentaje se mantiene estable respecto al tamaño de caché empleado en los servidores secundarios.

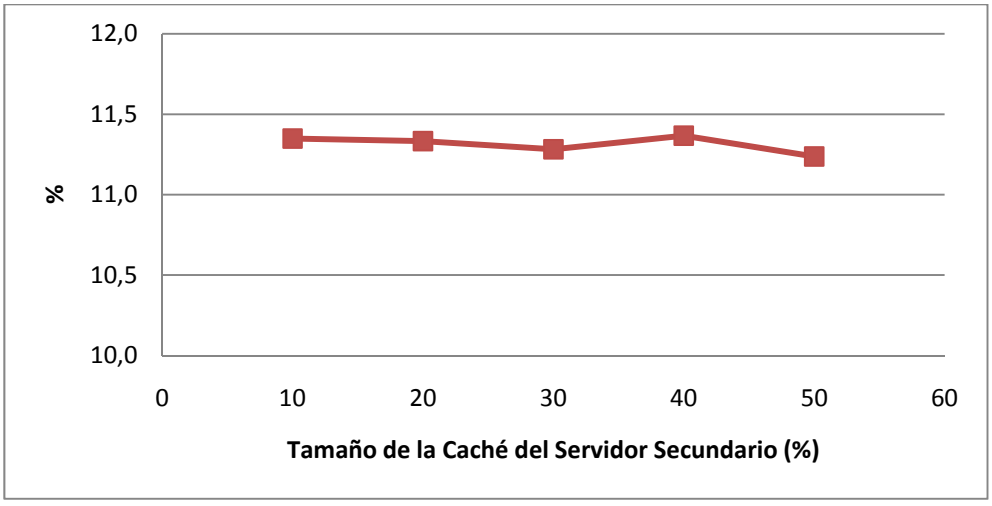
En la Gráfica 5.30 se muestra el número medio de precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por los clientes vecinos. En ella se observa como el tamaño de esta cola se mantiene estable respecto al uso de un diferente tamaño de caché por parte de los servidores secundarios.

Como se observa en las gráficas anteriores, el incremento del tamaño de la caché de los servidores secundarios no va a afectar al comportamiento de los clientes, ya que tanto la latencia de los precintos respondidos por los clientes, como las necesidades de los clientes de acceder a los servidores secundarios a por la información que requieren, y como la carga de los clientes, se mantienen más o menos estables. Esto pensamos que se debe a que el tamaño de la caché del servidor secundario es un parámetro que afecta únicamente al servidor secundario, por lo que no debe influir en el funcionamiento de los clientes, tal como se comprueba en los resultados anteriores.

5.4.4.3 – Sistema



Gráfica 5.31 – Latencia media global del intercambio de precintos en el Sistema



Gráfica 5.32 – Porcentaje medio de los precintos vueltos a pedir

En las gráficas anteriores se muestra la influencia que tiene en el sistema en su conjunto el tamaño de la caché que emplean los servidores secundarios.

En la Gráfica 5.31 se muestra la latencia media global del intercambio de los precintos en el sistema. En ella se puede observar como conforme el tamaño de la caché de los servidores

secundarios se incrementa, la latencia global del sistema disminuye ligeramente, aunque parece que a partir de una caché del 40% se estabiliza.

En la Gráfica 5.32 se muestra el número medio de precintos que los clientes han necesitado volver a pedir bien porque se han perdido, o bien porque han tardado más tiempo del permitido en obtenerse. En ella se observa como ese número de precintos se mantiene más o menos estable respecto al tamaño de la caché empleada por los servidores secundarios.

La disminución de la latencia media global del sistema conforme aumenta la caché de los servidores secundarios pensamos que se debe a que la latencia de respuesta de los precintos enviados por los servidores secundarios a los clientes es menor. Este hecho se produce porque los servidores secundarios van a tener que acudir en menor medida al servidor principal a por los precintos requeridos por los clientes.

El número de precintos que se pierden o no llegan a tiempo y se vuelven a pedir se mantiene más o menos estable, porque a nuestro entender, la cantidad de mensajes que se intercambian entre todos los elementos del sistema es similar, independientemente del tamaño de la caché empleada por los servidores secundarios, no influyendo en la cantidad de colisiones que se producen en las líneas de conexión.

5.4.4.4 – Conclusiones de la prueba

Como conclusión de este apartado podemos decir que, de igual forma que ocurría con el tamaño de la caché de los clientes, conforme mayor sea el tamaño de la caché que se emplee en los servidores secundarios, mejores resultados obtendremos. Sin embargo hay que plantearse si es necesario almacenar una réplica completa de la base de datos original en cada servidor secundario.

Nosotros consideramos que, ya que los servidores secundarios se van a especializar por regiones del terreno, que generalmente existirán zonas de la superficie del terreno almacenada en la base de datos que por carecer de interés para los usuarios no se van a necesitar nunca, y que como se muestra en los resultados, el incremento en el tamaño de la caché no aporta un descenso destacado del tiempo de latencia global, sobre todo a partir de un tamaño del 40%, que una caché del 50% del tamaño de la base de datos original para almacenar las superficies del terreno podría resultar suficiente y adecuada para un sistema de visualización de terrenos en tiempo real.

Por todo esto, para el resto de pruebas que se van a realizar para la evaluación del sistema, se ha decidido emplear un tamaño de caché para los servidores secundarios del 50% del tamaño de la base de datos original.

5.4.5 – Variación en el número de servidores secundarios conectados

Uno de los objetivos de nuestro sistema es distribuir la carga de las peticiones de los servidores secundarios tanto de forma espacial como de forma computacional. Para ello se necesitará emplear más de un servidor secundario para comprobar que esta distribución se lleva a cabo adecuadamente.

El objetivo de esta prueba es emplear un diferente número de servidores secundarios para comprobar si el mecanismo implementado para repartir la carga entre los servidores secundarios aporta un mayor rendimiento al sistema.

Se han llevado a cabo varias pruebas cuyos parámetros de configuración son los siguientes:

Número de clientes conectados = 35 (el número máximo de clientes disponibles).

Ruta: Distribución uniforme (toda la superficie del terreno tiene el mismo nivel de interés).

Tamaño de la caché de los servidores secundarios = 50% (independientemente del número de servidores secundarios).

Tamaño de la caché del cliente = 10% (consideramos este valor como aceptable dado el tamaño de la base de datos utilizada en la prueba).

Tamaño de la lista de clientes vecinos = 10 (consideramos este valor como aceptable dado el número máximo de clientes utilizados en la prueba).

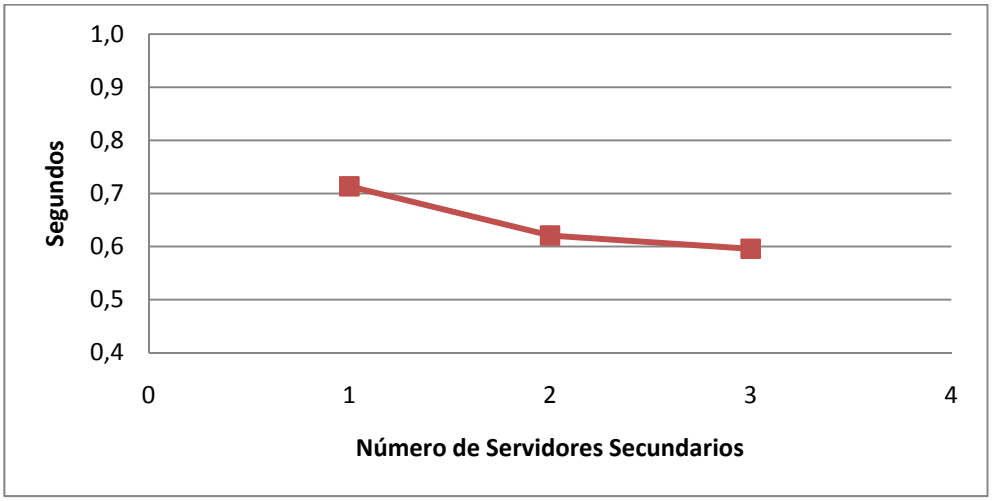
Las pruebas realizadas han sido:

Prueba_A: Número de servidores secundarios conectados = 1.

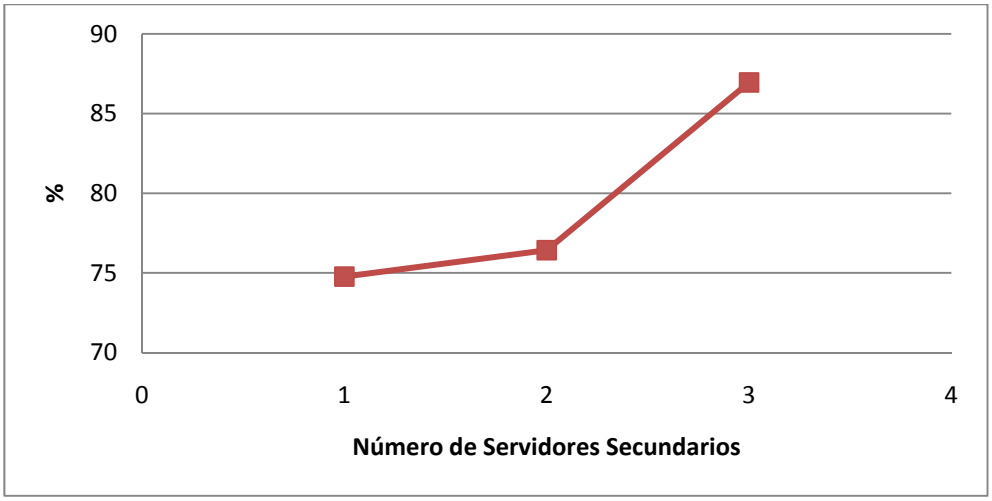
Prueba_B: Número de servidores secundarios conectados = 2.

Prueba_C: Número de servidores secundarios conectados = 3.

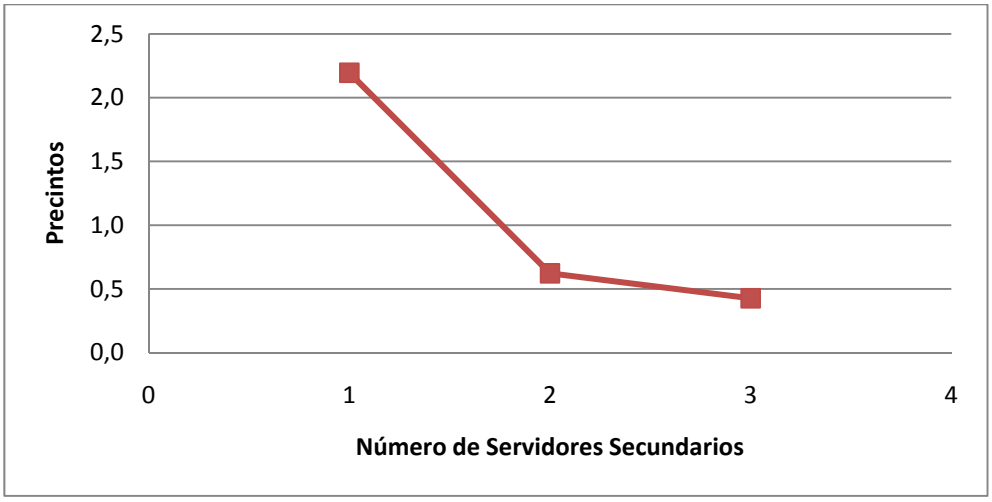
5.4.5.1 – Servidores Secundarios



Gráfica 5.33 – Latencia media de los precintos respondidos por los Servidores Secundarios



Gráfica 5.34 – Porcentaje medio de los precintos encontrados con éxito en la caché de los Servidores Secundarios



Gráfica 5.35 – Número de precintos medio en las colas de los Servidores Secundarios

En las gráficas anteriores se muestra la influencia que tiene en los servidores secundarios el número de servidores secundarios conectados al sistema.

En la Gráfica 5.33 se muestra la latencia media de los precintos respondidos por los servidores secundarios a los clientes. En ella se observa que conforme aumenta el número de servidores secundarios conectados al sistema, la latencia decrece.

En la Gráfica 5.34 se muestra el porcentaje medio de los precintos pedidos por los clientes a los servidores secundarios que se han encontrado con éxito en la caché de los servidores secundarios. En ella se observa como conforme se emplea un mayor número de servidores secundarios conectados al sistema, el porcentaje de precintos que son requeridos por los clientes que se encuentran con éxito en la caché de los servidores secundarios aumenta.

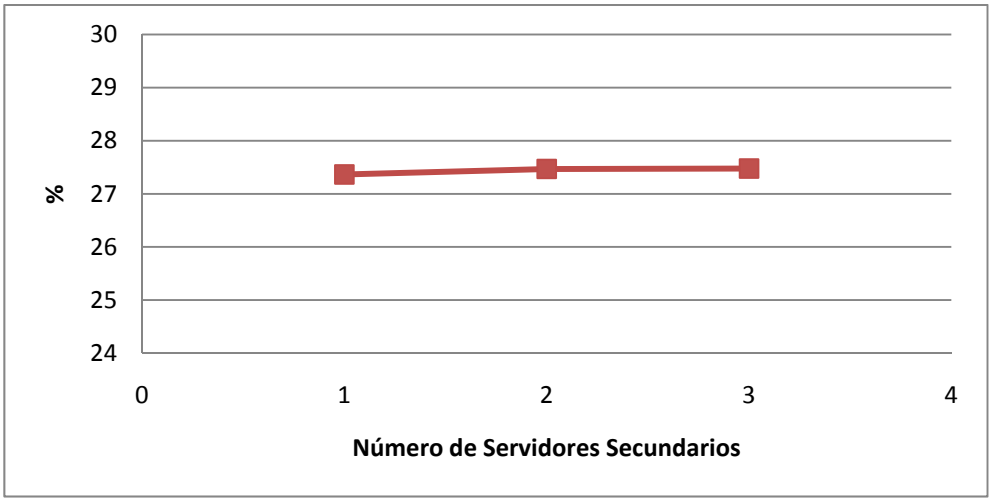
En la Gráfica 5.35 se muestra el número medio de precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por los servidores secundarios. En ella se observa que conforme haya un mayor número de servidores secundarios conectados al sistema, el tamaño de la cola disminuye, con lo que disminuye su carga. Si bien, hay que hacer notar que la diferencia de tamaño al usar dos y tres servidores secundarios es pequeña.

Obviamente, el hecho de existir un mayor número de servidores secundarios conectados en el sistema va a provocar que la carga de los mismos disminuya, debido a que el servidor de comunicaciones va a disponer de un mayor número de servidores entre los que repartir la misma carga. Este hecho se muestra en la Gráfica 5.35.

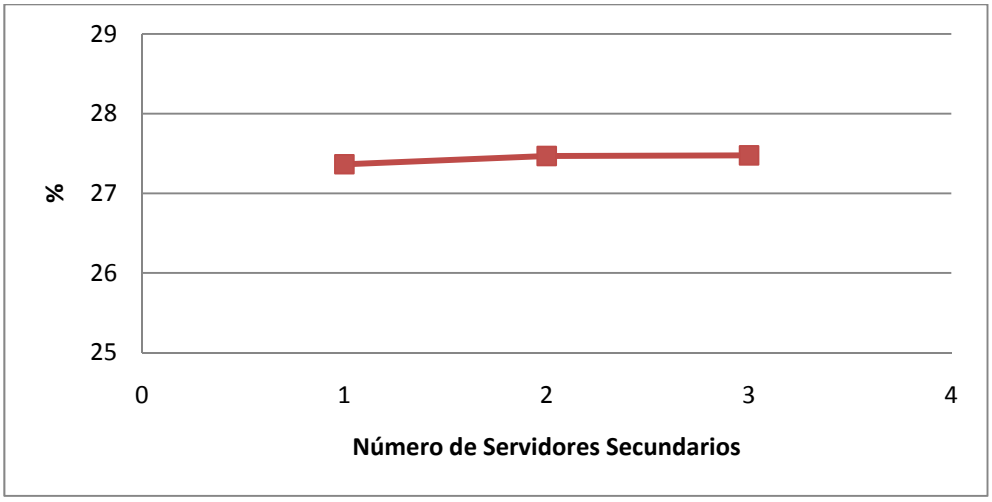
El mayor porcentaje de precintos pedidos por los clientes que se encuentran en la caché de los servidores secundarios a medida que hay más servidores secundarios conectados al sistema es debido, a nuestro entender, a que el servidor de comunicaciones puede distribuir espacialmente la carga de las peticiones de los clientes entre un número mayor de servidores secundarios, consiguiendo una mejor especialización de éstos por regiones del terreno.

La disminución de la carga y el aumento del porcentaje de precintos que se encuentran en la caché, pensamos que son los dos factores responsables de que la latencia de respuesta de los precintos de los servidores secundarios a los clientes sea ligeramente menor conforme aumenta el número de servidores secundarios conectados al sistema, tal y como se puede observar en la Gráfica 5.33.

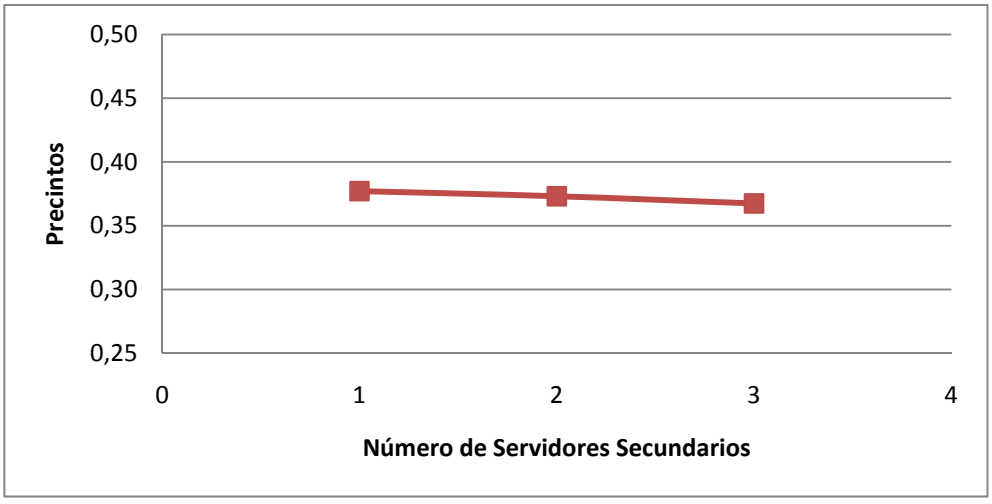
5.4.5.2 – Clientes



Gráfica 5.36 – Latencia media de los precintos intercambiados entre los Clientes



Gráfica 5.37 – Porcentaje medio de precintos respondidos por los Servidores Secundarios



Gráfica 5.38 – Número de precintos medio en las colas de los Clientes

En las gráficas anteriores se muestra la influencia que tiene en los clientes el número de servidores secundarios conectados en el sistema.

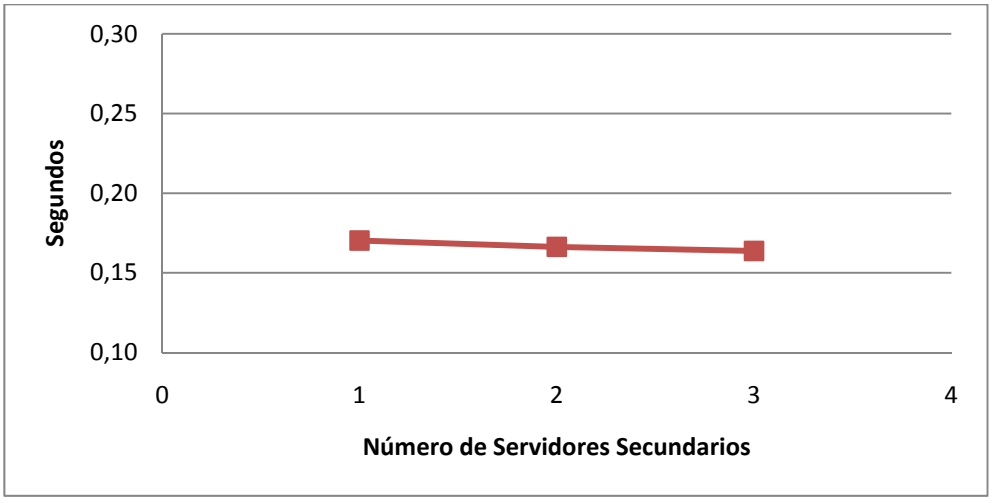
En la Gráfica 5.36 se muestra la latencia media de los precintos respondidos por los clientes. En ella se observa como esta latencia se mantiene bastante estable independientemente del número de servidores secundarios conectados al sistema.

En la Gráfica 5.37 se muestra el porcentaje medio de peticiones que realizan los clientes a los servidores secundarios. En ella se observa como este porcentaje se mantiene más o menos estable respecto al número de servidores secundarios conectados al sistema.

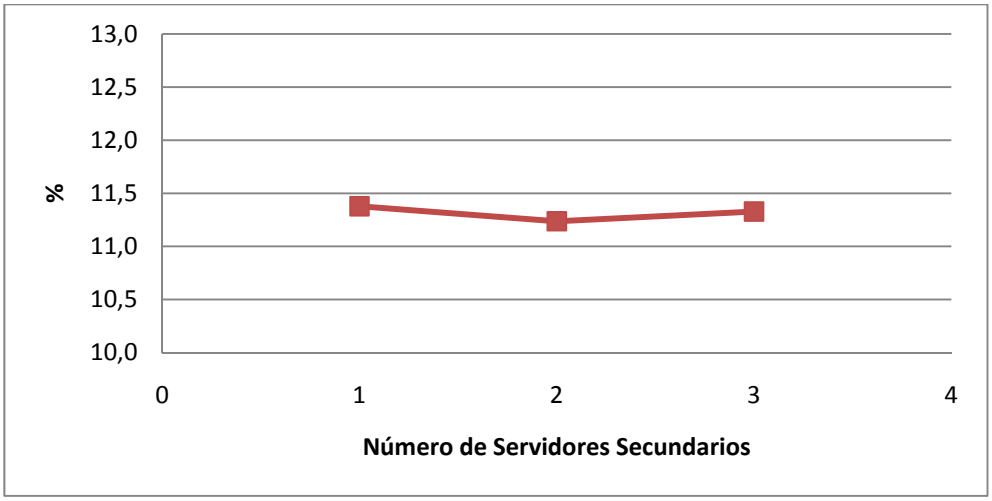
En la Gráfica 5.38 se muestra el número medio de precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por los clientes vecinos. En ella se observa como el tamaño de la cola se mantiene más o menos estable respecto al número de servidores secundarios conectados al sistema, por lo que su carga se mantiene estable.

Como se observa en las gráficas anteriores, el empleo de un diferente número de servidores secundarios parece que no afecta demasiado al comportamiento de los clientes, ya que tanto la latencia media de los precintos respondidos por los clientes, como las necesidades de los clientes de acceder a los servidores secundarios a por la información que requieren, así como la carga de los clientes, se mantienen más o menos estables. Este hecho pensamos que se debe a que la carga de los clientes se mantiene más o menos estable independientemente del número de servidores secundarios conectados al sistema, por lo que el funcionamiento de los clientes debe ser similar para las diferentes pruebas.

5.4.5.3 – Sistema



Gráfica 5.39 – Latencia media global del intercambio de precintos en el Sistema



Gráfica 5.40 – Porcentaje medio de los precintos vueltos a pedir

En las gráficas anteriores se muestra la influencia que tiene en el sistema en su conjunto el número de servidores secundarios conectados al mismo.

En la Gráfica 5.39 se muestra la latencia media global del intercambio de los precintos en el sistema. En ella se puede observar que esta latencia va disminuyendo ligeramente conforme aumenta el número de servidores secundarios conectados al sistema.

En la Gráfica 5.40 se muestra el número medio de precintos que los clientes han necesitado volver a pedir bien porque se han perdido, o bien porque han tardado más tiempo del permitido en obtenerse. En ella se observa como ese número de precintos se mantiene más o menos estable respecto al número de servidores secundarios conectados al sistema.

Como se observa en los resultados, el incremento en el número de servidores secundarios conectados al sistema proporciona un ligero descenso en la latencia general del sistema. Este hecho pensamos que se debe a la disminución de la latencia en el envío de los precintos de los servidores secundarios a los clientes que tiene lugar a medida que hay un mayor número de servidores secundarios conectados al sistema, como se observa en la Gráfica 5.33. Este descenso en la latencia global es ligero debido a que, a nuestro entender, el peso que tiene en su cálculo la latencia de los precintos enviados por los servidores secundarios a los clientes es menor que el de la latencia de los precintos enviados por los clientes, ya que el número de precintos intercambiados entre los clientes es superior, y como se observa en la Gráfica 5.36, esta latencia de intercambio de precintos entre clientes se mantiene estable.

La estabilidad en el número de precintos que se pierden o no llegan a tiempo y se vuelven a pedir, pensamos que se debe a que la cantidad de mensajes que se intercambian entre los elementos del sistema es similar, independientemente del número de servidores secundarios conectados al sistema, manteniéndose la cantidad de colisiones que se producen en las líneas de conexión.

5.4.5.4 – Conclusiones de la prueba

Como conclusión de esta apartado podemos decir que, para las pruebas que hemos llevado a cabo, cuanto mayor sea el número de servidores secundarios conectados al sistema, mejor va a ser su comportamiento. Esto pensamos que se debe, como hemos comentado en el apartado 5.4.5.1, a que el servidor de comunicaciones puede realizar un mejor reparto de la carga que han de soportar los servidores secundarios tanto de forma espacial como computacional. Sin embargo, cabe hacer notar que si la carga a distribuir no es muy elevada, la mejora en el rendimiento del sistema no será muy grande, por lo que habrá que mantener un compromiso entre esa mejora y el coste económico que conlleva añadir al sistema más servidores secundarios.

Para el resto de las pruebas a realizar para la evaluación del sistema se ha decido conectar al sistema dos servidores secundarios, ya que hemos considerado que ese número de servidores secundarios es suficiente dado el número de clientes que se van a conectar al sistema. Además, al emplear dos servidores secundarios, el servidor de comunicaciones podrá llevar a cabo la distribución de la carga de las peticiones entre ellos tanto de de forma espacial como computacional.

5.4.6 – Variación en el tamaño de la lista de clientes vecinos

Otro parámetro importante en el sistema es el tamaño de la lista de clientes vecinos que posee cada cliente, el cual, a priori, determinará la cantidad de información que puede obtener cada cliente de sus vecinos sin tener que acudir a los servidores secundarios.

El objetivo de esta prueba es estudiar cómo afecta el tamaño de la lista de clientes vecinos a las necesidades de acceso de los clientes a los servidores secundarios, y si se produce una mejora del funcionamiento general de sistema.

Se han llevado a cabo varias pruebas cuyos parámetros de configuración son los siguientes:

Número de servidores secundarios conectados = 2 (al menos son necesarios dos para que haya una distribución de la carga del sistema entre ellos).

Número de clientes conectados = 35 (el número máximo de clientes disponibles).

Ruta: Distribución uniforme (toda la superficie del terreno tiene el mismo nivel de interés).

Tamaño de la caché del servidor secundario = 50% (se reparte la base de datos entre los dos servidores secundarios).

Tamaño de la caché del cliente = 10% (consideramos este valor como aceptable dado el tamaño de la base de datos utilizada en la prueba).

Las pruebas realizadas han sido:

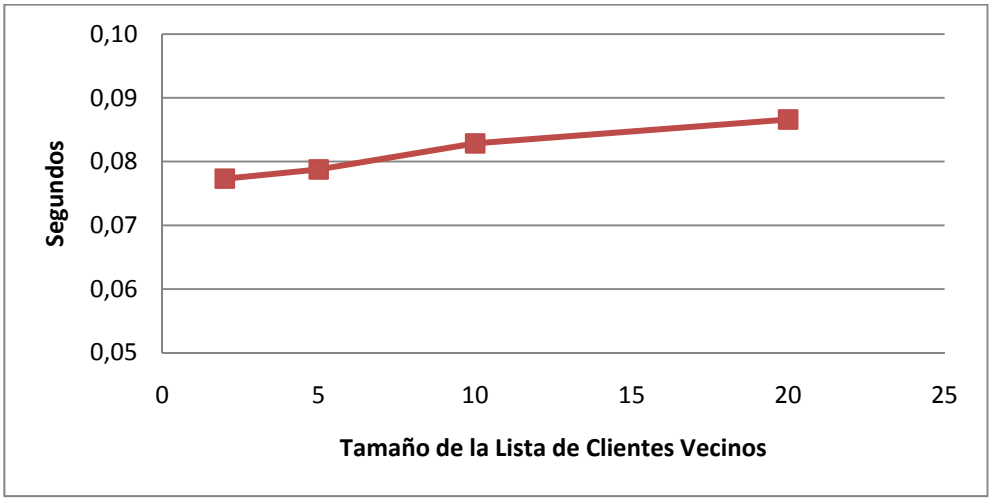
Prueba_A: Tamaño de la lista de clientes vecinos = 2.

Prueba_B: Tamaño de la lista de clientes vecinos = 5.

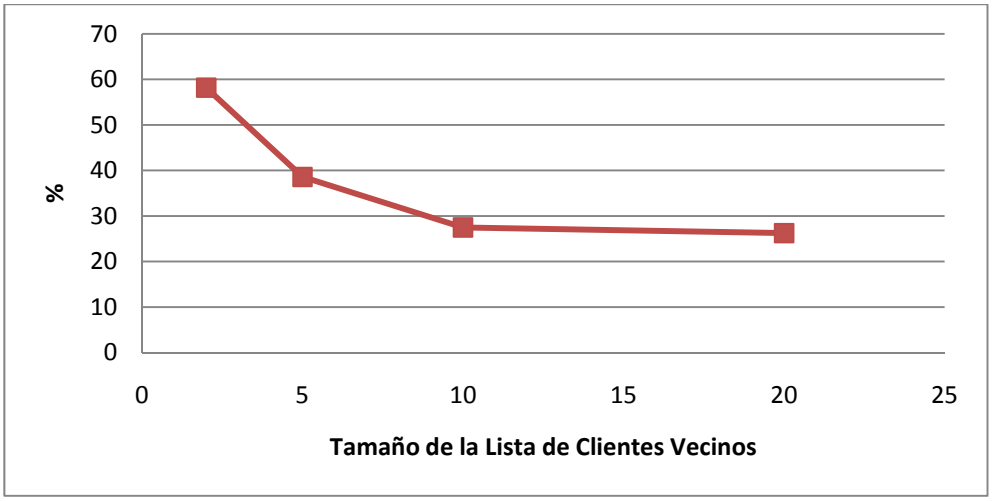
Prueba_C: Tamaño de la lista de clientes vecinos = 10.

Prueba_D: Tamaño de la lista de clientes vecinos = 20.

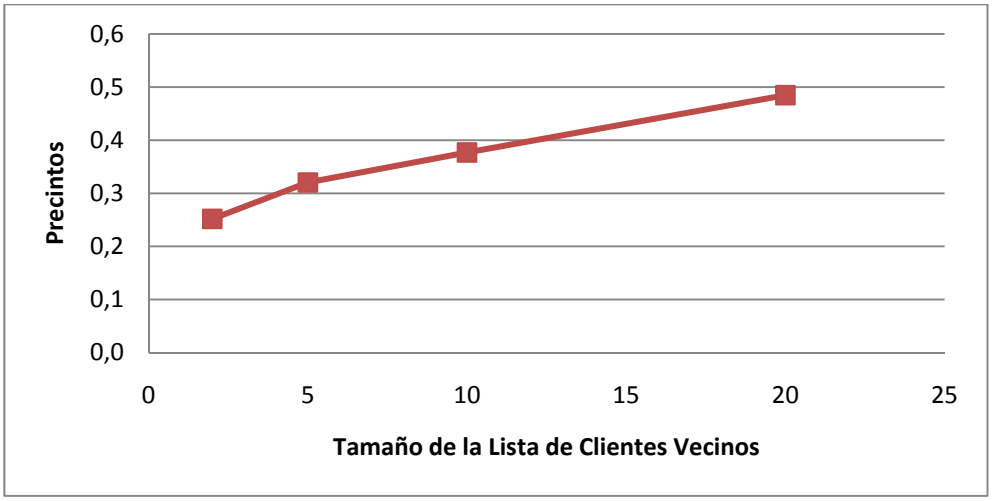
5.4.6.1 – Clientes



Gráfica 5.41 – Latencia media de los precintos intercambiados entre los Clientes



Gráfica 5.42 – Porcentaje medio de precintos respondidos por los Servidores Secundarios



Gráfica 5.43 – Número de precintos medio en las colas de los Clientes

En las gráficas anteriores se muestra la influencia que tiene en el cliente el tamaño de su lista de clientes vecinos.

En la Gráfica 5.41 se muestra la latencia media de los precintos respondidos por los clientes. En ella se observa como esta latencia aumenta ligeramente conforme aumenta el tamaño de la lista de clientes vecinos.

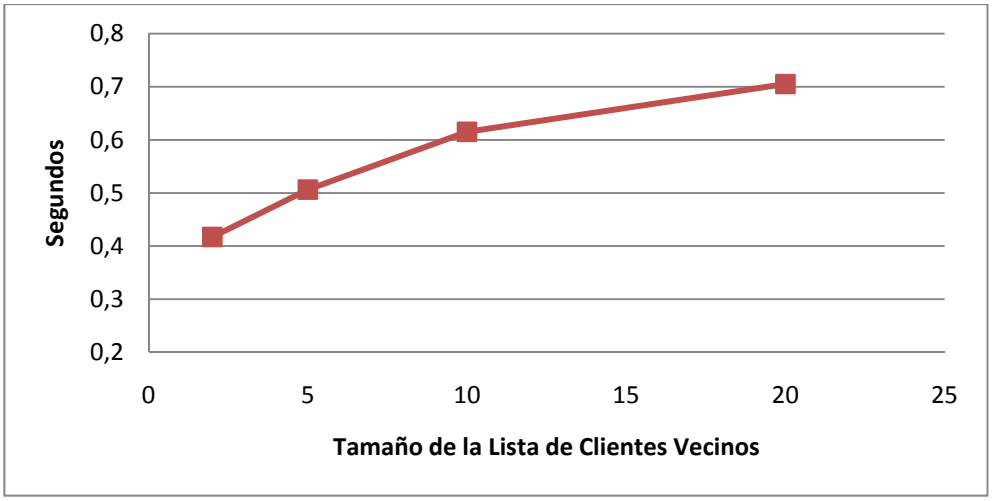
En la Gráfica 5.42 se muestra el porcentaje medio de peticiones que realizan los clientes a los servidores secundarios. En ella se observa como a medida que aumenta el tamaño de la lista de clientes vecinos, disminuye el porcentaje de peticiones realizadas a los servidores secundarios. Hay que destacar que a partir de un tamaño de 10 clientes vecinos el valor de ese porcentaje es muy parecido.

En la Gráfica 5.43 se muestra el número medio de precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por los clientes vecinos. En ella se observa como el incremento en el tamaño de la lista de los clientes vecinos provoca un aumento en el tamaño de la cola, con lo que aumenta su carga.

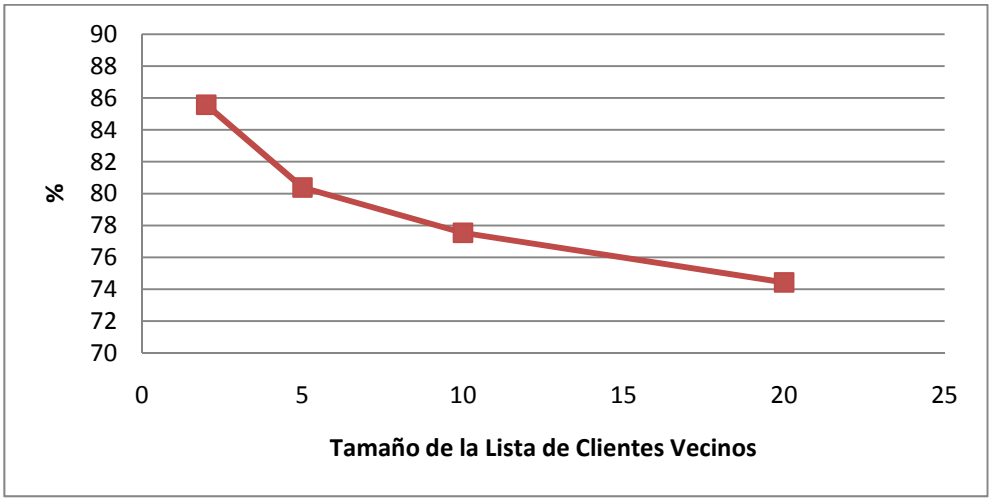
El incremento del tamaño de la lista de los clientes vecinos supone que los clientes van a tener acceso, a priori, a una mayor cantidad de información disponible por parte de los clientes vecinos. Esto va a suponer un descenso del número de peticiones que los clientes tienen que realizar a los servidores secundarios, como se observa en la Gráfica 5.42. Estas peticiones que no se realizan a los servidores secundarios, se realizarán entre los propios clientes, aumentándose por lo tanto el número de peticiones de precintos realizadas a los clientes vecinos, aumentando su carga, como se muestra en la Gráfica 5.43.

Este aumento de la carga de los clientes pensamos que es el responsable del ligero incremento de la latencia que se produce en el intercambio de precintos entre clientes, tal como se muestra en la Gráfica 5.41.

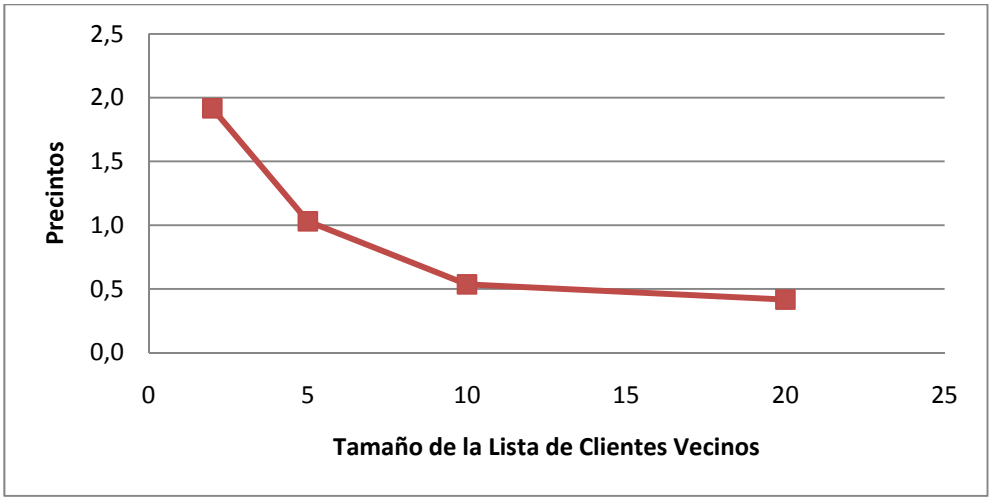
5.4.6.2 – Servidores Secundarios



Gráfica 5.44 – Latencia media de los precintos respondidos por los Servidores Secundarios



Gráfica 5.45 – Porcentaje medio de los precintos encontrados con éxito en la caché de los Servidores Secundarios



Gráfica 5.46 – Número de precintos medio en las colas de los Servidores Secundarios

En las gráficas anteriores se muestra la influencia que tiene en los servidores secundarios el tamaño de la lista de clientes vecinos de los clientes.

En la Gráfica 5.44 se muestra la latencia media de los precintos respondidos por los servidores secundarios a los clientes. En ella se observa como esta latencia se incrementa a medida que aumenta el tamaño de la lista de clientes vecinos.

En la Gráfica 5.45 se muestra el porcentaje medio de los precintos pedidos por los clientes a los servidores secundarios que se han encontrado con éxito en la caché de los servidores secundarios. En ella se observa como este porcentaje de éxito disminuye conforme aumenta el tamaño de la lista de clientes vecinos.

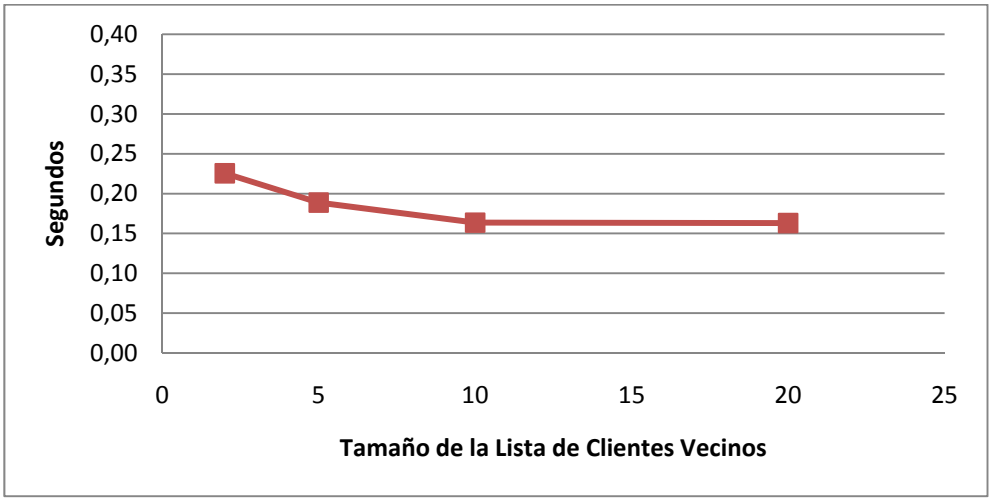
En la Gráfica 5.46 se muestra el número medio de precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por los servidores secundarios. En ella se observa como el tamaño de la cola disminuye conforme aumenta el tamaño de la lista de clientes vecinos, con lo que disminuye su carga. Si bien, se observa que a partir de un tamaño de la lista de 10 clientes vecinos el descenso en el tamaño de la cola es menor.

Como se ha comentado en el apartado 5.4.6.1, el incremento del tamaño de la lista de clientes supone un decremento del número de peticiones que los clientes realizan a los servidores secundarios. Este decremento se traduce en una disminución de la carga de los mismos, tal como se muestra en la Gráfica 5.46.

