Análisis de la arquitectura de CDN aplicada a MMOG

Autor: Jaime Alberto Díaz Pineda

Director: Carlos Enrique Palau Salvador

Resumen

A través de los años, los juegos en red multijugador (MMOG – Massively Multiplayer Online Game) han tenido un continuo y elevado crecimiento en Internet. Existen diversos tipos de MMOG dependiendo de la forma como el usuario interactúa con el mundo del juego. Los tipos de juego más comunes son MMORPG (Massively Multiplayer Online Role-Playing Game), MMORTS (Massively Multiplayer Online Real-Time Strategy) y MMOFPS (Massively Multiplayer Online First-Person Shooter). Las redes de distribución de contenidos (CDN, Content Delivery Networks) permiten reducir la latencia experimentada por los clientes y reducir la carga de los servidores. Diversos problemas han sido identificados en los MMOG, donde los de mayor importancia son la cantidad de usuarios que se pueden soportar simultáneamente, la variabilidad de la latencia de la red debido a la saturación de los servidores y el ancho de banda requerido para garantizar una calidad de servicio adecuada. Basados en estudios realizados con diferentes topologías, hemos seleccionado la CDN como solución a los problemas evidenciados en MMOG. En esta tesina simulamos la CDN para validar el algoritmo de redirección implementado.

Abstract

Over the years, the Massively Multiplayer Online Games (MMOG) have had a continuous and high growth on the Internet. Currently, there are different types of MMOG according to the interface of the player with the world of the game. The most common types of MMOG are the MMORPG (Massively Multiplayer Online Role-Playing Game), the MMORTS (Massively Multiplayer Online Real-Time Strategy) and the MMOFPS (Massively Multiplayer Online First-Person Shooter). The Content Delivery Networks (CDN) can reduce the latency experienced by the users and thus reduce the load on servers. Several problems have been identified in the MMOG, in which the most important problems are the number of users who can be online simultaneously, the variable latency of the network due to overload in the servers and the bandwidth required to ensure a good quality of service. Based on studies with different topologies, we have selected the CDN as a solution to the problems identified in MMOG. Also, we have simulated a CDN to validate the redirector algorithm.

Autor: Jaime Alberto Díaz Pineda, email: jaime.diaz@ibv.upv.es
Director: Carlos Enrique Palau Salvador, email: cpalau@dcom.upv.es
Fecha de entrega: 03-11-07

ÍNDICE

I. ESTADO DEL ARTE DE LOS JUEGOS EN RED Y LAS ARQUITECTURAS DISPONIBLES.

La posibilidad de utilizar un ordenador para jugar en red comenzó en 1979, cuando un grupo de estudiantes de la Universidad de Essex crearon una versión informática multiusuario de un juego de rol, basado en el uso de textos alfanuméricos. El primer juego multiusuario que incorporó imágenes se desarrolló en 1996, pero la verdadera revolución de los juegos en red se desarrolló con la aparición del internet. La rápida difusión del Internet como medio de entretenimiento facilitó la mejora de las tecnologías para la conexión en red de usuarios.

Gracias a las nuevas tecnologías de transmisión las redes han ido creciendo y ofreciendo nuevos servicios manejando grandes cantidades de tráfico. Los juegos en red en tiempo real hoy conocidos como MMOG (Massively Multiplayer Online Game) han ido incrementando su número de usuarios.