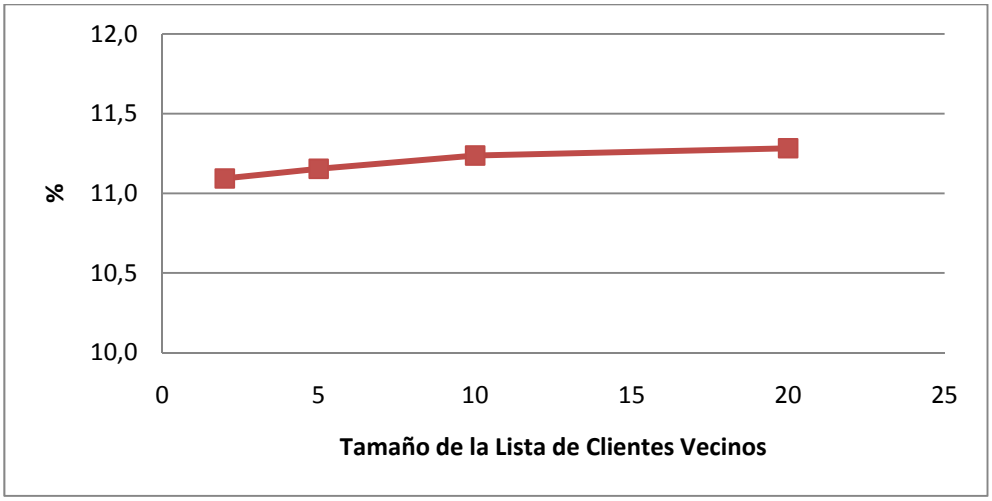
Además, como los clientes resuelven entre ellos una mayor cantidad de peticiones de precintos, a nuestro entender, los precintos que van a necesitar pedir a los servidores secundarios van a ser más "singulares", por lo que la probabilidad de que se encuentren en la caché de los servidores secundarios es menor, provocando un descenso en el porcentaje de éxito, tal como se muestra en la Gráfica 5.45.

Este descenso del porcentaje de éxito obligará a los servidores secundarios a realizar un mayor número de conexiones con el servidor principal para obtener los precintos que no dispone en su caché, provocando el aumento de la latencia de respuesta de los precintos enviados por los servidores secundarios a los clientes, hecho que se muestra en la Gráfica 5.44.

5.4.6.3 – Sistema



Gráfica 5.47 – Latencia media global del intercambio de precintos en el Sistema



Gráfica 5.48 – Porcentaje medio de los precintos vueltos a pedir

En las gráficas anteriores se muestra la influencia que tiene en el sistema en su conjunto el tamaño de la lista de clientes vecinos de los clientes.

En la Gráfica 5.47 se muestra la latencia media global del intercambio de los precintos en el sistema. En ella se puede observar que esta latencia global inicialmente disminuye conforme se emplea un tamaño de la lista de clientes vecinos mayor, para a partir de un tamaño de la lista de 10, mantenerse más o menos estable.

En la Gráfica 5.48 se muestra el número medio de precintos que los clientes han necesitado volver a pedir bien porque se han perdido, o bien porque han tardado más tiempo del permitido en obtenerse. En ella se observa como este número de precintos se incrementa ligeramente conforme aumenta el tamaño de la lista de clientes vecinos empleada.

El decremento inicial de la latencia global del intercambio de precintos pensamos que se debe a que, a pesar de que la latencia de respuesta de los precintos por parte de los servidores secundarios aumenta, y la de los clientes crece ligeramente, como el número de peticiones de precintos resueltos por los clientes aumenta (siendo la latencia de éstos menor que la de los servidores secundarios) y disminuye el de los servidores secundarios, esto hace que se produzca un decremento de la latencia global. A partir de una lista de clientes vecinos de tamaño 10, el porcentaje de peticiones que se realizan a los servidores secundarios es similar. Este hecho pensamos que es el motivo por el que la latencia a partir de este punto se estabiliza, en vez de continuar disminuyendo.

El incremento del número de precintos que se vuelven a pedir conforme aumenta el tamaño de la lista de clientes vecinos, pensamos que se debe, a priori, al mayor número de mensajes intercambiados entre los clientes debido a que éstos han de comunicarse con un número mayor de clientes vecinos, lo que afectará al número de colisiones que se producirán en las líneas de conexión.

5.4.6.4 – Conclusiones de la prueba

Como conclusión de esta prueba, se puede decir que el incremento en el tamaño de la lista de los clientes vecinos de los cuales un cliente puede descargar información, parece que provoca un descenso en la latencia global del intercambio de precintos en el sistema. Este hecho permite comprobar que la asignación que realiza el servidor de comunicaciones, para cada cliente, de los clientes vecinos más adecuados de los cuales descargar información funciona de forma correcta. Si bien, cabe hacer notar que emplear un número mayor a 10 clientes vecinos, parece que no aporta mayores beneficios al funcionamiento del sistema. Esto pensamos que se debe a que, para esta prueba, dado el tamaño de los datos y el número de clientes totales utilizados en la prueba, emplear un número de clientes vecinos mayor a 10 no supone un aumento suficiente de la cantidad de información disponible para cada cliente de sus clientes vecinos como para que aumente el rendimiento del sistema.

En vista de los resultados, se ha decidido emplear para el resto de pruebas realizadas para evaluar el sistema un tamaño de lista de clientes vecinos de 10 clientes, debido a que se ha observado que, a priori, un número mayor de clientes vecinos no aporta mayores beneficios al funcionamiento del sistema y si que aumenta el tráfico que se produce en la red, incrementando el número de precintos que se pierden o tardan un tiempo excesivo en llegar a los clientes.

5.4.7 – Prueba Cliente-Servidor

Con el objetivo de ver cuál de las dos arquitecturas, la arquitectura clásica cliente-servidor y la arquitectura mixta cliente-servidor / P2P, se adapta mejor a un sistema de visualización de terrenos con bases de datos remotas, se ha realizado una prueba en la cual los clientes siempre realizan las peticiones de datos a un servidor que dispone de toda la información, y no intercambian datos con otros clientes, siguiendo el comportamiento de una arquitectura cliente-servidor clásica.

Los parámetros de configuración para esta prueba son los siguientes:

Número de servidores conectados = 1 (un solo servidor con toda la información).

Número de clientes conectados = 35 (el número máximo de clientes disponible).

Ruta: Distribución uniforme (toda la superficie del terreno tiene el mismo nivel de interés).

Tamaño de la caché del servidor = 100% (toda la base de datos se encuentra en el servidor).

Tamaño de la caché del cliente = 10% (consideramos este valor como aceptable dado el tamaño de la base de datos utilizada en la prueba).

Tamaño de la lista de clientes vecinos = 0 (no se intercambia información con ningún cliente, toda la información se descarga del servidor).

Para realizar la comparativa de la arquitectura cliente-servidor clásica con la nueva arquitectura mixta cliente-servidor / P2P, se ha realizado también una prueba con esta última donde se ha elegido una configuración lo más similar posible a la arquitectura cliente-servidor clásica, de forma que los resultados obtenidos sean equiparables.

En ambas arquitecturas los clientes se van a encargar de la visualización de la información, la única diferencia estriba en que en la arquitectura cliente-servidor todos los datos se van a pedir al servidor, mientras que en la arquitectura mixta los datos se pedirán a los clientes vecinos, y en caso de que éstos no dispusieran de ellos, se pedirán al servidor secundario. El número máximo de clientes vecinos empleados en la arquitectura mixta será de 10, que consideramos un valor aceptable de acuerdo con el número máximo de clientes utilizados en la prueba.

En la arquitectura cliente-servidor se empleará un solo servidor del cual se va a descargar toda la información. En cambio, en la arquitectura mixta se empleará un servidor secundario del cual los clientes obtendrán la información que requieran, y además de este servidor secundario, existirá un servidor principal al que el servidor secundario podrá acceder en el caso de que no disponga de los precintos requeridos por los clientes en su caché, que será del 50% del tamaño de la base de datos original. Este tamaño de caché se considera un tamaño adecuado respecto al tamaño total de la base de datos utilizada en la prueba.

Además, para la arquitectura mixta se empleará un servidor de comunicaciones que permitirá conectar a los clientes con los clientes vecinos y el servidor secundario.

A continuación se muestra una tabla en la que se encuentran los resultados obtenidos en esta prueba para la arquitectura cliente servidor, y para la arquitectura mixta

Arquitectura	Latencia Global	Número de Precintos Medio Cola Servidor	% precintos respondidos por el Servidor
Cliente-Servidor	0,45	24,0	100,0
Mixta	0,18	2,2	27,6

Tabla 5.1 – Comparativa de la arquitectura Cliente-Servidor y de la arquitectura mixta Cliente-Servidor / P2P

En la Tabla 5.1 se muestra: la latencia media global del intercambio de los precintos en el sistema, el número medio de precintos pedidos por los clientes que se encuentran en espera de ser respondidos por los servidores y el porcentaje medio de los precintos pedidos por los clientes a los servidores.

Como se puede observar en la Tabla 5.1, la latencia global empleando la arquitectura cliente-servidor clásica es más del doble que con la arquitectura mixta. Además la carga del sistema es muy superior, siendo el número de precintos que espera en la cola unas 12 veces mayor.

También se observa en la tabla que obviamente, empleando la arquitectura cliente-servidor, el servidor tendrá que responder al 100% de las peticiones que realicen los clientes, a diferencia de cuando se emplea una arquitectura mixta, en la que el porcentaje de precintos que responde el servidor secundario se mantiene por debajo del 30%.

En ambas pruebas el porcentaje de precintos que se ha necesitado volver a pedir bien porque se han perdido, o bien porque han tardado más tiempo del permitido en obtenerse es similar, en torno al 11%.

Los mejores resultados obtenidos por la arquitectura mixta pensamos que se deben a que, en esta arquitectura, la carga global de peticiones del sistema se reparte entre los clientes conectados al sistema y el servidor secundario, mientras que en la arquitectura cliente-servidor, toda la carga global de peticiones ha de ser asumida únicamente por el servidor.

5.4.7.1 – Conclusiones de la prueba

Como conclusión de este apartado podemos decir que la arquitectura mixta cliente-servidor / P2P parece que se adapta mejor a un sistema de visualización de terrenos con bases de datos remotas que una arquitectura cliente-servidor clásica, que ya con un número no muy elevado de clientes da muestras de su falta de escalabilidad, lo que probablemente supondrá que a medida que se incremente este número de clientes conectados, sus prestaciones se reduzcan muy rápidamente y llegue a colapsarse.

5.4.8 – Distribución de la carga en el Sistema

Uno de los objetivos en nuestro sistema es distribuir la carga global del mismo entre todos los elementos del sistema. Para comprobar si esta distribución se ha realizado de forma adecuada tanto en los clientes como en los servidores secundarios, se han medido un conjunto de parámetros a lo largo del transcurso de la prueba:

- Para observar si la carga de los clientes se mantiene estable, se medirá en determinados instantes de la simulación, cuál es en esos instantes la latencia de los precintos respondidos por los clientes vecinos, y el número de precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por los clientes vecinos.

- Para observar si la carga de los servidores secundarios se mantiene estable, se medirá en determinados instantes de la simulación, cuál es en esos instantes el tiempo de proceso que necesitan éstos para responder los precintos a los clientes, y el número de precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por los servidores secundarios.

El objetivo de esta prueba es conocer si el reparto de la carga entre todos los elementos del sistema se hace de forma equitativa y se mantiene estable a lo largo de la prueba.

Los parámetros de configuración empleados son los siguientes:

Número de servidores secundarios conectados = 2 (al menos son necesarios dos para que haya una distribución de la carga del sistema entre ellos).

Ruta: Distribución uniforme (toda la superficie del terreno tiene el mismo nivel de interés).

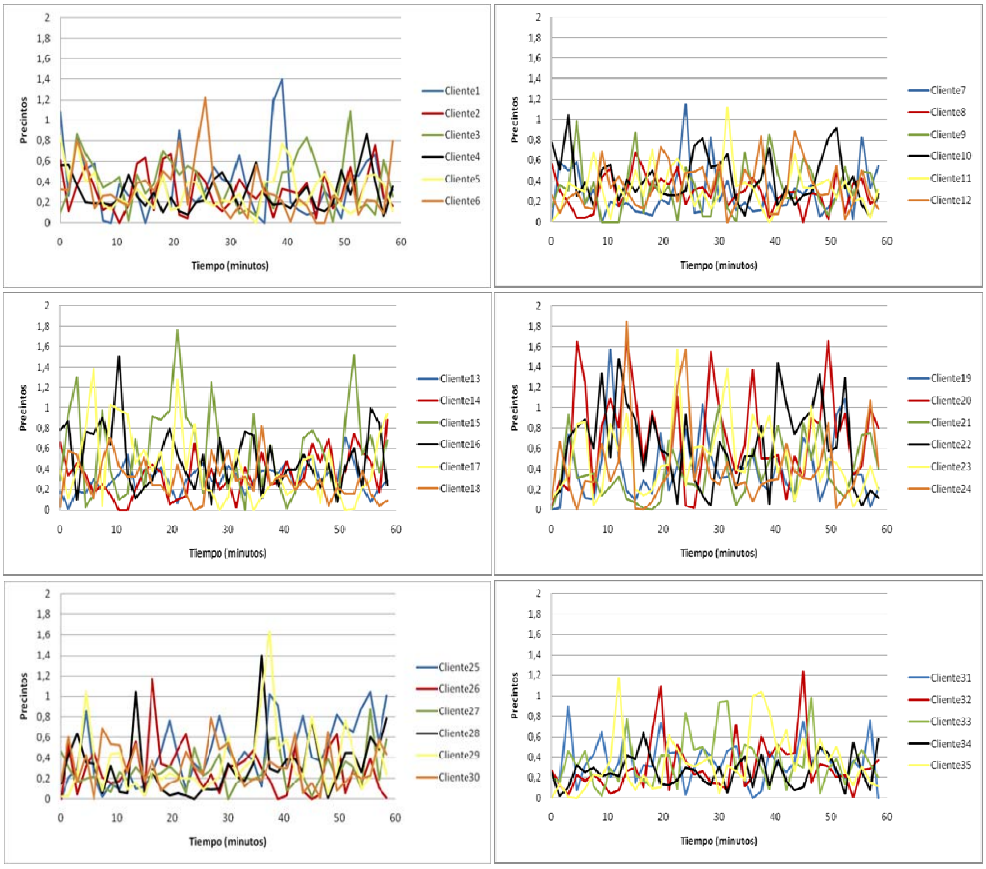
Tamaño de la caché del servidor secundario = 50% (se reparte la base de datos entre los dos servidores secundarios).

Tamaño de la caché del cliente = 10% (consideramos este valor como aceptable dato el tamaño de la base de datos utilizada en la prueba).

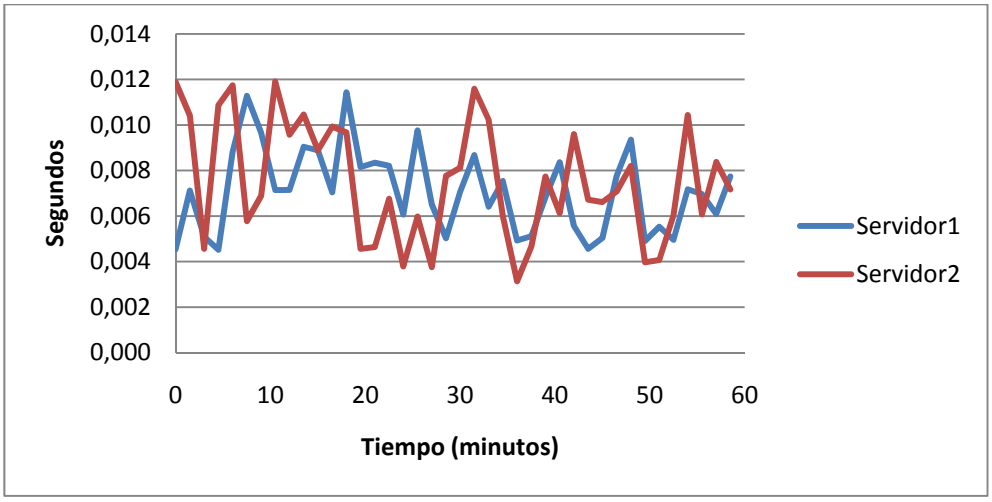
Tamaño de la lista de clientes vecinos = 10 (consideramos este valor como aceptable dado el número máximo de clientes utilizados en la prueba).

Número de clientes conectados = 35 (el número máximo de clientes disponible).

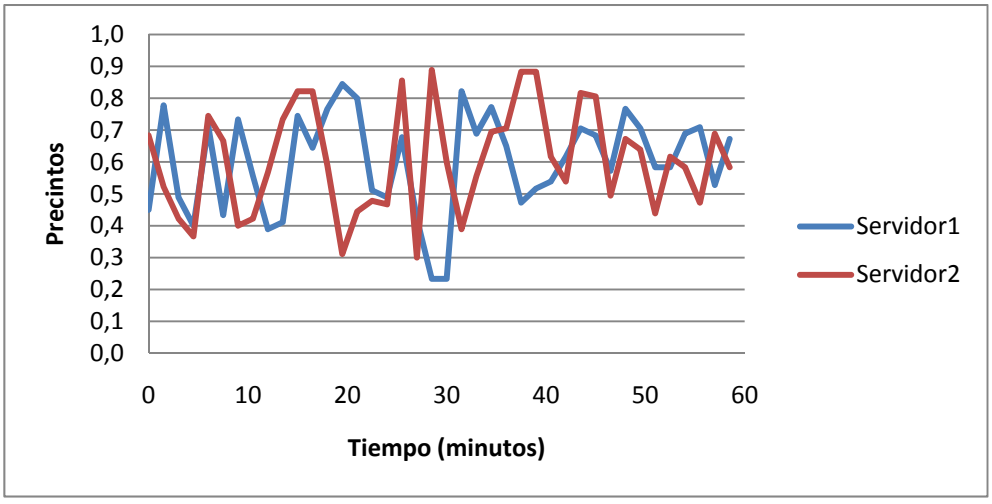
Gráfica 5.49 – Latencia de los precintos intercambiados entre los clientes



Gráfica 5.50 – Número de precintos en las colas de los Clientes



Gráfica 5.51 – Tiempo de proceso de los precintos en los Servidores Secundarios



Gráfica 5.52 – Número de precintos en las colas de los Servidores Secundarios

En la Gráfica 5.49 se agrupan un conjunto de seis gráficas que muestran la latencia media de los precintos respondidos por los clientes a lo largo de la prueba. En ellas se observa como aparecen dos grupos de ordenadores en los cuales la latencia se mantiene entre un intervalo de 0 y 0.25 segundos para el primer grupo, y entre 0.1 y 0.5 segundos para el segundo grupo.

En la Gráfica 5.50 se agrupan un conjunto de seis gráficas que muestran el número de los precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por los clientes vecinos a lo largo de la prueba. En ellas se observa que el tamaño de la cola de los clientes se mantiene entre el intervalo de 0 y 1.8 precintos. Si bien, en la mayoría de los clientes estos valores se mantienen entre 0 y 0.6 precintos.

En la Gráfica 5.51 se muestra el tiempo de proceso que necesitan los servidores secundarios para enviar los precintos a los clientes a lo largo de la prueba. En ella se observa como este tiempo se mantiene dentro del intervalo de 0.003 y 0.012 segundos.

En la Gráfica 5.52 se muestra el número de los precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por los servidores secundarios a lo largo de la prueba. En ella se muestra que el tamaño de la cola de los servidores secundarios se mantiene entre 0.2 y 0.9 precintos.

Los picos puntuales que se producen en la Gráfica 5.49, pensamos que se deben a que las peticiones realizadas por los clientes se han dirigido a algunos clientes vecinos que, en ese momento, tienen una mayor carga de la habitual, (tal como se observa en los picos de la Gráfica 5.50), y por eso tardan ligeramente más en responder los precintos. Si bien, como se observa en los resultados, rápidamente se corrige esa situación, debido a que el cliente detectará ese aumento de la carga del cliente o clientes vecinos a los que les ha pedido información, y a la hora de realizar nuevas peticiones, escogerá a otros de los clientes vecinos que tenga en su lista con una carga menor para que le sirvan la información que requiera. Esto provocará a su vez, que se reduzca la carga de estos clientes que se encontraban más cargados.

Como nota se puede destacar que en la Gráfica 5.49 se distinguen dos conjuntos de clientes respecto al valor de la latencia de intercambio de precintos entre clientes. A priori, todos los equipos tienen unas características hardware similares y están conectados a la misma red local, por lo tanto, esta diferencia que se observa pensamos que se debe a que, a pesar de que todos los clientes pertenecen a la misma red local, éstos se encuentran separados físicamente en varias habitaciones, por lo que utilizan diferentes elementos intermedios de conexión y diferentes líneas de conexión, las cuales puede que supongan diferentes retardos y velocidades de transmisión.

En cuanto a la Gráfica 5.51 y a la Gráfica 5.52, se observa como los valores obtenidos se mantienen en intervalos parecidos y centrados en torno a la media de los mismos.

Por lo tanto, como se ha observado en las gráficas anteriores, el sistema es capaz de mantener distribuida equitativamente la carga del sistema entre los servidores secundarios y entre los clientes. No se observa que ningún elemento del sistema mantenga una carga más elevada que el resto durante un largo período de tiempo, si no que si en algún momento

aumenta demasiado, posteriormente el sistema lo corrige y se observa una disminución de la misma.

Por todo ello, podemos comentar que el sistema ha sido capaz de ajustarse a las diferentes prestaciones que ofrecen los clientes, distinguiendo las capacidades de unos y otros, para mantener una carga equitativa entre todos ellos.

5.4.8.1– Conclusiones de la prueba

Como se ha comprobado con los resultados, se puede concluir que el servidor de comunicaciones funciona de forma correcta, repartiendo la carga del sistema equitativamente entre los diferentes elementos del sistema y evitando la sobrecarga de cualquiera de ellos.

5.4.9 – Validación de la función de probabilidad empleada

En el apartado 5.2.2.6.2 se definió la función que se ha empleado en el sistema para predecir la probabilidad de que un precinto se encuentre en la caché de un servidor secundario. Esta función se ha determinado de forma empírica a través de diversas pruebas del sistema.

El objetivo de estas pruebas es averiguar si existe alguna relación entre la distancia a la que se encuentra un precinto de la posición media de los datos almacenados en un servidor secundario, y la probabilidad de que ese precinto se encuentre en la caché de ese servidor secundario.

En caso de que exista alguna relación, otro objetivo de las pruebas será encontrar la función más adecuada que permita predecir la probabilidad de que un precinto se encuentre en la caché de un servidor secundario.

A continuación se van a mostrar los resultados de algunas de esas pruebas, donde los parámetros de configuración empleados han sido los siguientes:

Número de servidores secundarios conectados = 2 (al menos son necesarios dos para que haya una distribución de la carga del sistema entre ellos).

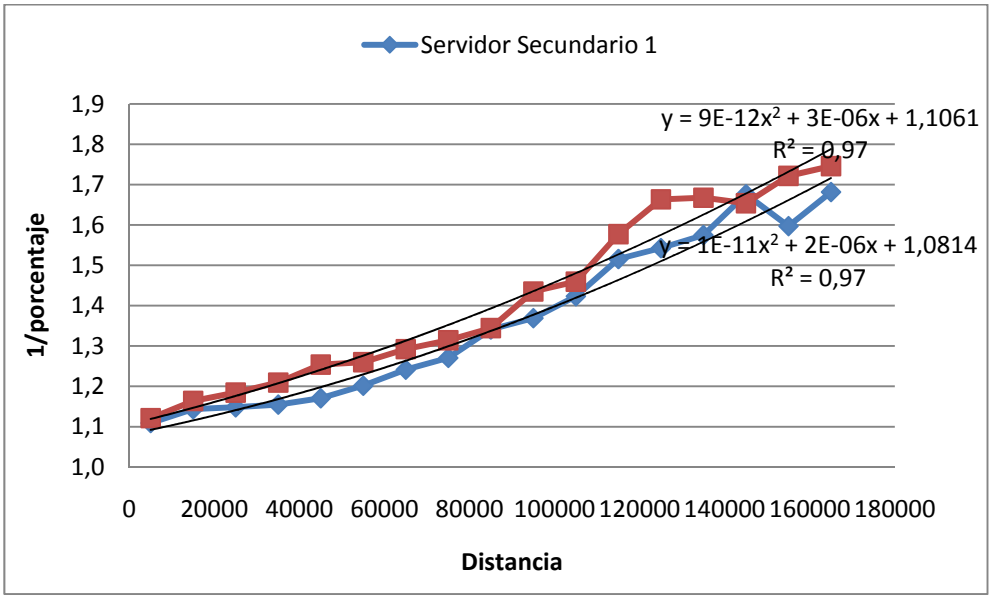
Ruta: Distribución uniforme (toda la superficie del terreno tiene el mismo nivel de interés).

Tamaño de la caché del cliente = 10% (consideramos este valor como aceptable dato el tamaño de la base de datos utilizada en la prueba).

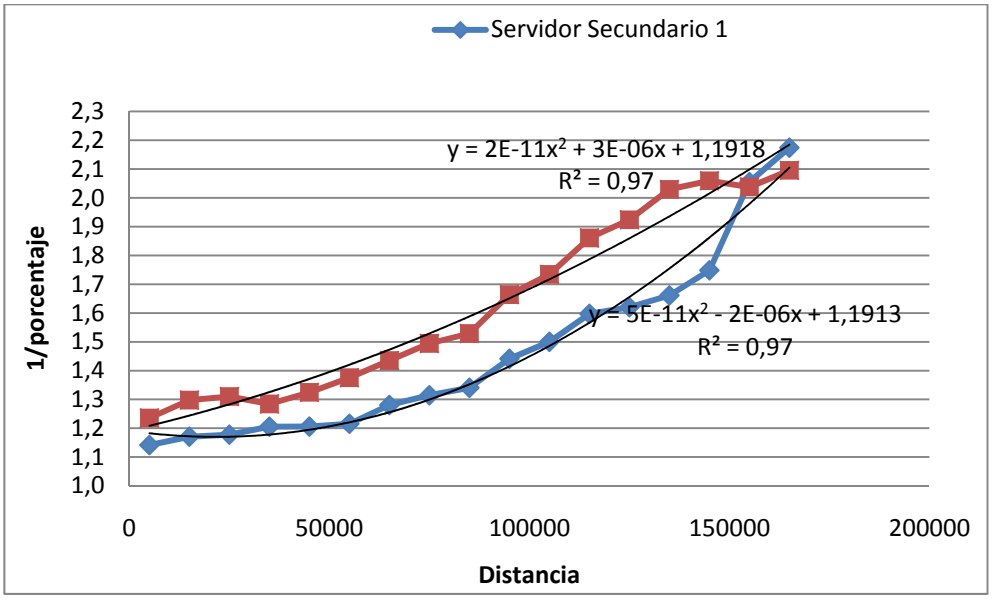
Tamaño de la lista de clientes vecinos = 10 (consideramos este valor como aceptable dado el número máximo de clientes utilizados en la prueba).

Número de clientes conectados = 35 (el número máximo de clientes disponible).

En estas pruebas, se ha ido variando el tamaño de la caché del servidor secundario empleada. En las gráficas que se muestran a continuación, se ha utilizado un tamaño de caché del 50% del tamaño de la base de datos (Gráfica 5.53) y un tamaño de 25% (Gráfica 5.54).



Gráfica 5.53 – Inversa del porcentaje de éxito de que un precinto se encuentre en la caché del servidor secundario respecto a la distancia del precinto pedido al centro de la base de datos del servidor secundario.



Gráfica 5.54 – Inversa del porcentaje de éxito de que un precinto se encuentre en la caché del servidor secundarios respecto a la distancia del precinto pedido al centro de la base de datos del servidor secundario.

	Lineal	Cuadrática	Cúbica
Servidor 1	0,96	0,97	0,98
Servidor 2	0,95	0,97	0,98

Tabla 5.2 – Coeficiente de correlación al cuadrado de diferentes tipos de funciones de regresión para la Gráfica 5.53.

	Lineal	Cuadrática	Cúbica
Servidor 1	0,95	0,97	0,98
Servidor 2	0,87	0,97	0,98

Tabla 5.3 – Coeficiente de correlación al cuadrado de diferentes tipos de funciones de regresión para la Gráfica 5.54

En la Gráfica 5.53 y en la Gráfica 5.54 se muestra el valor de la inversa del porcentaje de éxito de que un precinto se encuentre disponible en la caché del servidor secundario, respecto a la distancia existente entre la posición del precinto dentro de la escena y la posición media de los datos almacenados en la caché del servidor secundario.

También se muestra una de las funciones de regresión calculadas, concretamente la función de regresión cuadrática, que se ajusta a los resultados obtenidos en la prueba junto con su coeficiente de correlación al cuadrado.

En la Tabla 5.2 y en la Tabla 5.3 se muestran los coeficientes de correlación al cuadrado de las diferentes tipos de funciones de correlación que se han calculado, con el objetivo de averiguar cuál de ellas se ajusta mejor a los resultados obtenidos en las pruebas.

En la Gráfica 5.53 y en la Gráfica 5.54 se observa que, como era de esperar, a medida que los precintos que se piden a los servidores secundarios se encuentran más alejados de la posición media de los datos almacenados en la caché del servidor secundario, el porcentaje de éxito de encontrar esos precintos en la caché del servidor secundario disminuye.

En la Tabla 5.2 y en la Tabla 5.3 se observa que el tipo de función que se ajusta más a los resultados de las pruebas es la cúbica, seguida de la cuadrática y por último de la lineal. El coeficiente de correlación al cuadrado es muy alto para todos los tipos de función, sin embargo, cabe destacar que para la función lineal, aparecen casos en los que la diferencia respecto a los otros tipos de funciones es considerable, como en la Tabla 5.3, donde este coeficiente tiene un valor para el servidor secundario 2 de 87, mientras que los otros tipos de funciones obtienen valores de 97 y 98.

5.4.9.1– Conclusiones de la prueba

El objetivo de esta prueba era averiguar si existía alguna relación entre la distancia a la que se encuentra un precinto de la posición media de los datos almacenados en un servidor secundario, y la probabilidad de que ese precinto se encontrará en la caché de ese servidor secundario. Además, en caso afirmativo, se pretendía encontrar una función analítica adecuada que permita predecir la probabilidad de que un precinto se encuentre en la caché de un servidor secundario.

Como conclusión a esta prueba se puede decir que parece que existe dicha relación. Además, este tipo de relación parece que se ajusta bien a las funciones cuadráticas y cúbicas (la función lineal en ocasiones no se ajusta demasiado al comportamiento real). De entre estos dos tipos de funciones, no se aprecia demasiada diferencia en el coeficiente de correlación calculado, si bien, para la función cúbica es ligeramente superior.

Por lo tanto, debido a que el ajuste para ambos tipos de funciones es similar, y a que el cálculo de la función cuadrática es más simple y tiene un coste computacional menor, se ha decidido emplear la función cuadrática para predecir la probabilidad de que un precinto se encuentre en la caché de un servidor secundario.

5.5 – Conclusiones

Se han llevado a cabo un conjunto de pruebas para evaluar la arquitectura mixta cliente-servidor / P2P diseñada en esta tesis ante diferentes condiciones de funcionamiento. A continuación vamos a destacar las conclusiones más importantes obtenidas en estas pruebas.

Una de las características exigidas a la hora de diseñar la arquitectura mixta cliente-servidor / P2P era que fuera escalable con el número de usuarios conectados al mismo. Como se ha podido ver en los resultados (apartado 5.4.3), parece que el sistema es escalable con el número de clientes, ya que es capaz de repartir el aumento de la carga que produce el incremento del número de clientes conectados al sistema, entre los clientes y los servidores secundarios del mismo, manteniendo la misma calidad de servicio. Además, este reparto se realiza de forma equitativa, evitando la sobrecarga de los diferentes elementos del sistema y manteniendo un nivel de carga similar para cada uno de ellos, tal como se ha observado en los resultados mostrados en el apartado 5.4.8.

Se ha podido comprobar que el sistema tiene un comportamiento mixto acorde a lo esperado. Es decir, cuando existe un número reducido de clientes conectados al sistema, éste se comporta como una arquitectura cliente-servidor, donde una parte importante de la carga del sistema recaerá en los servidores secundarios, mientras que conforme aumenta el número de clientes conectados, el sistema pasa a comportarse como una arquitectura P2P, donde los clientes se hacen cargo de la mayor parte de la carga del sistema, liberando de ella a los servidores secundarios.

También se ha podido comprobar en los resultados, que el servidor de comunicaciones realiza una buena selección de los servidores secundarios y de los clientes vecinos que han de emplear los clientes ante las diferentes condiciones de funcionamiento empleadas en las pruebas. Además, cabe destacar que la selección será mejor a medida que haya un mayor número de clientes y/o servidores secundarios conectados al sistema, ya que en esos casos el servidor de comunicaciones va a disponer de un mayor número de posibilidades para asignar a cada cliente el servidor secundario y los clientes vecinos más adecuados.

Otra conclusión que se ha extraído de las pruebas es que cuanto mayor sea el tamaño de la caché empleada, tanto para el caso de los clientes como para el caso de los servidores secundarios, a priori, tanto mejor será la eficiencia global del sistema. Sin embargo, como

ya se comentó en los apartados 5.4.1.4 y 5.4.4.4, hay que llegar a un compromiso entre eficiencia y tamaño de caché, debido a que las bases de datos de terrenos que se suelen emplear en estos sistemas de visualización en tiempo real son muy extensas, y puede no ser viable o conveniente emplear un tamaño de caché muy elevado, sobre todo en el caso de los clientes.

También se ha comprobado en los resultados que el sistema funciona de forma correcta ante diferentes comportamientos de los usuarios conectados al mismo. Se han simulado, a través de un conjunto de distribuciones aleatorias, diferentes comportamientos de los usuarios, de las cuales, las que han obtenido unos mejores tiempos de latencia globales han sido las distribuciones normales que empleaban puntos calientes, ya que por su naturaleza, van a facilitar la especialización de los servidores secundarios por regiones del terreno, lo que proporciona un funcionamiento más rápido y eficiente de los mismos, afectando positivamente al funcionamiento global del sistema.

Otra circunstancia que se ha podido comprobar en los resultados es que la arquitectura mixta cliente-servidor / P2P proporciona unas mejores prestaciones que la arquitectura clásica cliente-servidor. A pesar de que la arquitectura cliente-servidor no llega a colapsarse debido a que no se emplea un número de clientes conectados al sistema muy elevado, si que comienzan a mostrarse signos de su falta de escalabilidad en los resultados, lo que previsiblemente provocará su saturación en el instante en el que el número de clientes conectados al sistema sea considerable.

Este sistema ha sido diseñado para soportar una gran carga de trabajo provocada por la conexión al mismo de un elevado número de clientes. Como hemos comentado, el número de clientes conectado al sistema empleado en las pruebas no es muy elevado, lo que provoca que en ciertas pruebas no sea posible extraer tantas conclusiones acerca del funcionamiento del sistema como las que se hubiera deseado.

Debido a que es complicado disponer y hacer uso de un mayor conjunto de equipos conectados al sistema para realizar las pruebas oportunas en la realidad, se ha procedido a diseñar un simulador de este sistema real, de forma que nos permita evaluar las prestaciones del sistema ante el uso de un número más elevado de clientes, y ante el uso de una mayor variedad de los parámetros iniciales de las pruebas. En el siguiente capítulo se estudiará con más detalle este simulador y los resultados obtenidos con el mismo.

Como conclusión final, se podría decir que dentro de los límites físicos que nos ha impuesto el número de ordenadores disponibles para las pruebas, el sistema, bajo las condiciones descritas de funcionamiento y el entorno de ejecución empleado, cumple con las características requeridas por un sistema de visualización de terrenos en tiempo real para entornos distribuidos.

Capítulo 6 – Simulador del Sistema

6.1 – Introducción

El sistema diseñado en esta tesis ha sido pensado para poder soportar una elevada carga computacional provocada por la conexión simultánea de un elevado número de clientes dentro de un entorno de red distribuido. Debido a que resulta complicado disponer de un elevado conjunto de equipos para realizar las pruebas del sistema en un entorno real, a que resulta complejo modificar las propiedades de las líneas de conexión empleadas dentro de una red de uso público, y a que el coste temporal y computacional de la realización de las pruebas es elevado, para poder extraer conclusiones sobre el funcionamiento del sistema con un número elevado de clientes conectados en un entorno de red distribuido se ha decidido diseñar un simulador del sistema.

En este capítulo primeramente se va a llevar a cabo el diseño del simulador, donde se van a definir los elementos que van a formar parte del mismo, así como las diferencias y las similitudes de simulador respecto al sistema real. Posteriormente se definirá la metodología que se va a emplear para validar el funcionamiento del simulador y se procederá a la validación del mismo. Una vez validado el simulador se procederá a realizar un conjunto de pruebas con el mismo, para posteriormente hacer un análisis de los resultados y extraer conclusiones de los mismos.

6.2 – Diseño

La arquitectura del simulador consta de los mismos elementos que la arquitectura del sistema real descrita en el Capítulo 5. En la Figura 6.1 se muestran los diferentes módulos del simulador y cómo se interconectan entre sí.

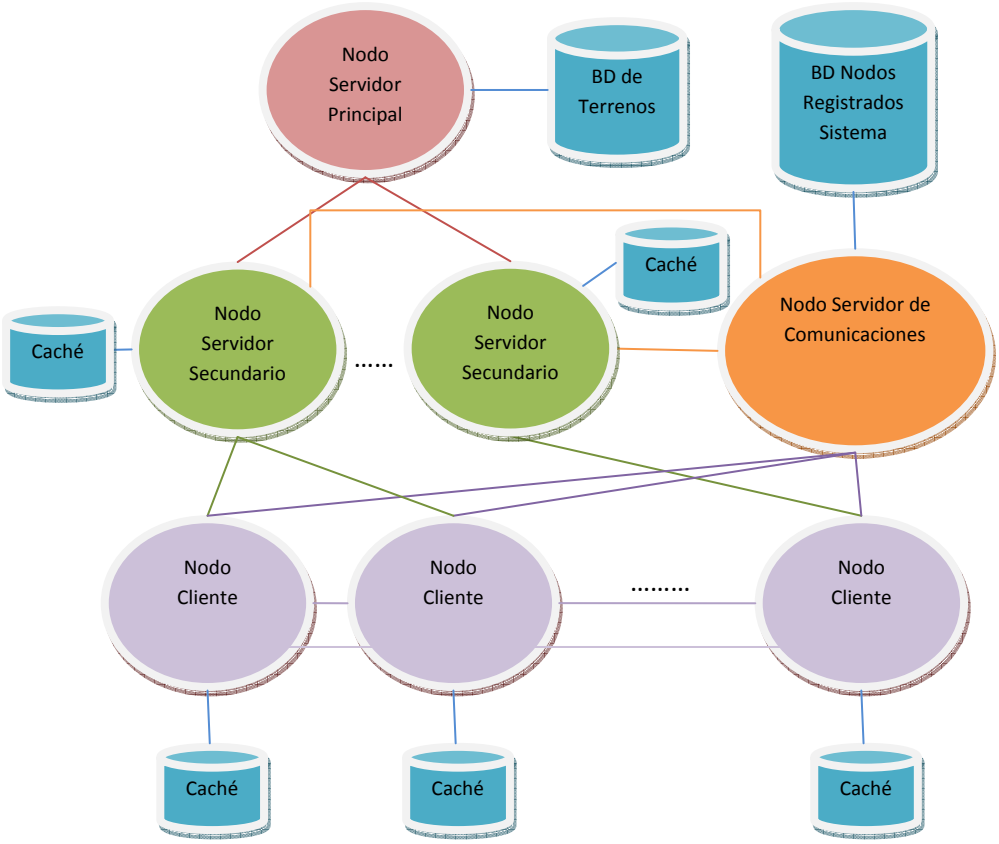


Figura 6.1 – Esquema de la arquitectura del Simulador

6.2.1 – Esquema general

La arquitectura diseñada para el simulador está compuesta por los mismos cuatro tipos de nodos utilizados en el sistema real, los cuales se encuentran definidos con detalle en el apartado 5.1.

Recordemos brevemente cuales eran estos tipos de nodos:

- El nodo servidor principal, que es el único que tiene acceso a toda la información de la base de datos del terreno ofreciendo una disponibilidad continua de toda la información a los servidores secundarios.

- El nodo servidor secundario, que envía la información del terreno a los clientes que la soliciten. Si el servidor secundario no dispone de la información requerida por los clientes, accederá al servidor principal para obtenerla, almacenándola en su caché para usarla en posteriores peticiones de los clientes.

- El nodo cliente, es el nodo encargado de obtener la información del terreno requerida por cada usuario. Para ello realizará peticiones de precintos a sus clientes vecinos y, en el caso de que éstos no dispongan de esa información o no puedan ofrecérsela dentro de unos límites de tiempo adecuados, dirigirán sus peticiones hacia el servidor secundario. Este nodo cliente actuará a su vez como un servidor, recibiendo y sirviendo peticiones de otros clientes, ejerciendo de esa forma el rol típico de un nodo que pertenece a una arquitectura P2P.

- El nodo servidor de comunicaciones, es el nodo encargado de gestionar la topología del sistema. Selecciona para cada cliente los clientes vecinos y el servidor secundario de los que puede obtener la información del terreno. Esta selección se realizará tratando de mantener en todo momento la carga del sistema repartida de forma equilibrada entre los diferentes elementos del sistema.

6.2.2 – Diferencias globales del Simulador respecto del Sistema Real

El funcionamiento y los procesos que llevan a cabo los diferentes elementos que forman parte de la arquitectura empleada en el simulador son los mismos que los del sistema real (Capítulo 5).

Sin embargo, en el simulador, todos los elementos del sistema se van a ejecutar de forma simultánea en el mismo equipo, a diferencia del sistema real, en el que cada uno se ejecutaba en un equipo diferente. Este hecho conlleva una serie de diferencias en el funcionamiento del simulador respecto al del sistema real, que comentamos a continuación.

6.2.2.1 – Tiempo

El funcionamiento del sistema real está condicionado por un factor temporal. Cada elemento del sistema es capaz de llevar a cabo un conjunto de procesos en un tiempo concreto, el cual viene determinado por las características particulares de los equipos y de las líneas de conexión utilizadas. El simulador no tiene esa restricción temporal, por lo que

es necesario establecer algún mecanismo equivalente, que en nuestro caso van a ser las iteraciones y el tiempo de CPU por iteración.

El funcionamiento del simulador se basa en iteraciones. En cada iteración se simulará una vez la ejecución de cada uno de los elementos conectados al sistema. Cada elemento del sistema se compone de un conjunto de procesos. Cada uno de estos procesos tiene asociado un coste temporal de ejecución. Para determinar la cantidad de procesos que puede ejecutar cada elemento del sistema en una iteración, se ha definido un tiempo de CPU por iteración.

Inicialmente en cada iteración, cada elemento del sistema dispone de todo el tiempo de CPU disponible. Este tiempo irá disminuyendo a medida que se ejecuten los procesos asociados a ese elemento del sistema. Cuando ese tiempo de CPU se agote, ese elemento del sistema tendrá que esperar a que se inicie la siguiente iteración para continuar con la ejecución de sus procesos. Cuando se ha realizado una vez la ejecución de todos los elementos conectados al sistema, finalizará la iteración actual y comenzará una nueva.

Los costes temporales asociados a cada uno de los procesos de los diferentes elementos del sistema y el tiempo de CPU por iteración, se extraerán de las medidas empíricas obtenidas de las pruebas realizadas con el sistema real.

6.2.2.2 – Líneas de Conexión

Una de las diferencias fundamentales entre el simulador y el sistema real radica en la ausencia, en el simulador, de las líneas físicas de comunicación entre los diferentes elementos del sistema. En el simulador, estas líneas de transmisión se implementan empleando colas de entrada y de salida de mensajes para cada uno de los elementos del sistema.

En las colas de salida se colocarán los mensajes que los diferentes elementos del sistema envían. Estos mensajes permanecerán en esas colas hasta que transcurra un tiempo de transmisión determinado. Una vez transcurrido este tiempo, el mensaje se eliminará de esa cola de salida y se colocará en la cola de entrada del elemento del sistema al cual el mensaje va dirigido.

Dentro del simulador, para las pruebas donde se evalúa el funcionamiento del mismo, para fijar el tiempo de transmisión de la forma más acorde al estado actual de las líneas de conexión empleadas en Internet, se ha partido de un estudio global de la calidad de la banda ancha [38][39] realizado por la Universidad de Oxford [130] en verano del 2009, en la que ha colaborado la Universidad de Oviedo [129]. En este estudio se han analizado 24

millones de transmisiones, en las que se ha calculado la velocidad media de descarga, la velocidad media de subida y la latencia media para las líneas de banda ancha de un conjunto diversos de países de todo el mundo.

En el simulador se ha optado por tomar como tiempo medio de transmisión, la latencia media obtenida en el estudio, que se corresponde con la latencia media de las líneas de banda ancha empleadas en Alemania (100 milisegundos).

Por otro lado, en las líneas de conexión suelen aparecer problemas de comunicación que en un sistema real provocan que se pierdan precintos o que éstos lleguen fuera del tiempo permitido a su destino y sea necesario volverlos a pedir. En el estudio mencionado anteriormente no se hace referencia a estos problemas. Por ello, en el simulador se opta por simular este problema, descartando mensajes de forma aleatoria, manteniendo un porcentaje similar al porcentaje de error obtenido en las pruebas realizadas con el sistema real.

6.2.2.3 – Base de datos

La base de datos con la información del terreno se divide por un lado en la información de las texturas y por otro lado en la de alturas de la superficie del terreno. La base de datos empleada en el sistema real tenía un tamaño reducido (apartado 5.3.2), para mantener una densidad espacial (en el espacio del terreno) aceptable, dado que el número de clientes que se utilizaban era reducido como consecuencia de la limitación que se tenía a la hora de conseguir máquinas que actuaran como tales clientes conectados al sistema real. En el simulador, esta limitación física no existe, y se podrá emular un mayor número de clientes conectados al sistema. Se utilizará, por ello, una base de datos mayor para mantener una densidad de usuarios en la superficie del terreno similar a la que había en el sistema real. En concreto, se multiplicará por 16 la extensión de la base de datos empleada, llegando a abarcar una superficie mayor de 1300 km x 1300 km.

Por otro lado, para simular el intercambio de los datos del terreno entre los elementos del sistema, no es necesario emplear información real. Por ello, para simplificar el funcionamiento del simulador, en su lugar, se emplearán únicamente los índices que hacen referencia a la información incluida dentro del quadtree que almacena toda la información descargada, los cuales se calcularán y se almacenarán en los diferentes elementos del sistema como si fueran los datos reales.

Por último, en el simulador, para realizar la transferencia de la información entre los diferentes elementos del sistema, se asigna un tiempo de transmisión a los mensajes de

envío que afecta al tiempo que permanecen los mensajes en la cola de salida de mensajes antes de ser traspasados a la cola de mensajes de entrada del elemento destino del mensaje, tal y como se ha comentado en el apartado 6.2.2.2.

6.2.2.4 – Nodo Servidor Principal

El servidor principal es el encargado de almacenar toda la base de datos del terreno. Como se ha comentado en el apartado 6.2.2.3, en el simulador no se va a emplear una base de datos real, únicamente se va a hacer referencia a los datos a través del índice del quadtree que los determina en la escena.

Por ello, la función de este servidor en el simulador se limita simplemente a responder a los servidores secundarios los valores de estos índices transcurrido un tiempo de espera determinado. El tiempo medio de espera será extraído de las medidas empíricas obtenidas de las pruebas con el sistema real. El servidor principal empleará las colas de mensajes descritas en el apartado 6.2.2.2 para llevar a cabo el intercambio de información con los servidores secundarios.

6.2.2.5 – Nodo Servidor Secundario

El servidor secundario es el único tipo de nodo que puede acceder al servidor principal, actuando como intermediario entre el servidor principal y los clientes del sistema en el intercambio de la información del terreno. En el simulador, el servidor secundario empleará las colas de mensajes descritas en el apartado 6.2.2.2 para comunicarse con el resto de elementos del sistema.

El funcionamiento del servidor secundario es el siguiente:

- Recibe peticiones de precintos por parte de los clientes.

- Comprueba si es esos precintos se encuentran almacenados en su caché, en cuyo caso los devolverá de manera inmediata.

- En caso contrario, pide al servidor principal estos precintos, retrasando su envío al cliente hasta que se reciban del servidor principal.

En el simulador, los mensajes de respuesta de los precintos hacia los clientes esperarán en la cola de salida un tiempo diferente dependiendo de si se encuentran en la caché del servidor secundario o no. Los tiempos medios de espera en ambos casos se extraerán de las medidas empíricas obtenidas de las pruebas realizadas con el sistema real.

Al igual que en el sistema real, en el simulador, el servidor secundario utiliza dos cachés, una para almacenar la información de las alturas del terreno y otra para almacenar la información de las texturas. Como se ha comentado en el apartado 6.2.2.3, la información que se almacena en estas cachés es el índice que identifica a esa información dentro del quadtree.

Estas cachés pueden tener un tamaño determinado. Para especificarlo, se utilizará el número de precintos que se pueden almacenar en la caché, en vez de utilizar el tamaño en bytes, como se empleaba en el sistema real.

El servidor secundario se comunica periódicamente con el servidor de comunicaciones para informarle acerca de su estado con el objetivo de que éste pueda balancear la carga del sistema adecuadamente entre los diferentes servidores secundarios. En el simulador, esta periodicidad se define en función de un número predeterminado de iteraciones. Para realizar estas comunicaciones periódicas se emplearán las colas de mensajes descritas en el apartado 6.2.2.2.

6.2.2.6 – Nodo Servidor de Comunicaciones

Este nodo se encarga de gestionar la topología de la red P2P. Al igual que en el sistema real, en el simulador tanto los clientes como los servidores secundarios se conectan inicialmente con él para registrarse en el sistema.

Los servidores secundarios se conectan periódicamente con el servidor de comunicaciones para ofrecerle información actualizada sobre su estado. Los clientes también se van a conectar de forma periódica con el servidor de comunicaciones para obtener los clientes vecinos y el servidor secundario más adecuados en cada momento de los que podrá obtener la información del terreno. En el simulador, este periodo de tiempo se define en función de un número predeterminado de iteraciones.

Además, en el simulador se emplearán las colas de mensajes descritas en el apartado 6.2.2.2 para realizar estas comunicaciones del servidor de comunicaciones con los clientes y con los servidores secundarios.

6.2.2.7 – Nodo Cliente

Este nodo realiza todos los procesos necesarios para permitir al usuario llevar a cabo la visualización de los datos del terreno. En el simulador, el único proceso que no se llevará a cabo respecto a los que se ejecutan en el sistema real, será el de dibujado, debido a que no es factible realizar este proceso de forma simultánea en un mismo equipo para todos los clientes conectados al sistema.

En el simulador, los clientes emularán el desplazamiento de los usuarios por la escena empleando rutas que siguen el mismo tipo de distribuciones que las que se han empleado en el sistema real. Siguiendo estas rutas, realizarán todo el proceso de cálculo de los precintos necesarios para la visualización de la escena, determinando qué precintos necesitan descargar y cuales ya tienen disponibles en su caché.

Los precintos que cada cliente requiera serán pedidos en primer lugar a sus clientes vecinos. Si ninguno de sus clientes vecinos dispone de esa información o ninguno de ellos responde dentro de un tiempo aceptable, los precintos serán pedidos al servidor secundario que tenga asignado.

En el simulador, los clientes emplearán las colas de mensajes descritas en el apartado 6.2.2.2 para realizar la comunicación con los clientes y con el servidor secundario. Además, también emplearán estas colas para comunicarse periódicamente con el servidor de comunicaciones para obtener con ello, la lista de los clientes vecinos y el servidor secundario más adecuado en cada momento.

Al igual que en el sistema real, en el simulador cada cliente emplea una caché para almacenar la información del terreno que obtiene de los clientes vecinos y del servidor secundario. Cada cliente dispondrá de dos cachés, una para almacenar la información de las alturas del terreno y otra para almacenar la información de las texturas. Como se ha comentado en el apartado 6.2.2.3, en el simulador la información que se almacena en estas cachés es el índice que identifica a esa información dentro del quadtree.

Estas cachés pueden tener un tamaño predeterminado. En el simulador, se utilizará el número de precintos que se pueden almacenar en la caché para especificar ese tamaño, en vez de utilizar el tamaño en bytes, como se empleaba en el sistema real.

6.3 – Metodología de Evaluación

La metodología que se va a emplear para evaluar el sistema utilizando el simulador es similar a la descrita para el sistema real (apartado 5.3). En este apartado se va a realizar un repaso de esa metodología, centrándonos en las diferencias existentes.

6.3.1 – Parámetros de las pruebas

Igual que en el capítulo anterior, para evaluar el simulador se ha planteado un modelo de evaluación de prestaciones atendiendo a la variación de diversos parámetros fundamentales. La mayor parte de estos parámetros son los mismos que los utilizados para evaluar el sistema real, los cuales fueron descritos en el apartado 5.3.1. A estos parámetros se les añadirá un nuevo conjunto de ellos que será necesario configurar para el caso del simulador.

6.3.1.1 – Parámetros comunes al sistema real

Primeramente se van a enunciar los parámetros empleados que son comunes a los utilizados para evaluar el sistema real. Una descripción más detallada se puede encontrar en el apartado 5.3.1:

- **Número de clientes conectados**. Indica el número de clientes que se van a conectar al sistema.

- **Número de servidores secundarios conectados**. Indica el número de servidores secundarios conectados al sistema.

- **Ruta**. Establece la trayectoria que va a recorrer un cliente dentro de la escena durante las pruebas.

- **Tamaño de la caché**. Indica la cantidad de información medida en precintos que se puede almacenar en una caché. Se especificará un tamaño para la información de las texturas del terreno y otro para la de las alturas.

- **Tamaño de la lista de clientes vecinos**. Indica el número máximo de clientes vecinos a los cuales el cliente puede solicitar información.

6.3.1.2 – Parámetros propios del simulador

6.3.1.2.1 – Parámetros globales

Para configurar el simulador inicialmente, son necesarios una serie de parámetros que son extraídos en su mayor parte de las medidas empíricas obtenidas de las pruebas llevadas a cabo sobre el sistema real, y que son los siguientes:

- **El tiempo de CPU de cada elemento del sistema**. Indica el tiempo de que dispone cada elemento del sistema para realizar los procesos asociados a su ejecución.

En el sistema real, cada elemento del mismo se ejecuta en un equipo diferente, de forma que todo el tiempo de CPU del equipo se dedica a ese elemento. En el simulador, todos los elementos del sistema se ejecutan en la misma máquina. Por lo tanto, el tiempo de CPU de la máquina ha de repartirse entre todos ellos.

Sin embargo, no resulta adecuado repartir el tiempo de CPU por igual entre los diferentes elementos del sistema, ya que en un entorno real, es habitual que las diferentes maquinas utilizadas tengan una potencia computacional diferente. Por ello, con este parámetro, se puede especificar un tiempo de CPU diferente para cada elemento del sistema. Este tiempo se extraerá de las pruebas realizadas con el sistema real, observando el tiempo que emplea cada elemento del sistema en procesar el conjunto de procesos que lo integran.

- **El tiempo medio para la transmisión de un precinto**. Indica el tiempo necesario para realizar la transmisión de un precinto entre dos elementos del sistema a través de una línea de conexión.

En el sistema real, la transmisión de un mensaje a través de la línea de conexión tardará un determinado tiempo dependiendo del tipo de la línea de conexión empleada y de su estado. Como en el simulador no se dispone de unas líneas de conexión reales, habrá que asignar un tiempo de transmisión predeterminado que simule el retardo en la transmisión de esa información.

Este tiempo de transmisión del precinto se calculará de forma aleatoria empleando una distribución normal centrada en un tiempo medio que será especificado como un parámetro inicial del simulador, y utilizando una dispersión que también se especificará como un parámetro inicial.

A la hora de fijar el valor de estos dos parámetros: el tiempo medio y la dispersión, como se ha comentado en el apartado 6.2.2.2, se empleará el estudio global de la calidad de banda ancha [38][39] realizado por la Universidad de Oxford.

- **El porcentaje de precintos vueltos a pedir**. Indica el porcentaje de precintos que se van a tener que volver a pedir en la prueba, simulando los problemas en las líneas de conexión o en los clientes que se producen en el sistema real.

En el sistema real, el porcentaje de precintos que es necesario volver a pedir debido a que, bien no llegan a su destino, o bien lo hacen fuera de un tiempo aceptable, es consecuencia, a priori, a la pérdida o retraso de los mensajes, o a problemas puntuales de los clientes, que les impiden devolver la información dentro de unos límites de tiempo predeterminados. Este tipo de problemas no se van a producir en el simulador, ya que las líneas de conexión son simuladas.

Por ello, se introduce este parámetro en el simulador con el fin de emular este problema. Como se comentó en el apartado 6.2.2.2, cuando un mensaje llega a una de las colas de entrada que simulan las líneas de conexión, se procede a decidir de forma aleatoria si ese mensaje se descarta, emulando con ello la pérdida o el retraso excesivo del mismo. El porcentaje de precintos que se descartará se ajustará al valor de este parámetro, el cual se extraerá empíricamente de las pruebas realizadas en el sistema real.

6.3.1.2.2 – Parámetros particulares del servidor principal

Como se ha comentado en el apartado 6.2.2.4, la función de este servidor se limita a responder las peticiones de precintos realizadas por los servidores secundarios dentro de un tiempo determinado. Este tiempo será el único parámetro que se especificará inicialmente para este servidor:

- **El tiempo medio necesario para realizar el proceso de descarga de un precinto desde el servidor principal**. Indica el tiempo que requiere el servidor principal para realizar el proceso de búsqueda de un precinto en la base de datos, su extracción y su preparación para el envío. Dentro de este tiempo no se incluye el tiempo de transmisión, que se especifica como un parámetro aparte (apartado 6.3.1.2.1).

Este tiempo se calculará de forma aleatoria empleando una distribución normal centrada en un tiempo medio, que será especificado como un parámetro inicial del simulador, y utilizando una dispersión que también se especificará como un parámetro inicial. De esta forma se puede obtener un tiempo de descarga diferente para cada petición realizada al servidor principal, intentando simular el proceso tal como ocurre en la realidad, donde el tiempo de respuesta se ve afectado por las condiciones particulares del equipo en el que se ejecuta el servidor.

6.3.1.2.3 – Parámetros particulares del servidor secundario

El servidor secundario se va a encargar de ofrecer la información del terreno a los clientes. Los parámetros iniciales que se aportarán para la configuración en el simulador del servidor secundario son los siguientes:

- El tiempo medio necesario para realizar el proceso de descarga de un precinto cuando se encuentra disponible en la caché del servidor secundario. Indica el tiempo que requiere el servidor secundario para realizar el proceso de búsqueda de un precinto en su caché, su extracción y su preparación para el envío. Dentro de este tiempo no se incluye el tiempo de transmisión.

- El tiempo medio necesario para realizar el proceso de descarga de un precinto cuando no se encuentra disponible en la caché del servidor secundario. Indica el tiempo que requiere el servidor secundario para realizar el proceso de búsqueda de un precinto en su caché, la petición del mismo al servidor principal, la espera a recibirlo y su preparación para el envío. Dentro de este tiempo no se incluye el tiempo de transmisión.

Estos dos tiempos de descarga se calcularán de forma aleatoria empleando una distribución normal centrada en unos tiempos medios que serán especificados como unos parámetros iniciales del simulador, y utilizando unas dispersiones que también se especificarán como unos parámetros iniciales. De esta forma se pueden obtener unos tiempos de descarga diferentes para cada petición realizada a los servidores secundarios, intentando simular el proceso tal como ocurre en la realidad, donde el tiempo de respuesta se ve afectado por las condiciones particulares de los equipos en los que se ejecutan los servidores.

- El tamaño de la caché empleada para almacenar la información de las texturas y de las alturas. Indica la cantidad de información medida en precintos que puede almacenar el servidor secundario en su caché.

Como se comentó en el apartado 6.2.2.3, no se van a almacenar en la caché datos reales. En su lugar se almacena el índice que referencia a los datos en el quadtree. Por ello, este tamaño se define en función del número de precintos que se puede almacenar como máximo en la caché. Este número de precintos se especificará por separado para la información de las texturas y para la de las alturas.

6.3.1.2.4 – Parámetros particulares del servidor de comunicaciones

El servidor de comunicaciones se encarga de gestionar la topología de la arquitectura empleada en el sistema. Para ello asignará, a cada cliente, el servidor secundario y los clientes vecinos más adecuados en cada instante.

Como parámetro inicial para la configuración del servidor de comunicaciones tenemos:

- El tiempo medio necesario para realizar el proceso de envío de los clientes vecinos y el servidor secundario al cliente. Indica el tiempo que requiere el servidor de comunicaciones para realizar el proceso de selección y envío, a un cliente, de los clientes vecinos y del servidor secundario de los que podrá descargar la información del terreno. Dentro de este tiempo no se incluye el tiempo de transmisión.

Este tiempo de envío se calculará de forma aleatoria empleando una distribución normal centrada en un tiempo medio que será especificado como un parámetro inicial del simulador, y utilizando una dispersión que también se especificará como un parámetro inicial del mismo. De esta forma se puede obtener un tiempo de envío diferente para cada petición realizada al servidor de comunicaciones, intentando simular el proceso tal como ocurre en la realidad, donde el tiempo de respuesta se ve afectado por las condiciones particulares del equipo en el que se ejecuta el servidor.

6.3.1.3 – Parámetros particulares del cliente

El cliente es el encargado de obtener la información que será requerida en cada instante para la visualización de la superficie del terreno. Los parámetros iniciales para la configuración del cliente son los siguientes:

- El tiempo medio necesario para realizar el proceso de descarga de un precinto desde un cliente. Indica el tiempo que requiere el cliente para realizar el proceso de búsqueda de un precinto en su caché, su extracción y su preparación para el envío. Dentro de este tiempo no se incluye el tiempo de transmisión.

Este tiempo se calculará de forma aleatoria empleando una distribución normal centrada en un tiempo medio que será especificado como un parámetro inicial del simulador, y utilizando una dispersión que también se especificará como un parámetro inicial. De esta forma se puede obtener un tiempo de envío diferente para cada petición realizada a los clientes, intentando simular el proceso tal como ocurre en la realidad, donde el tiempo de

respuesta se ve afectado por las condiciones particulares del equipo en el que se ejecuta el cliente.

- El tamaño de la caché empleada para almacenar la información de las texturas y de las alturas. Indica la cantidad de información medida en precintos que puede almacenar el cliente en su caché.

Como se comentó en el apartado 6.2.2.3, no se van a almacenar en la caché datos reales. En su lugar se almacena el índice que referencia a los datos en el quadtree. Por ello, este tamaño se define en función del número de precintos que se puede almacenar como máximo en la caché. Este número de precintos se especificará de forma separada para la información de las texturas y para la de las alturas.

6.3.2 – Distribuciones

El simulador empleará el mismo conjunto de distribuciones que se usaban en el sistema real para obtener los puntos intermedios por los que transcurre la ruta que describen los usuarios mientras se mueven por la escena. Las distribuciones empleadas están definidas con detalle en el apartado 5.3.3 y son: la distribución uniforme, la distribución normal con un punto caliente y la distribución normal con varios puntos calientes.

6.3.3 – Métricas

Los valores medidos para la evaluación del simulador serán prácticamente los mismos que los que se emplearon en la evaluación del sistema real. La única diferencia radica en que no se va a realizar el cálculo de los precintos vueltos a pedir (pvp), que tal como se comentó en el apartado 6.2.2.2, no se puede medir en el simulador puesto que este valor está relacionado con las condiciones particulares que se producen en las líneas de conexión y en los equipos en un entorno real. Por ello, en el simulador se va a introducir ese error como un parámetro inicial de cada prueba, el cual se obtendrá empíricamente de las pruebas realizadas en el sistema real.

El resto de valores medidos se encuentran detallados en el apartado 5.3.4. A continuación nos vamos a limitar a enunciarlos:

- **latPreClientes**: es el valor de la latencia media de descarga de un precinto para un cliente cuando los precintos del terreno han sido obtenidos de sus clientes vecinos.

- **latPreServidores**: es el valor de la latencia media de descarga de un precinto para un cliente cuando los precintos del terreno han sido obtenidos finalmente de los servidores secundarios, midiendo el tiempo desde que se inicia la petición del precinto por parte del cliente, hasta que el precinto es recibido finalmente por éste.

- **latPreGlobal**: es el valor de la latencia media global de descarga de un precinto para un cliente, teniendo en cuenta tanto los precintos del terreno que el cliente obtiene de sus clientes vecinos como de los servidores secundarios.

- **prSS (precintos respondidos por el servidor secundario)**: es el porcentaje de precintos del terreno pedidos por los clientes que han sido respondidos por los servidores secundarios.

- **pESS (precintos encontrados con éxito en la caché del servidor secundario)**: es el porcentaje de precintos del terreno pedidos por los clientes que ya se encuentran en la caché de los servidores secundarios, por lo que no se ha tenido que acudir al servidor principal a por ellos.

- **nColaCL (número de precintos medio en la cola del cliente)**: es el número medio de precintos que se encuentran en la cola del cliente esperando a ser procesados.

- **nColaSS (número de precintos medio en la cola del servidor secundario)**: es el número medio de precintos que se encuentran en la cola del servidor secundario esperando a ser procesados.

6.4 – Resultados

En este apartado se van a mostrar los resultados de las pruebas realizadas para validar el simulador y para evaluar el sistema usando el simulador bajo condiciones que no pudieron probarse en el sistema real.

6.4.1 – Validación del simulador

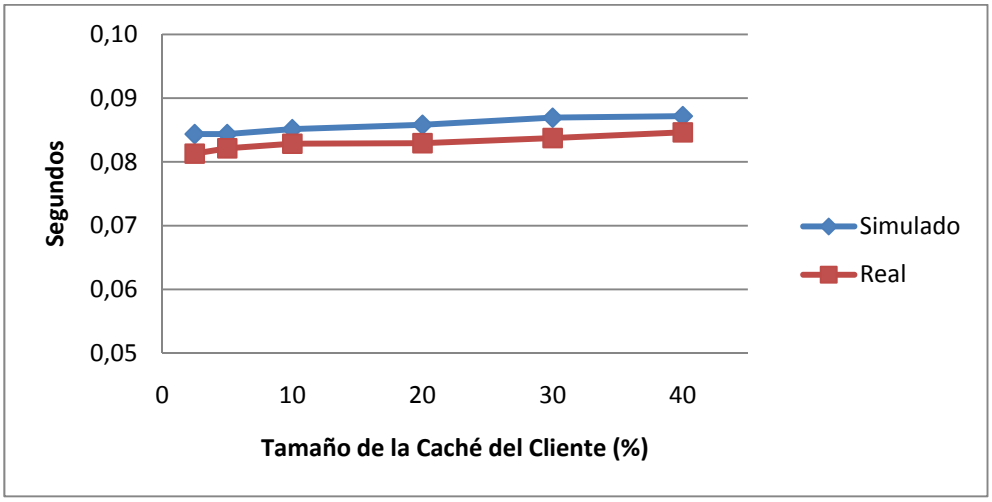
Se va a comprobar que el simulador se comporta de la misma manera que el sistema real bajo las mismas condiciones (número de clientes conectados, número de servidores conectados, etc.), de forma que los resultados obtenidos con el mismo, bajo unas condiciones distintas, se puedan extrapolar al sistema real.

Para llevar a cabo la validación del simulador se han realizado con él las mismas pruebas que se han llevado a cabo con el sistema real, empleando los mismos parámetros iniciales. Con los resultados obtenidos, se puede comparar el comportamiento del simulador frente al del sistema real.

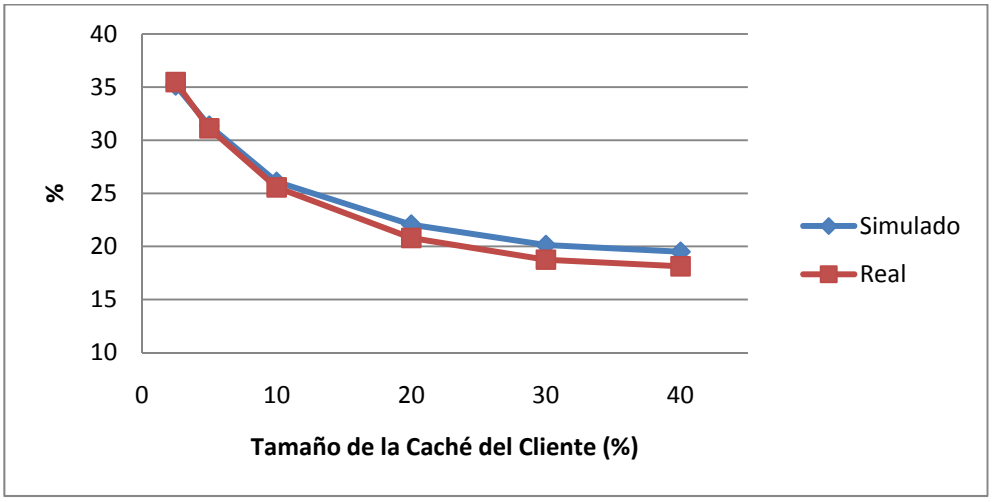
En este apartado sólo se incluyen los resultados obtenidos en las pruebas donde se variaba el tamaño de la caché empleada por los clientes. Pensamos que estas pruebas son unas de las más representativas, ya que la variación del tamaño de la caché del cliente afecta significativamente al funcionamiento de todo el sistema. Los detalles de esta prueba se explicaron en el apartado 5.4.1. Como para el resto de las pruebas realizadas se obtienen unos resultados similares, tanto en el caso del simulador como en el del sistema real, se ha creído conveniente no incluirlos dentro de este apartado, y en su lugar incluirlos en el apéndice A de la memoria.

6.4.1.1– Variación en el tamaño de la caché de los clientes

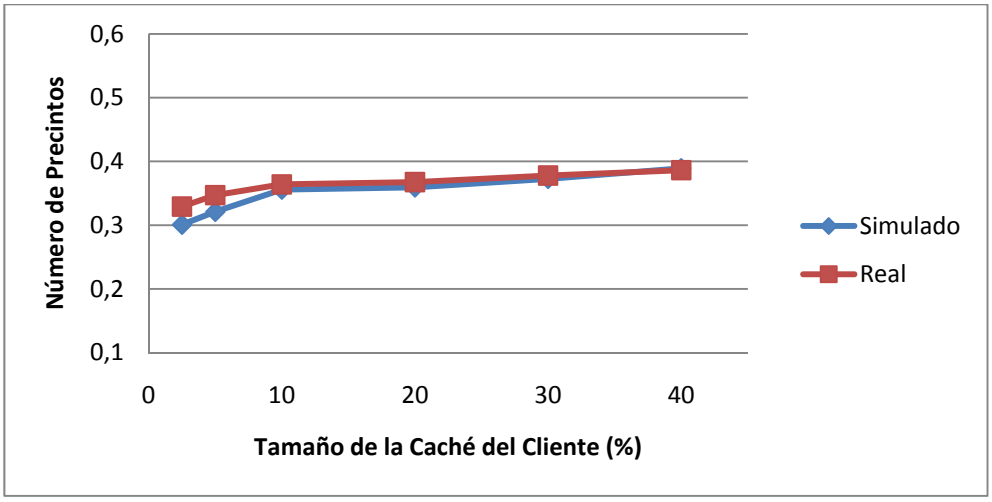
6.4.1.1.1 – Clientes



Gráfica 6.1 – Comparativa de la latencia media de los precintos intercambiados entre los Clientes

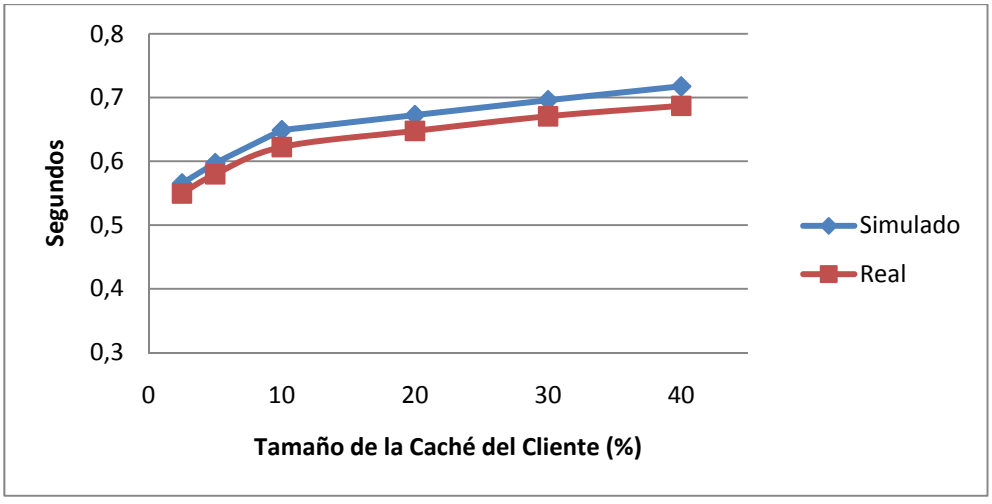


Gráfica 6.2 – Comparativa del porcentaje medio de precintos respondidos por los Servidores Secundarios

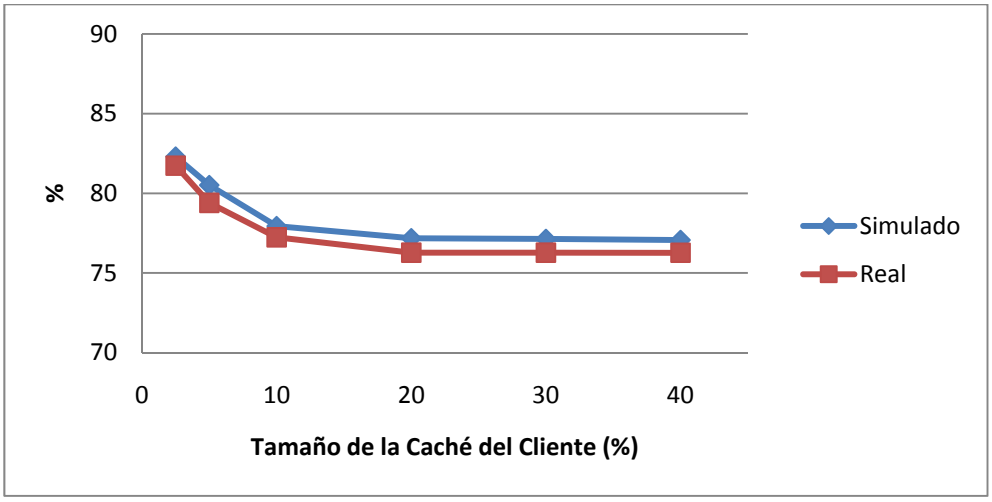


Gráfica 6.3 – Comparativa del número de precintos medio de las colas de los Clientes

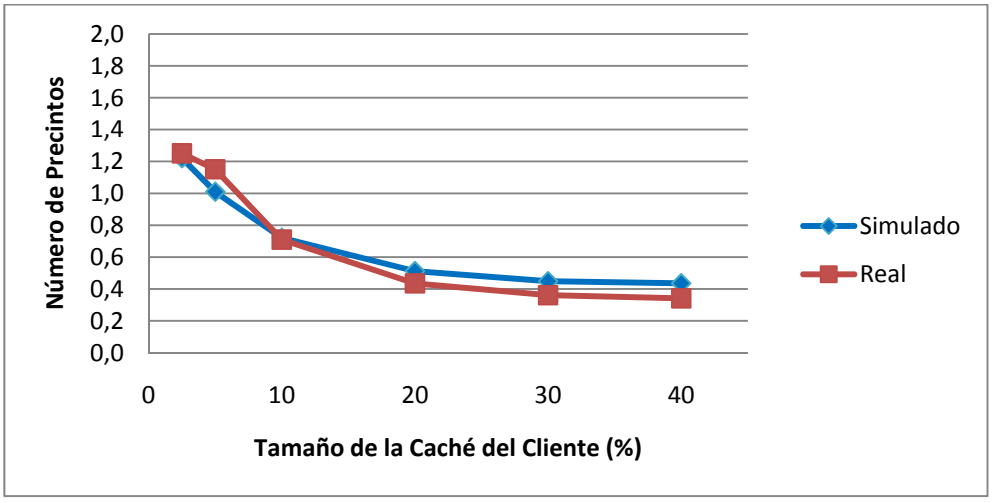
6.4.1.1.2 – Servidores Secundarios



Gráfica 6.4 – Comparativa de la latencia media de los precintos respondidos por los Servidores Secundarios

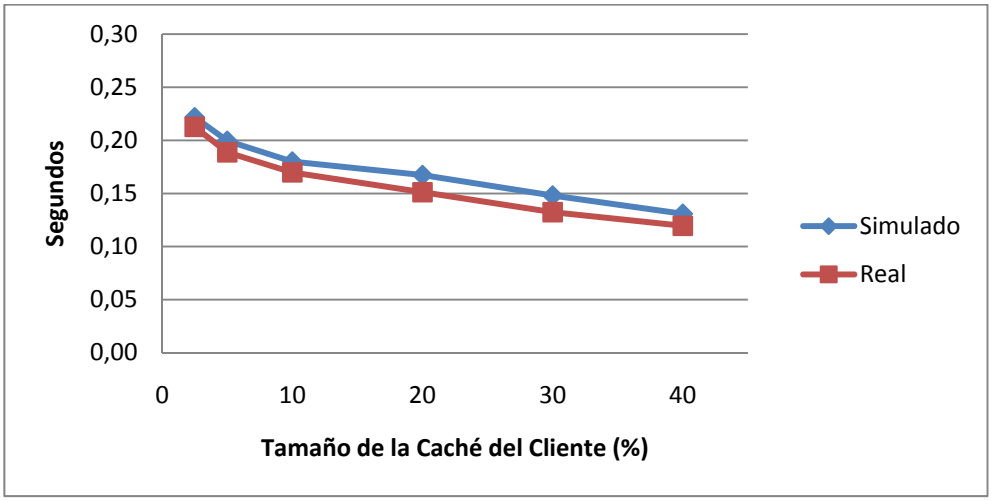


Gráfica 6.5 – Comparativa del porcentaje medio de los precintos encontrados con éxito en la Caché de los Servidores Secundarios

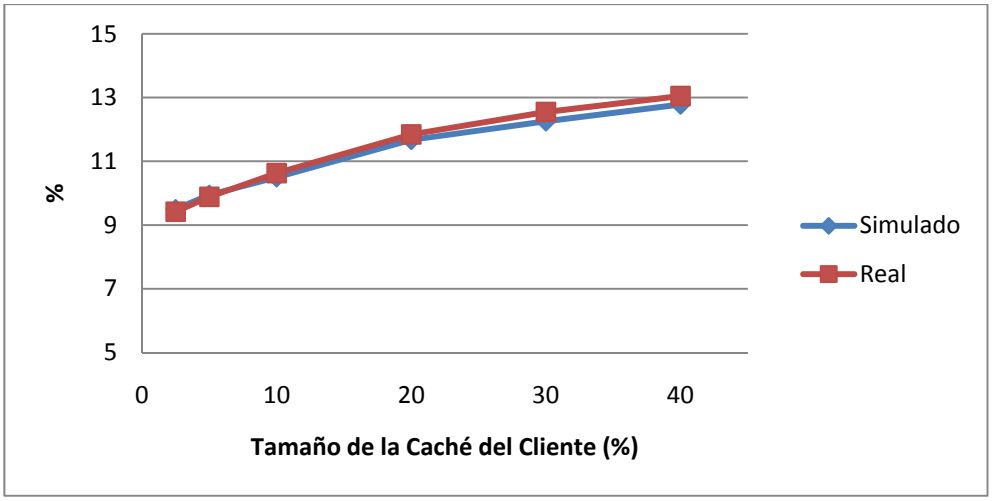


Gráfica 6.6 – Comparativa del número de precintos medio de las colas de los Servidores Secundarios

6.4.1.1.3 – Sistema



Gráfica 6.7 – Comparativa de la latencia media global del intercambio de precintos en el Sistema



Gráfica 6.8 – Comparativa del porcentaje medio de precintos vueltos a pedir

6.4.1.2 – Conclusiones

Las gráficas del apartado 6.4.1 y del apéndice A muestran las comparativas de los resultados obtenidos en las pruebas realizadas en el simulador y en las pruebas realizadas con el sistema real. En ambas pruebas se han empleado el mismo número de clientes y una base de datos del terreno equivalente de forma que los resultados obtenidos fueran equiparables.

Los parámetros adicionales que han sido empleados en el simulador, como por ejemplo los tiempos de transmisión y los tiempos de descarga de los diferentes elementos del sistema, se han extraído de forma empírica de las pruebas del sistema real, intentando que las condiciones de las pruebas fueran lo más parecidas posibles.

Como se puede observar en las gráficas, los resultados obtenidos con el simulador son parecidos a los que se obtuvieron con el sistema real. Si bien en ocasiones los valores numéricos difieren en algunas pruebas, el comportamiento observado en las gráficas es el mismo, y es debido a las mismas razones que se daban en el sistema real.

De acuerdo a todo esto, se puede decir que el simulador parece que se comporta de forma parecida a como lo hace el sistema real, y por lo tanto, parece razonable pensar que los nuevos resultados que se obtengan con el simulador, podrían ser extrapolados al funcionamiento real del sistema.

6.4.2 – Nuevas pruebas realizadas con el simulador

Una vez que se ha validado el funcionamiento del simulador, se va a proceder a realizar un nuevo conjunto de pruebas con el objetivo de obtener resultados adicionales a los calculados con el sistema real, que nos permitan conocer mejor el comportamiento del sistema bajo condiciones que no pudieron probarse en el sistema real por diversos motivos.

6.4.2.1 – Cambios respecto al sistema real

Una de las principales limitaciones que se tenía a la hora de probar el sistema real era disponer de un elevado número de clientes conectados simultáneamente, ya que no se disponía físicamente de los equipos para llevar a cabo esta prueba. Empleando el simulador, se van a poder realizar pruebas con un número más elevado de clientes.

Ante el uso de un mayor número de clientes, para mantener una densidad parecida de usuarios sobre la superficie del terreno a la empleada en el sistema real, tal como se especificó en el apartado 6.2.2.3, se va utilizar una base de datos del terreno de mayor tamaño, que abarcará una superficie mayor de 1300km x 1300 km.

Otra de las limitaciones a la hora de probar el sistema en un entorno real era la ubicación de las máquinas empleadas, las cuales estaban conectadas a la red local de la Universidad de Valencia. Con el simulador se podrán establecer las condiciones de las líneas de conexión que se deseen. En particular, como se ha comentado en el apartado 6.2.2.2, para especificar las condiciones de las líneas de conexión se empleará el estudio global de la calidad de la banda ancha [38][39] realizado por la Universidad de Oxford.

6.4.2.2 – Escalabilidad del sistema

Uno de los objetivos propuestos a la hora de realizar el diseño de la nueva arquitectura mixta cliente-servidor / P2P desarrollada en esta tesis era que fuera escalable con el número de clientes conectados al sistema.

En las pruebas desarrolladas con el sistema real, parecía que éste sí que era escalable, si bien, estas pruebas sólo se habían podido realizar con un número reducido de clientes.

Con la ayuda del simulador se va a probar el sistema con un número mayor de clientes conectados, con el objetivo de conocer la capacidad de escalabilidad del mismo en función de este número.

Se han llevado a cabo para ello varias pruebas cuyos parámetros de configuración son los siguientes:

Número de servidores secundarios conectados = 2 (al menos son necesarios dos para que haya una distribución de la carga del sistema entre ellos).

Ruta: Distribución uniforme (toda la superficie del terreno tiene el mismo nivel de interés).

Tamaño de la caché del servidor secundario = 50% (se reparte la base de datos entre los dos servidores secundarios).

Tamaño de la caché del cliente = 10% (consideramos este valor como aceptable dato el tamaño de la base de datos utilizada en la prueba).

Tamaño de la lista de clientes vecinos = 10 (se mantiene este número de vecinos puesto que se consideró un número adecuado en las pruebas realizas con el sistema real).

Porcentaje de precintos que vuelven a ser pedidos = alrededor del 11% (este valor ha sido extraído de las pruebas reales como media de los precintos que deben ser pedidos de nuevo debido a que se han perdido en el envío o han tardado más del tiempo permitido en llegar a su destino).

Las pruebas realizadas han sido:

Prueba_A: Número de clientes conectados = 10.

Prueba_B: Número de clientes conectados = 25.

Prueba_C: Número de clientes conectados = 50.

Prueba_D: Número de clientes conectados = 100.

Prueba_E: Número de clientes conectados = 200.

Prueba_F: Número de clientes conectados = 300.

Prueba_G: Número de clientes conectados = 400.

Prueba_H: Número de clientes conectados = 500.

Prueba_I: Número de clientes conectados = 600.

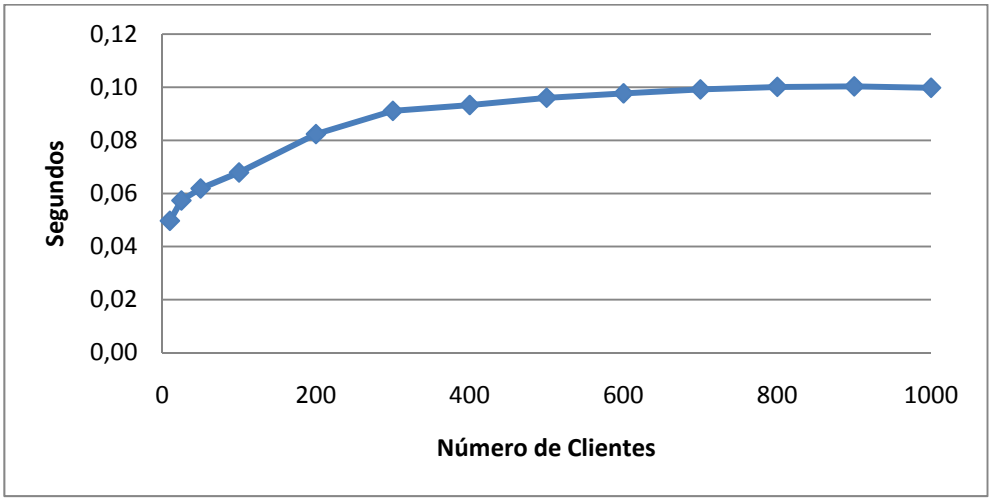
Prueba_J: Número de clientes conectados = 700.

Prueba_K: Número de clientes conectados = 800.