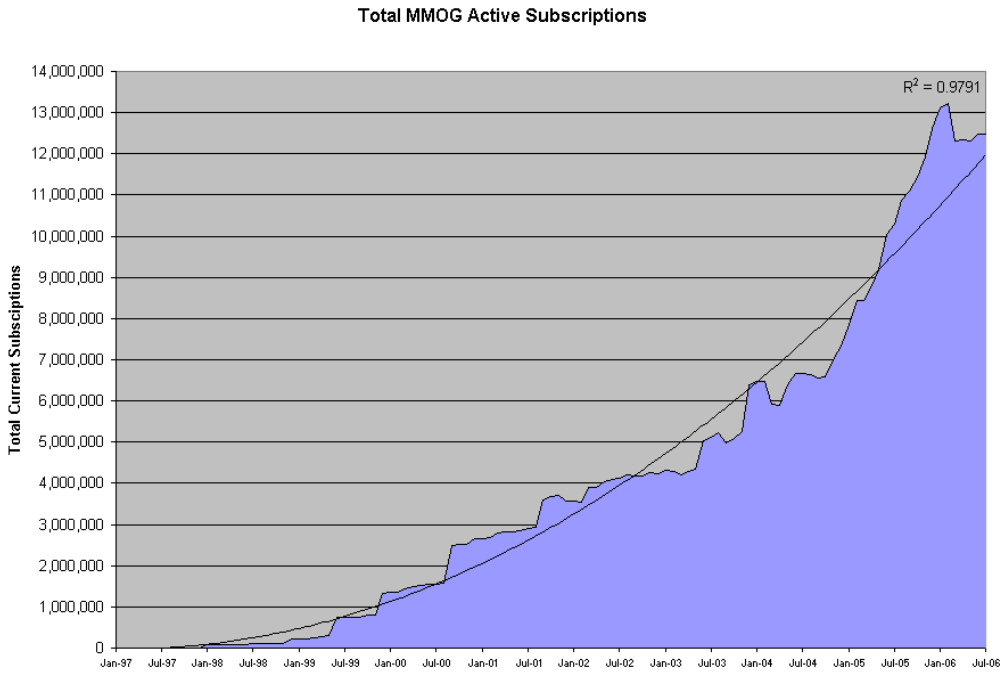

Fig. 1. Total de subscriptores de MMOG [1]

Existen diversos tipos de MMOG, los cuales van desde simples juegos basados en texto hasta la incorporación de gráficos complejos en mundos virtuales habitados por múltiples jugadores simultáneamente.

I.1. *TIPOS DE MMOG*

Existen diversos tipos de juegos que dependen de la representación gráfica del personaje que utiliza el jugador, o conjunto de ellos, y la forma de interactuar con los otros usuarios y el entorno.

Los tipos de juegos más comunes se describen a continuación.

- *MMORPG (Massively Multiplayer Online Role-Playing Game)*

  También llamados como juegos “SLOW ACTION” (Acción lenta). En este tipo de juegos, un gran número de jugadores interactúa con otros en un mundo virtual. Los jugadores, que usan el programa cliente, son representados en el mundo del juego a través de una representación gráfica del personaje con que juegan.

  El mundo en el que habitan los jugadores es ubicado en un servidor al que acceden los clientes. Dependiendo del número de jugadores y la arquitectura del sistema, el juego puede disponer de múltiples servidores, cada uno representando un mundo independiente, donde los jugadores de un servidor no pueden interactuar con aquellos de otro.

  Como se puede apreciar en este documento, los estudios realizados no se centran en este tipo de juegos debido a que no son considerados como los más críticos ya que no se manejan elevados flujos de información y la desconexión de un jugador no afecta a ninguno de los demás participantes.

- *MMORTS (Massively Multiplayer Online Real-Time Strategy)*

  También llamados como juegos “STRATEGIC” (De estrategia). Esta categoría combina estrategia en tiempo real con un gran número de jugadores. Aquí el jugador asume el rol de líder y dirige un ejército. El problema que presenta este tipo de juegos es que no tolera fallos de conexión.

  Si en un MMORPG, un jugador se desconecta, simplemente desaparece del mundo del juego. Mientras que en un MMORTS, si un jugador se desconecta, los otros jugadores no tendrían oponente, por lo que obtendrían la victoria al estar solos en el mundo del juego. En síntesis, no tiene sentido que el “líder” de un ejército desaparezca.

- *MMOFPS (Massively Multiplayer Online First-Person Shooter)*

  También llamados como juegos “FAST ACTION” (Acción rápida). Esta categoría combina una presentación desde la perspectiva visual del jugador con un gran número de jugadores simultáneos.

  La calidad de servicio de este tipo de juegos está altamente condicionada por diferentes variables como la potencia de procesamiento del servidor, ancho de banda tanto del cliente como del servidor, promedio o peor caso de ping con el cliente, promedio o peor caso de potencia de procesamiento del cliente y la selección del tipo de actualización (por parte del servidor, por predicción del cliente, por tiempos establecidos para cada cliente, por un algoritmo, o alguna combinación de estos).

Un gran número de usuarios conectados simultáneamente reduce las prestaciones de los servidores por lo que es muy frecuente que éstos fijen la cantidad de usuarios conectados al mismo tiempo.

Generalmente, los servidores destinados a aplicaciones MMOFPS fijan la latencia que debe tener cada usuario para estar conectado. Esta latencia depende del tipo de juego y de la cantidad de información a transmitir necesaria para ofrecer una calidad de servicio óptima a todos los jugadores.

I.2. *ARQUITECTURAS DE RED*

En los juegos en red en tiempo real, la calidad del servicio depende de diversos factores como la tolerancia a fallos, respuesta del juego y al sistema de protección anti trampas. Estos aspectos están altamente influenciados por la arquitectura de red utilizada en la aplicación.

Las siguientes son las arquitecturas disponibles para este tipo de aplicaciones:

- *TOPOLOGIA CLIENTE SERVIDOR*

  En este tipo de topología el servidor es el encargado de modificar el estado del juego, realizando todos los cálculos requeridos para la transición de estados, recibiendo las entradas de cada cliente e informándolas a los demás para actualizar los estados continuamente.