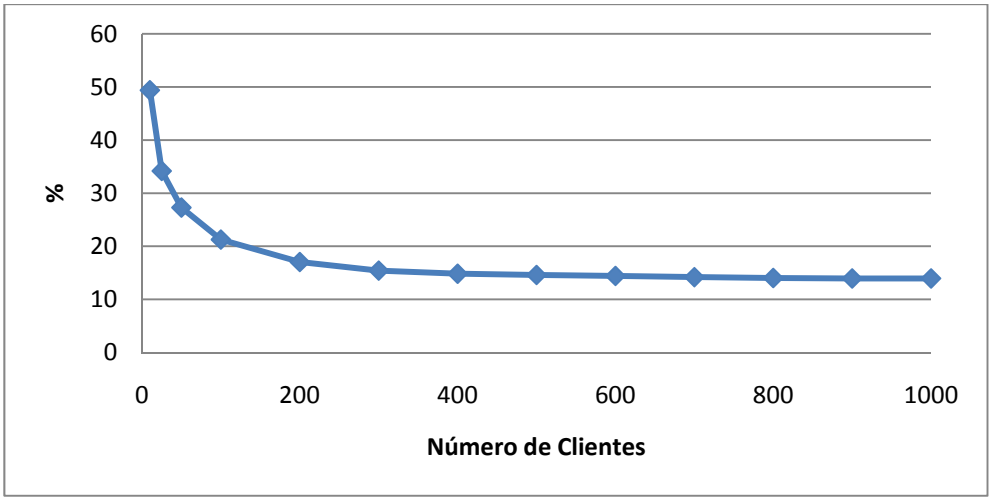
Prueba_L: Número de clientes conectados = 900.

Prueba_M: Número de clientes conectados = 1000.

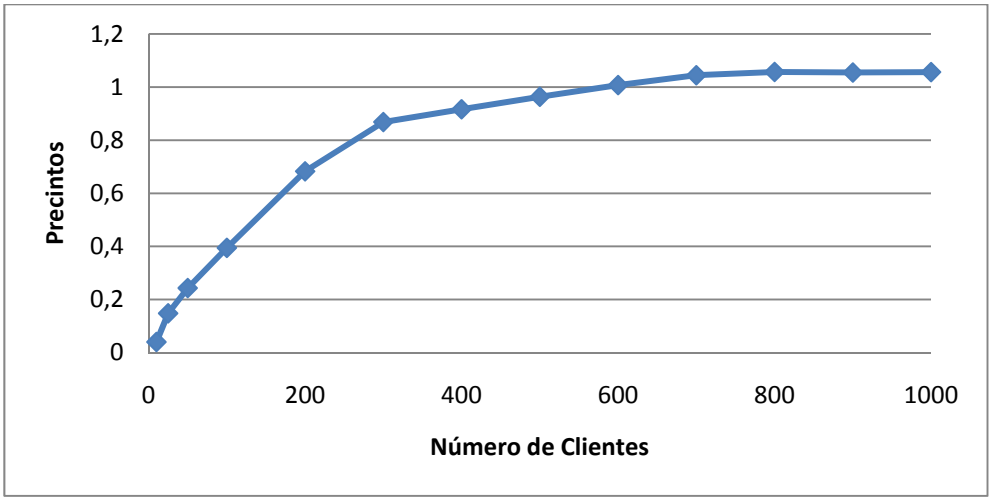
6.4.2.2.1 – Clientes



Gráfica 6.9 – Latencia media de los precintos intercambiados entre los Clientes



Gráfica 6.10 – Porcentaje medio de precintos respondidos por los Servidores Secundarios



Gráfica 6.11 – Número de precintos medio en las colas de los Clientes

En las gráficas anteriores se muestra la influencia que tiene en los propios clientes la cantidad de clientes conectados al sistema.

En la Gráfica 6.9 se muestra la latencia media de los precintos respondidos por los clientes. En ella se observa como el incremento del número de los clientes conectados al sistema provoca un ligero aumento de la latencia. Sin embargo, hay que hacer notar que este incremento cada vez es menor, y a partir de 800 clientes conectados parece que la latencia más o menos se estabiliza.

En la Gráfica 6.10 se muestra el porcentaje medio de peticiones que realizan los clientes a los servidores secundarios. En ella se observa como este porcentaje disminuye conforme aumenta el número de clientes conectado al sistema, si bien, a partir de 400 clientes el porcentaje disminuye ligeramente, y a partir de 800 clientes parece que más o menos se estabiliza.

En la Gráfica 6.11 se muestra el número medio de precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por los clientes vecinos. En ella se observa como el incremento del número de clientes conectados al sistema provoca un aumento del tamaño de esta cola, por lo que aumenta su carga, sin embargo a medida que hay un mayor número de clientes conectados este aumento es menor, y a partir de 800 clientes el tamaño de la cola parece que más o menos se estabiliza.

Conforme aumenta el número de clientes conectados al sistema, la información almacenada en la base de datos se distribuirá en mayor medida entre los clientes. De esta forma los clientes podrán obtener una mayor cantidad de información de sus clientes vecinos, evitando tener que acudir a los servidores secundarios. Este hecho provoca que el porcentaje de peticiones realizadas a los servidores secundarios por parte de los clientes disminuya al aumentar el número de clientes conectados al sistema, tal como se muestra en la Gráfica 6.10.

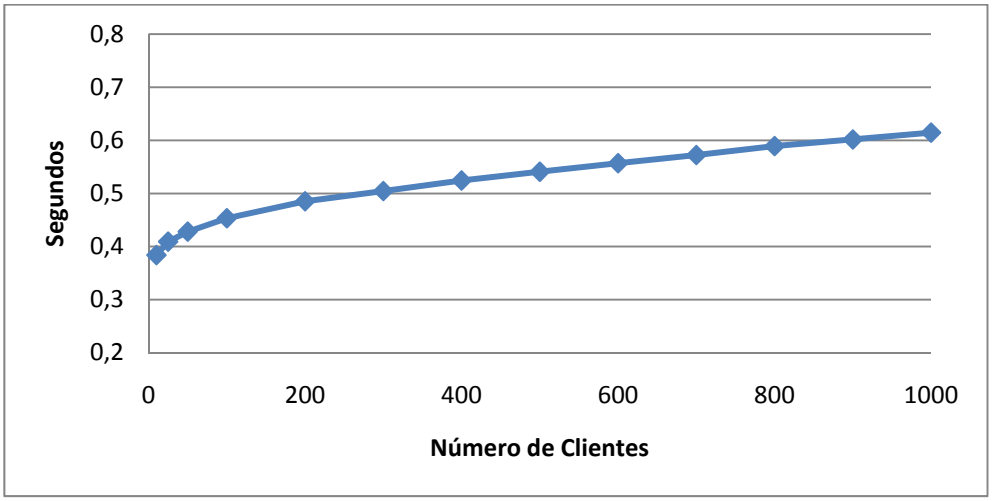
A su vez, como consecuencia de este aumento de peticiones resueltas entre los clientes, se produce un aumento de la carga de los mismos, tal como se muestra en la Gráfica 6.11. Además, este aumento en la carga va a suponer a su vez un aumento de la latencia de respuesta de los precintos por parte de los clientes (Gráfica 6.9).

En las gráficas anteriores se observa que a partir de un número de 800 clientes conectados al sistema, los valores mostrados en ellas parece que más o menos se estabilizan. Este comportamiento es acorde al que suele darse en una arquitectura P2P, donde la carga del sistema se reparte entre los clientes conectados manteniéndose ésta más o menos estable. Si bien, para alcanzar esta estabilidad en las arquitecturas P2P, hace falta un número suficiente de clientes.

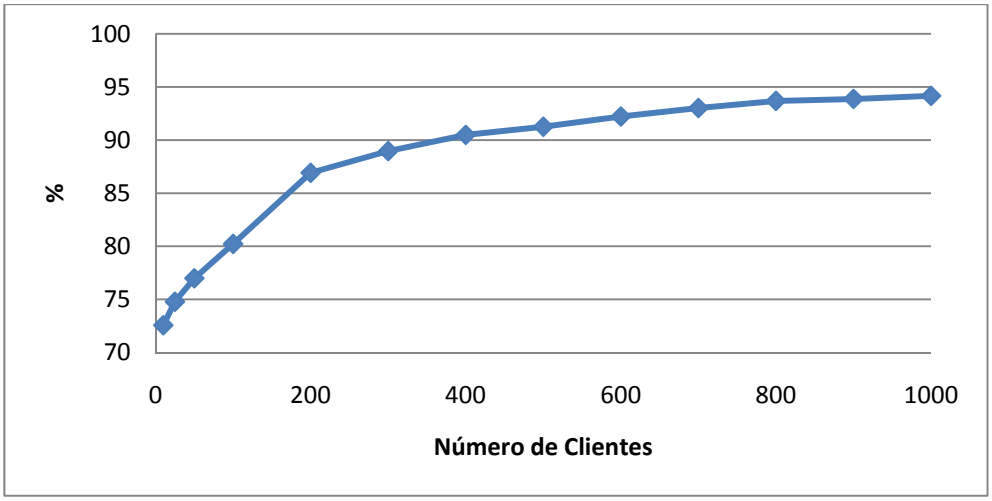
Como en el sistema se emplea una arquitectura mixta cliente-servidor / P2P, cuando no haya un número suficiente de clientes conectados en el sistema, éste se comportará como una arquitectura cliente-servidor, recayendo la mayor parte de la carga del sistema en los servidores secundarios. Conforme aumente el número de clientes conectados, éstos serán capaces de hacerse cargo de una mayor carga del sistema, liberando de ella a los servidores secundarios, hasta llegar a una situación de estabilidad.

Estos hechos se muestran en la Gráfica 6.9 y en la Gráfica 6.11, donde hasta los 800 clientes conectados, tanto la latencia como la carga en los clientes aumentan debido a que éstos se están haciendo cargo de una mayor cantidad de carga. A partir de 800 clientes, el sistema parece que se estabiliza, comportándose como una arquitectura P2P pura, debido a que parece que existe un número suficiente de clientes conectados para realizar una distribución adecuada de la carga del sistema entre todos ellos.

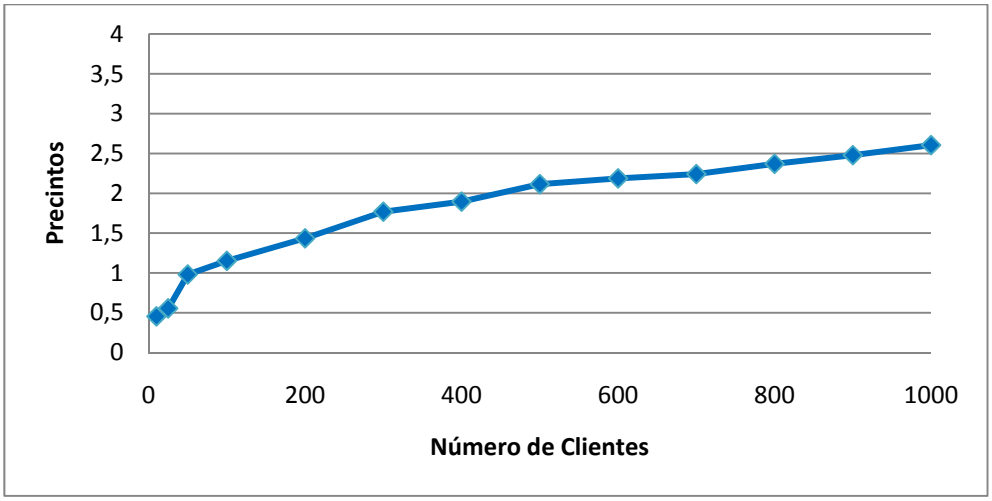
6.4.2.2.2 – Servidores Secundarios



Gráfica 6.12 – Latencia media de los precintos respondidos por los Servidores Secundarios



Gráfica 6.13 – Porcentaje medio de los precintos encontrados con éxito en la caché de los Servidores Secundarios



Gráfica 6.14 – Número de precintos medio en las colas de los Servidores Secundarios

En las gráficas anteriores se muestra la influencia que tiene en los servidores secundarios el aumento del número de clientes conectados al sistema.

En la Gráfica 6.12 se muestra la latencia media de los precintos respondidos por los servidores secundarios a los clientes. En ella se observa que la latencia se incrementa conforme aumenta el número de clientes conectados al sistema. Cabe recordar, que al igual que en los resultados obtenidos para el sistema real, esta latencia comprende el tiempo desde que un cliente inicia la petición de un precinto hasta que lo obtiene finalmente. En esta latencia se incluye el tiempo que se tarda en las diferentes peticiones del precinto que se hayan tenido que realizar, ya que por problemas en la transmisión o en los clientes, puede haber sido necesario repetir la petición del precinto. Este hecho explica porque el valor obtenido en los resultados para esta latencia es siempre superior a lo que se mide en el caso de la latencia de la transferencia entre clientes.

En la Gráfica 6.13 se muestra el porcentaje medio de los precintos pedidos por los clientes a los servidores secundarios que se han encontrado con éxito en la caché de los servidores secundarios. En ella se observa como el porcentaje de éxito aumenta conforme se incrementa el número de clientes conectados al sistema.

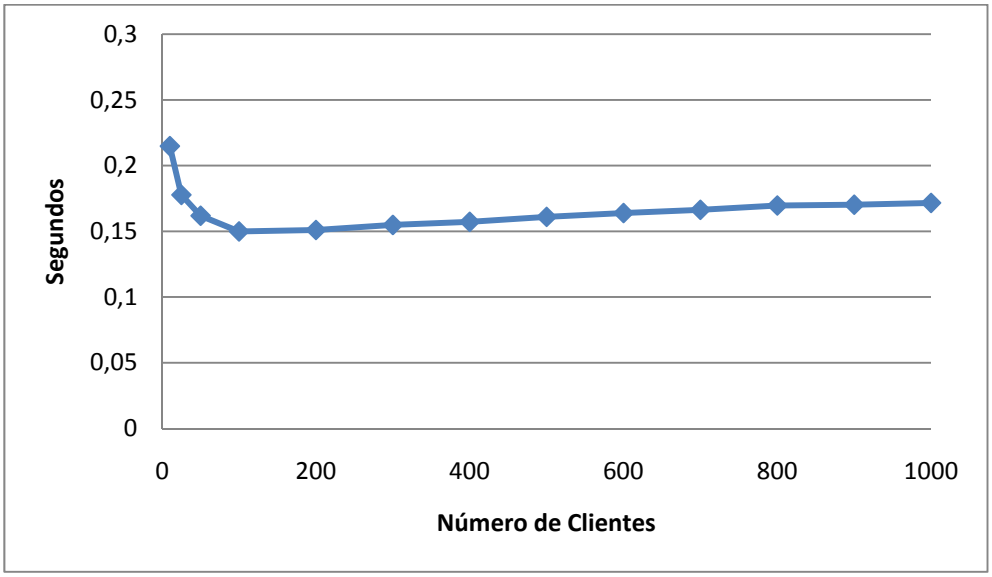
En la Gráfica 6.14 se muestra el número medio de precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por los servidores secundarios. En ella se observa que conforme el número de clientes conectado al sistema aumenta, el tamaño de esta cola aumenta, con lo que aumenta su carga.

A medida que se incrementa el número de clientes conectados al sistema, la carga global del mismo aumenta. A pesar de que, como se observa en la Gráfica 6.10 del apartado 6.4.2.2.1, el porcentaje de peticiones que realizan los clientes a los servidores secundarios decrece conforme hay un mayor número de clientes conectados al sistema, este descenso no es lineal con el número de clientes conectados, sino que tiende a estabilizarse en torno al 14%. Por lo tanto, a medida que aumenta el número de clientes conectados, la carga que aporta cada nuevo cliente que se conecta al sistema no será absorbida en su totalidad por el resto de clientes conectados al mismo, teniéndose que hacer cargo de ella los servidores secundarios, por lo que aumentará su carga, tal como se puede observar en la Gráfica 6.14.

El objetivo del servidor de comunicaciones en el sistema era el de repartir la carga global del sistema entre los clientes y los servidores secundarios tanto de forma espacial como de forma computacional. Para el caso de los servidores secundarios, el aumento en el número de clientes conectados al sistema permite al servidor de comunicaciones realizar una mejor distribución de la carga global de los servidores secundarios siguiendo estos dos criterios, ya que dispondrá de un mayor conjunto de clientes que podrán ser asignados al servidor secundario más especializado en la región del terreno en la que se encuentren, manteniendo a su vez equilibrada la carga global de los diferentes servidores secundarios. Este hecho explica, a nuestro entender, el aumento en el porcentaje de éxito de precintos pedidos por los clientes que se encuentran en la caché de los servidores secundarios conforme hay un mayor número de clientes conectados al sistema, tal como se muestra en la Gráfica 6.13.

El aumento de la carga de los servidores secundarios pensamos que es la que provoca el aumento de la latencia de respuesta de los precintos por parte de los servidores secundarios que se observa en la Gráfica 6.12. Si bien el porcentaje de éxito de que los precintos se encuentren en la caché aumenta conforme hay un mayor número de clientes conectados al sistema, este aumento no es suficiente para compensar el incremento de la carga de los servidores secundarios.

6.4.2.2.3 – Sistema



Gráfica 6.15 – Latencia media global del intercambio de precintos en el Sistema

En la gráfica anterior se muestra la influencia que tiene en el sistema en su conjunto el aumento del número de clientes conectados al mismo.

En la Gráfica 6.15 se muestra la latencia media global que se produce en el intercambio de precintos en el sistema. En ella se puede observar como conforme el número de clientes conectados al sistema aumenta, la latencia se decrementa inicialmente, para a partir de 100 clientes conectados aumentar ligeramente.

El decremento inicial de la latencia global que se produce hasta llegar a 100 clientes conectados pensamos que se debe a que, a pesar del incremento que hasta ese número de clientes conectados se produce en las latencias de respuesta de los precintos tanto por parte de los clientes como de los servidores secundarios, como aumenta el número de precintos servidos por los clientes (cuya latencia de respuesta es menor que la de los servidores secundarios) y disminuye el de los servidores secundarios, esto hace que se produzca un decremento de la latencia global.

A partir de 100 clientes conectados, el hecho de que el porcentaje de peticiones de los clientes que se dirigen a los servidores secundarios se decremente en menor medida hasta llegar a estabilizarse, pensamos que provoca que no se llegue a contrarrestar suficientemente el aumento de las latencias de respuesta de precintos de los servidores secundarios y de los clientes, lo que hace que haya un ligero aumento de la latencia global del sistema.

6.4.2.2.4 – Conclusiones de la prueba

Como conclusión de esta prueba, se puede decir que el sistema se comporta bastante bien con un número elevado de clientes conectados al mismo. Si bien, se observa un ligero incremento de la latencia general del sistema provocada por el aumento de la latencia de los servidores secundarios, ya que la de los clientes parece que llega a estabilizarse. Por lo tanto, si esta latencia general se incrementara en exceso sería necesario añadir algún servidor secundario adicional que permitiera reducir la carga, y con ello la latencia del resto de los servidores secundarios del sistema.

Por este motivo, para esta prueba en particular, no se puede afirmar que el sistema sea 100% escalable con el número de usuarios conectados, ya que los servidores secundarios pueden llegar a saturarse. Lo que sí se puede afirmar, es que se ha aumentado considerablemente la escalabilidad del sistema de visualización respecto de aquellos que emplean una arquitectura cliente-servidor clásica, ya que como se ha visto en los resultados, el porcentaje de peticiones de los que se han de hacer cargo los servidores secundarios, para esta prueba, se sitúa en torno al 14%, mientras que empleando una arquitectura cliente-servidor clásica, éste sería del 100%. Por lo tanto, se ha aumentado la escalabilidad del sistema en más de un 500%, lo que conlleva un importante ahorro económico.

6.4.2.3 – Comparativa de las arquitecturas

Con el objetivo de ver cuál de las dos arquitecturas, la arquitectura clásica cliente-servidor o la arquitectura mixta cliente-servidor / P2P, se adapta mejor a un sistema de visualización de terrenos con bases de datos remotas, se han realizado varias pruebas.

Para llevar a cabo estas pruebas, se han realizado las modificaciones convenientes en el simulador de forma que emule el funcionamiento de una arquitectura cliente-servidor pura.

Las modificaciones realizadas son las siguientes:

- Se ha modificado el módulo del cliente de forma que los clientes realicen siempre las peticiones de los precintos al servidor principal, sin intercambiar ningún tipo de información con otros clientes.

- Se han anulado los módulos del servidor secundario y del servidor de comunicaciones, y se ha modificado el módulo del servidor principal para que atienda directamente las peticiones de los clientes.

Se van a realizar pruebas para la arquitectura mixta y para la arquitectura cliente-servidor pura empleando los mismos parámetros iniciales de forma que las pruebas realizadas sean equiparables. Los resultados obtenidos en estas pruebas se compararán para evaluar la escalabilidad de ambas arquitecturas.

En ambas arquitecturas los clientes se van a encargar de la visualización de la información, la única diferencia estriba en que en la arquitectura cliente-servidor todos los datos son pedidos al servidor principal, mientras que en la arquitectura mixta los datos son pedidos primero a los clientes vecinos, y en caso de que éstos no dispusieran de ellos, son pedidos al servidor secundario.

Además, a diferencia de la arquitectura cliente-servidor, para la arquitectura mixta se continuará utilizando el servidor de comunicaciones que permitirá conectar a los clientes con los clientes vecinos y el servidor secundario.

Los parámetros de configuración para las pruebas de la arquitectura cliente-servidor son los siguientes:

Número de servidores conectados = 1 (un solo servidor con toda la información).

Ruta: Distribución uniforme (toda la superficie del terreno tiene el mismo nivel de interés).

Tamaño de la caché del servidor = 100% (toda la base de datos se encuentra en el servidor).

Tamaño de la caché del cliente = 10% (consideramos este valor como aceptable dado el tamaño de la base de datos utilizada en la prueba).

Tamaño de la lista de clientes vecinos = 0 (no se intercambia información con ningún cliente, toda la información se descarga del servidor).

Porcentaje de precintos que vuelven a ser pedidos = alrededor del 11% (el mismo valor extraído de las pruebas reales como media de los precintos que deben ser pedidos de nuevo debido a que se han perdido en el envío).

Las pruebas realizadas han sido:

Prueba_A: Número de clientes conectados = 10.

Prueba_B: Número de clientes conectados = 25.

Prueba_C: Número de clientes conectados = 50.

Prueba_D: Número de clientes conectados = 100.

Prueba_E: Número de clientes conectados = 200.

Los parámetros de configuración para las pruebas de la arquitectura mixta cliente-servidor / P2P son los siguientes:

Número de servidores secundarios conectados = 1 (un solo servidor secundario, equivalente al servidor de la arquitectura cliente-servidor).

Ruta: Distribución uniforme (toda la superficie del terreno tiene el mismo nivel de interés).

Tamaño de la caché del servidor secundario = 50% (consideramos este valor como aceptable dado el tamaño de la base de datos utilizada en la prueba).

Tamaño de la caché del cliente = 10% (consideramos este valor como aceptable dato el tamaño de la base de datos utilizada en la prueba).

Tamaño de la lista de clientes vecinos = 10 (se mantiene este número de vecinos puesto que se consideró un número adecuado en las pruebas realizas con el sistema real).

Porcentaje de precintos que vuelven a ser pedidos = alrededor del 11% (este valor ha sido extraído de las pruebas reales como media de los precintos que deben ser pedidos de nuevo debido a que se han perdido en el envío o han tardado más del tiempo permitido en llegar a su destino).

Las pruebas realizadas han sido:

Prueba_A: Número de clientes conectados = 10.

Prueba_B: Número de clientes conectados = 25.

Prueba_C: Número de clientes conectados = 50.

Prueba_D: Número de clientes conectados = 100.

Prueba_E: Número de clientes conectados = 200.

Prueba_F: Número de clientes conectados = 300.

Prueba_G: Número de clientes conectados = 400.

Prueba_H: Número de clientes conectados = 500.

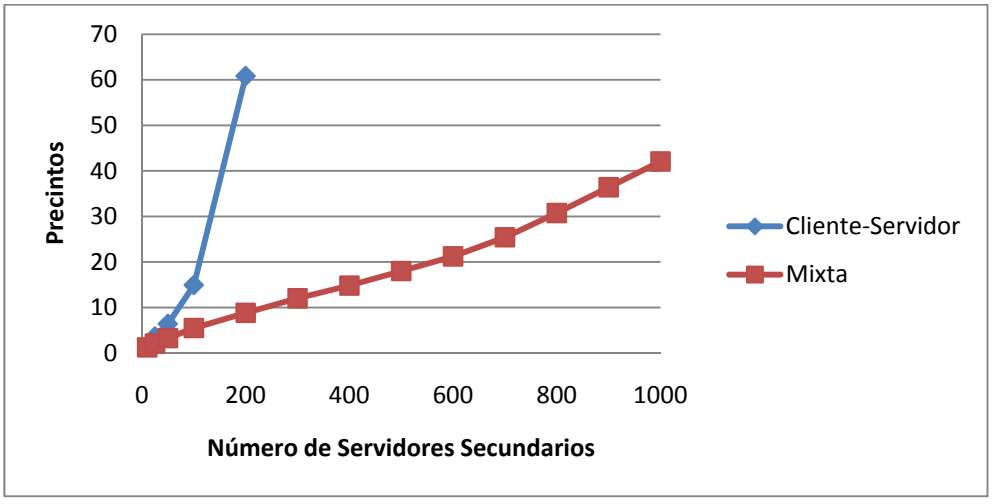
Prueba_I: Número de clientes conectados = 600.

Prueba_J: Número de clientes conectados = 700.

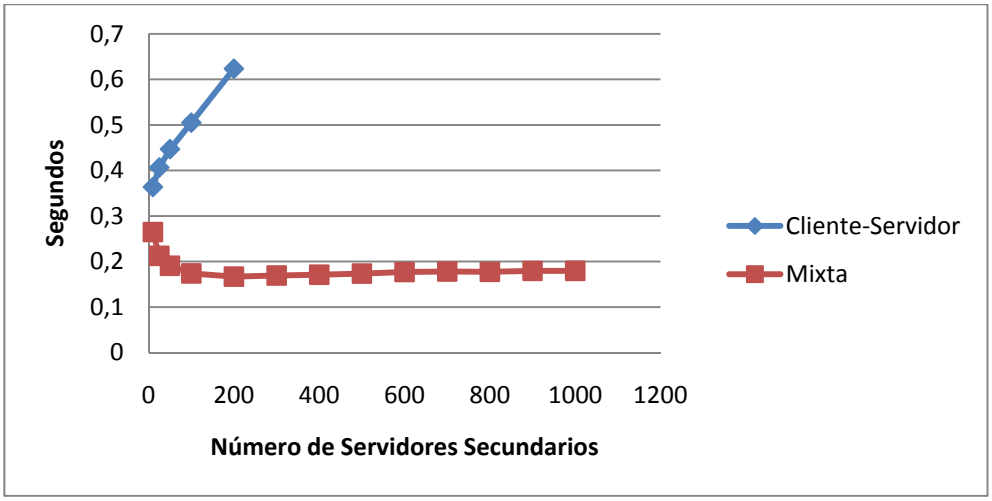
Prueba_K: Número de clientes conectados = 800.

Prueba_L: Número de clientes conectados = 900.

Prueba_M: Número de clientes conectados = 1000.



Gráfica 6.16 – Comparativa del número de precintos medio de la cola del Servidor



Gráfica 6.17 – Comparativa de la latencia media global del intercambio de precintos en el Sistema

En las gráficas anteriores se muestra la comparativa entre la arquitectura cliente-servidor y la arquitectura mixta cliente-servidor / P2P para un sistema de visualización de terrenos en tiempo real.

En la Gráfica 6.16 se muestra el número medio de precintos pedidos por los clientes que se encuentran en cola esperando a ser respondidos por el servidor. En ella se observa que conforme aumenta el número de clientes conectados al sistema, para la arquitectura cliente-servidor el número de precintos en cola esperando a ser procesados aumenta rápidamente, mientras que para la arquitectura mixta este aumento es mucho menor.

En la Gráfica 6.17 se muestra la latencia media global del intercambio de precintos en el sistema. En ella se puede observar como conforme el número de clientes conectados al sistema aumenta, para la arquitectura cliente-servidor la latencia aumenta rápidamente, sin embargo para la arquitectura mixta la latencia inicialmente decrece, para después ir incrementándose pero sólo ligeramente.

A medida que aumenta el número de clientes conectados al sistema, la carga que ha de soportar el mismo aumenta. Para la arquitectura cliente-servidor, toda esta carga ha de ser asumida por el servidor, lo que provoca que la carga del mismo se incremente rápidamente (Gráfica 6.16), y tenga como consecuencia que aumente la latencia global (Gráfica 6.17). Estos hechos prueban la falta de escalabilidad de la arquitectura cliente-servidor. En las

pruebas realizadas, a partir de 200 clientes conectados, el servidor se satura y no es capaz de responder los precintos a los clientes dentro de un tiempo aceptable.

Para la arquitectura mixta cliente-servidor / P2P, el aumento de carga producido por el aumento del número de clientes conectados al sistema se reparte entre el servidor y los clientes, lo que hace que la carga del servidor se incremente en menor medida que en el caso de la arquitectura cliente-servidor clásica. Además, la latencia general del sistema se incrementa de forma muy ligera, a diferencia de la arquitectura cliente-servidor clásica, debido a que conforme aumenta el número de clientes, éstos van a poder resolver un mayor porcentaje de peticiones de precintos entre sus clientes vecinos, disminuyendo de esa forma la necesidad de acceso al servidor. Gracias a ello, el sistema permite que haya 1000 clientes conectados sin que se sature el servidor.

6.4.2.3.1 – Conclusiones de la prueba

Como conclusión de esta prueba podemos decir que la arquitectura mixta cliente-servidor / P2P se va a adaptar mejor a un sistema de visualización de terrenos con bases de datos remotas que una arquitectura cliente-servidor clásica, debido principalmente a su mayor escalabilidad, que soporta un número de clientes conectados al sistema bastante superior al que es capaz de soportar la arquitectura clásica cliente-servidor antes de saturarse.

6.4.2.4 – Otras pruebas

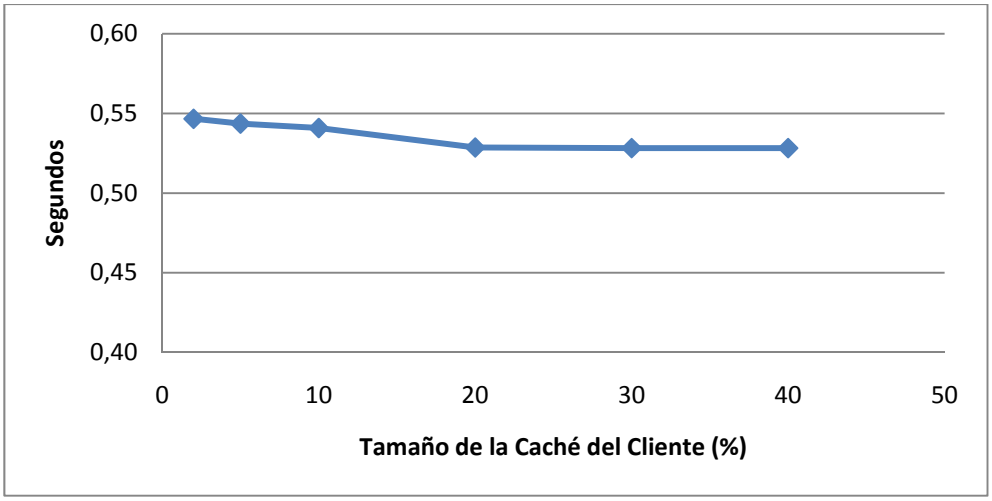
Se han realizado con el simulador las mismas pruebas que se han llevado a cabo con el sistema real (apartado 5.4), pero en esta ocasión aplicando los cambios especificados en el apartado 6.4.2.1.

Los resultados obtenidos en estas pruebas son similares a los obtenidos con el sistema real, reafirmando la mayor parte de las conclusiones obtenidas en su momento para el mismo. Por ello, a continuación únicamente se van a explicar las diferencias observadas en esas pruebas, sin llegar a realizar una explicación exhaustiva de todo el conjunto de gráficas resultantes de las pruebas.

6.4.2.4.1 – Variación en el tamaño de la caché de los clientes

Esta prueba realizada con el simulador es la equivalente a la realizada con el sistema real que se ha mostrado en el apartado 5.4.1. Aparte de los cambios especificados en el apartado 6.4.2.1, los parámetros iniciales empleados van a ser los mismos que para aquella prueba, la única diferencia radica en el uso de un mayor número de clientes, que en este caso será de 500.

Del conjunto de gráficas que se va a mostrar a continuación, la única en la que se puede observar un comportamiento diferente al visto en las gráficas obtenidas para el sistema real es en la Gráfica 6.18.



Gráfica 6.18 – Latencia media de los precintos respondidos por los Servidores Secundarios

En la Gráfica 6.18 se muestra la latencia media de los precintos respondidos por los servidores secundarios a los clientes. En ella se observa como a medida que se incrementa el tamaño de la caché de los clientes, disminuye la latencia. Sin embargo, en la prueba del sistema real se mostraba un incremento en esa latencia (Gráfica 5.4).

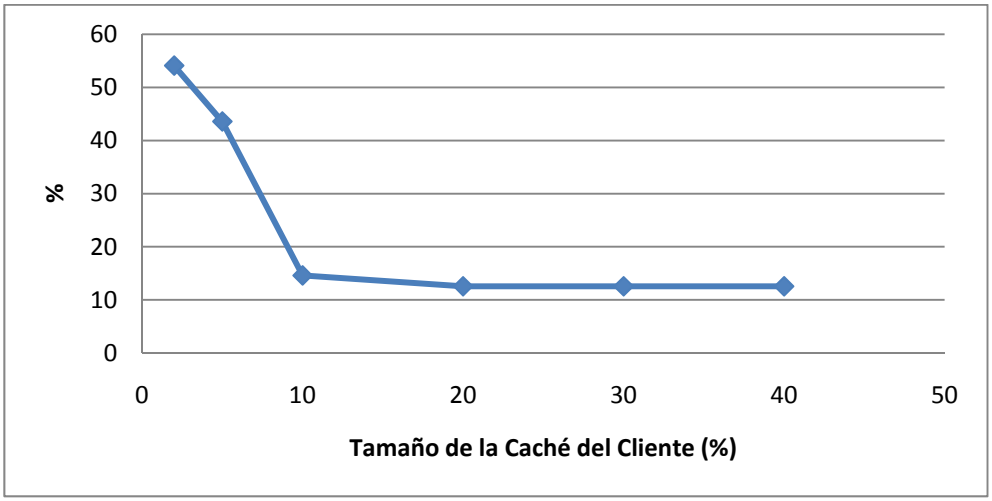
En la prueba del sistema real, se dedujo que este incremento en la latencia era debido al descenso del porcentaje de éxito de que los precintos pedidos por los clientes se encontraran en la caché del servidor secundario, lo que suponía una mayor necesidad de acceso al servidor principal a por la información, y con ello una mayor latencia.

A pesar de que la carga de los servidores secundarios era menor conforme mayor era el tamaño de la caché empleada en el cliente, debido a que los clientes tenían disponible una

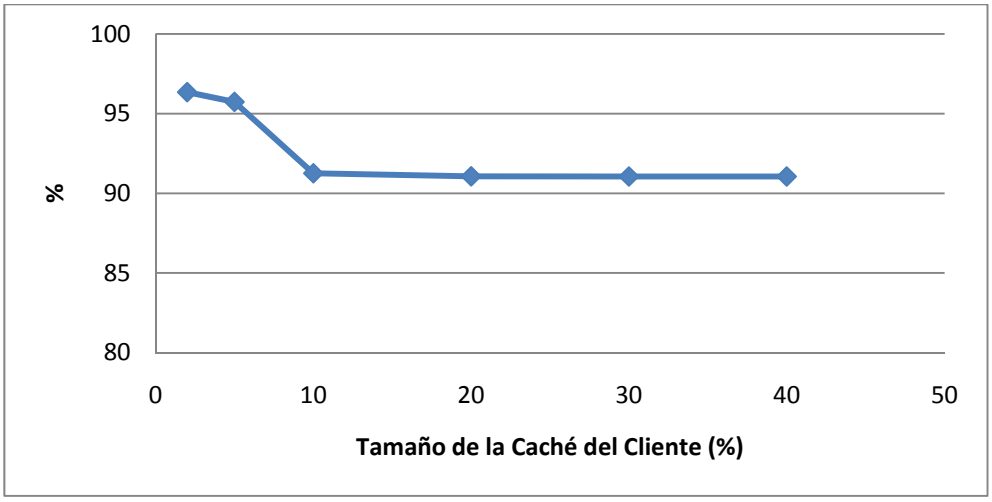
mayor cantidad de información en su caché que podían intercambiar con sus clientes vecinos en vez de acudir a los servidores secundarios, este descenso de carga no era suficiente para compensar el aumento de latencia.

En la prueba realizada con el simulador, se observa que esa latencia disminuye ligeramente en vez de aumentar. Esto pensamos que se produce porque, esta vez, el número de clientes conectados al sistema es mayor (500), por lo que el impacto que tiene el aumento del tamaño de la caché del cliente en la necesidad de éstos de acudir al servidor secundario a por los precintos (Gráfica 6.19) es mayor que en la prueba del sistema real. Por lo tanto, pensamos que el descenso en la carga de los servidores secundarios, en esta ocasión, sí que compensa el aumento de latencia que se produce por el descenso del porcentaje de éxito de los servidores secundarios (Gráfica 6.20), lo que supone globalmente una latencia de respuesta de precintos por parte del servidor secundario ligeramente inferior a medida que aumenta el tamaño de la caché empleada en los clientes.

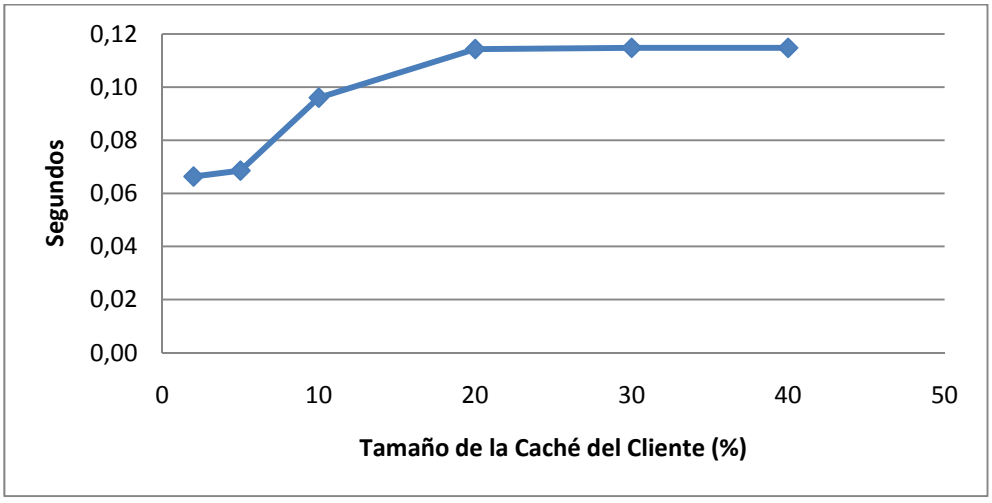
A continuación se van a mostrar el resto de gráficas de esta prueba. La explicación del comportamiento mostrado en ellas es la misma que la expuesta en la prueba del sistema real del apartado 5.4.1, por lo que no se van a volver a comentar de nuevo aquí.