  Cuando los datos enviados al cliente exceden el ancho de banda disponible existen mecanismos de priorización para omitir información que no es necesaria y reducir el volumen de datos.

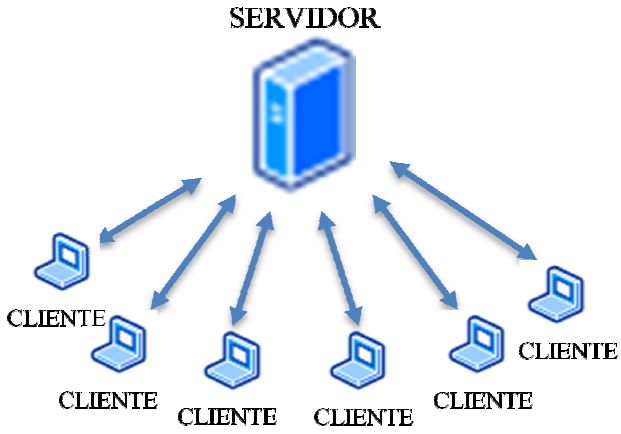

Fig. 2. Topología cliente-servidor

Este tipo de topología es la más usada en los juegos MMOFPS para permitir la participación de clientes tanto con gran, como con bajo ancho de banda. El principal problema es que toda la carga la soporta un servidor por lo que el número de usuarios depende de su capacidad de procesamiento computacional.

- *TOPOLOGIA PEER-TO-PEER*

  En este tipo de topología no existe un servidor central para mantener el estado del juego. Cada participante actúa al mismo tiempo como cliente y servidor y está conectado a los

otros para sincronizar su estado local del juego. La sincronización la realiza el cliente enviando los cambios locales a los otros usuarios, los cuales actualizan sus estados de acuerdo a esta información y viceversa.

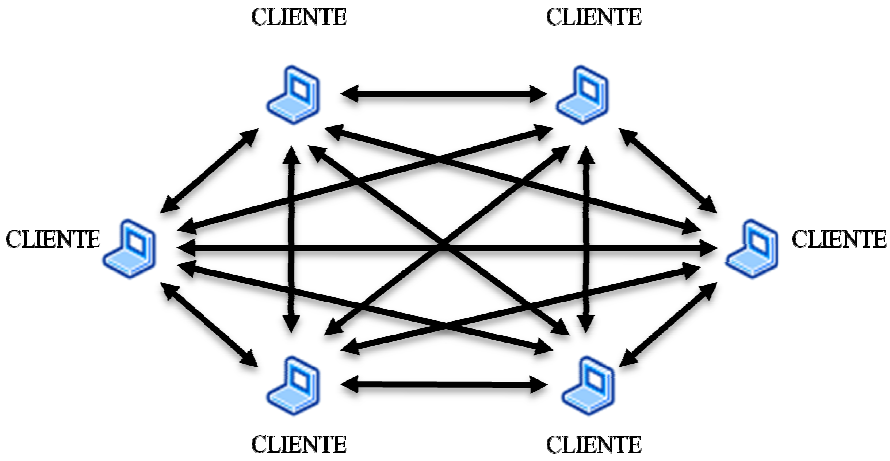

Fig. 3. Topología Peer-to-Peer

El principal problema de este sistema es que requiere un elevado ancho de banda para comunicar los cambios de estado a los clientes y cada cliente tiene que procesar el estado de todos los demás, lo cual representa mucho gasto computacional.

Esta arquitectura puede ser usada por juegos MMOFPS o MMORTS pero la forma de sincronización debe ser usada en función de las características del juego. Entre más personajes tenga el juego, mayor es la frecuencia con la que se debe sincronizar la información, razón por la cual no es utilizado para los MMORPG.

- *TOPOLOGIA MULTI-SERVIDOR*

Los juego multi-jugador de rol (MMORPG) soportan una topología multiservidor en la cuales los participantes son distribuidos a través de los servidores.

Esta topología, también conocida como topología de servidor proxy utiliza una interconexión vía multicast.

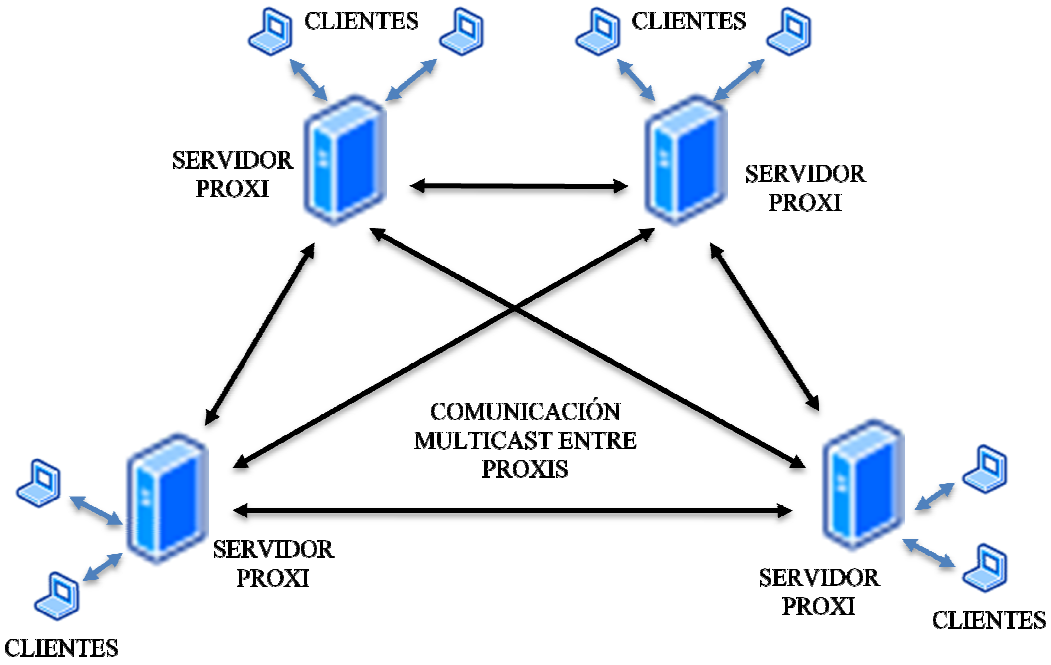

Fig. 4. Topología multi-servidor

Cada uno de los proxies tiene una visión completa del estado del juego. Generalmente se segmenta el mundo del juego y se distribuye en los servidores, por lo que los clientes deben cambiar de servidor si se mueven de una región a otra.

Este tipo de topología no es posible en juegos MMOFPS y MMORTS ya que estos requieren información de todo el mundo del juego, por ejemplo, jugadores que usan rifles de francotirador y disponen de una visión de largo alcance o cuando el jugador cuenta con muchos elementos sobre todas las regiones. Para permitir esto, el cliente debería mantener conexión con varios servidores con lo cual se consume mucho ancho de banda y se incrementa la carga en los servidores. Por esta razón, distribuir la información entre los servidores limita al usuario a disponer de la información solamente de su rango de vista.

I.3. *CARACTERISTICAS DEL TRÁFICO DE LOS MMOG*

Los juegos en red se caracterizan porque su protocolo de transporte es UDP, donde un estudio realizado en el juego “Counter-Strike” del tipo MMOFPS, considerado uno de los más popular del mercado en internet y altamente sensitivo a la calidad del servicio, la longitud de las sesiones que se inician en un servidor de juegos es exponencial y el tiempo que hay entre llegadas de nuevos usuarios es muy pequeño, por lo que se generan largas colas. Sin embargo, a medida que aumenta el número de usuarios, los tiempos entre llegadas disminuyen por lo que existe la hipótesis de que el número de jugadores es determinante en la decisión de otros para unirse al juego. [2]

El ancho de banda promedio usado por un usuario de juegos en internet es de 40 Kbps con lo que queda en evidencia la facilidad de saturar un servidor, motivo por el cual se limita la cantidad de usuarios simultáneos o de lo contrario no se podría garantizar la calidad del servicio. [3]

Para aplicaciones interactivas de voz o vídeo (telefonía por internet o video conferencia), se requiere una latencia máxima de 300 ms y la variación de 50 ms no es apreciable para el usuario. Los juegos de “elevada acción” como los MMOFPS y MMORTS son muy sensibles a la latencia de la red que los soporta, donde el retraso que puede ser tolerado por un jugador varía entre 100 y 150 ms. [4 - 5]. Es necesario resaltar que la diferencia de 50 ms en un juego en red puede determinar quién gana o pierde en el juego. Sin embargo, muchos estudios han demostrado que los jugadores se adaptan ellos mismos a la latencia por lo que los límites podrían variar. [6]

Los estudios realizados en [7] sirven como referencia para conocer el flujo de información durante una sesión de juegos:

- *LONGITUD DE LOS PAQUETES*

  Servidor a cliente: La longitud de este paquete depende del mapa que se esté jugando. Su tamaño varía entre los 60 y 400 bytes pero la mayoría de los paquetes están entre los 140 y 180 bytes.