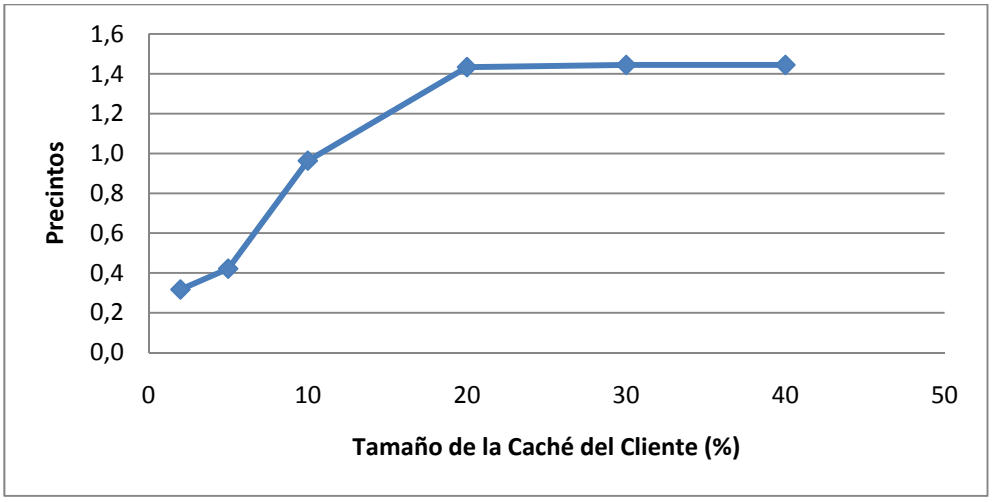
Gráfica 6.19 – Porcentaje medio de precintos respondidos por los Servidores Secundarios



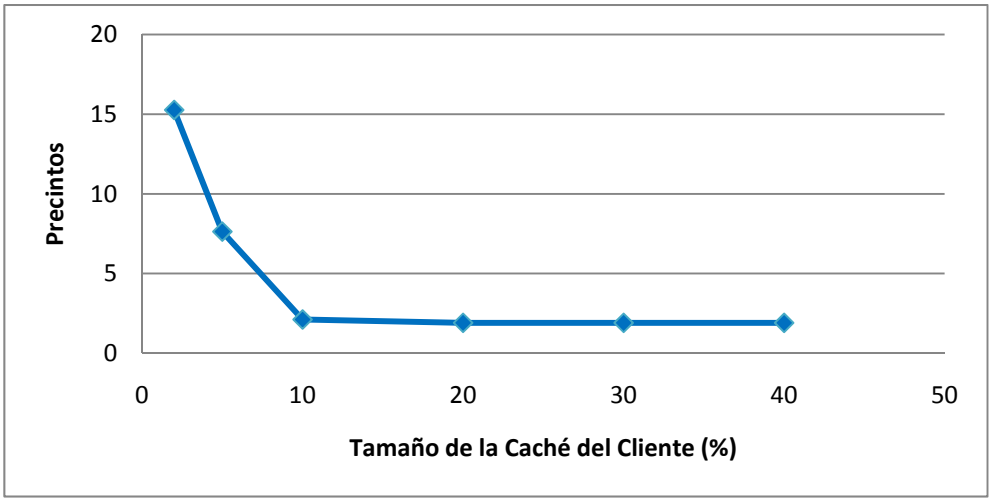
Gráfica 6.20 – Porcentaje medio de los precintos encontrados con éxito en la Caché de los Servidores Secundarios



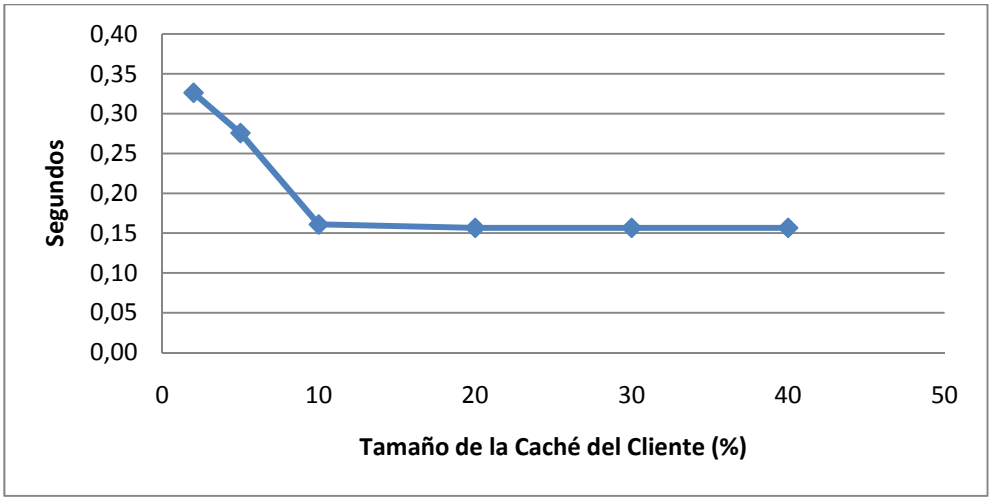
Gráfica 6.21 – Latencia media de los precintos intercambiados entre los Clientes



Gráfica 6.22 – Número de precintos medio de las colas de los Clientes



Gráfica 6.23 – Número de precintos medio de las colas de los Servidores Secundarios



Gráfica 6.24 – Latencia media global del intercambio de precintos en el Sistema

Conclusión

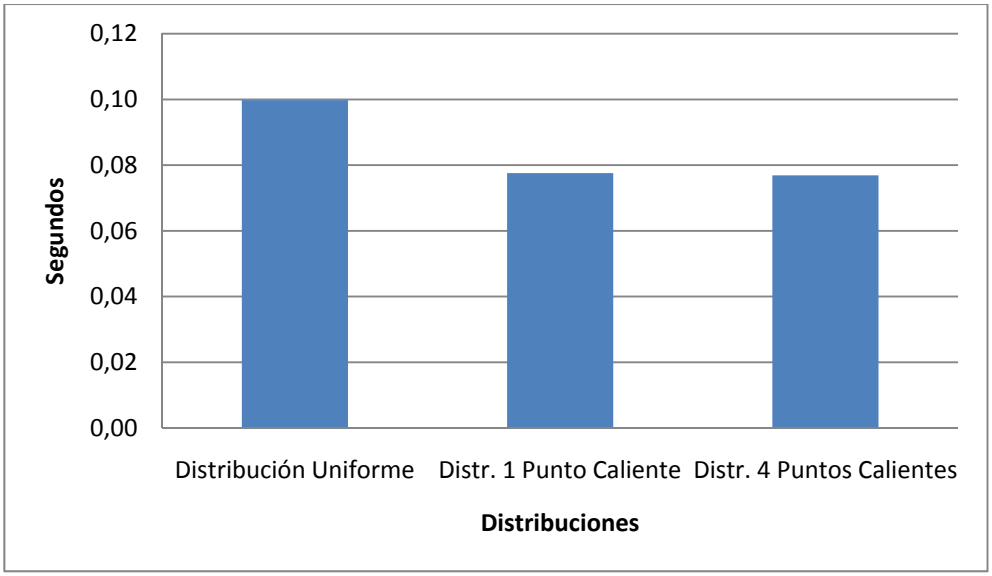
Como conclusión de esta prueba se puede decir que, al igual que se observó para el sistema real empleando un número reducido de clientes, a priori, cuando se conecten un número más elevado de ellos, parece que el sistema seguirá comportándose mejor conforme mayor sea la caché de los clientes (Gráfica 6.24). Sin embargo, como se comentó para el sistema real, habrá que llegar a un compromiso entre almacenamiento y prestaciones, ya que las bases de datos de terrenos suelen tener un gran tamaño y por lo tanto el cliente no va a poder almacenarla de forma completa.

Además, como se observa para esta prueba, parece que un tamaño de caché del 10% del tamaño de la base de datos del terreno empleada, parece ser suficiente para el desarrollo de la prueba, ya que un tamaño mayor apenas aporta una mejora de las prestaciones y en cambio sí que aumenta los requerimientos de almacenamiento de los clientes.

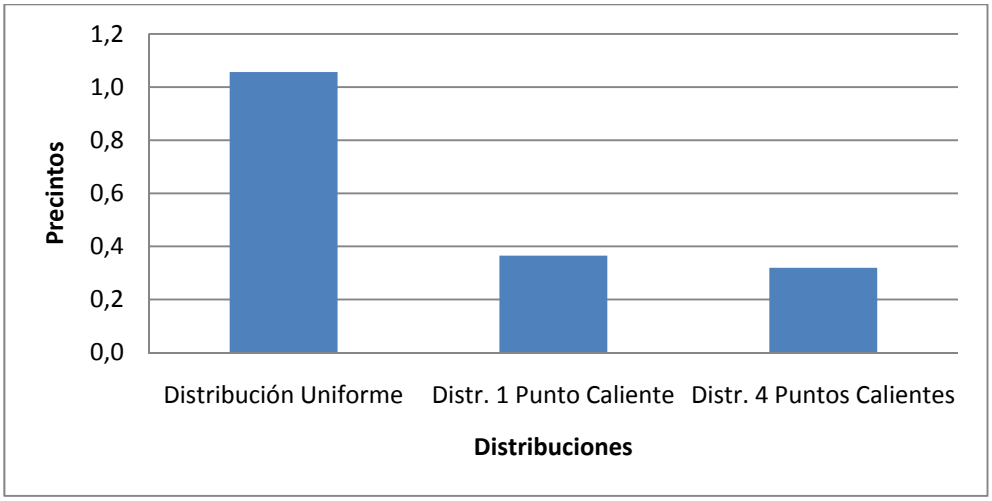
6.4.2.4.2 – Variación en la distribución de la ruta

Esta prueba realizada con el simulador es la equivalente a la realizada con el sistema real que se ha mostrado en el apartado 5.4.2. Aparte de los cambios especificados en el apartado 6.4.2.1, los parámetros iniciales empleados van a ser los mismos que para aquella prueba, la única diferencia radica en el uso de un mayor número de clientes, que en este caso será de 1000.

Del conjunto de gráficas que se va a mostrar a continuación, las únicas en la que se puede observar un comportamiento diferente al visto en las gráficas obtenidas para el sistema real son la Gráfica 6.25 y la Gráfica 6.26.



Gráfica 6.25 – Latencia media de los precintos intercambiados entre los Clientes



Gráfica 6.26 – Número de precintos medio de las colas de los Clientes

En la Gráfica 6.25 se muestra la latencia media de los precintos respondidos por los clientes. En ella se observa como cuando se emplean distribuciones normales de uno y cuatro puntos calientes, disminuye la latencia respecto a la distribución uniforme. Sin embargo, en la prueba del sistema real, se mostraba un ligero incremento en esa latencia (Gráfica 5.9).

En la Gráfica 6.26 se muestra el número medio de precintos pedidos por los clientes que se encuentran en la cola esperando a ser respondidos por los clientes vecinos. En ella se observa como el tamaño de la cola es menor para las distribuciones con uno y cuatro puntos calientes. Sin embargo, en la prueba del sistema real, se mostraba un ligero incremento en el tamaño de la cola para esas distribuciones (Gráfica 5.11).

En la prueba del sistema real, se dedujo que debido a que en las distribuciones normales los clientes se desplazaban alrededor de las mismas regiones del terreno y no alrededor de toda la superficie del terreno, era más probable que éstos contuvieran en su caché los precintos que van a ser necesitados por otros clientes vecinos, por lo que el porcentaje de peticiones que se realizaban a los servidores secundarios era menor. Esto suponía que aumentaba el número de peticiones que se resolvían entre los clientes, aumentando su carga y el tiempo de respuesta de los precintos por parte de ellos.

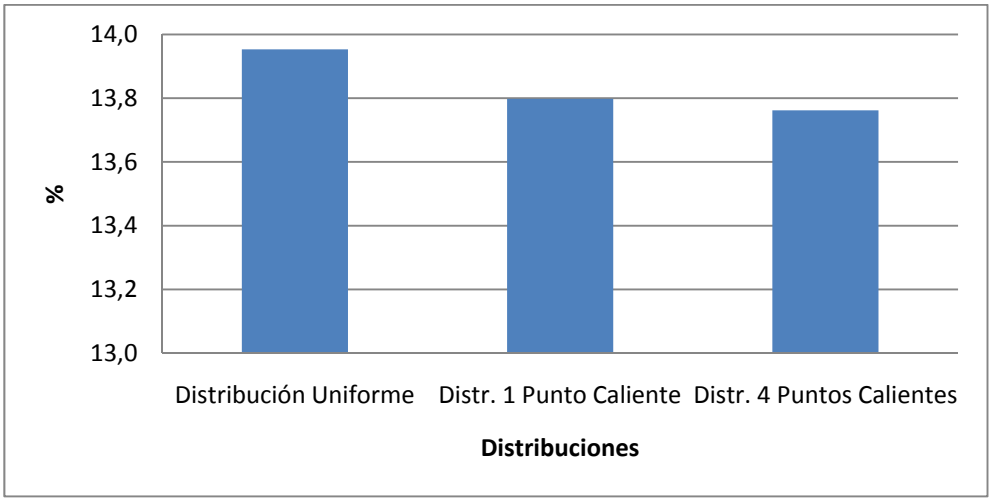
En la prueba realizada con el simulador, se observa que cuando se emplean distribuciones de tipo normal, tanto la carga de los clientes como la latencia de respuesta de los precintos disminuye en vez de aumentar, a pesar de que al igual que en el sistema real, el porcentaje

de peticiones que se realizan a los servidores secundarios. Si bien, hay que hacer notar que este descenso es más pequeño que el que se producía en el sistema real.

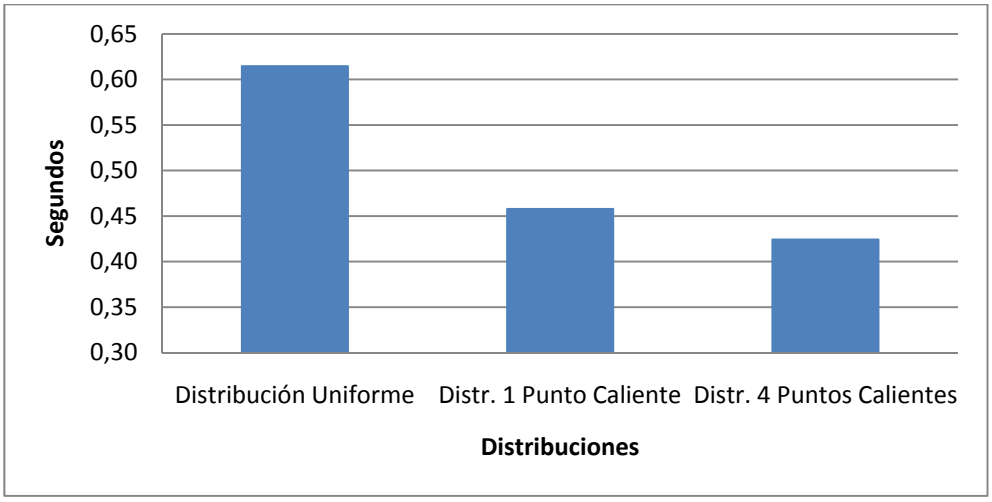
Cuando el cliente se mueve utilizando un tipo de distribución normal, además del menor porcentaje de peticiones que se realizan a los servidores secundarios, también se produce un descenso de la cantidad de peticiones de precintos que cada cliente ha de realizar, ya que al desplazarse por las mismas regiones, ya habrá descargado previamente parte de la información que necesita para llevar a cabo la visualización de la superficie del terreno.

En la prueba real, pensamos que debido a que el número de clientes era reducido, esta disminución de peticiones se contrarrestaba con el mayor número de peticiones que se podían resolver entre los clientes sin acudir a los servidores secundarios, por lo que finalmente, tanto la latencia como la carga de los clientes aumentaban ligeramente. Sin embargo, para la prueba realizada con el simulador se han utilizado 1000 clientes, lo que va a suponer un descenso considerable del número de peticiones global del sistema cuando se emplee un tipo de distribución normal. Por lo tanto, pensamos que este descenso en el número de peticiones globales, en esta ocasión no se compensa con el mayor número de peticiones de precintos que pueden intercambiarse entre los clientes, lo que provoca que tanto la carga de los clientes como la latencia de respuesta de precintos por su parte desciendan en vez de aumentar.

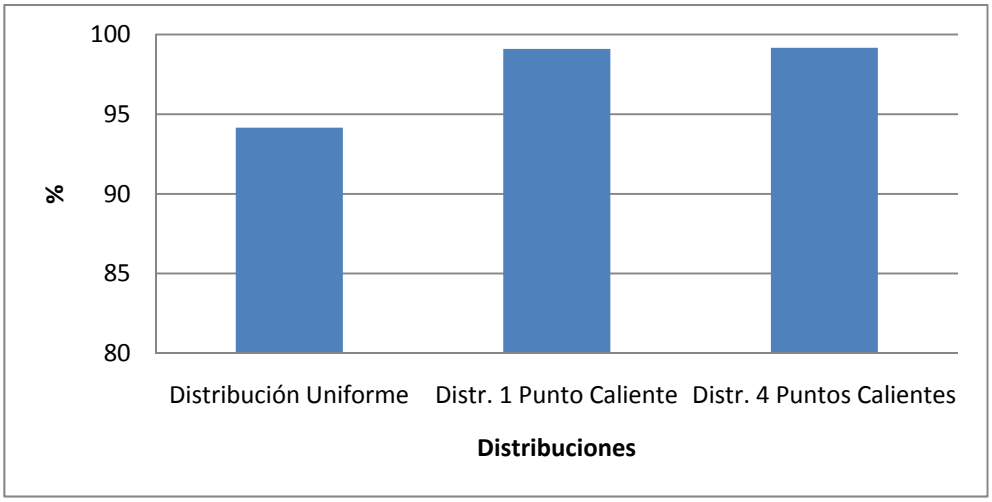
A continuación se van a mostrar el resto de gráficas de esta prueba. La explicación del comportamiento mostrado en ellas es la misma que la expuesta en la prueba del sistema real del apartado 5.4.2, por lo que no se van a volver a comentar de nuevo aquí.



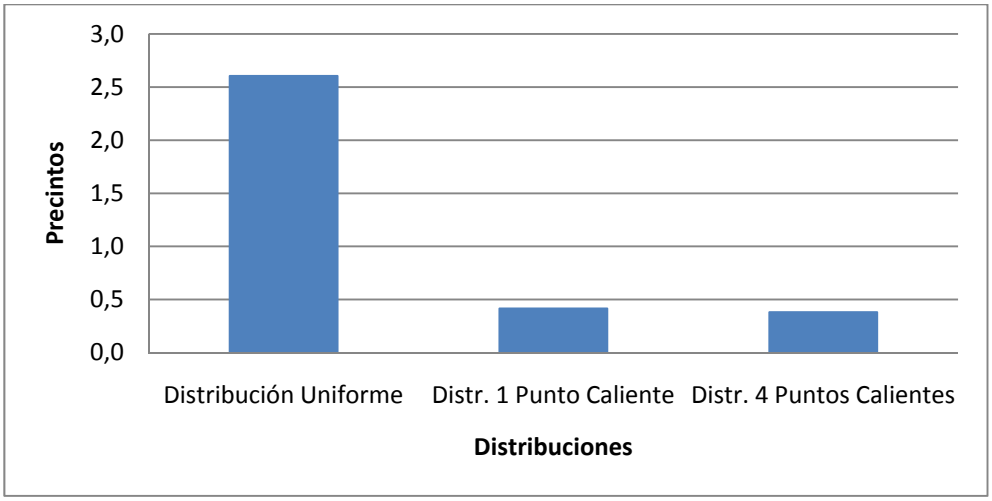
Gráfica 6.27 – Porcentaje medio de precintos respondidos por los Servidores Secundarios



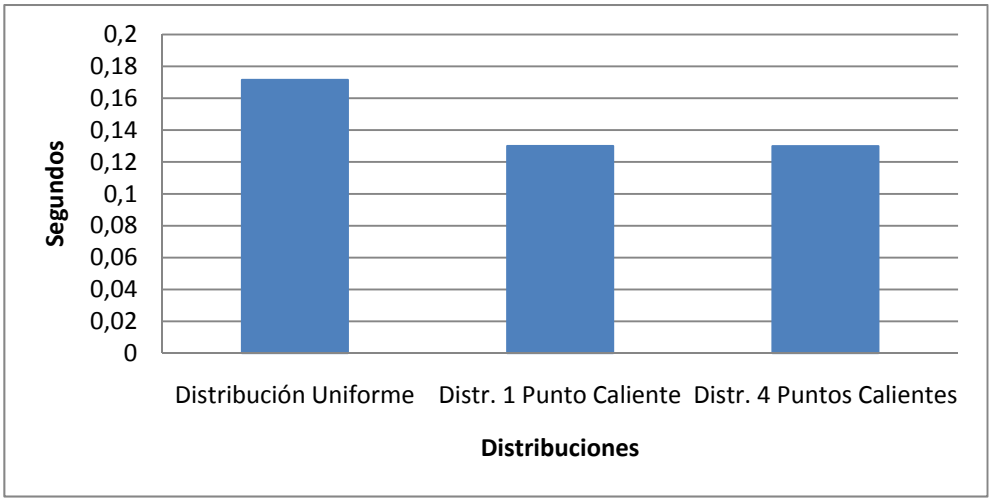
Gráfica 6.28 – Latencia media de los precintos respondidos por los Servidores Secundarios



Gráfica 6.29 – Porcentaje medio de los precintos encontrados con éxito en la Caché de los Servidores Secundarios



Gráfica 6.30 – Número de precintos medio de las colas de los Servidores Secundarios



Gráfica 6.31 – Latencia media global del intercambio de precintos en el Sistema

Conclusión

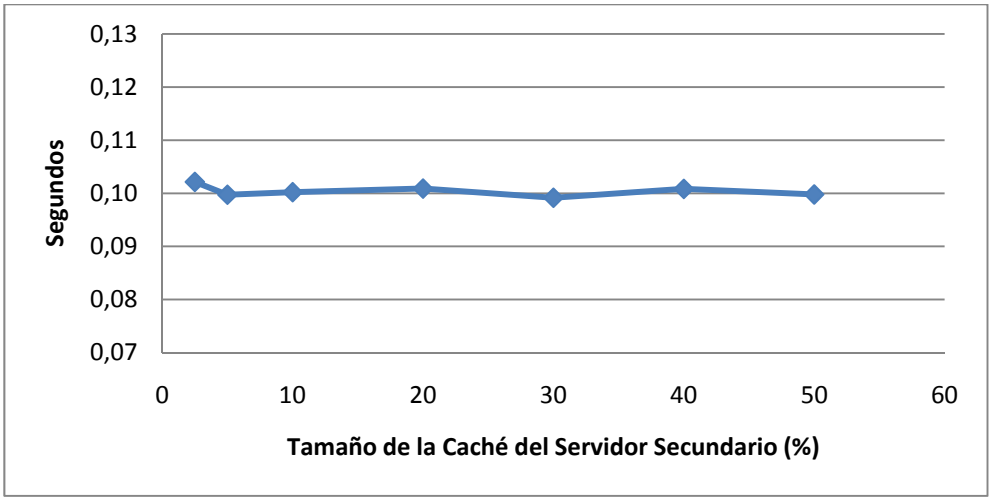
Como conclusión de esta prueba se puede decir que, al igual que para el sistema real con un número reducido de clientes conectados, parece que, a priori, el sistema funciona adecuadamente con diferentes supuestos de comportamiento de los usuarios cuando el número de ellos conectados simultáneamente es elevado.

Además, parece que la especialización por regiones del terreno llevada a cabo por los servidores secundarios sigue funcionando adecuadamente cuando hay un número elevado de clientes conectados, obteniendo unos mejores resultados cuando se emplean varias zonas de interés que concentran a los usuarios, ya que en ese caso la especialización por regiones es más evidente.

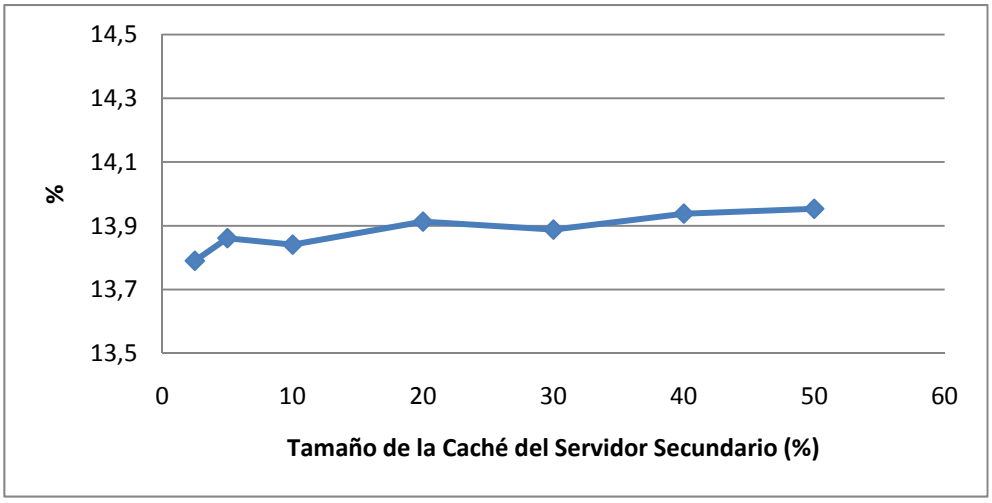
6.4.2.4.3 – Variación en el tamaño de la caché de los servidores secundarios

Esta prueba realizada con el simulador es la equivalente a la realizada con el sistema real que se ha mostrado en el apartado 5.4.4. Aparte de los cambios especificados en el apartado 6.4.2.1, los parámetros iniciales empleados van a ser los mismos que para aquella prueba, la única diferencia radica en el uso de un mayor número de clientes, que en este caso será de 1000, y en el empleo un mayor rango de tamaños de caché en las pruebas, ya que el tamaño de la base de datos del terreno utilizada es mayor. Estos tamaños de caché van a ser del 2.5%, 5%, 10%, 20%, 30%, 40% y 50% del tamaño de la base de datos del terreno.

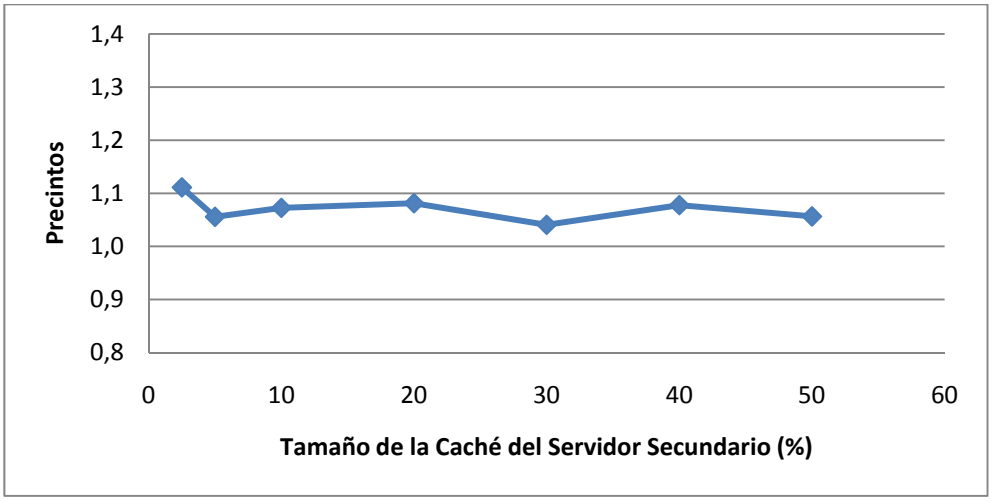
Del conjunto de gráficas que se va a mostrar a continuación, la explicación del comportamiento mostrado en ellas es la misma que la expuesta en la prueba del sistema real del apartado 5.4.4, por lo que no se van a volver a comentar de nuevo aquí.



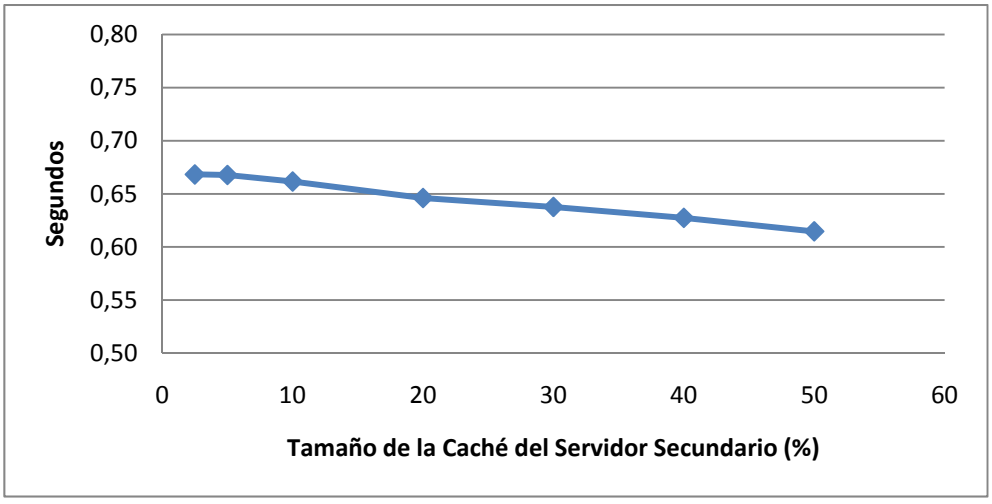
Gráfica 6.32 – Latencia media de los precintos intercambiados entre los Clientes



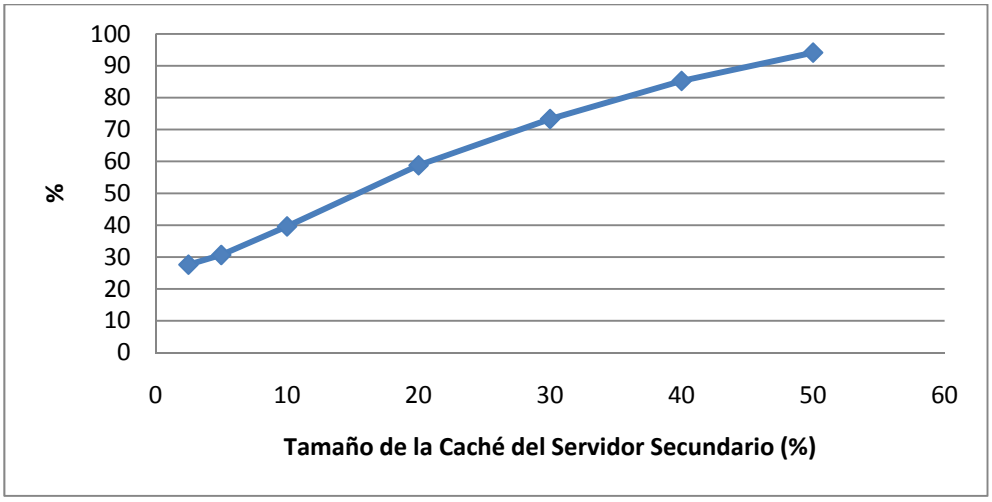
Gráfica 6.33 – Porcentaje medio de precintos respondidos por los Servidores Secundarios



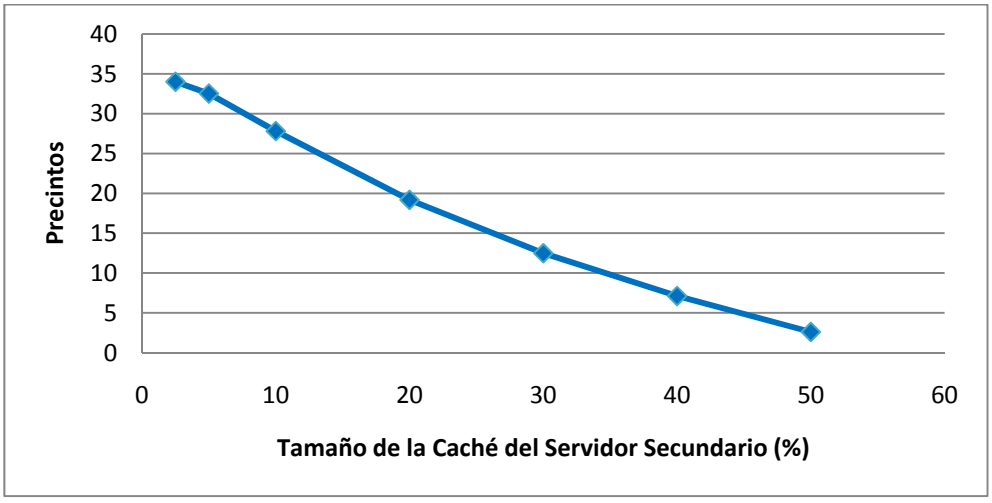
Gráfica 6.34 – Número de precintos medio de las colas de los Clientes



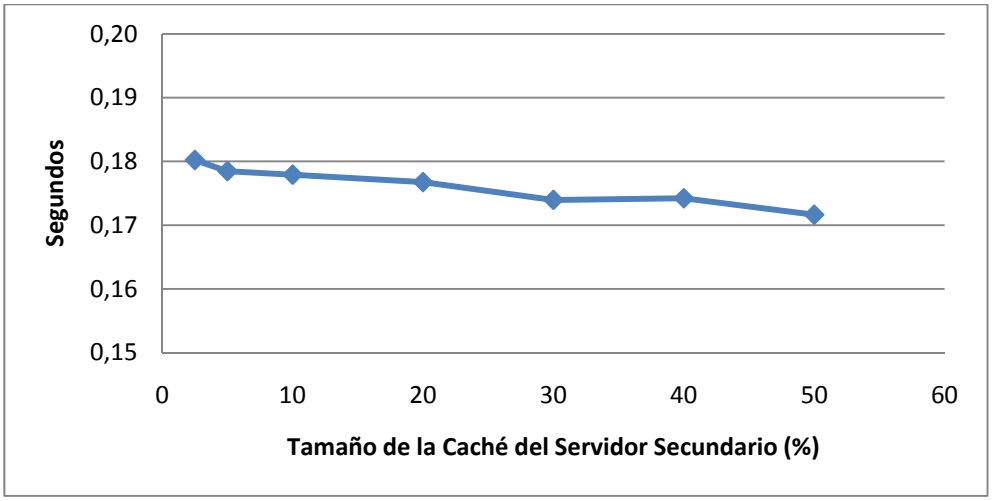
Gráfica 6.35 – Latencia media de los precintos respondidos por los Servidores Secundarios



Gráfica 6.36 – Porcentaje medio de los precintos encontrados con éxito en la Caché de los Servidores Secundarios



Gráfica 6.37 – Número de precintos medio de las colas de los Servidores Secundarios



Gráfica 6.38 – Latencia media global del intercambio de precintos en el Sistema

Conclusión

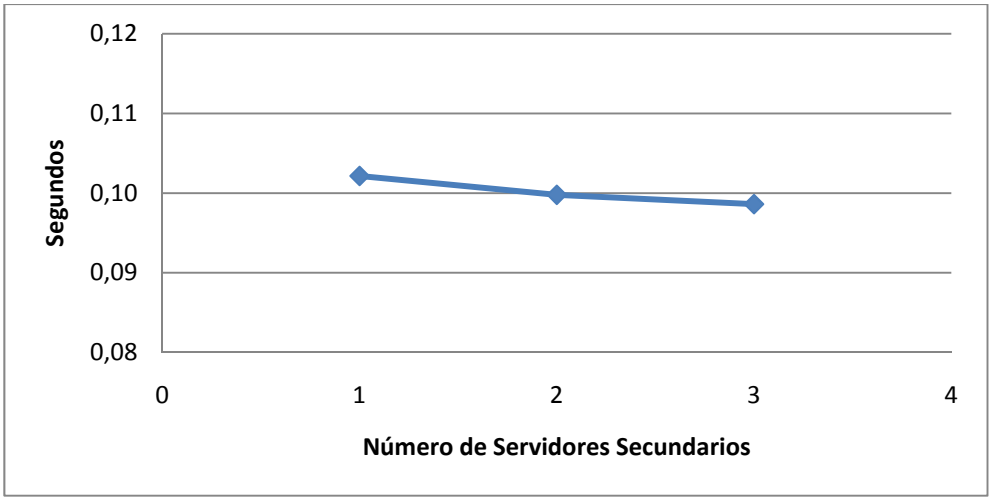
Como conclusión de esta prueba se puede decir que, al igual que para el sistema real con un número reducido de clientes conectados, parece que, a priori, cuando hay un número de clientes conectados elevado, el sistema continua funciona mejor conforme mayor sea el tamaño de la caché empleada en los servidores secundarios.

Al igual que se comentó en las conclusiones de la prueba del sistema real (apartado 5.4.4), habrá que plantearse si es necesario una réplica completa de la base de datos en cada servidor secundario. Como se comentó en ese apartado, un tamaño de caché del 50% del tamaño de la base de datos del terreno parece adecuada porque: los servidores secundarios se van a especializar por regiones del terreno, existirán zonas de la superficie del terreno almacenada en la base de datos que, por carecer de interés para los usuarios, no se a van a necesitar nunca, y el incremento en el tamaño de la caché no aporta un descenso destacado del tiempo de latencia global.

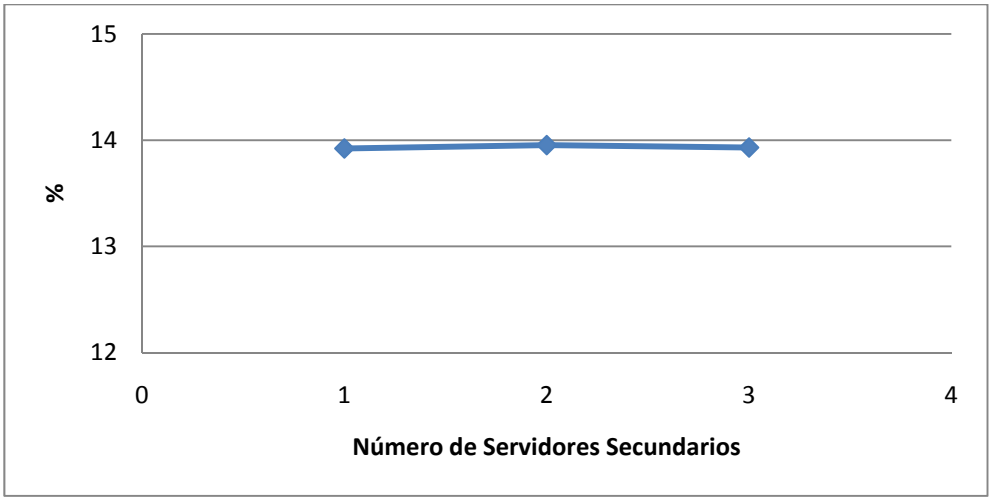
6.4.2.4.4 – Variación en el número de servidores secundarios conectados

Esta prueba realizada con el simulador es la equivalente a la realizada con el sistema real que se ha mostrado en el apartado 5.4.5. Aparte de los cambios especificados en el apartado 6.4.2.1, los parámetros iniciales empleados van a ser los mismos que para aquella prueba, la única diferencia radica en el uso de un mayor número de clientes, que en este caso será de 1000.

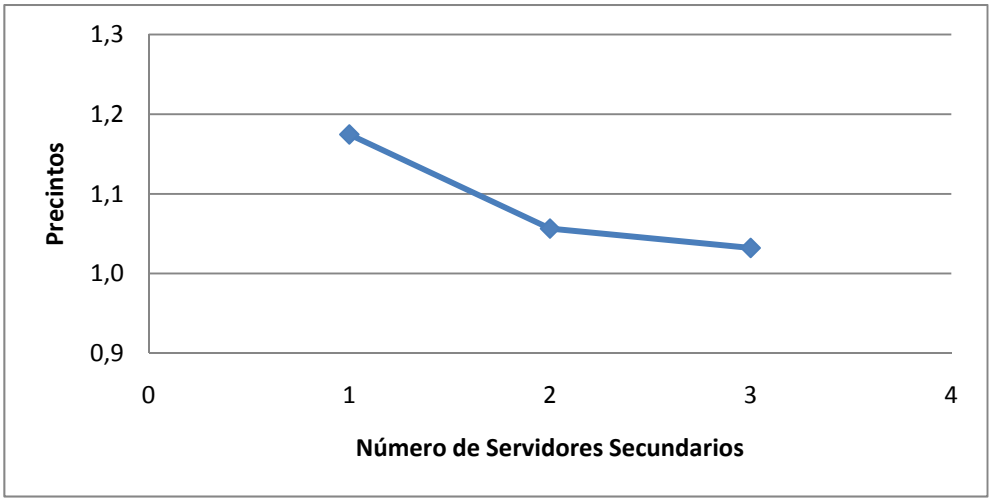
Del conjunto de gráficas que se van a mostrar a continuación, la explicación del comportamiento mostrado en ellas es la misma que la expuesta en la prueba del sistema real del apartado 5.4.5, por lo que no se van a volver a comentar de nuevo aquí.



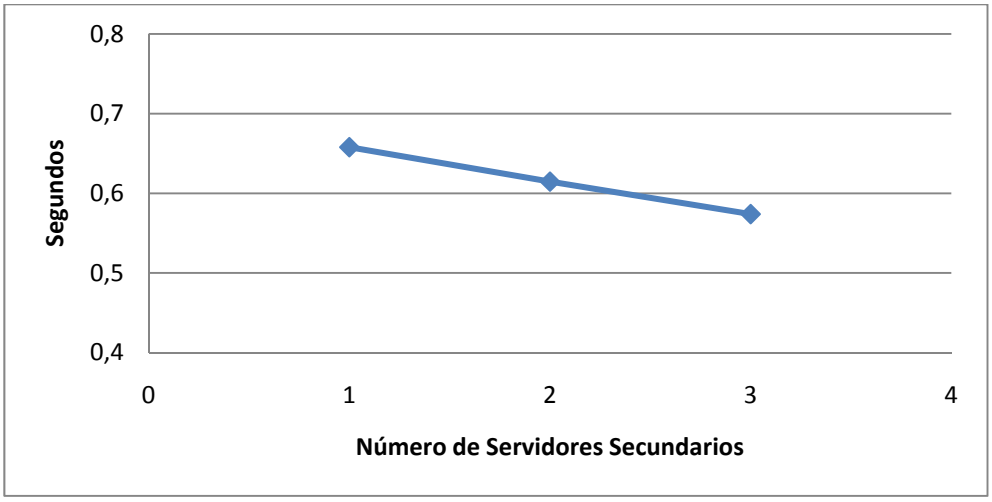
Gráfica 6.39 – Latencia media de los precintos intercambiados entre los Clientes



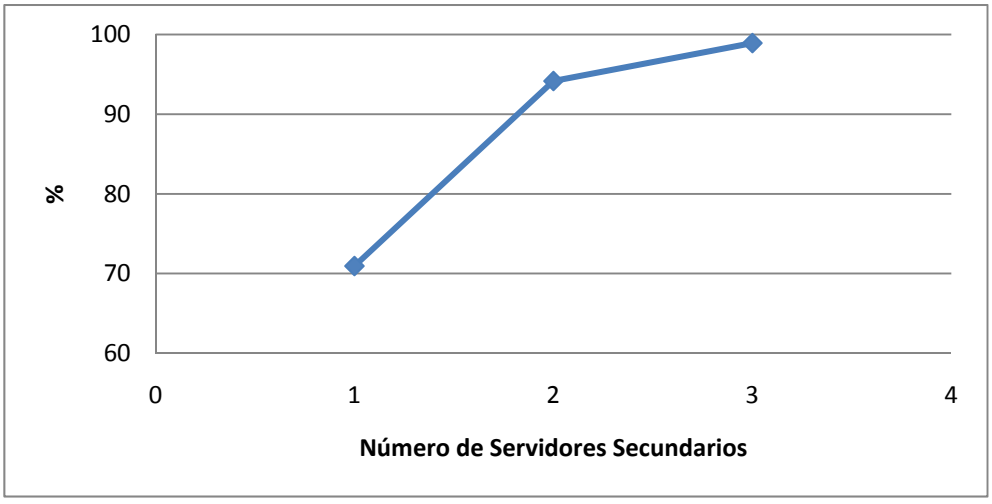
Gráfica 6.40 – Porcentaje medio de precintos respondidos por los Servidores Secundarios



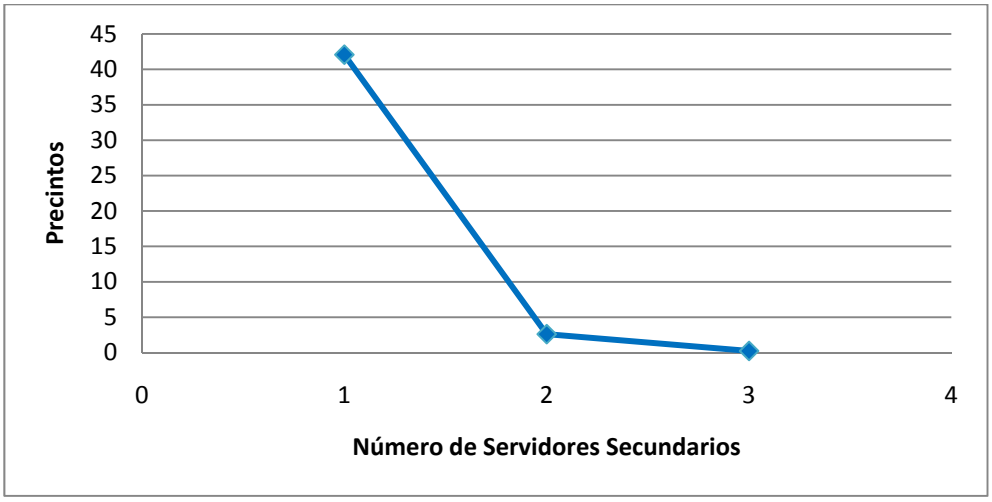
Gráfica 6.41 – Número de precintos medio de las colas de los Clientes



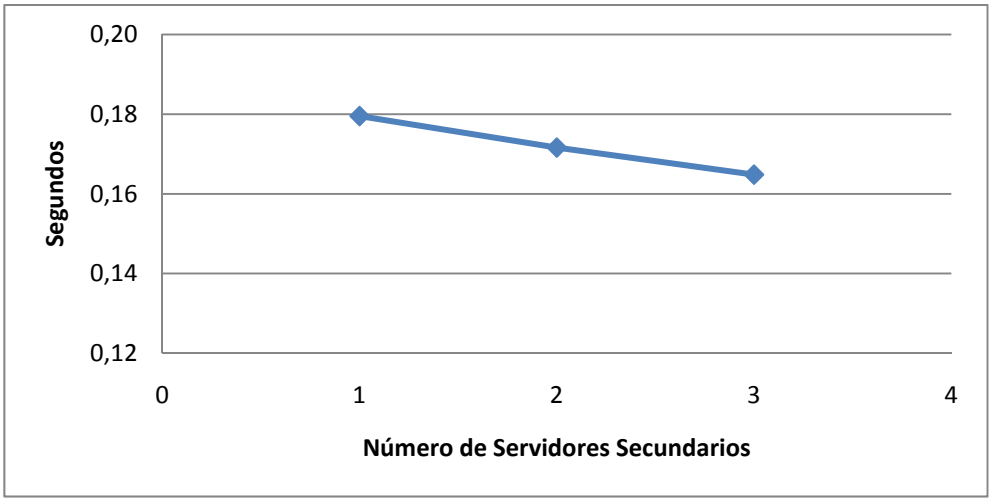
Gráfica 6.42 – Latencia media de los precintos respondidos por los Servidores Secundarios



Gráfica 6.43 – Porcentaje medio de los precintos encontrados con éxito en la Caché de los Servidores Secundarios



Gráfica 6.44 – Número de precintos medio de las colas de los Servidores Secundarios



Gráfica 6.45 – Latencia media global del intercambio de precintos en el Sistema

Conclusión

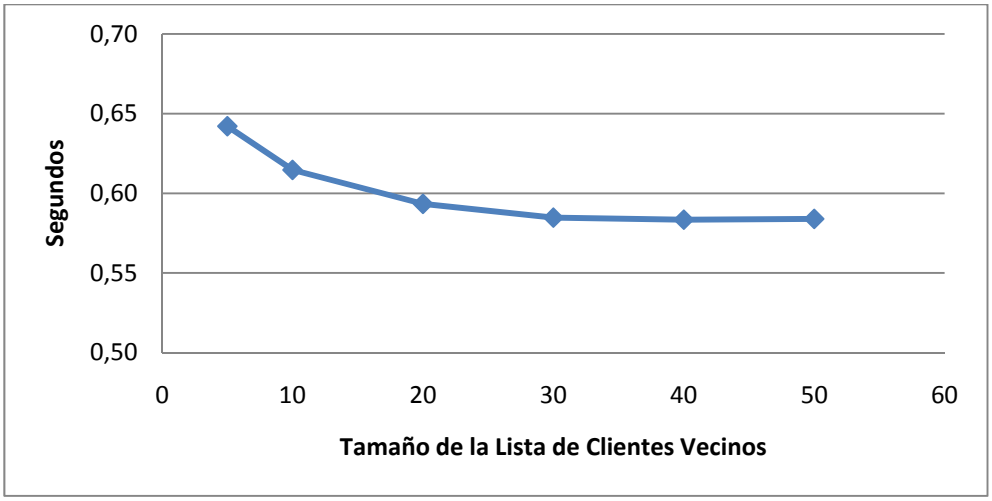
Como conclusión de esta prueba se puede decir que, al igual que para el sistema real con un número reducido de clientes conectados, parece que, a priori, cuando hay un número de clientes conectados elevado, el sistema continua funcionando mejor conforme mayor sea el número de servidores secundarios conectados al mismo.