  Cliente a servidor: La longitud de este paquete depende de todos los parámetros (tipo de acción del usuario, mapa, cantidad de jugadores, etc). Los paquetes más cortos son de 60 bytes y los más largos son de 90 bytes.

- *TIEMPO ENTRE LLEGADAS DE PAQUETES*

  Servidor a cliente: El tiempo entre llegada de paquetes es muy regular. El servidor envía una actualización al cliente cada 60 ms.

  Cliente a servidor: La transmisión de paquetes del cliente al servidor depende del software usado para la renderización gráfica. (OpenGL, Direct3D y Software). Puede variar entre los 30 y 50 ms.

- *PAQUETES POR SEGUNDO Y TASA DE DATOS*

  Servidor a cliente: La media de transmisión de paquetes por segundo es de 16,3. La tasa de datos depende de los paquetes por segundo transmitidos y de la longitud del paquete por lo que también se hace dependiente del tipo de mapa. Para un mapa promedio con 6 jugadores simultáneos, la tasa de datos varía entre 80 y 190 Kbps.

  Cliente a servidor: El número de paquetes por segundo transmitidos varía entre 20 y 24. La tasa de transferencia de paquetes varía aleatoriamente centrada alrededor de 13 Kbps por jugador.

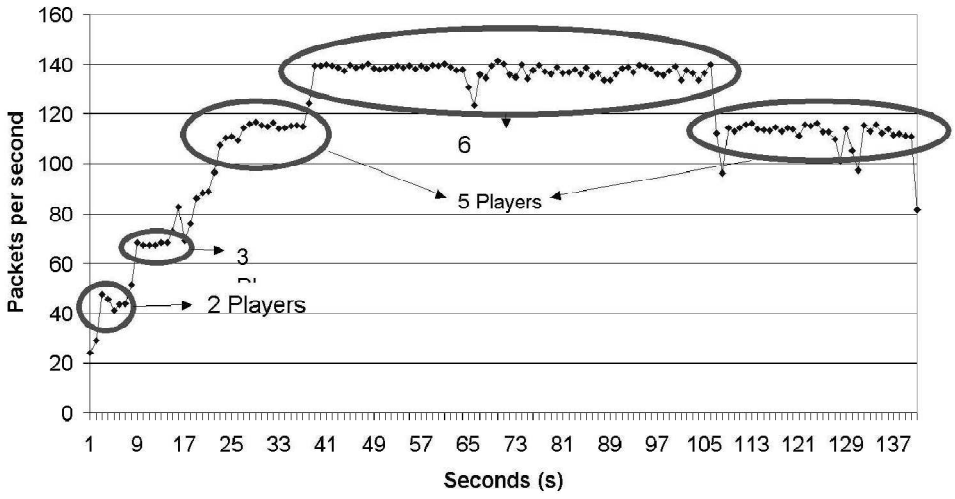

Fig. 5. Paquetes por segundo, de todos los clientes al servidor. [7]

II. MECANISMOS DE DISTRIBUCION DE CONTENIDOS.

Con el paso de los años, los niveles de información que se mueven por las redes han ido aumentando y haciendo más y más congestionado el flujo de datos. El mercado demanda el acceso de contenidos a mínima latencia con diversas aplicaciones con datos complejos de diferentes tipos que van desde aplicaciones de telefonía IP hasta los juegos en línea multijugador, los cuales disponen de imágenes de alta resolución a una taza muy elevada de transferencia.

Estas necesidades, sumadas al crecimiento de la población de usuarios hace necesaria la adopción de diferentes soluciones para escalar la distribución de contenidos. Los sistemas más comunes para dicha distribución son el multicast, web caching, las estructuras P2P y las redes de distribución de contenidos (CDN).

II.1. *MULTICAST*

Multicast es una tecnología que reduce el tráfico mediante la entrega simultánea de un solo stream a miles de usuarios. Entre las aplicaciones en las cuales se usa este método se encuentra la videoconferencia, comunicaciones corporativas, educación a distancia y la distribución de software y noticias.

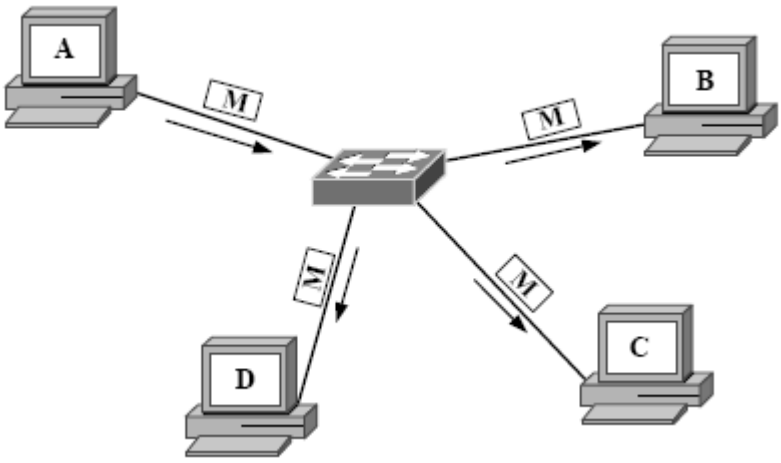

Fig. 6. Distribución de contenidos Multicast

Multicast entrega el tráfico a múltiples receptores sin añadir ninguna carga en la información original, usando un mínimo ancho de banda de la red.

Todas las alternativas multicast implican que la fuente envíe más de una copia de los datos. Algunos incluso obligan a la fuente a enviar una copia a cada receptor. Las aplicaciones de gran ancho de banda, como video MPEG, pueden requerir una gran proporción del ancho de banda para la transmisión, por lo que si hay varios usuarios lo mejor es implementar multicast para proporcionarles el video a todos en un solo stream.

Multicast se basa en el concepto de grupos. Un grupo de receptores expresa el interés de recibir un stream de datos determinado. Este grupo no tiene ninguna frontera geográfica o física, lo que significa que el host puede estar alojado en cualquier lugar en Internet. Los hosts que están

interesados en recibir un stream de datos para un grupo en particular deben ingresar al grupo usando el protocolo IGMP. Para poder recibir el stream de datos, los host deben ser miembros del grupo. [8]

II.2. *WEB CACHING*

El web caching consiste en almacenar la información que ha sido recientemente consultada para que cuando se vuelva a solicitar se pueda acceder más rápido a ella. En ocasiones se almacena información que no ha sido consultada pero que se estima que será utilizada posteriormente. Este sistema se utiliza cuando el costo de almacenamiento de la información es menor que el costo de solicitarla nuevamente al servidor.

Con este servicio se predice el acceso futuro basado en los accesos anteriores. Cuando la predicción es correcta, se mejora significativamente el rendimiento para acceder a la información.
El principio con el cual trabaja el caché se llama “localidad de referencia”. Hay dos tipos de localidad: temporal y espacial. La localidad temporal significa que algunas piezas de los datos son más populares que otras, por ejemplo, la página de Google se cachea debido a su popularidad frente a otras. La localidad espacial significa que las peticiones para determinadas piezas son probables a ocurrir juntas, por ejemplo, los diez primeros resultados de una búsqueda se cachean porque se estima que el usuario hará uso de ellos. [9]

El cacheo puede realizarse en el navegador del cliente web o en el proxy a diferentes niveles. El caché en el navegador no es compartido por lo que únicamente beneficia al cliente.