Cabe hacer notar que, al haber un mayor número de clientes conectados al sistema, la mejora observada en el rendimiento del mismo es mayor que la que se producía en la prueba del sistema real, tal como se observa en el mayor descenso que se produce de la latencia global del intercambio de precintos (Gráfica 6.45).

6.4.2.4.5 – Variación en el tamaño de la lista de clientes vecinos

Esta prueba realizada con el simulador es la equivalente a la realizada con el sistema real que se ha mostrado en el apartado 5.4.6. Aparte de los cambios especificados en el apartado 6.4.2.1, los parámetros iniciales empleados van a ser los mismos que para aquella prueba, la única diferencia radica en el uso de un mayor número de clientes, que en este caso será de 1000.

Del conjunto de gráficas que se va a mostrar a continuación, la única en la que se puede observar un comportamiento diferente al visto en las gráficas obtenidas para el sistema real es en la Gráfica 6.46.



Gráfica 6.46 – Latencia media de los precintos respondidos por los Servidores Secundarios

En la Gráfica 6.46 se muestra la latencia media de los precintos respondidos por los servidores secundarios a los clientes. En ella se observa como a medida que se incrementa el tamaño de la lista de clientes vecinos, disminuye la latencia. Sin embargo, en la prueba del sistema real, se observaba un incremento en esa latencia (Gráfica 5.44).

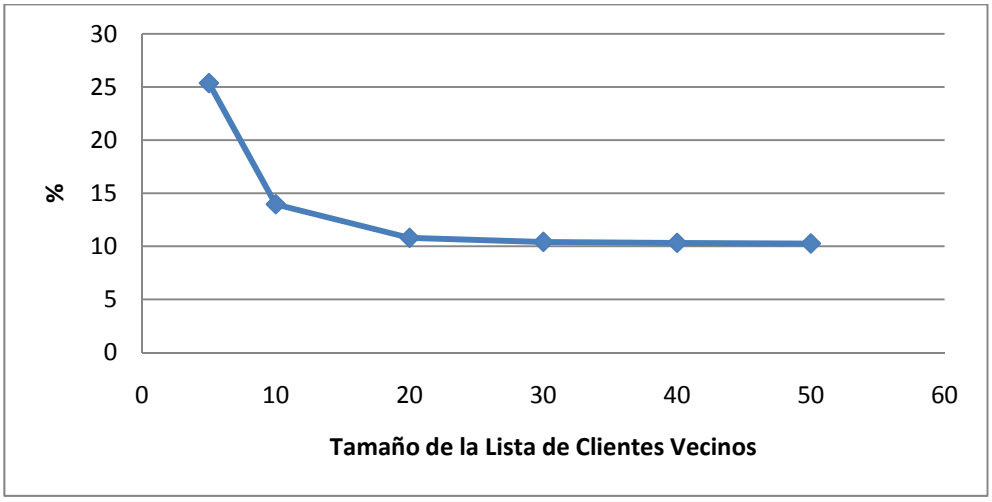
En la prueba del sistema real, se dedujo que este incremento en la latencia era debido al descenso del porcentaje de éxito de que los precintos pedidos por los clientes se encontraran en la caché del servidor secundario, lo que suponía una mayor necesidad de acceso al servidor principal a por la información, y con ello una mayor latencia.

A pesar de que la carga de los servidores secundarios era menor conforme mayor tamaño de la lista de clientes vecinos se empleaba en los clientes, debido a que los clientes tenían disponible una mayor cantidad de información en su caché y la podían intercambiar con sus clientes vecinos en vez de acudir a los servidores secundarios, este descenso de carga no era suficiente para compensar el aumento de latencia.

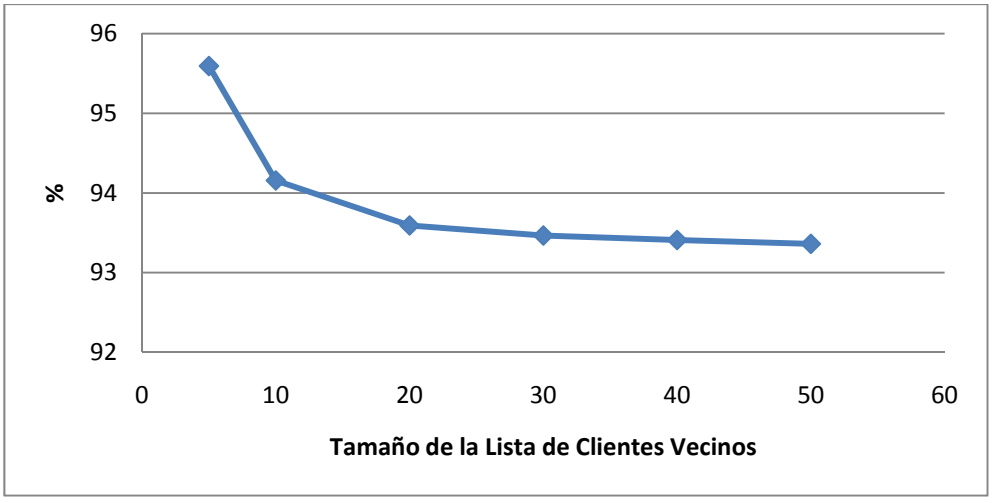
En la prueba realizada con el simulador, se observa que esa latencia disminuye ligeramente en vez de aumentar. Esto pensamos que se produce porque, debido al mayor número de clientes conectados al sistema, el impacto que tiene el aumento del tamaño de la lista de clientes vecinos en la necesidad de éstos de acudir al servidor secundario a por los precintos (Gráfica 6.47) es mayor que en la prueba del sistema real y, por lo tanto, pensamos que el descenso en la carga de los servidores secundarios, en esta ocasión, sí que compensa el aumento de latencia que se produce por el descenso del porcentaje de éxito de los

servidores secundarios (Gráfica 6.48), lo que supone globalmente una latencia de respuesta de precintos por parte del servidor secundario ligeramente inferior a medida que aumenta el tamaño de la lista de vecinos empleada en los clientes.

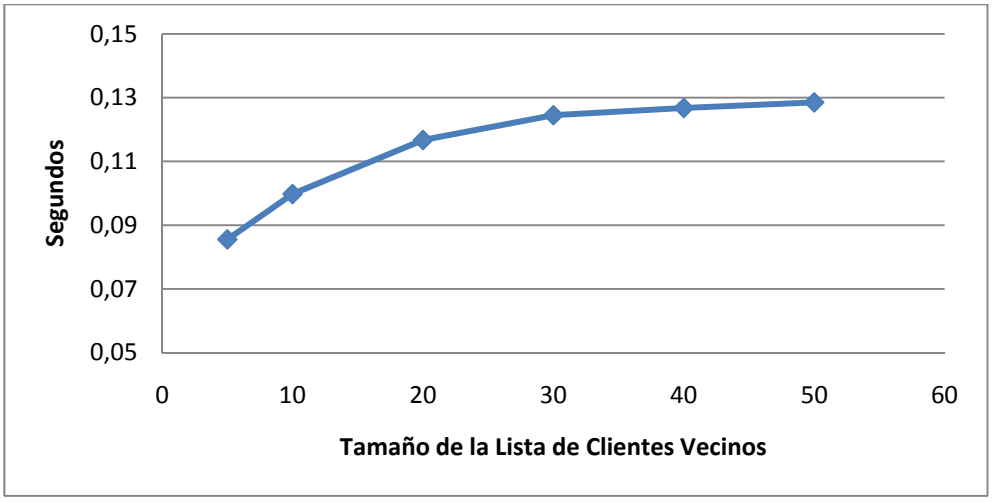
A continuación se van a mostrar el resto de gráficas de esta prueba. La explicación del comportamiento mostrado en ellas es la misma que la expuesta en la prueba del sistema real del apartado 5.4.6, por lo que no se van a volver a comentar de nuevo aquí.



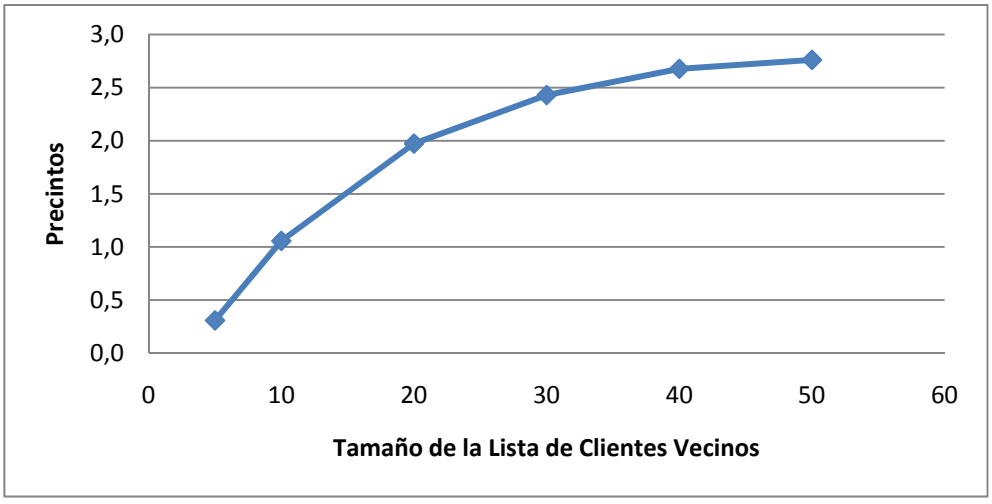
Gráfica 6.47 – Porcentaje medio de precintos respondidos por los Servidores Secundarios



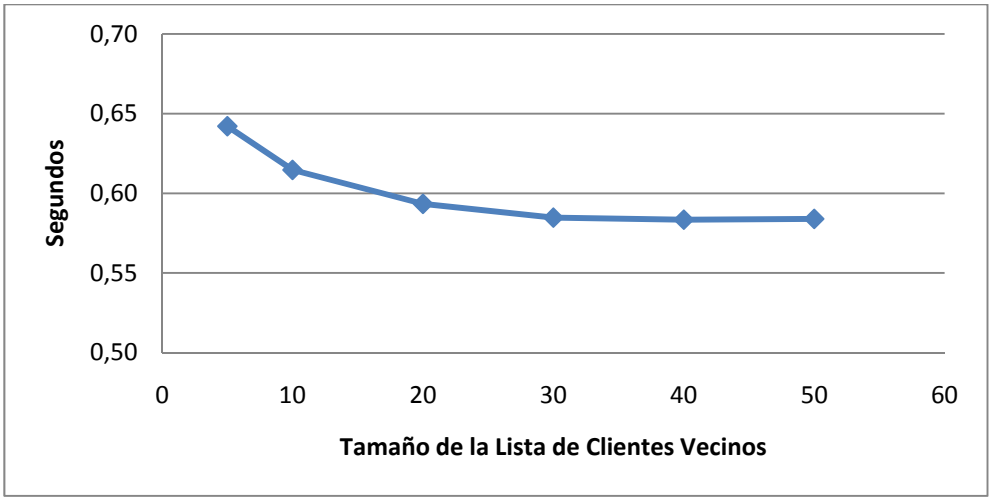
Gráfica 6.48 – Porcentaje medio de los precintos encontrados con éxito en la Caché de los Servidores Secundarios



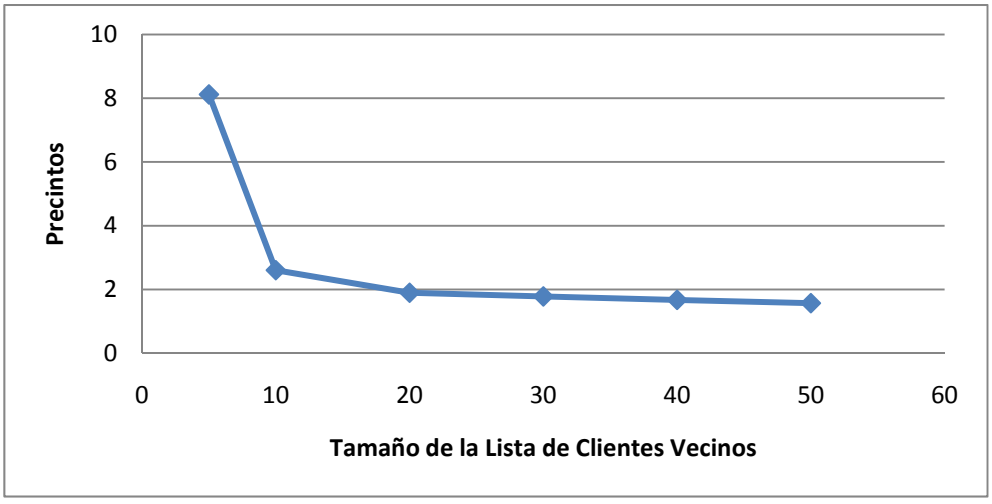
Gráfica 6.49 – Latencia media de los precintos intercambiados entre los Clientes



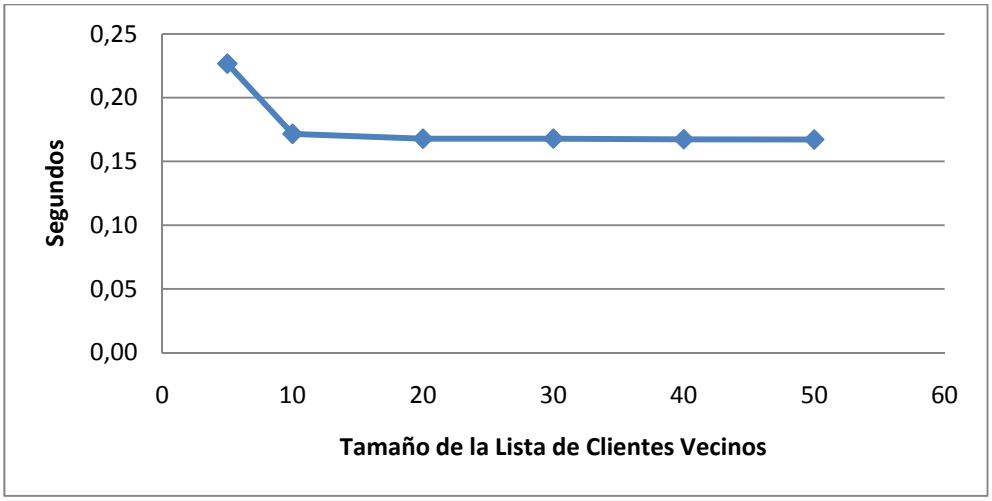
Gráfica 6.50 – Número de precintos medio de las colas de los Clientes



Gráfica 6.51 – Latencia media de los precintos respondidos por los Servidores Secundarios



Gráfica 6.52 – Número de precintos medio de las colas de los Servidores Secundarios



Gráfica 6.53 – Latencia media global del intercambio de precintos en el Sistema

Conclusión

Como conclusión de esta prueba se puede decir que, al igual que se observó para el sistema real empleando un número reducido de clientes, cuando se conecten un número más elevado de ellos, el sistema, a priori, seguirá comportándose mejor conforme mayor sea el tamaño de la lista de los clientes vecinos (Gráfica 6.53). Sin embargo, como se comentó para el sistema real, parece que un tamaño de lista de clientes vecinos superior a 10 apenas aporta una mejora de las prestaciones y en cambio sí que va a aumentar el tráfico que se produce en la red.

6.4.3 – Conclusiones de las pruebas del Simulador

El objetivo por el que se ha llevado a cabo el diseño de este simulador era poder realizar pruebas del sistema ante un conjunto diferente de condiciones de funcionamiento que no era posible llevar a cabo con el sistema real. Entre ellas, el uso del sistema con un número elevado de clientes conectados simultáneamente al mismo.

Para ello, una vez diseñado e implementado el simulador, se ha realizado un proceso de validación del mismo que nos ha permitido comprobar que el simulador, a priori, parece que se comporta de la misma forma que el sistema real. Este hecho nos va a permitir extrapolar los resultados obtenidos en las pruebas llevadas a cabo en el simulador, al funcionamiento esperado del sistema real.

Una vez realizada la validación del simulador, se ha procedido a realizar un conjunto de pruebas, las cuales eran básicamente las mismas que las llevadas a cabo para el sistema real, pero empleando un mayor número de usuarios conectados de forma simultánea. De estas pruebas se han extraído un conjunto de resultados que apoyan las conclusiones que se extrajeron de las pruebas del sistema real. Por lo tanto, se puede afirmar que el sistema que se ha llevado a cabo en esta tesis cumple con los objetivos planteados inicialmente, y permite una visualización interactiva de terrenos en entornos distribuidos empleando una arquitectura mixta cliente-servidor / P2P que aporta una mayor escalabilidad que los sistemas empleados habitualmente para este tipo de aplicaciones que emplean la arquitectura cliente-servidor.

Capítulo 7 – Conclusiones y Trabajo Futuro

A lo largo de esta tesis se ha diseñado un sistema de visualización interactiva de terrenos eficiente para entornos distribuidos. El diseño está enfocado a hacer frente a los nuevos retos que han supuesto el aumento de la cantidad y precisión de la información de las bases de datos del terreno, y el aumento de las líneas de conexión de banda ancha y la facilidad de acceso a las mismas, que han provocado una gran demanda de aplicaciones de este tipo en los últimos años.

7.1 – Conclusiones

El objetivo principal de la tesis desarrollada era el diseño de un sistema de visualización interactiva de terrenos eficiente para entornos distribuidos. Para ello se han tenido que resolver una serie de problemas asociados a este tipo de sistemas:

- Representación y visualización de la información del terreno. Los últimos avances en los dispositivos de adquisición de los datos del terreno han provocado un considerable aumento en la cantidad y en la precisión de la información almacenada en las bases de datos del terreno. Toda esta información no se puede representar y visualizar de forma interactiva sin llevar a cabo algún tipo de gestión o adaptación de la misma, puesto que suele sobrepasar con creces las capacidades de memoria y procesamiento de los equipos actuales. Por lo tanto, uno de los problemas que debe afrontar este tipo de sistemas es cómo llevar a cabo la gestión de toda esa información.

Para solucionar este problema, primero, en el Capítulo 1, se ha realizado un repaso de los algoritmos de representación y visualización de terrenos habitualmente usados en la literatura para este tipo de sistemas. De este repaso, se han seleccionado para su implementación y prueba aquellos que podrían adaptarse mejor a los requerimientos del tipo de sistema que se trata en esta tesis.

Después, en el Capítulo 3, se ha realizado una descripción más detallada de los algoritmos seleccionados, y se ha realizado una comparativa entre ellos con el objetivo de seleccionar el que cumpliera de una forma más adecuada los requerimientos del sistema.

Para la implementación de los algoritmos seleccionados, se han tenido en cuenta las nuevas capacidades técnicas que ofrecen las tarjetas gráficas actuales, las cuales no existían cuando se realizó la descripción original de algunos de ellos.

En las pruebas realizadas con estos algoritmos ya adaptados, se han empleado bases de datos locales, debido a que se quería evaluar las prestaciones que ofrecían cada uno de ellos, evitando que los resultados obtenidos estuvieran afectados por problemas en la transmisión de los datos.

En este ámbito, tras comparar los resultados obtenidos, se han podido extraer de ellos algunas conclusiones importantes acerca de su funcionamiento. Algunos de estos algoritmos ofrecen unas aproximaciones de mayor calidad a costa de conseguir unas velocidades de refresco bajas (algoritmo de SOAR), y al contrario, hay otros algoritmos que proporcionan mayores velocidades de refresco a costa de unas aproximaciones de menor calidad (algoritmo Geometry Clipmap). Por último, existen otros algoritmos que se encuentran en una situación intermedia (algoritmos NRQT y ROAM). Además, en función de la rugosidad de la superficie del terreno a visualizar, se ha constatado que algunos algoritmos ofrecen unos rendimientos mejores o peores.

De las pruebas con bases de datos locales se ha podido concluir que todos los algoritmos son más o menos adecuados para un sistema de visualización interactivo en tiempo real. Sin embargo, cuando se emplean bases de datos distribuidas, sólo uno de ellos cumple con todos los requisitos que impone el sistema sin necesidad de llevar a cabo modificaciones importantes en el funcionamiento del mismo. Este algoritmo es NRQT. Por ello, como conclusión de este capítulo, se decidió emplear el algoritmo de triangulación NRQT dentro del sistema desarrollado en esta tesis.

- Transmisión y almacenamiento de la información del terreno. A pesar del aumento que se ha producido en los últimos años en el ancho de banda de las líneas de conexión, la cantidad de información del terreno a transmitir desde los servidores a los clientes sigue siendo demasiado elevada y puede llegar a colapsar estas conexiones, provocando que el cliente no pueda acceder a la información con la rapidez adecuada para realizar una visualización del terreno interactiva en tiempo real. Además, esta información transmitida ha de ser almacenada en los clientes, por lo que éstos tendrán que disponer de un considerable espacio de almacenamiento dedicado a ello.

Para minimizar el problema, será necesario emplear algún esquema de compresión que permita la transmisión de la misma de forma más eficiente y reducir el tamaño de la información almacenada.

Para ello, primero, en el Capítulo 1, se ha realizado un repaso de los métodos de compresión y transmisión habitualmente usados en un sistema de visualización de terrenos. Estos sistemas emplean habitualmente esquemas de compresión basados en la transformada wavelet, ya que son los que mejores prestaciones les ofrecen.

En el Capítulo 4, de entre los esquemas introducidos en el Capítulo 1, nos hemos centrado para su estudio en profundidad en el esquema de compresión estándar Jpeg2000, ya que este esquema podía cumplir, a priori, con las condiciones requeridas por el sistema de visualización desarrollado en esta tesis. Sin embargo, durante el estudio se observó que el estándar presentaba una serie de problemas, como la aparición de artefactos visuales en las fronteras de las regiones cuando se reconstruían de forma independiente, o como que resultaba complicada su integración con la pirámide de cuadrantes empleada en la organización de los datos dentro del sistema.

Por ello, se decidió también llevar a cabo el diseño de un nuevo esquema de compresión y transmisión que estaba basado en el estándar Jpeg2000, de forma que heredaba parte de sus propiedades, pero al que se le han hecho una serie de modificaciones para evitar los problemas que presentaba éste.

Una vez elaborado este nuevo esquema, se compararon las prestaciones de ambos esquemas, tanto para las situaciones particulares que se presentan dentro del sistema de visualización de terrenos remoto, como para el resto de situaciones generales en las que es necesario el uso de un esquema de compresión.

Tras el análisis de los resultados obtenidos en estas pruebas, se vio que, para el sistema de visualización de terrenos remoto, el nuevo esquema obtenía unas reconstrucciones de mejor calidad y una mayor tasa de compresión que el esquema estándar, solucionando el problema de la aparición de artefactos visuales en las fronteras de las regiones del terreno reconstruidas de forma independiente, y generando reconstrucciones del terreno que se integraban de forma directa con la pirámide de cuadrantes. Para las pruebas de uso general del esquema de compresión, se vio que el nuevo esquema ofrecía prestaciones similares que el esquema estándar, siendo también adecuado su uso en estas situaciones. Debido a todo esto, se decidió que este nuevo esquema de compresión sería el que se iba a utilizar en el sistema desarrollado en esta tesis.

- Escalabilidad del sistema. Los sistemas de visualización de terrenos remotos cuentan en la actualidad con un gran número de usuarios, y se prevé que esa cantidad de usuarios aumente a medida de que las bases de datos sean más precisas y se elaboren nuevas aplicaciones para las mismas. La arquitectura que se ha empleado habitualmente en ese tipo de sistemas ha sido la arquitectura cliente-servidor que, por su naturaleza, se muestra como una arquitectura con una escalabilidad limitada, debido a que los servidores tienen que soportar toda la carga del sistema. Existen otras arquitecturas, como las arquitecturas P2P, que son inherentemente escalables con el número de usuarios, pero que tienen asociados otro tipo de problemas, como una mayor complejidad en la gestión y transmisión de la información, o la necesidad de que exista un número mínimo de usuarios conectados a estas redes que dispongan de la información a compartir. Por lo tanto, otro de los problemas de

este tipo de sistemas es diseñar una arquitectura que sea escalable con el número de usuarios conectados al sistema y, a ser posible, que no tenga un coste económico elevado.

Para solucionar este problema, se decidió diseñar una arquitectura híbrida basada en estas dos arquitecturas, la arquitectura cliente-servidor y la arquitectura P2P, que incorporaba las ventajas de ambas arquitecturas y evitaba sus problemas.

El diseño de esta arquitectura mixta se ha desarrollado en el Capítulo 5. En este capítulo se han explicado detalladamente cada uno de los elementos que forman parte de esta arquitectura y la metodología que se ha empleado para probar la misma en un entorno real. Para realizar la prueba de esta arquitectura, se ha realizado la implementación completa del sistema de visualización, integrando el algoritmo de triangulación NRQT, el nuevo esquema de compresión que se ha elaborado a partir del esquema estándar Jpeg2000 y la arquitectura mixta cliente-servidor / P2P.

Para poder comparar la escalabilidad de las dos arquitecturas, la nueva arquitectura mixta y la arquitectura clásica cliente-servidor, también se ha llevado a cabo la implementación de esta última.

A la hora de realizar las pruebas del sistema en un entorno real se han tenido algunas limitaciones, tales como el número de equipos disponibles, que ha restringido el número de clientes que podían conectarse de forma simultánea en el sistema, o la ubicación de los equipos, que ha restringido el tipo de las líneas de conexión utilizadas.

Para poder llevar a cabo un conjunto más amplio de pruebas del sistema evitando esas limitaciones, en el Capítulo 6 se ha realizado el diseño y la implementación de un simulador del sistema. Los resultados obtenidos con él, han servido para confirmar en parte los resultados obtenidos con el sistema real, apoyando las conclusiones obtenidas con éste.

Del conjunto de pruebas realizadas con el sistema real y con el simulador, se han obtenido unos resultados que indican que, empleando la arquitectura mixta cliente-servidor / P2P, el sistema tiene una mayor escalabilidad con el número de usuarios que haciendo uso de la arquitectura clásica cliente-servidor.

También se ha podido comprobar que el sistema adapta su comportamiento al número de clientes conectados al mismo, comportándose como una arquitectura cliente-servidor cuando el número de clientes es reducido, y aproximando su comportamiento al de una arquitectura P2P a medida que el número de clientes conectados aumenta.

Por lo tanto, como conclusión final, se puede afirmar que en esta tesis se ha desarrollado un sistema de visualización interactiva eficiente para entornos distribuidos, que cumple con los objetivos planteados inicialmente para el mismo:

- Emplea un algoritmo de representación y visualización de terrenos en tiempo real adecuado para bases de datos distribuidas.
- Utiliza un nuevo esquema de compresión y transmisión más cercano a la solución óptima para bases de datos del terreno distribuidas.
- Incorpora una nueva arquitectura de red apropiada para el acceso eficiente a los datos del terreno, que aporta una mayor escalabilidad con el número de usuarios conectados que las arquitecturas empleadas habitualmente en este tipo de sistemas.

7.2 – Resumen de las aportaciones

Las aportaciones más importantes realizadas esta tesis son:

- Se ha hecho un estudio comparativo del funcionamiento de los algoritmos de representación y visualización de terrenos utilizados habitualmente en la literatura. En este estudio, además de evaluar estos algoritmos cuando acceden a bases de datos locales, se ha prestado especial atención al caso en el que el acceso se realiza a bases de datos distribuidas, con el objetivo de encontrar el algoritmo más cercano a la solución óptima en ese caso. Además, se han adaptado estos algoritmos a las nuevas características técnicas implementadas en las tarjetas gráficas actuales.

- Se ha diseñado un nuevo esquema de compresión basado en el estándar Jpeg2000 que realiza una compresión óptima de los datos del terreno. Este esquema permite realizar una transmisión progresiva de los datos por niveles de detalle y por regiones independientes, generando reconstrucciones del terreno que se integran de forma directa con la estructura de pirámide de cuadrantes que se emplea habitualmente para organizar la información en los sistemas de visualización interactivos de terrenos extensos. Este esquema también evita la aparición de artefactos visuales en las fronteras de las regiones del terreno cuando son reconstruidas de forma independiente proporcionando unas reconstrucciones de mayor calidad visual, y proporciona unas tasas de compresión superiores al estándar Jpeg2000 para este tipo de datos. Además, este nuevo esquema mantiene unas prestaciones similares a las del estándar Jpeg2000 cuando se emplea fuera del ámbito de la visualización de terrenos, en los ámbitos habituales donde se utiliza el estándar.

- Se ha diseñado una nueva arquitectura de red mixta cliente-servidor / P2P para el acceso a las bases de datos del terreno distribuidas, que varía su comportamiento en función del número de usuarios conectados a la misma. El comportamiento es parecido al de una arquitectura cliente-servidor cuando el número de clientes es reducido, y se aproxima al de una arquitectura P2P cuando el número de clientes aumenta. Este doble comportamiento permite a la nueva arquitectura disfrutar de las ventajas que ofrecen ambas arquitecturas

evitando sus limitaciones, consiguiendo entre otras cosas, una mayor escalabilidad con el número de clientes conectados que con la arquitectura cliente-servidor, y asegurar la disponibilidad de toda la información del terreno en cada momento, a diferencia de una arquitectura P2P pura.

- Todos los elementos diseñados se integran perfectamente para crear un sistema de visualización interactivo eficiente para entornos distribuidos. El sistema incorpora un algoritmo de representación y visualización del terreno eficiente, un esquema de compresión y transmisión más cercano a la solución óptima para bases de datos del terreno distribuidas y una arquitectura de red eficiente para el acceso a los datos del terreno.

- En referencia a otras propuestas existentes como Google Earth, se ha desarrollado un sistema de código abierto que emplea una nueva arquitectura mixta cliente-servidor-P2P que permite reducir drásticamente el número de servidores necesarios para proporcionar una similar calidad de servicio a los usuarios, lo que disminuye considerablemente el coste económico de la instalación y el mantenimiento del mismo. Además, este sistema almacena de forma innata información relativa a los usuarios que se encuentran en la misma región del terreno, lo que simplifica la integración de aplicaciones sociales que lleven a cabo la interacción entre esos usuarios.

7.3 – Trabajo futuro

La realización de esta tesis deja abiertas varias líneas de trabajo que se podrían desarrollar para mejorar el sistema.

- **Empleo de conexiones dinámicas entre el servidor principal y los servidores secundarios**. Actualmente cada servidor secundario establece con el servidor principal dos líneas de conexiones estáticas, una para la descarga de la información de las texturas y otra para la de las alturas. Estas dos conexiones son las únicas establecidas entre cada servidor secundario y el servidor principal y siempre están activas independientemente de que se haga o no uso de ellas para obtener información del servidor principal. Esto puede provocar que haya líneas de conexión sin usarse, mientras que haya servidores secundarios que tengan las líneas de conexión muy ocupadas. Por lo tanto, para solucionar este problema, otra posible línea de trabajo futuro sería la utilización de líneas de conexión dinámicas, que se asignen a cada servidor secundario en función de las necesidades de comunicación del mismo con el servidor principal. Esta asignación dinámica permitirá aprovechar mejor el ancho de banda de las conexiones entre el servidor secundario y el servidor principal, reduciendo la carga de los servidores secundarios.

- **Desarrollo de servidores secundarios más escalables**. Los servidores secundarios en el sistema actual, pueden saturarse debido a que siempre hay un porcentaje de peticiones de precintos que deben asumir, tal como se ha observado en los resultados. Si el número de clientes conectados es elevado, puede llegar el momento en que los servidores secundarios no sean capaces de responder a las peticiones de precintos dentro de un tiempo aceptable. Por eso, otra línea de trabajo futuro sería buscar mecanismos que aumenten la escalabilidad de los servidores secundarios, como por ejemplo, enviando a los clientes, junto al conjunto de precintos que han pedido, un conjunto de precintos adicional que se estime que vaya a ser necesitado por el cliente en un breve periodo de tiempo, reduciendo el número de futuras peticiones que el cliente tenga que realizar al servidor secundario.

- **Desarrollo de nuevas técnicas de selección de clientes vecinos y servidores secundarios para el servidor de comunicaciones**. En la actualidad, el servidor de comunicaciones realiza, para cada cliente, la selección de los clientes vecinos y del servidor secundario más adecuado en cada momento. La selección del servidor secundario se realiza en función de la carga del servidor secundario y de una estimación del tiempo esperado de respuesta de los precintos por parte del servidor secundario. Dentro de esta estimación se emplea, entre otras variables, la predicción de que un precinto se encuentre o no en la caché del servidor secundario. La selección de los clientes vecinos se realiza en función de la distancia que existe entre la posición del cliente en la escena y la del cliente vecino, prediciendo que la presencia de los precintos que va a requerir un cliente va a ser más probable en los clientes vecinos más cercanos. En ambos casos se hace una selección en base a una predicción que puede no ser correcta. Por lo tanto, otra posible línea de trabajo sería la utilización de otras estrategias a la hora de realizar esta selección, como por ejemplo, que el servidor de comunicaciones conozca exactamente la información que tienen disponible en su caché los clientes y los servidores secundarios, en vez de usar predicciones. De esta forma, se aseguraría que la selección realizada es la más óptima posible.

- **Prueba de la escalabilidad del sistema en un entorno real**. A la hora de realizar pruebas en un entorno real con el sistema desarrollado en esta tesis, se ha tenido la limitación del número de equipos disponibles, que ha provocado que el número de clientes conectados al sistema fuera reducido. Para poder obtener resultados del funcionamiento del sistema con un número elevado de clientes conectados, se ha tenido que emplear un simulador del sistema real. Sin embargo, resultaría interesante comprobar que los resultados obtenidos con el simulador concuerdan con los realizados en un entorno real con un elevado número de clientes. Para ello, otra línea de trabajo podría ser la puesta en marcha del sistema en internet, con la adecuada publicitación del mismo, para atraer a usuarios desde cualquier punto del mundo. La información obtenida del uso del sistema por parte de estos usuarios sería muy útil para la corroboración de las conclusiones obtenidas con el simulador, y también para la mejora y la optimización del sistema.

- **Funcionalidades adicionales.** Actualmente, el sistema desarrollado en esta tesis sólo permite la representación y visualización de información del relieve y de texturas del terreno. Habitualmente, los sistemas de representación de terrenos como Google Earth o Virtual Earth 3D, incorporan otros elementos de información, como objetos 3D superpuestos sobre la superficie del terreno o capas de información superpuestas a la superficie del terreno como información topográfica, información de las calles, información meteorológica, etc. Por lo tanto, otra línea de trabajo sería la inclusión de estos nuevos elementos dentro del sistema, para lo cual habría que estudiar cómo realizar la codificación y la transmisión de esa información de forma eficiente, y cómo realizar la integración de la misma con la superficie del terreno para su visualización.

- **Adaptación del sistema a dispositivos móviles.** Actualmente el sistema desarrollado en esta tesis funciona sobre clientes ejecutados en ordenadores de sobremesa. El aumento en las capacidades gráficas de los dispositivos móviles como pdas o teléfonos móviles que se ha producido en los últimos años, unido al desarrollo de las líneas de conexión móviles y al abaratamiento y accesibilidad de las mismas, permite pensar en el uso del sistema en estos dispositivos. Por lo tanto, otra línea de trabajo futuro podría ser la integración de estos dispositivos al sistema, para lo que habría que adaptar entre otras cosas, los protocolos de comunicaciones y los mecanismos de visualización utilizados.

7.4 – Publicaciones relacionadas

R. Olanda, M. Pérez, X. Benavent, “Tiling of the Wavelet Lowpass Subbands for Progressive Browsing of Images”, Signal Processing Letters, IEEE, 680-683, 2006.

M. Pérez, R. Olanda, M. Fernández, “Visualization of Large Terrain Using Non-restricted Quadtree Triangulations”, Proceedings of the International Conference in Computational Science and Its Application (ICCSA’2004) – Lecture Notes in Computer Science, volume 3044, 671-681, 2004.

R. Olanda, M. Pérez, P. Morillo, M. Fernández, S. Casas, “Entertainment virtual reality system for simulation of spaceflights over the surface of the planet Mars”, Proceedings of de ACM symposium on Virtual Reality Software and Technology, 123-132, Limassol, Cyprus, 2006.

R. Olanda, M. Pérez, M. Fernández, “Triangulación basada en quadtrees no restrictivos para la representación de terrenos en tiempo real”, CEIG 2004, Sevilla, España.

R. Olanda, M. Pérez, I. Coma, “Texturado de terrenos empleando pirámides de cuadrantes progresivas y uniformes”, CEDI 2005, Granada, España.

[1] K. Aberer, "P-Grid: A self-organizing access structure for P2P information systems", Conference on Cooperative Information Systems, 2001.

[2] P. Alliez, C. Gostman, "Recent advances in compression of 3d meshes", Advances in Multirresolution for Geometric Modelling, Springer, 2005.

[3] A. Asirvatham, H. Hoppe, "Terrain Rendering Using GPU Based Geometry Clipmaps", Pharr, M., Ed. "GPU Gems 2". Addison-Wesley, 2005.

[4] N. Beatrice, S. Antonio, L. Rynson, L. Frederick, "A multiserver architecture for distributed virtual walkthrough", Proceedings ACM VRST'02, 163-170, 2002.

[5] F. Bettio, E. Gobetti, F. Marton, G. Pintore, "High quality networked terrain rendering from compressed bitstreams", Proceedings Web3D International Symposium ACM Press 37-44, 2007.

[6] J. Blow, "Terrain rendering at high levels of detail", Proceedings of the 2000 Game Developers Conference, 2000.

[7] Bosque J., Sistemas de Información Geográfica, Ediciones Rialp, Madrid, 1992.

[8] H. Bryhni, E. Klovning, O.Kure, "A comparison of load balancing techniques for scalable web servers", IEEE Network, July 2000.

[9] E. Buyukkaya, M. Abdallah, R. Cavagna, "VoroGame: A Hybrid P2P Architecture for Massively Multiplayer Games", IEEE Consumer Communications and Networking Conference (CCNC), 2009.

[10] V. Cardellini, M. Colajanni, P. Yu, "Dynamic load balancing on web-server systems", IEEE Internet Computing, 28-29, June 1999.

[11] N. A. Carr, J. Hoberock, K. Crane, J.C. Hart, "Fast GPU ray tracing of dynamic meshes using geometry images", Graphics Interface (2006), 203-209, 2006.

[12] R. Cavagna, C. Bouville, J. Royan, "P2P Network for Very Large Virtual Environment," *Proc. ACM Virtual Reality Software and Technology*, ACM Press, 269-276, 2006.

[13] C. Cebenoyan, M. Wloka, "Optimizing the Graphics Pipeline", Nvidia, GDC 2003,http://developer.nvidia.com/docs/IO/8230/GDC2003_PipelinePerformance.pdf, 2003.

[14] P. Cignoni, E. Puppo, R Scopigno. "Representation and visualization of terrain surfaces at variable resolution". The Visual Computer, 13(5): 199-217, 1997.

[15] P. Cignoni, F. Ganovelli, E. Gobetti , F. Marton, F. Ponchio, R. Scopigno, "Bdam - Batched Dynamic Adaptive Meshes for High Performance Terrain Visualization" . Eurographics 2003. Computer Graphics Forum 22, 3 (2003), 505 – 514, 2003.

[16] P. Cignoni, F. Ganovelli, E. Gobetti , F. Marton, F. Ponchio, R. Scopigno, "Planet-Sized Batched Dynamic Adaptive Meshes (P-BDAM)" , IEEE Visualization 2003, 147-154, 2003.

[17] J. Clark, "Hierarchical geometric models for visible surface algorithms", Communications of the ACM, 547-554, 1976.

[18] M. Clasen, C.H. Hege, "Terrain rendering using spherical clipmaps", Proceedings EuroVis, 91.-98, 2006.

[19] D. Cline, P. Egbert, "Interactive display of very large textures. In Proceedings of IEEE Visualization'98" (1998), 343-350.