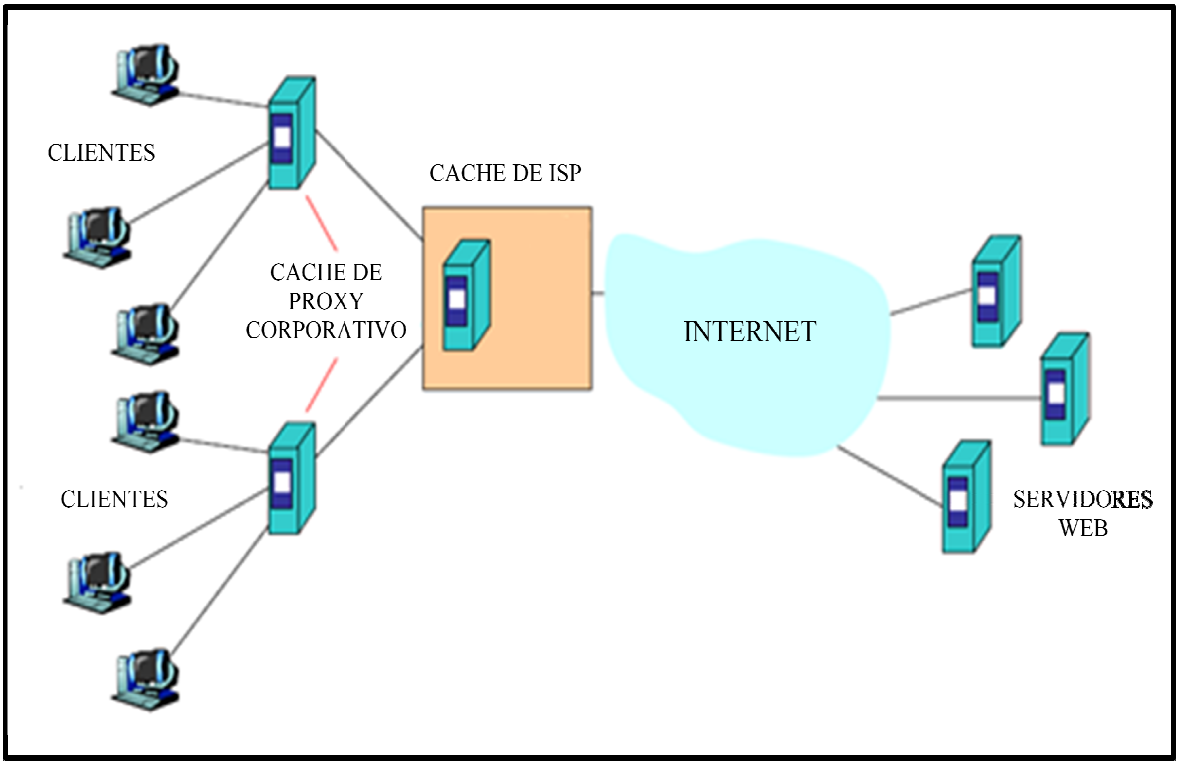

Fig. 7. Distribución de contenidos con Web Caching

Los componentes de una arquitectura de caché son los siguientes:

- *CLIENTES Y SERVIDORES.*

  Un servidor web controla y proporciona acceso a un conjunto de recursos, que pueden ser archivos de texto e imágenes, o algo mucho más complejo como bases de datos relacionales. Los clientes son conocidos como “agentes”, los cuales inician una comunicación enviando una petición al servidor, éste la procesa y envía una respuesta de regreso al cliente.

  Las descargas hechas por los clientes desde el servidor representan la mayor cantidad de transacciones realizadas en internet. En este caso, las peticiones son muy pequeñas comparadas con la información retornada desde el servidor por lo que es muy recomendado para servicios de internet por cable o por satélite.

- *PROXIES.*

  Un proxy es un intermediario en una transacción. Es una aplicación que se ubica entre el cliente y el servidor origen. En ocasiones son usados sobre firewalls para proporcionar seguridad.

  Mediante políticas de cacheo configurables, el servidor proxy selecciona la información que debe cachear, favoreciendo el acceso a dicha información a los clientes más próximos en sus próximas peticiones.

  Un proxy se comporta como un cliente y como un servidor. Recibe peticiones de los clientes y las procesa, para luego reenviarlas a los servidores origen. El proxy es una puerta de enlace de la capa de aplicación, caracterizado por tener dos conexiones TCP: una para el cliente y otra para el servidor.

- *OBJETOS WEB.*

  Al decir objeto nos referimos a toda la información intercambiada entre un cliente y un servidor. Estos tienen diversas características como tamaño (numero de bytes), tipo (HTML, imagen, audio, etc), tiempo de creación y tiempo de última modificación.

  Los recursos web pueden ser considerados como dinámicos o estáticos. Los recursos dinámicos son aquellos que cambian continuamente por lo que sus respuestas son generadas después de cada solicitud. Las respuestas para los recursos estáticos, son generadas previamente, independientemente de si el cliente las pide o no.

El principal objetivo del cache es almacenar algunas de las respuestas recibidas del servidor origen. Una respuesta se puede almacenar en caché si puede ser usada para una petición futura.

Un cache decide si una respuesta se debe cachear observando diferentes componentes de la petición y la respuesta. En particular, esta examina lo siguiente:

- El código de estado de la respuesta

- El método de petición
- Las directivas de *"control de cache"* para las respuestas
- Un validador de respuesta
- Autenticación de la petición

Estos factores interactúan para determinar si la información se puede cachear, pero el control final lo tiene la directiva de "control de cache" que son las que validan o no el cacheo.

II.3. *PEER TO PEER (P2P)*

A diferencia de la sección anterior, donde se describió el uso de la arquitectura P2P para la aplicación específica de MMOG, en esta sección describiremos los principios de esta arquitectura para la distribución de contenidos.