[20] D. Cline, P. Egbert, "Terrain Decimation through Quadtree Morphing", IEEE Transactions on Visualization and Computer Graphics, pp62-69, 2001.

[21] D. Cohen-Or, E. Rich, U. Lerner, V. Shenkar, "A real-time photo-realistic visual fly-through". IEEE Transactions on Visualization and Computer Graphics 2,3 (1996), 1077-2626.

[22] H. Date, S. Kanais, T. Kishinami, "An adaptive lod control method for textured digital terrain model using wavelet-based multiresolution representation". In IEEE 2001 International GeoScience and Remote Sensing Symposium (IGARSS'01) (2001), vol. 4, 1847 - 1849

[23] I. Daubechies, "Ten Lectures on Wavelets". Philadelphia, PA: SIAM, 1992, vol. 61, CBMS-NSF Regional Conf. Series in Applied Mathematics.

[24] D. Davis, T.Y. Jiang, W. Ribarsky, N. Faust, ”Intent, perception and out-of-core visualization applied to terrain”. In Proceedings of IEEE Visualization'98 (1998), 455 - 458.

[25] L. De Floriani, “A pyramidal data structure for triangle-based surface description”, IEEE Computer Graphics & Applications, 67-78, 1989.

[26] L. De Floriani, E. Puppo, “Hierarchical triangulation for multirresolution surface description”, ACM Tansactions on Graphics, 363-411, 1995.

[27] L. De Floriani, P. Marzano, E. Puppo, “Multirresolution models for topographic surface description”, The Visual Computer, 317-345, 1996.

[28] L. De Floriani, P. Magillo, E. Puppo, “Building and traversing a surface at variable resolution”, IEEE Visualization, 103-110, 1997.

[29] L. De Floriani, L. Kobbelt, E. Puppo, “A survey on Data Structures for Level-Of-Detail Models”, Advances in Multiresolution for Geometric Modelling”, Mathematics and Visualization, 49-74. Springer Verlag, 2004.

[30] S. Deb, P. Narayanan, “Streaming terrain rendering”, Proceedings SIGGRAPH 2006 Sketches, 2006.

[31] B. Delaunay. “sur la sphere vide. A la memoire de Georges Voronoi”, Izv. Akad. Nauk SSSR, Otdelenie Matematicheskih i Estestvennyh Nauk, 793-800, 1934.

[32] C. Dick, J. Schneider, R. Westermann, “Efficient geometry compression for GPU-based decoding in realtime terrain rendering”, Computer Graphics Forum 28, 2009.

[33] C. Dick, J. Schneider, R. Westermann, “GPU Ray-Casting for Scalable Terrain Rendering”, Eurographics 2009 – Area Papers, 43-50, 2009.

[34] J. Döllner, K. Baumann, K. Hinrichs, “Texturing techniques for terrain visualization”. In Proceedings of IEEE Visualization'00 (2000), 227 - 234.

[35] J. Duato, S. Yalamanchili, L. Ñi. “Interconnection Networks: An Engineering Approach”, IEEE Computer Society Press, 1997.

[36] M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich and M. B. Minnev-Weinstein, "ROAMing terrain: Real-Time optimally adapting meshes", Proceedings of the IEEE Visualization'97", 81-88, 1997

[37] M. Duchaineau. "ROAM algorithm version 2.0 – work in progress", http://www.cognigraph.com/ROAM_homepage/ROAM2.

[38] Estudio del Ancho de banda – Report:"http://www.sbs.ox.ac.uk/newsandevents/Documents/Broadband%20Quality%20Study%202009%20Press%20Presentation%20(final).pdf", 2009

[39] Estudio del Ancho de banda – Apéndice: "http://www.sbs.ox.ac.uk/newsandevents/Documents/Broadband%20Quality%20Study%202009%20Appendix.pdf", 2009.

[40] W. Evans, D. Kirkpatrick, G Townseed, "Right triangulated irregular networks", Algorithmica, 30(2): 264-286, 2001.

[41] Everquest, "http://everquest.station.sony.com".

[42] E. Frecon, M. Stenius, "Dive: A scalable network architecture for distributed virtual environments", Distributed Systems Engineering Journal, 91-100, September 1998.

[43] D. Frey, J. Royan, R. Piegay, A.M. Kermarrec, E. Anceaume, F. Le Fessant, "Solipsis: A Decentralized Architecture for Virtual Environments", International Workshop on Massively Multiuser Virtual Environtments (MVVE), 29-33, 2008.

[44] M. Garland, P. S. Heckbert, "Fast polygonal approximation of terrains and height fields", Technical Report 97-09, Department of Computer Science, University of Arizona, 1997.

[45] M. Garland, P. S. Heckbert, "Surface simplification using quadric error metrics", SIGGRAPH '97, pp 209-216, 1997.

[46] T. Gerstner, "Multiresolution visualization and compression of global topographic data", Technical report 29, Institut für Angewandte Mathematik, UniversityBoon 1999.

[47] P. Gioia, O. Aubault, and C. Bouville, "Real-Time Reconstruction of Wavelet-Encoded Meshes for View-Dependent Transmission and Visualization," IEEE Trans. Circuits Systems and Video Technology, vol. 14, no. 7, 2004, 1009-1020.

[48] E. Gobbetti, F. Marton, P. Cignoni, M. D. Benedetto, F. Ganovelli, "C-BDAM – Compressed batched dynamic adaptative meshes for terrain rendering", Computer Graphics Forum 25,3, Proceedings Eurographics 2006.

[49] GoogleEarth, "http://earth.google.es", 2010.

[50] GoogleMaps, "http://maps.google.es", 2010.

[51] G. Guennebaud , L. Barthe, M. Paulin, "Real-time soft shadow mapping by backprojection", Eurographics Symposium on Rendering (2006), 227-234, 2006.

[52] gvSIG, "http://www.gvsig.gva.es", 2010.

[53] gvSIG 3D, "http://gvsig3d.blogspot.com", 2010.

[54] M. E. Goss, K. Yuasa, "Texture tile visibility determination for dynamic texture loading". In In Proceedings of the EUROGRAPHICS/SIGGRAPH workshop on Graphics Hardware (1998), 55-60.

[55] M. H. Gross, R Gatti, O. G. Staadt, "Fast multiresolution surface meshing", In Proceedings Visualization 95, IEEE Computer Society Press, 135-142.

[56] P. Haeberli, M. Segal, "Texture Mapping as a Fundamental Drawing Primitive", Proceedings of the 4º Eurographics Workshop on Rendering, 259-266, 1993.

[57] H.V. Herzen, A. Barr, "Accurate triangulations of deformed intersecting surfaces", Siggraph'87, Computer Graphics Proceedings, 103-110, 1987.

[58] H. Hoppe, "Progressive Meshes", Proceedings SIGGRAPH 96, 99-108, ACM SIGRGRAPH, 1996.

[59] H. Hoppe, "View-dependant refinement of progressive meshes". In SIGGRAPH 97 Conference Proceedings (1997), 189–198.

[60] H. Hoppe, "Smooth view-dependent level-of-detail control and its application to terrain rendering", IEEE Proceedings Visualization 98, 35-42, 1998.

[61] S.Y. Hu, S. C. Chang, and J. R. Jian, "Voronoi State Management for Peer-to-Peer Massively Multiplayer Online Games". IEEE Network, 1134-1138, 2008.

[62] S.Y. Hu, J. F. Chen, and T. H. Che, "Von: A scalable peer-to-peer network for virtual environments". IEEE Network, 2006, 22-31.

[63] S.Y. Hu, “A case for 3d streaming on peer-to-peer networks”. Web3D 2006 Symposium proceedings, 2006.

[64] S.Y. Hu, T.H. Huang, S.C. Chang, W.L. Sung, J. R. Jiang, and B. Y. Chen, “FLod: A Framework for Peer-to-Peer 3D Streaming”, INFOCOM 2008.

[65] T. Hüttner, “High resolution textures”. Proceedings of IEEE Visualization'98 (1998), 13-17.

[66] L. M. Hwa, M. A. Duchaineau, K. I. Joy, “Real-time optimal adaptation for planetary and texture: 4-8 tile hierarchies”, IEEE Transactions on Visualization and Computer Graphics 11, 4, 335-368, 2005.

[67] W. Pennebaker, J. Mitchell, “Jpeg: Still image compression standard”, 1993.

[68] JPEG2000 - ISO/IEC, ISO/IEC 15444-1: Information Technology, JPEG2000 Image Coding System: Core Coding System, Jpeg Committee 2004.

[69] JPEG2000-9 - ISO/IEC 15444. Information Technology – JPEG2000, Image Coding System – Part 9: Interactive tools, APIs and protocols, 2003.

[70] R. L. C. JR., G. M. Davis, W. Sweldens, R. G. Baraniuk, “Nonlinear wavelet transform for image coding via lifting”, IEEE Transactions on Image Processing 12, 12, 1449-1459, 2003.

[71] Kakadu, “www.kakadusoftware.com”, 2010.

[72] Y. Kawahara, T. Aoyama, H. Morikawa, “A peer-to-peer message Exchange scheme for large scale networked virtual environments”, Telecommunication Systems, 353-370, 2004.

[73] J. Keller and G. Simon. “Solipsis: A massively multi-participant virtual world”, Parallel and Distributed Processing Techniques and Applications (PDPTA), 2003.

[74] J. K. Kim, J. B. Ra, “A real-time terrain visualization algorithm using wavelet-based compression”, The Visual Computer, 67-95, 2004.

[75] J. Kovacevic, W. Sweldens, “Wavelet families of increasing order in arbitrary dimensions”, IEEE Transactions on Image Processing 9, 3, 480-496, 2000.

[76] F. Kuhl, R. Weatherly, J. Dahmann, “Creating Computer Simulation Systems: An Introduction to the High Level Architecture”, 1999.

[77] R. Lario, R. Pajarola, F. Tirado, "HyperBlock-QuadTIN: Hyper-block Quadtree based Triangulated Irregular Networks", IASTED International Conference on Visualization, Imaging and Image Processing, 733-738, 2003.

[78] J. Lee, "Comparison of existing methods for building triangular irregular network models of terrain from grid digital elevation models", International Journal of Geographic Information Systems, 267-285, 1991.

[79] J. Levenberg, "Fast View-Dependant Level-of-Detail Rendering Using Cached Geometry", Proceedings IEEE Visualization'02, 259-266, 2002.

[80] N.-S. Lin, T.-H. Huang, B.-Y. Chen, "3d model streaming based on jpeg 2000", *IEEE TCE*, vol. 53, no. 1, 2007.

[81] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. Faust and G. A. Turner. "Real-Time, Continuous Level of Detail Rendering of Height Fields". In SIGGRAPH 96 Conference Proceedings, pp. 109-118, Aug 1996.

[82] P. Lindstrom and V. Pascucci, "Visualization of large terrains made easy", Proceedings of the IEEE Visualization 2001.

[83] P. Lindstrom, and V. Pascucci, "Terrain simplification simplified: A general framework for view-dependant out-of-core visualization", IEEE Transactions on Visualization and Computer Graphics, 8 (2):239-354, 2002.

[84] F. Lossaso and H. Hoppe, "Geometry clipmaps: terrain rendering using nested regular grids" ACM transactions on Graphics (2004), 769-776.

[85] D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, R. Huebner, "Level of Detail for 3D Graphics", Morgan Kaufman Publisher, 2003.

[86] J. C. S. Lui, M.F. Chan, "An efficient partitioning algorithm for distributed virtual environment systems", IEEE Trans. Parallel and Distributed Systems, 2002.

[87] H. Malvar, 2000. "Fast Progressive Image Coding without Wavelets", Data Compression Conference (DCC '00), 243-252.

[88] M. Matijasevic, K.P. Valavanis, D, Gracanin, I. Lovrek, "Application of a multiuser distributed virtual environment framework to mobile robot teleoperation over the internet", Machine Intelligence & Robotic Control, 11-26, 1999.

[89] P. Morillo, J.M. Orduña, M. Fernández, "Workload characterization in multiplayer online games", Lecture Notes in Computer Science, 490-499, 2006.

[90] P. Morillo, J.M. Orduña, M. Fernández, J. Duato, "On the characterization of avatars in distributed virtual worlds", Eurographics 2003 Workshops, 215-220, The Eurographics Association, 2003.

[91] P. Morillo, J.M. Orduña, M. Fernández, J. Duato, "On the characterization of distributed virtual environtment systems", Euro-Par' 2003 – Lecture Notes in Computer Science 2790, 1190-1198, ACM, 2003.

[92] P. Morillo, J.M. Orduña, M. Fernández, J. Duato, "Improving the performance of distributed virtual environment systems" IEEE Transactions on Parallel and Distributed Systems, 637-649, 2005.

[93] M. Pérez, R. Olanda, M. Fernández, "Visualization of Large Terrain Using Non-restricted Quadtree Triangulations", Proceedings of the International Conference in Computational Science and Its Application (ICCSA'2004) – Lecture Notes in Computer Science, volume 3044, 671-681, 2004.

[94] K. Oh, H. Ki, C.H. Lee, "Pyramidal displacement mapping: A GPU based artefacts-free ray tracing through an image pyramid", ACM symposium on Virtual Reality Software and Technology, 75-82, 2006.

[95] R. Olanda, M. Pérez, P. Morillo, M. Fernández, S. Casas, "Entertainment virtual reality system for simulation of spaceflights over the surface of the planet Mars", Proceedings of de ACM symposium on Virtual Reality Software and Technology, 123-132, Limassol, Cyprus, 2006.

[96] R. Olanda, M. Pérez, X. Benavent, "Tiling of the Wavelet Lowpass Subbands for Progressive Browsing of Images", Signal Processing Letters, IEEE, 680-683, 2006.

[97] R. Pajarola, "Large scale Terrain Visualization using the Restricted Quadtree Triangulation". Proceedings IEEE Visualization '98, 19-26, 1998.

[98] R. Pajarola, "Overview of quadtree based terrain triangulation and visualization", Technical Report UCI-ICS TR 02-01, Department of Information, Computer and Sciencie University of California, Irvine, 2002.

[99] R. Pajarola, M Antonijuan, R Lario, "QuadTIN: Quadtree based Triangulated Irregular Networks", Proceedings IEEE Visualization 2002, 395-402, 2002.

[100] R. Pajarola, E. Gobbetti, "Survey of semi-regular multirresolution models for interactive terrain rendering", Journal The Visual Computer, 583-605, 2007.

[101] J. Peng, C. S. Kim, C. C. J. Kuo, "Technologies for 3D mesh compression: A Survey", Journal of Visual Communication and Image Representation, 2005.

[102] M. Pérez, "Utilización de la Transformada Wavelet para la representación de terrenos en entornos de simulación", Tesis doctoral, Valencia 2002.

[103] A. A. Pomeranz, "ROAM using Surface Triangle Clusters (RUSTiC)", Master's thesis, University of California, 2000.

[104] G.V. Popescu, C.F. Codella, "An Architecture for QoS Data Replication in Network Virtual Environments," Proc. IEEE Virtual Reality Conf., IEEE CS Press, 2002, 41-48.

[105] Alturas y texturas del modelo "Puget sound" ubicadas en la dirección url "http://www.cc.gatech.edu/projects/large_models/ps.html".

[106] Quake, "http://www.idsoftware.com/games/quake".

[107] B. Rabinovich, C. Gotsman, "Visualization of large terrains in resource-limited computing environments". In Proceedings of IEEE Visualization'97 (1997), 97-102.

[108] M. Reddy, Y. Leclerc, L. Iverson, N. Blettern, "Terravision II: Visualizing massive terrain databases in vrml". IEEE Computer Graphics & Applications 19, 2 (1999), 30-38.

[109] W. Ribarsky, T. Wasilewski, N. Faust, "From urban terrain models to visible cities", IEEE Computer Graphics & Applications, 231-238, 2002.

[110] Röttger, S, W Heidrich, P Slussallek, and H-P Seidel. Real-Time Generation of Continuous Levels of Detail for Height Fields. Proceedings of 1998 International Conference in Central Europe on Computer Graphics and Visualization. pp. 315–322. 1998.

[111] J. Royan, P. Gioia, R. Cavagna, C. Bouville, "Network-based visualization of 3D landscapes and city Models", IEEE Computer Graphics and Applications 2007, 70-79, 2007.

[112] J. Royan, P. Gioia, R. Cavagna, C. Bouville, "Peer-to-Peer visualization of very large 3d landscape and city models using Mpeg4", 3DTV Conference: The True Vision – Capture, Transmission and Display of 3D Video (2008), 93-96, 2008.

[113] H. Samet, "Applications of Spatial Data Structures". Addison-Wesley. Reading, 1990.

[114] J. Schneider, R. Westermann, "GPU-Friendly High-Quality Terrain Rendering", Journal of WSCG, Plzen, Czech Republic, 2006.

[115] A. Skodras, C. Christopoulos, Ebrahimit, "The jpeg2000 still image compression standard." IEEE Signal Processing Magazine (2001), 36.58.

[116] S. Singhal, M. Zyda, "Networked Virtual Environment", ACM Press, 1999.

[117] P. Lindstrom, V. Pascucci, "SOAR Terrain Engine", "http://www-static.cc.gatech.edu/~lindstro/software/soar".

[118] SketchUP, "http://sketchup.google.com", 2010.

[119] Shuttle Radar Topography Mission, "http://www2.jpl.nasa.gov/srtm".

[120] S. Singhal, M. Zyda, "Networked Virtual Environments", ACM Press 1999.

[121] J.Spitzer, "Graphics Performance Optimisation", Developer NVIDIA documents, "http://developer.nvidia.com/docs/IO/8343/Performance-optimisation.pdf".

[122] R. Sudhakar, R. Karthiga, s. Jayaraman, "Image Compression using Coding of Wavelet Coefficients – A survey", GVIP journal, 5, 6, 2005.

[123] W. L. Sung, S.Y. Hu, J.R. Jiang, "Selection Strategies for Peer-to-Peer 3D Streaming", NOSSDAV 2008, Germany, 2008.

[124] C. Tanner, C. J. Migdal, M. T. Jones, "The clipmap: A virtual mipmap. In SIGGRAPH'1998" Computer Graphics Proceedings (1998), 151 - 159.

[125] D.S. Taubman. "High Performance Scalable Image Compression with EBCOT", IEEE Transactions on image processing, 1158–1170, July 2000.

[126] D. S. Taubman and M. W. Marcellin,"JPEG2000: Image Compression Fundamentals; Standards and Practice". New York: Springer, 2002.

[127] D. Taubman and R. Prandolini, "Architecture, philosophy, and performance of JPIP: Internet protocol standard for JPEG 2000," presented at the SPIE Conf. Visual Communication and Image Processing Lugano, Switzerland, 2003.

[128] A. Tevs, I. Ihrke, H.P Seidel, "Maximum mipmaps for fast, accurate and scalable dynamic height field rendering", ACM symposium on Interactive 3D Graphics and games (2008), 183-190, 2008.

[129] Universidad de Oviedo, Departamento de Economía Aplicada, http://www.uniovi.es/ecoapli/.

[130] Universidad de Oxford, Saïd Business School, http://www.sbs.ox.ac.uk/Pages/default.aspx.

[131] A. Voigtmann, L Becker, K Hinrichs, "Hierarchical surface representations using constrained Delaunay triangulations", Proceedings 6th International Symposium on Spatial Data Handling, 848-867, 1994.

[132] L. Williams, "Pyramidal parametrics". In SIGGRAPH'83, Computer Graphics Proceedings (1983), 1 - 11.

[133] E. Yusov, V. Turlapov, "Dynamic terrain simplification based on Haar transform and vertices quadtree", Proceedings of Conference on Computer Graphics and Applications - GraphiCon, 2006.

[134] S. Zheng, Z. Yu, Z. Li, L. Gau "PeerTR: a Peer-to-Peer Terrain Roaming Architecture", Advanced parallel Processing Technologies (2007), LNCS 4847, 292-300, 2007.

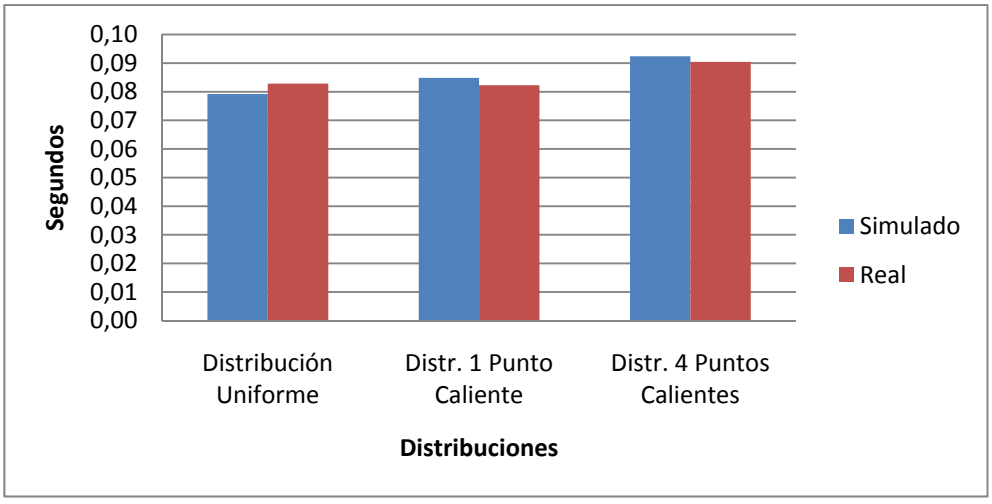
[135] Virtual Earth 3D, "http://www.microsoft.com/virtualearth", 2010.

[136] 3DVIA, "http://www.3dvia.com", 2010.

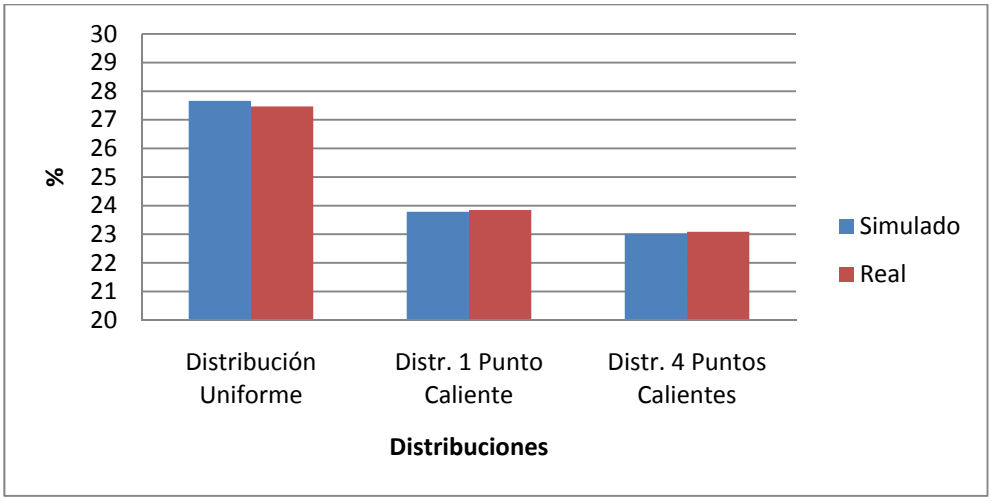
APÉNDICE A – Ajuste del Simulador

A continuación se muestran los resultados de las pruebas que se realizaron para evaluar el sistema real de visualización de terrenos en tiempo real comparados con los resultados de esas mismas pruebas pero empleando el simulador.

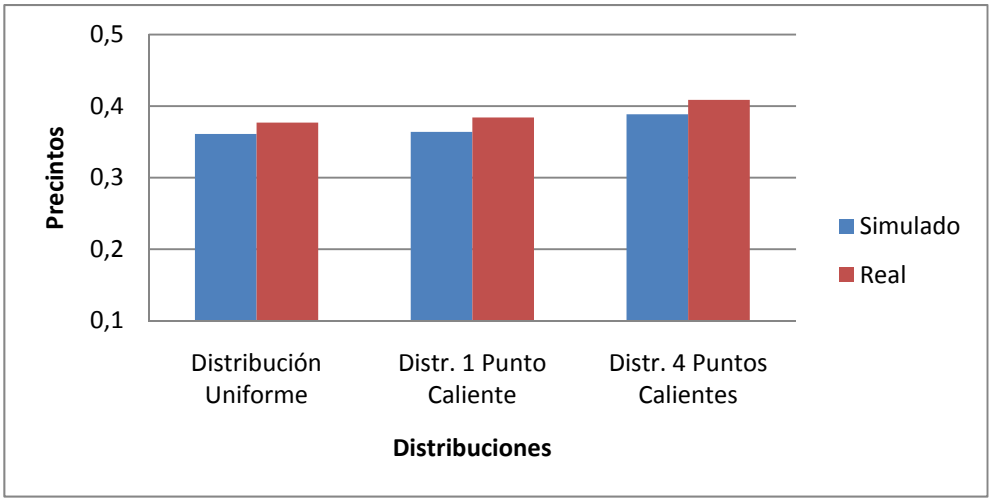
A.1 – Variación en la distribución de la ruta



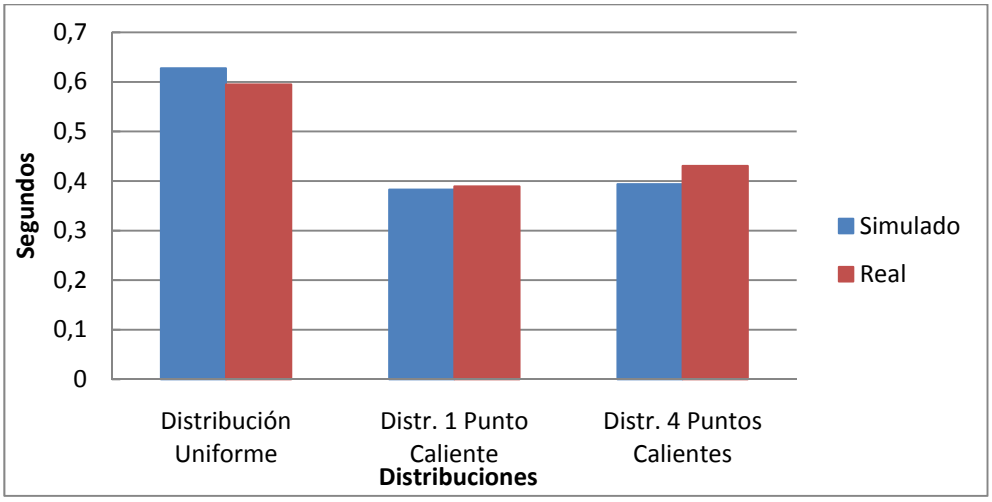
Gráfica A.1 – Comparativa de la latencia media de los precintos intercambiados entre los Clientes



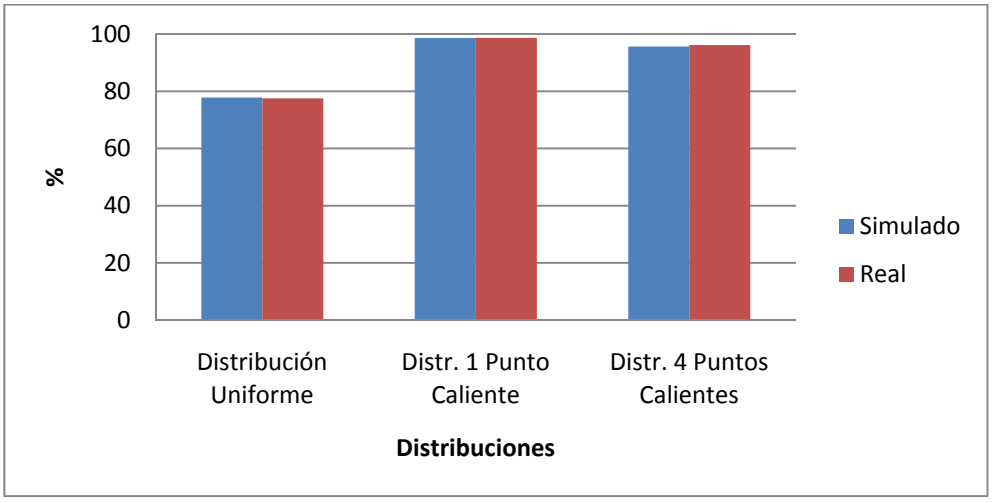
Gráfica A.2 – Comparativa del porcentaje medio de precintos respondidos por los Servidores Secundarios



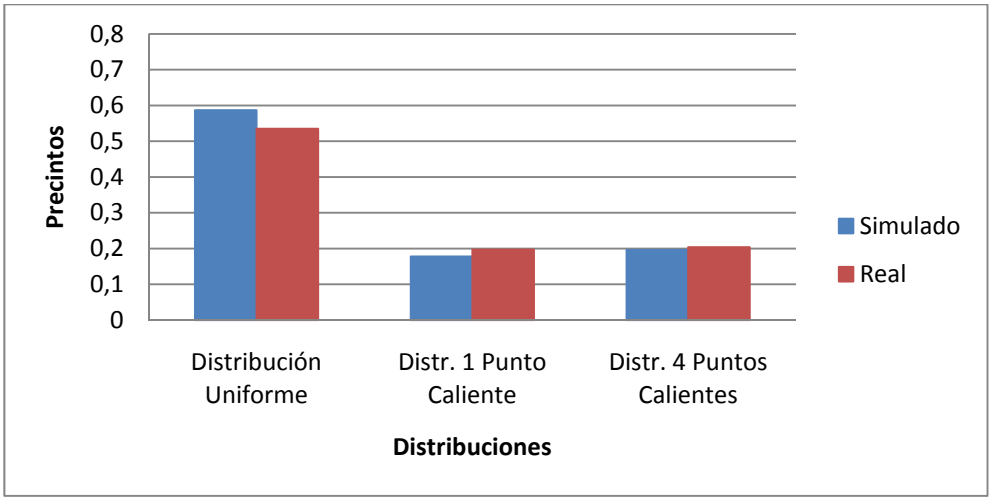
Gráfica A.3 – Comparativa del número de precintos medio de las colas de los Clientes



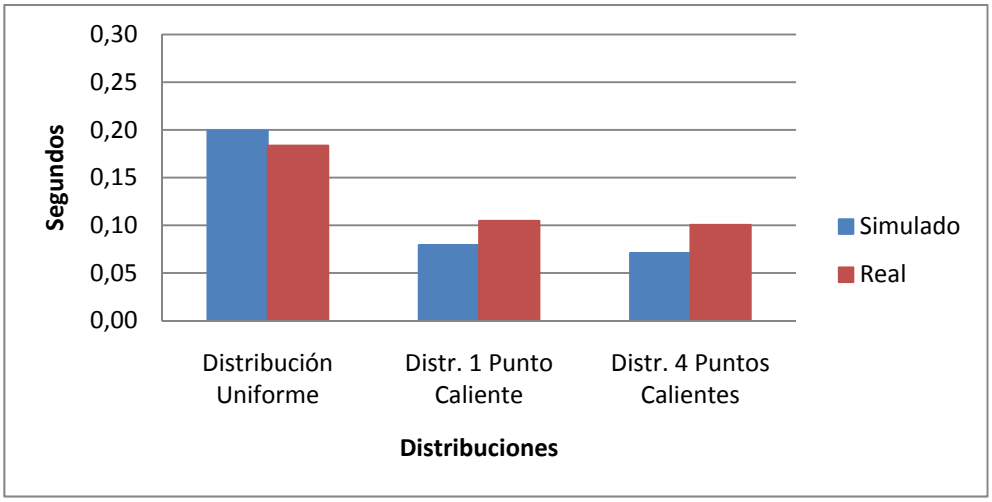
Gráfica A.4 – Comparativa de la latencia media de los precintos respondidos por los Servidores Secundarios



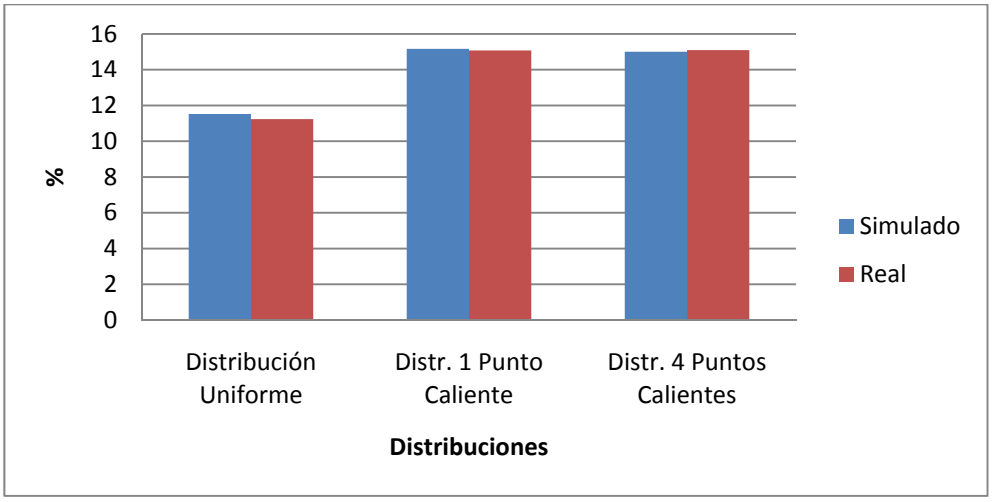
Gráfica A.5 – Comparativa del porcentaje medio de los precintos encontrados con éxito en la Caché de los Servidores Secundarios



Gráfica A.6 – Comparativa del número de precintos medio de las colas de los Servidores Secundarios

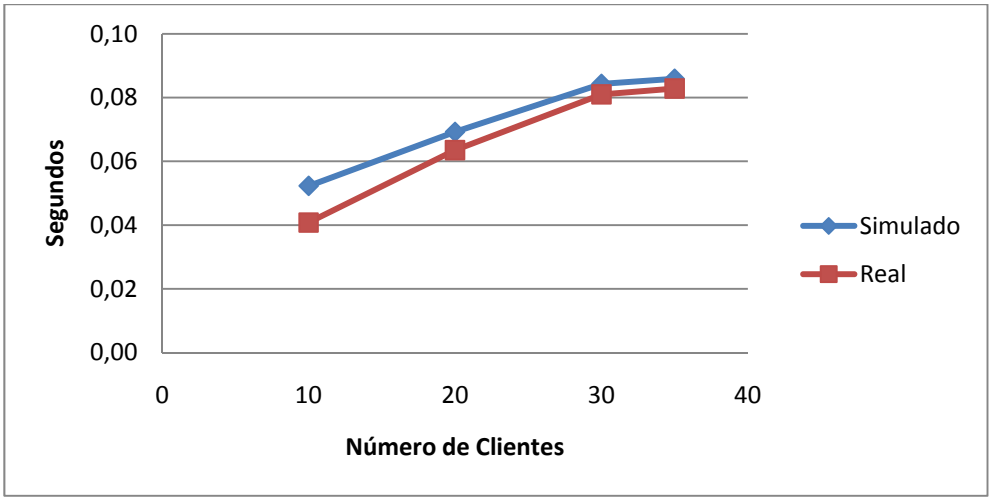


Gráfica A.7 – Comparativa de la latencia media global del intercambio de precintos en el Sistema

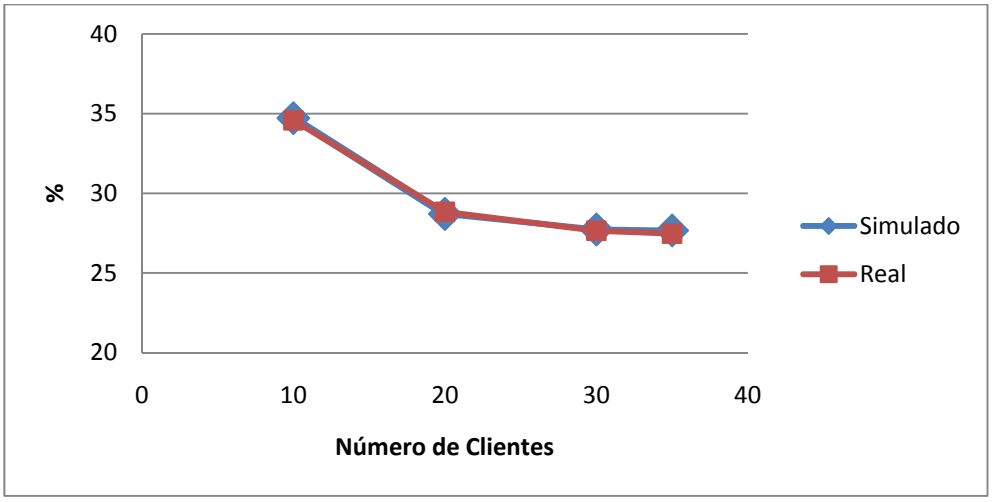


Gráfica A.8 – Comparativa del porcentaje medio de precintos vueltos a pedir

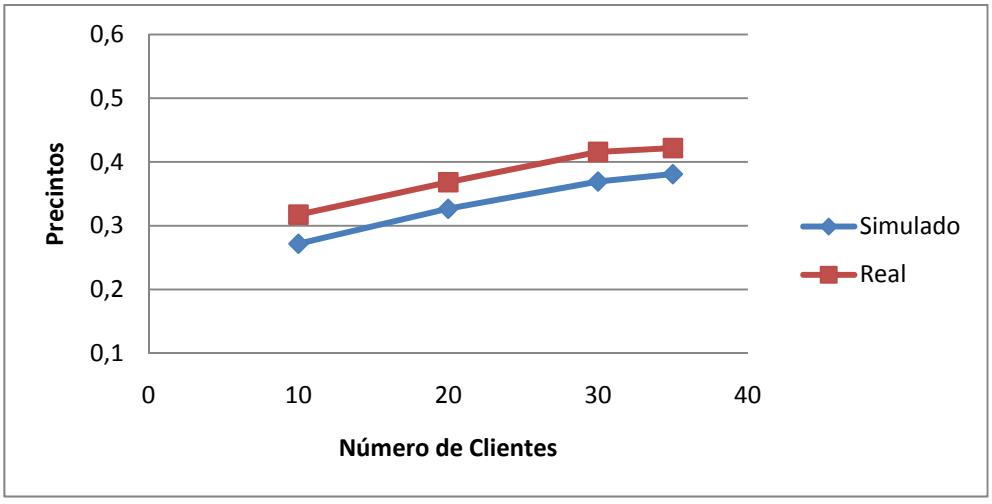
A.2 – Variación en el número de clientes conectados



Gráfica A.9 – Comparativa de la latencia media de los precintos intercambiados entre los Clientes

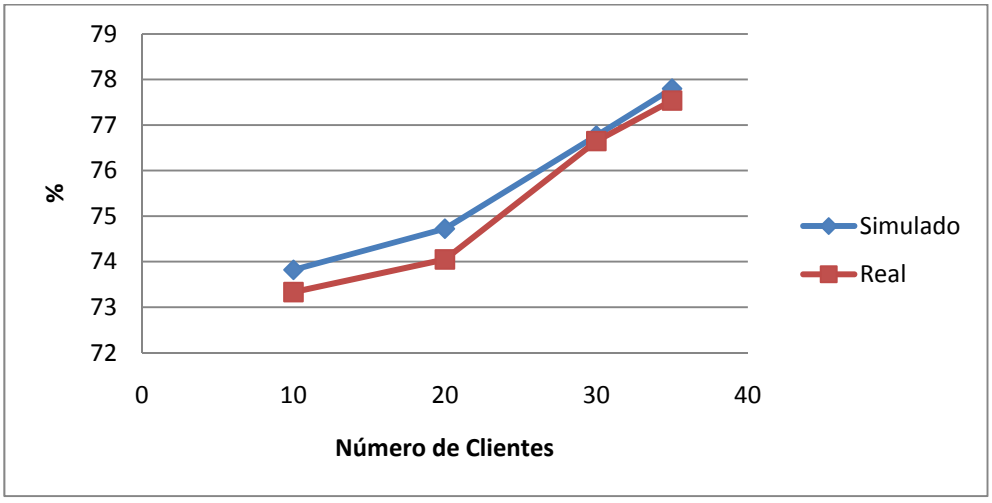


Gráfica A.10 – Comparativa del porcentaje de precintos medio respondidos por los Servidores Secundarios

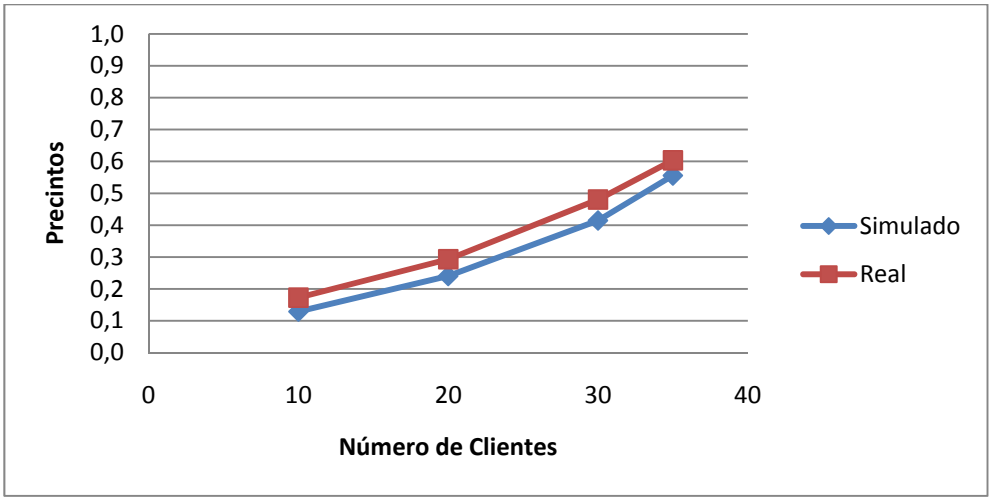


Gráfica A.11 – Comparativa del número de precintos medio de las colas de los Clientes

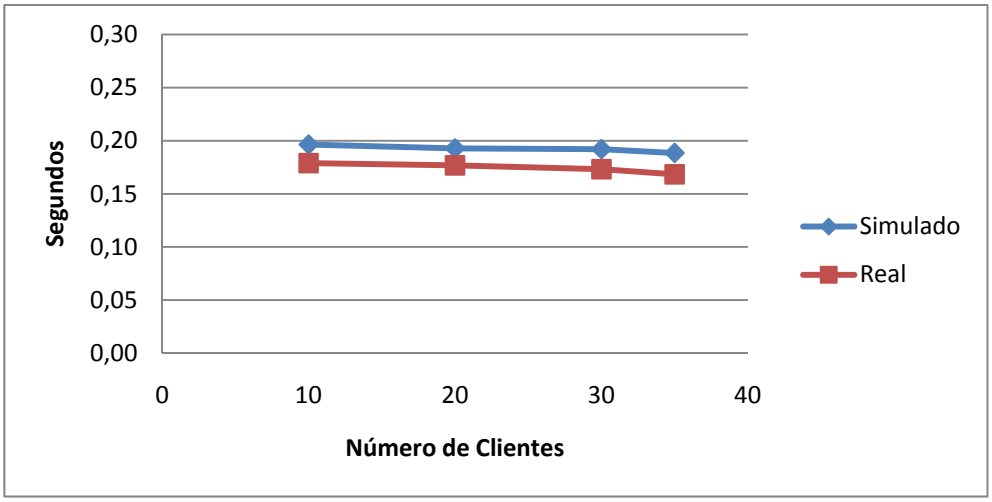
Gráfica A.12 – Comparativa de la latencia media de los precintos respondidos por los Servidores Secundarios



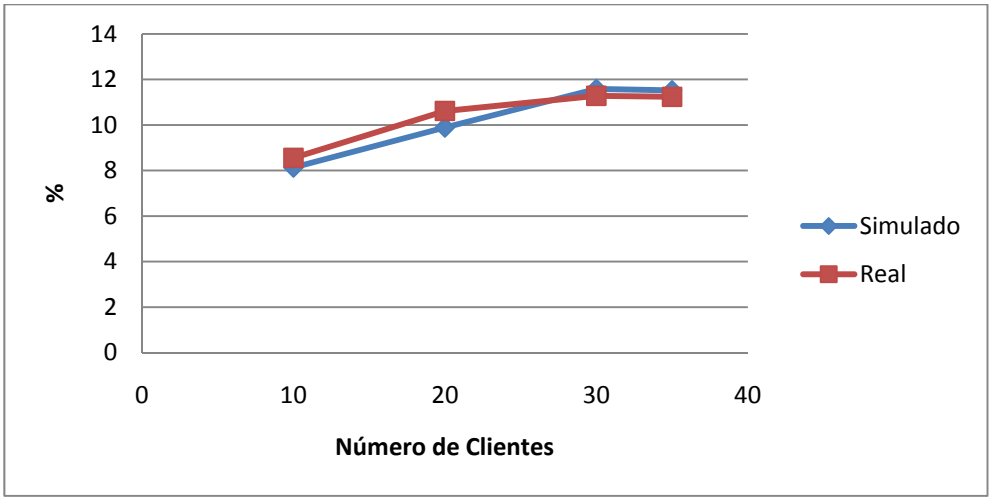
Gráfica A.13 – Comparativa del porcentaje medio de los precintos encontrados con éxito en la Caché de los Servidores Secundarios



Gráfica A.14 – Comparativa del número de precintos medio de las colas de los Servidores Secundarios

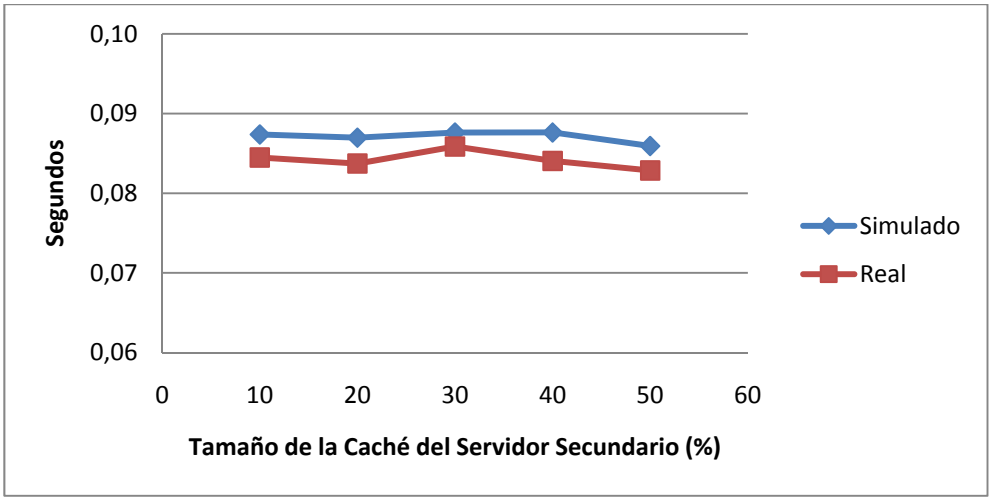


Gráfica A.15 – Comparativa de la latencia media global del intercambio de precintos en el Sistema

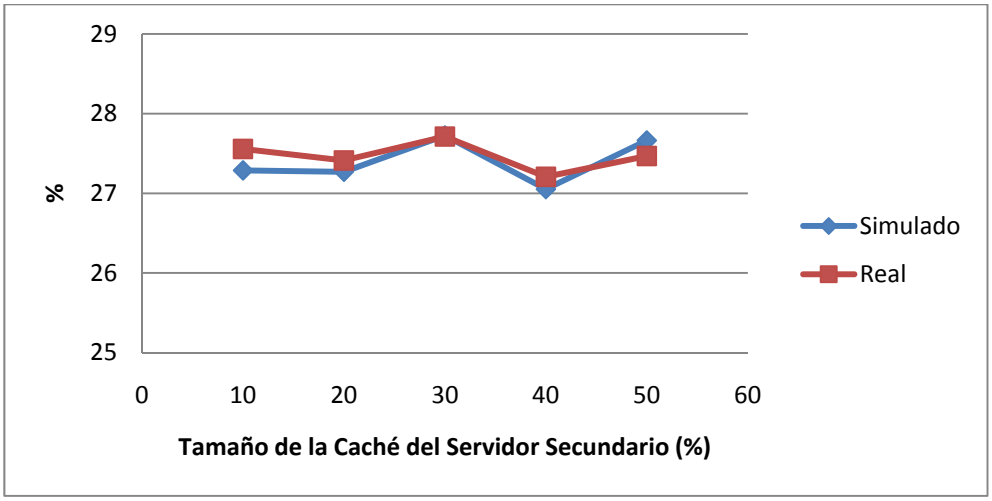


Gráfica A.16 – Comparativa del porcentaje medio de precintos vueltos a pedir

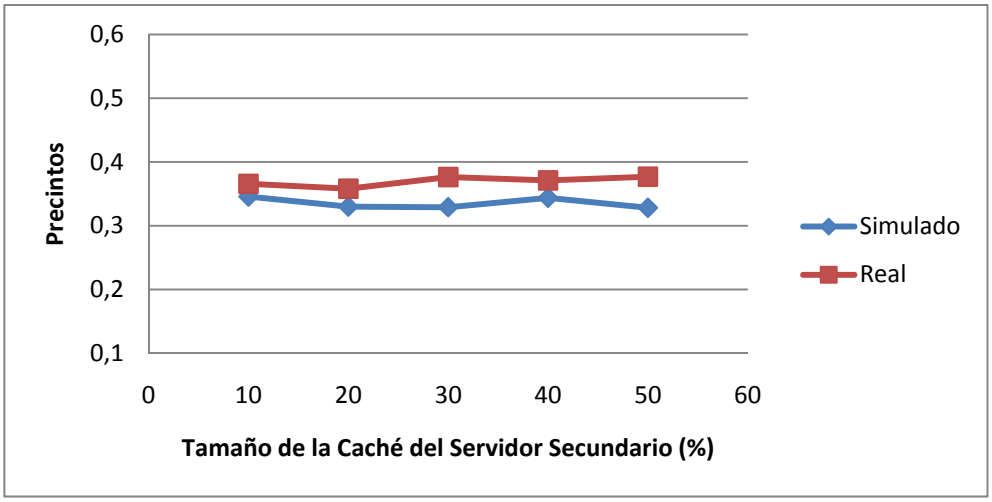
A.3 – Variación en el tamaño de la caché de los servidores secundarios



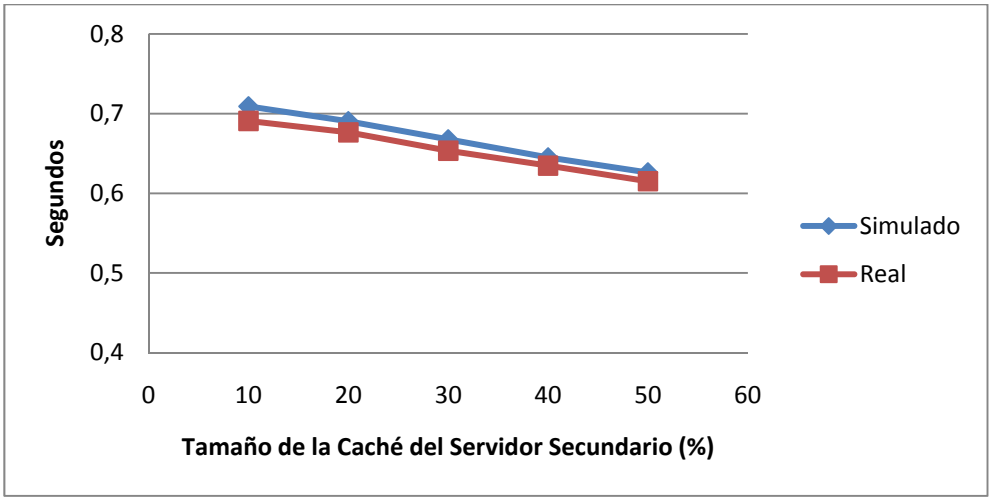
Gráfica A.17 – Comparativa de la latencia media de los precintos intercambiados entre los Clientes



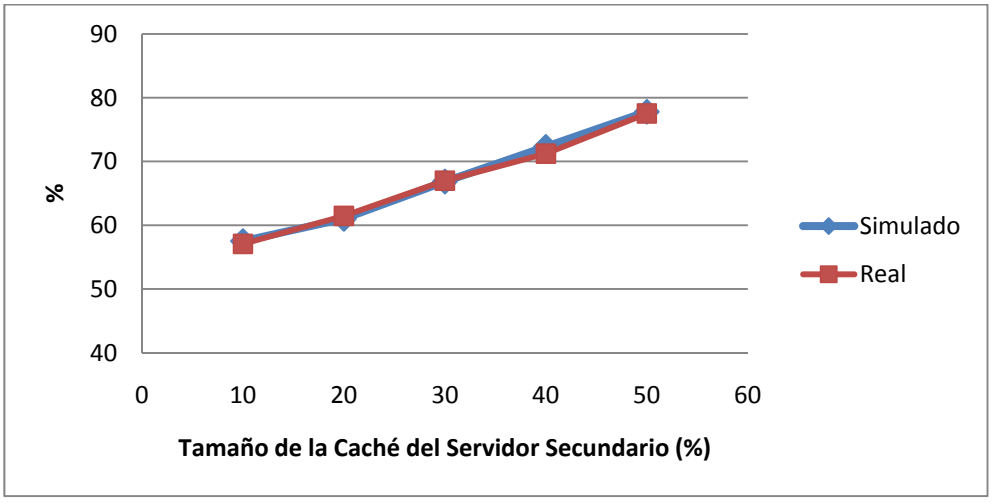
Gráfica A.18 – Comparativa del porcentaje medio de precintos respondidos por los Servidores Secundarios



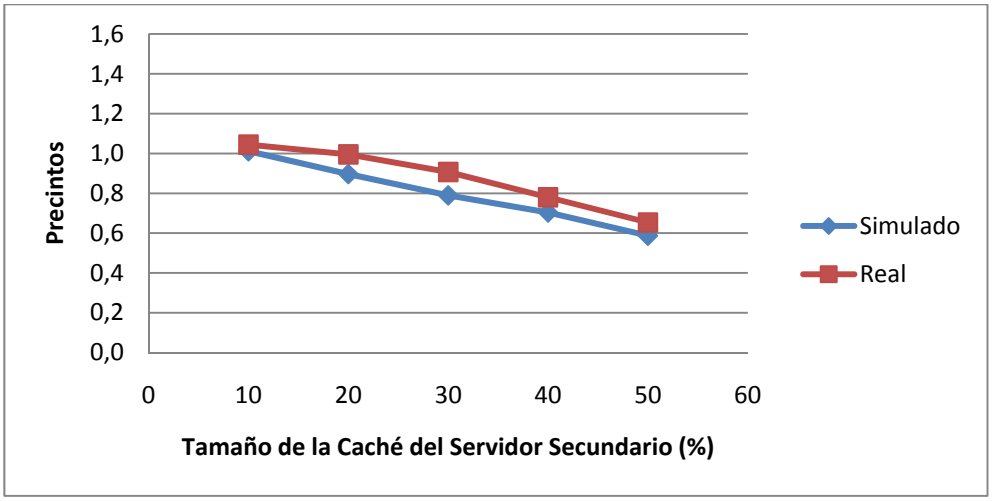
Gráfica A.19 – Comparativa del número de precintos medio de las colas de los Clientes



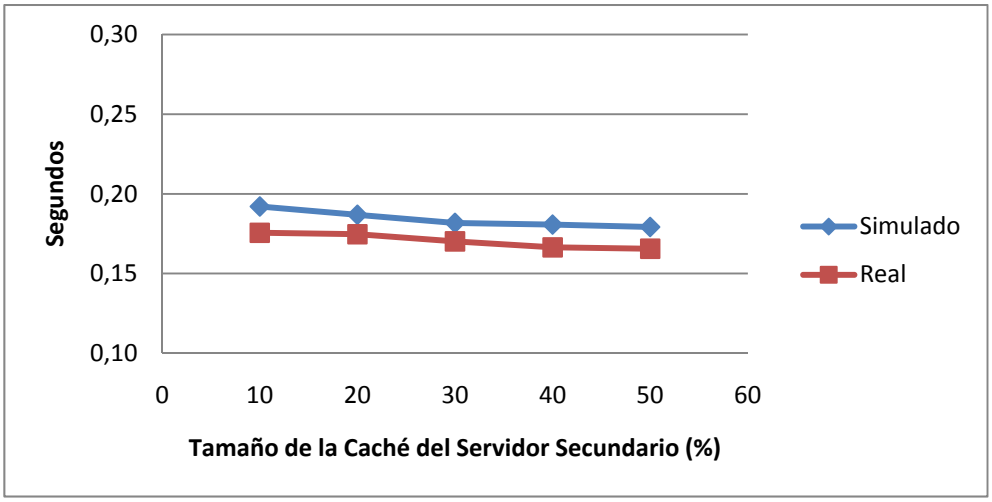
Gráfica A.20 – Comparativa de la latencia media de los precintos respondidos por los Servidores Secundarios



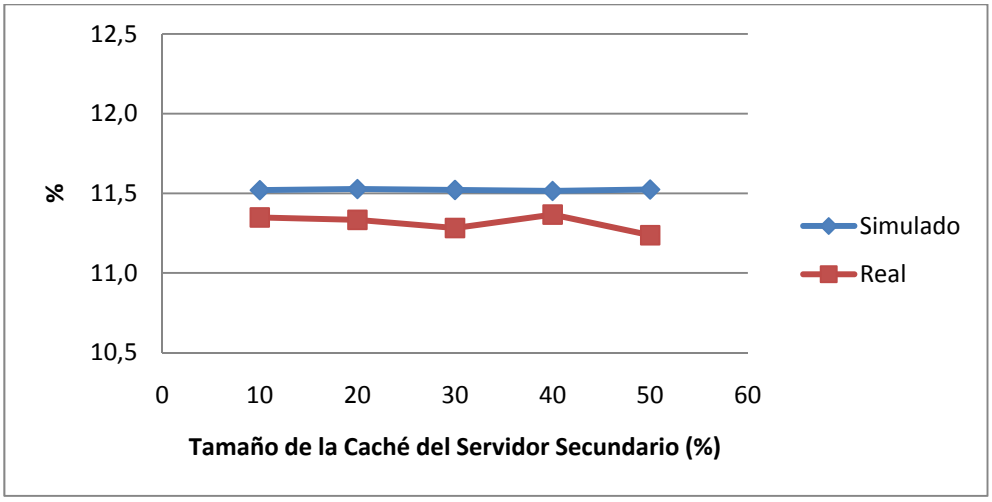
Gráfica A.21 – Comparativa del porcentaje medio de los precintos encontrados con éxito en la Caché de los Servidores Secundarios



Gráfica A.22 – Comparativa del número de precintos medio de las colas de los Servidores Secundarios

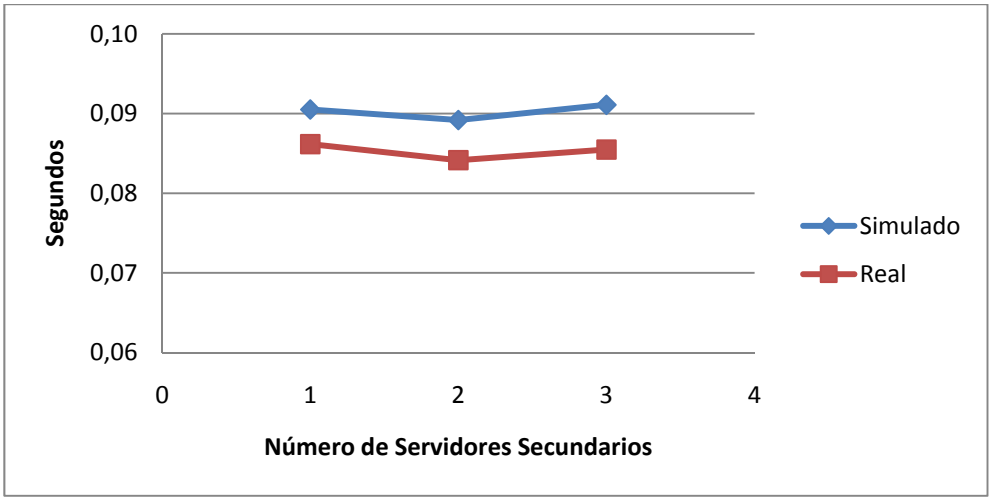


Gráfica A.23 – Comparativa de la latencia media global del intercambio de precintos en el Sistema

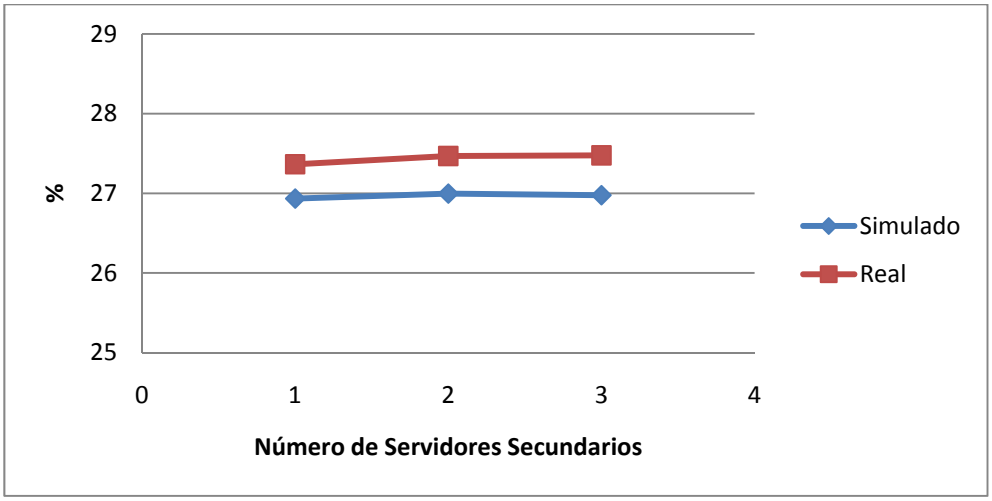


Gráfica A.24 – Comparativa del porcentaje medio de precintos vueltos a pedir

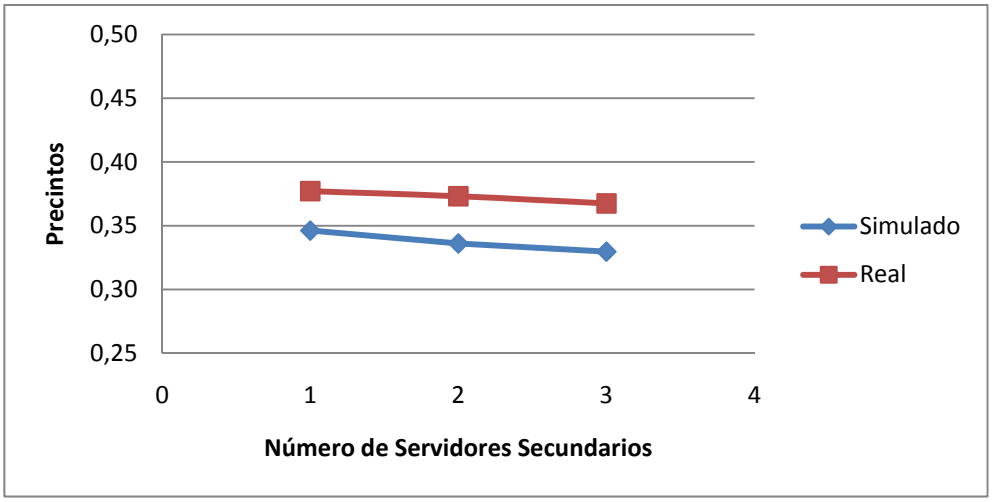
A.4 – Variación en el número de servidores secundarios conectados



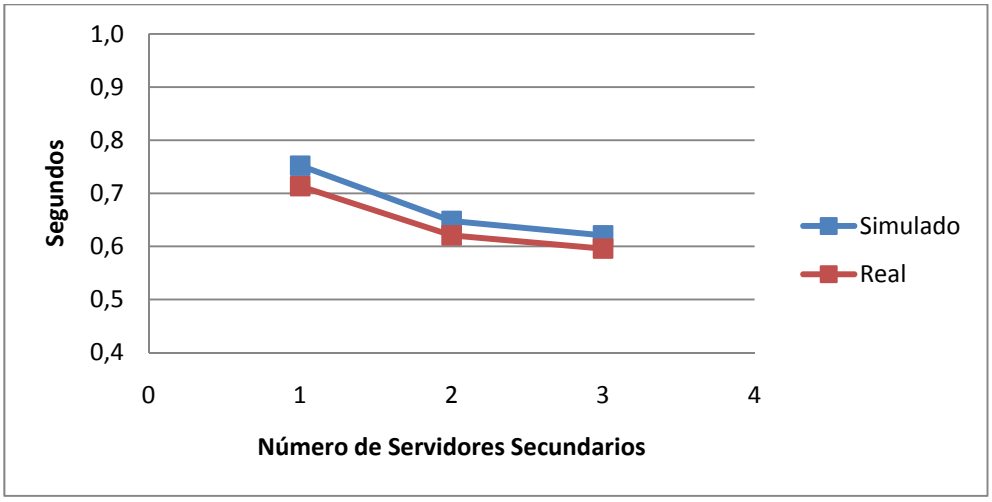
Gráfica A.25 – Comparativa de la latencia media de los precintos intercambiados entre los Clientes



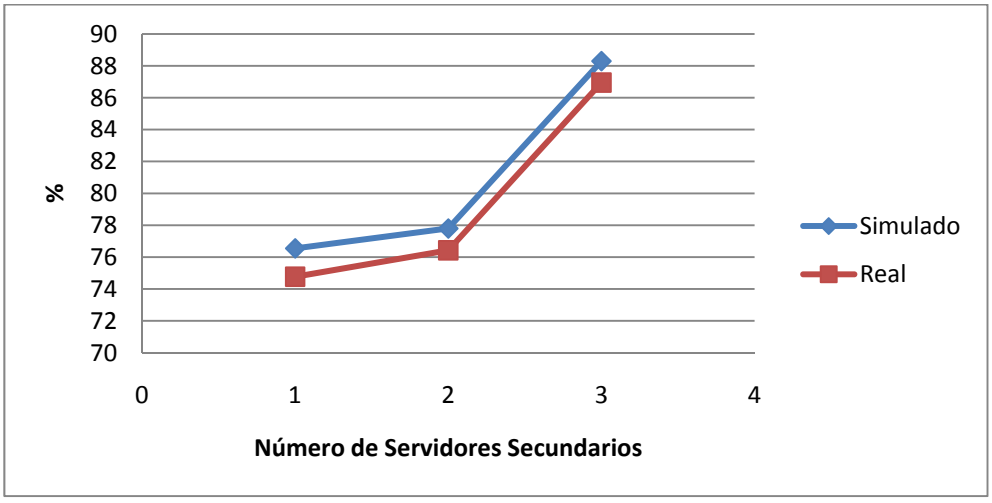
Gráfica A.26 – Comparativa del porcentaje medio de precintos respondidos por los Servidores Secundarios



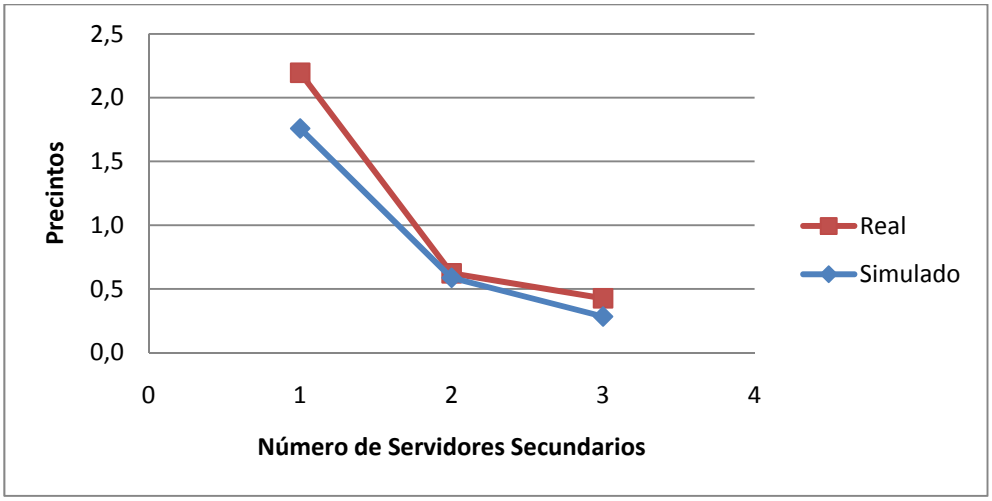
Gráfica A.27 – Comparativa del número de precintos medio de las colas de los Clientes



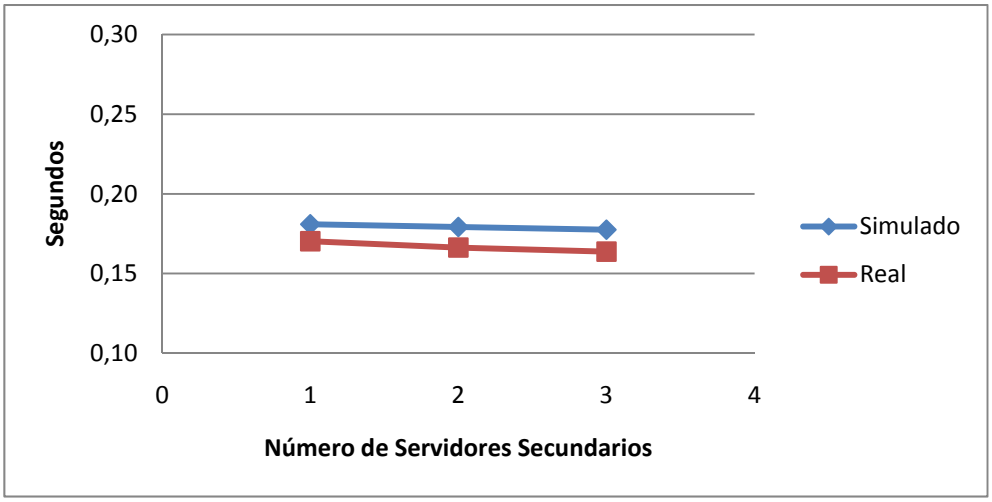
Gráfica A.28 – Comparativa de la latencia media de los precintos respondidos por los Servidores Secundarios



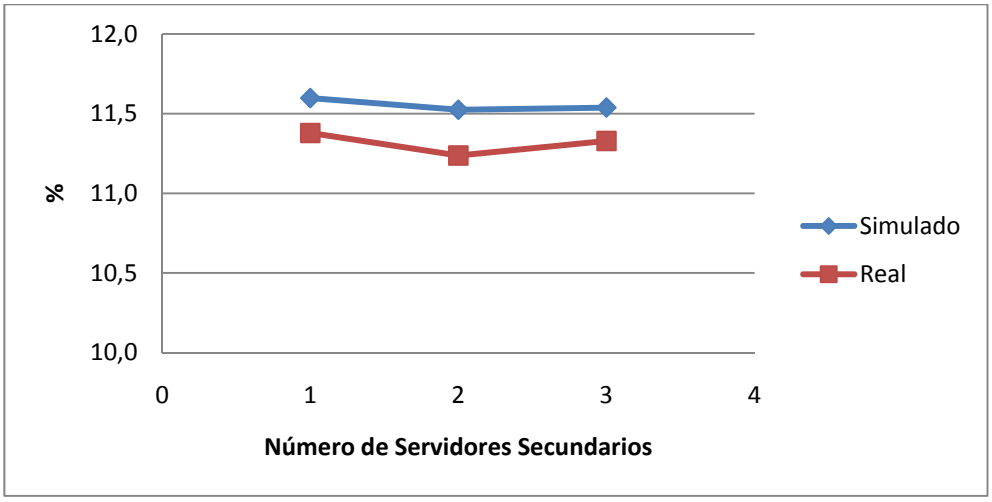
Gráfica A.29 – Comparativa del porcentaje medio de los precintos encontrados con éxito en la Caché de los Servidores Secundarios



Gráfica A.30 – Comparativa del número de precintos medio de las colas de los Servidores Secundarios

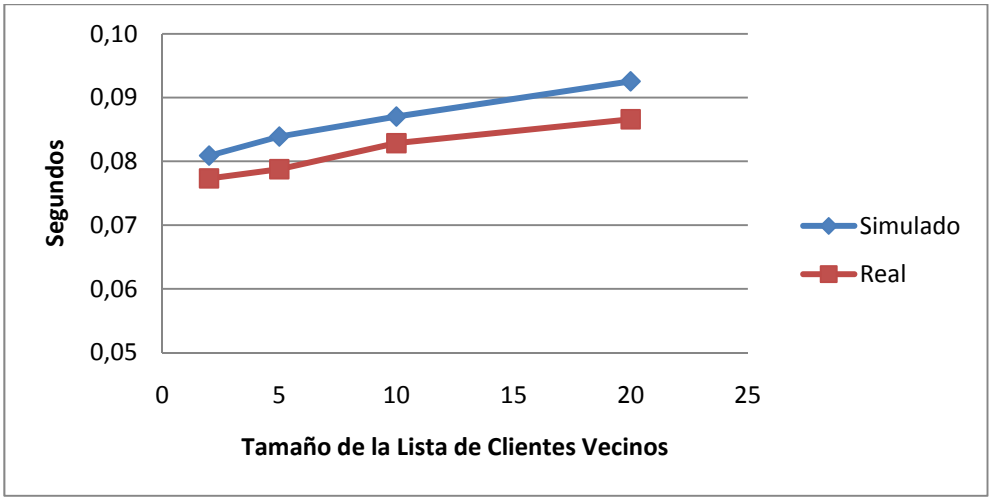


Gráfica A.31 – Comparativa de la latencia media global del intercambio de precintos en el Sistema

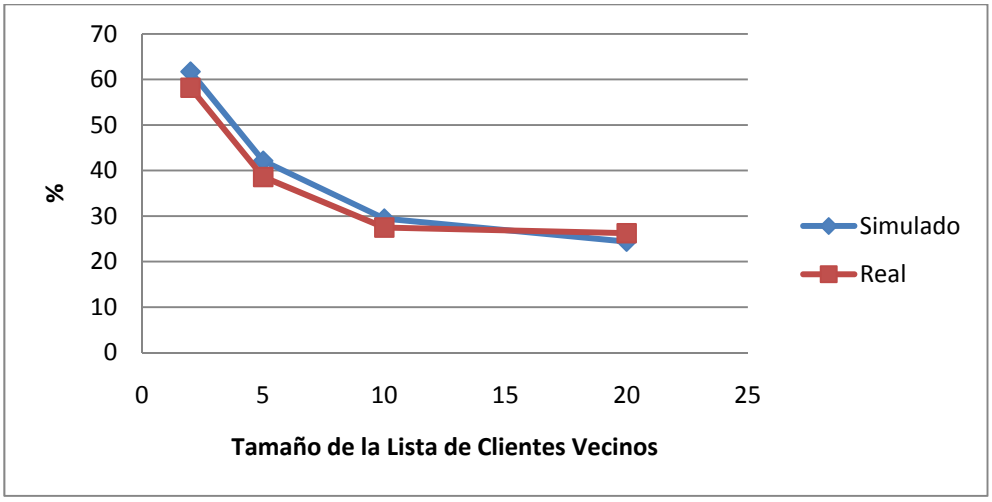


Gráfica A.32 – Comparativa del porcentaje medio de precintos vueltos a pedir

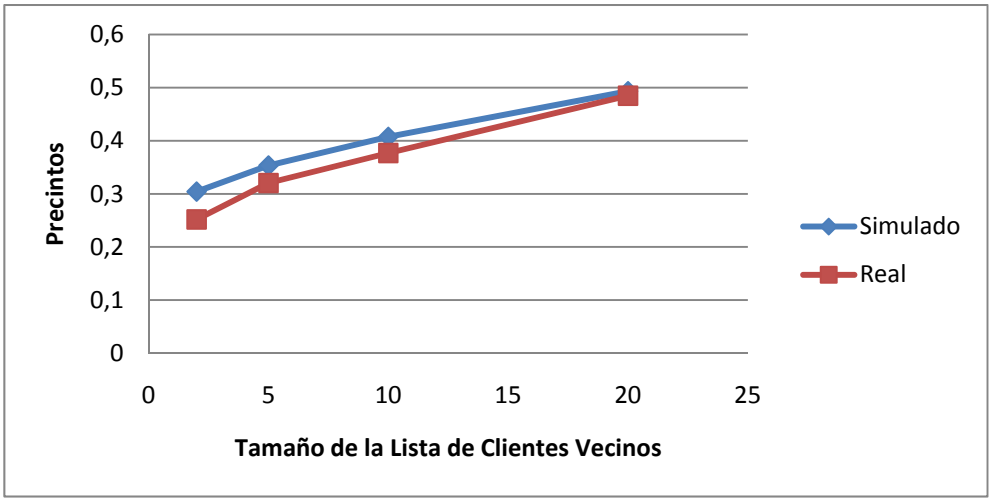
A.5 – Variación en el tamaño de la lista de los clientes vecinos



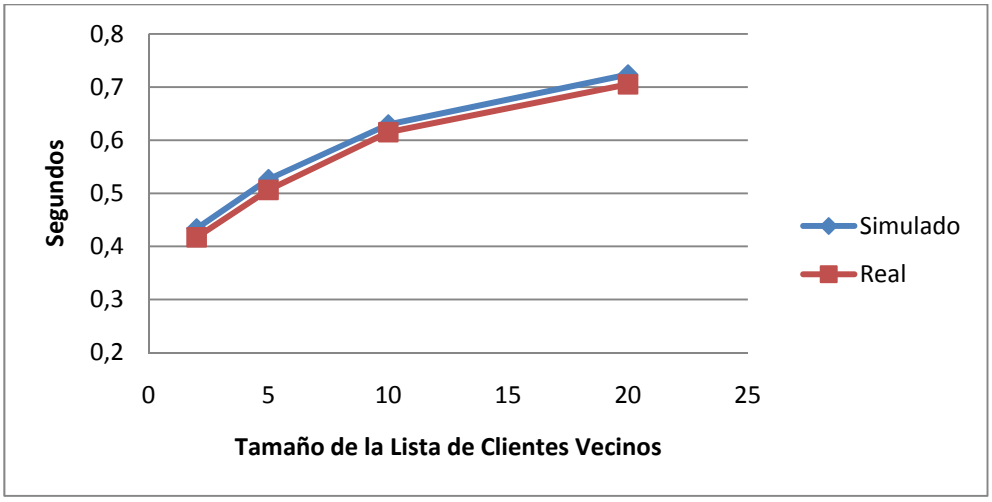
Gráfica A.33 – Comparativa de la latencia media de los precintos intercambiados entre los Clientes



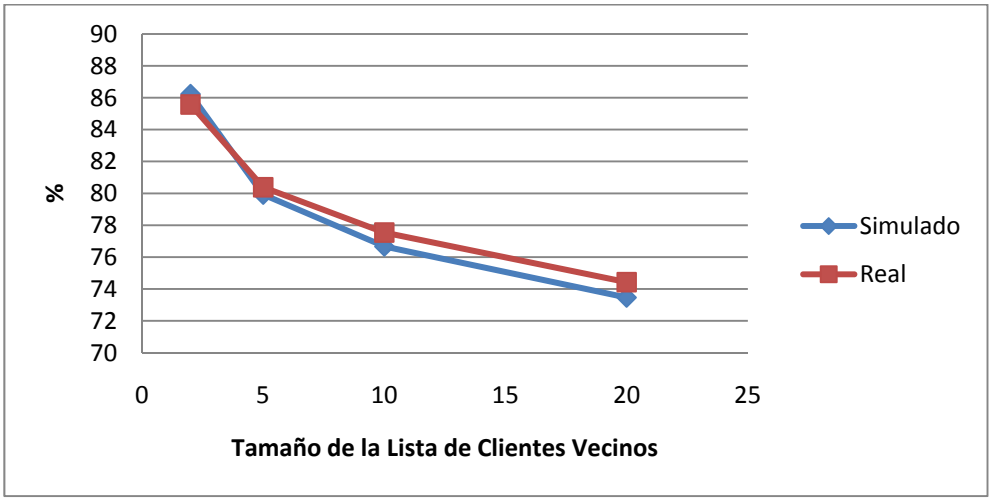
Gráfica A.34 – Comparativa del porcentaje medio de precintos respondidos por los Servidores Secundarios



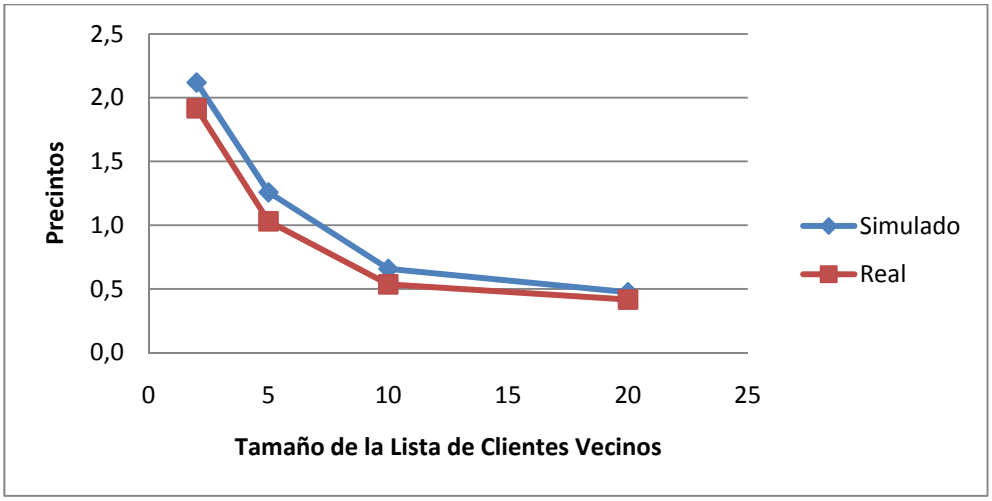
Gráfica A.35 – Comparativa del número de precintos medio de las colas de los Clientes



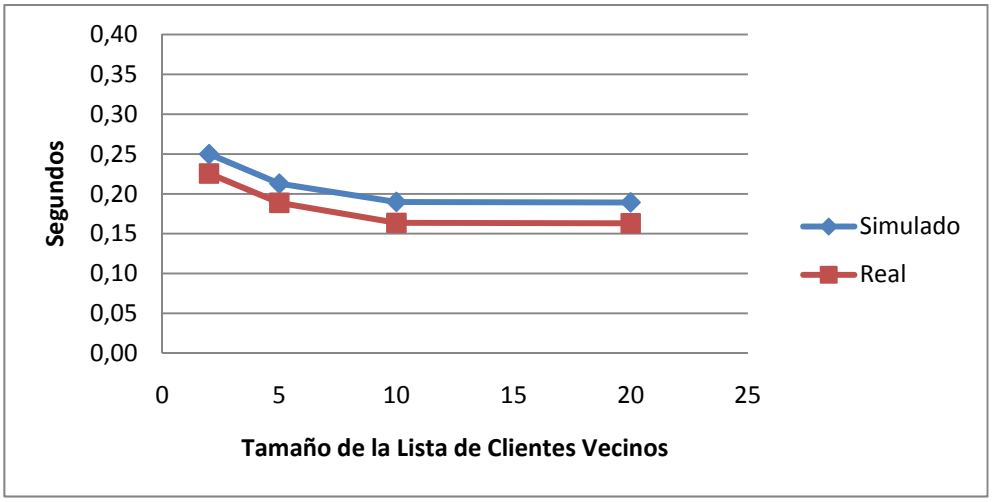
Gráfica A.36 – Comparativa de la latencia media de los precintos respondidos por los Servidores Secundarios



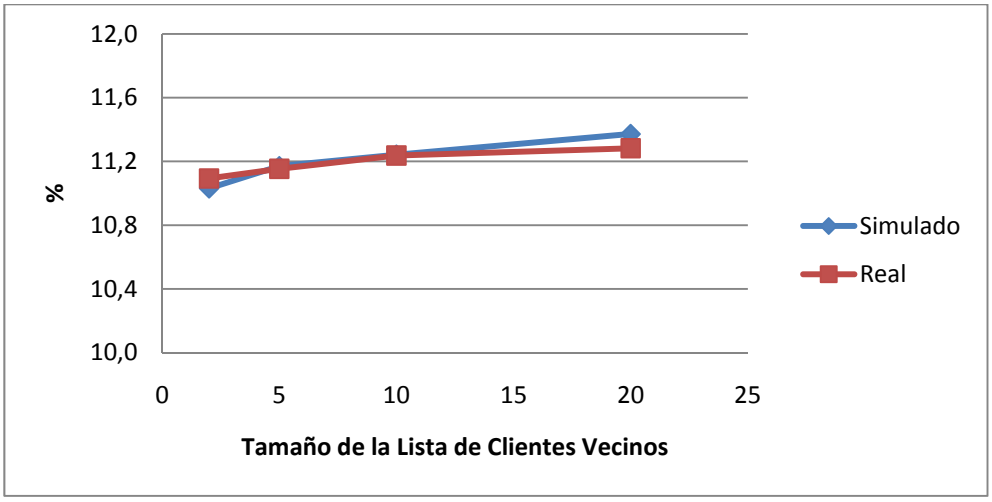
Gráfica A.37 – Comparativa del porcentaje medio de los precintos encontrados con éxito en la Caché de los Servidores Secundarios



Gráfica A.38 – Comparativa del número de precintos medio de las colas de los Servidores Secundarios



Gráfica A.39 – Comparativa de la latencia media global del intercambio de precintos en el Sistema



Gráfica A.40 – Comparativa del porcentaje medio de precintos vueltos a pedir

A.6 – Prueba Cliente-Servidor

	Latencia Global	precintos Medio Cola Servidor	% precintos vueltos a pedir
Real	0,45	24,15	10,98
Simulado	0,41	17,73	11,12

Tabla A.1 – Comparativa de la latencia global del intercambio de precintos, del número de precintos medio de las colas de los Servidores Secundarios y del porcentaje medio de precintos vueltos a pedir