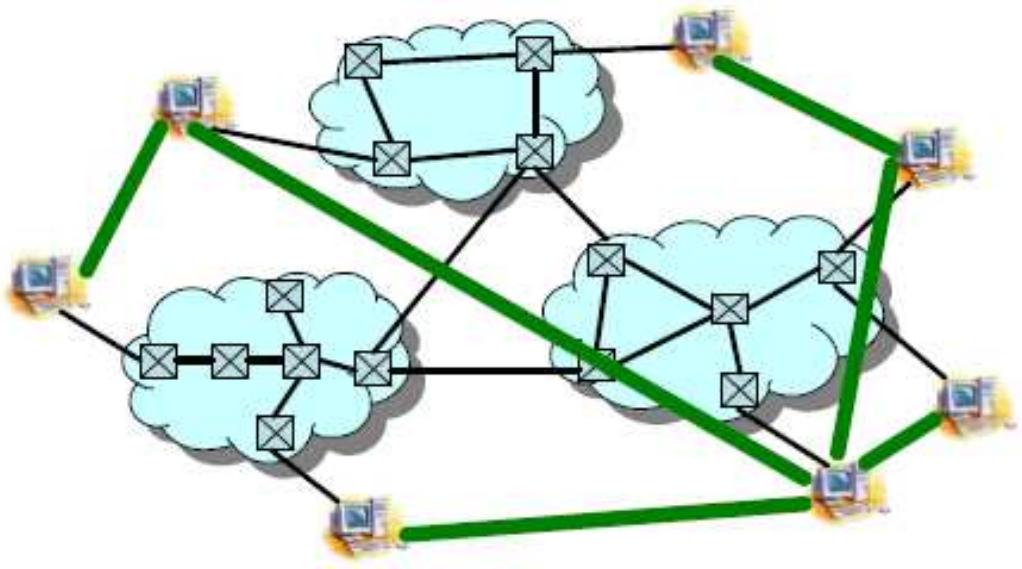

Fig. 8. Distribución de contenidos con P2P

El termino P2P se refiere a redes y aplicacionesque emplean recursos distribuidos (potencia de computación, almacenamiento de datos, ancho de banda de red, etc.) para desarrollar distintas funciones de una manera descentralizada. En general, es una red entre peers (pares), que no tiene clientes y servidores fijos, sino una serie de equipos que se comportan como clientes y como servidores de los demás equipos de la red. Cualquier equipo puede iniciar o completar una transacción compatible. Los equipos pueden diferir en configuración local, velocidad del proceso, ancho de banda de su conexión a la red y capacidad de almacenamiento. [10]

Debido a que los clientes modifican su IP continuamente, es necesaria una entidad que conozca la ubicación de la información en la red.

La solución habitual es realizar una conexión a un servidor (o servidores) con dirección conocida, que se encarga de mantener la relación de direcciones IP de los clientes de la red, de los demás servidores e información adicional, como un índice de la información de que disponen los

clientes. Con esto, los clientes disponen de información sobre la red y pueden intercambiar información entre sí, sin ayuda de los servidores.

Esta tecnología es la base de diversas aplicaciones como compartir contenidos, envío de mensajes instantáneos, herramientas de colaboración, acceso remoto a otras máquinas y el control de archivos.

Los objetivos que buscan las aplicaciones P2P son:

- *REDUCIR EL COSTO DE LOS PROCESOS*

  Una arquitectura P2P puede ayudar a distribuir el coste compartiéndolo entre los equipos. Un ejemplo de esto es el sistema NAPSTER, en donde se comparte el espacio de almacenamiento de ficheros para compartir, manteniendo el índice requerido para dicha transferencia.

- *PERMITIR LA AGREGACIÓN DE RECURSOS E INTEROPERABILIDAD*

  Cada nodo en el sistema P2P trae consigo ciertos recursos como la potencia computacional o el espacio de almacenamiento. Las aplicaciones que se benefician de todos estos recursos como las simulaciones computacionales intensivas o los sistemas de distribución de archivos, conducen todos estos recursos para solucionar un problema mayor. Sistemas como SETI @ Home o Distributed.net son ejemplos de este sistema. La agregación de diversos recursos para un mismo objetivo requiere que exista interoperabilidad entre los usuarios.

- *MEJORAR LA ESCALABILIDAD Y FIABILIDAD*

  Debido que no hay una autoridad central para los pares autónomos, la mejora de la escalabilidad y fiabilidad es un objetivo importante. Debido a esto, se han desarrollado y se sigue investigando en algoritmos para realizar búsquedas y encontrar recursos, con lo cual se han desarrollado diferentes plataformas de P2P. La escalabilidad y fiabilidad se definen en el sistema distribuido, como el uso del ancho de banda, cuantos sistemas pueden acceder a un nodo, cuantos sistemas pueden ser soportados, cuantos usuarios pueden ser soportados y cuanto almacenamiento puede ser usado. La fiabilidad está relacionada a los fallos del sistema y la red, desconexión, disponibilidad, recursos, etc.

- *INCREMENTAR LA AUTONOMIA*

  En muchos casos, los usuarios de un sistema distribuido no están dispuestos a confiar en cualquier proveedor de servicios centralizados. En lugar de ello, prefieren que todos sus datos y trabajo realizados en su nombre se desarrollen a nivel local. El sistema P2P soporta este nivel de autonomía. Un ejemplo de esto es el sistema de compartir archivos como Napster, Gnutella y FreeNet. En cada caso, los usuarios son capaces de obtener archivos que no están disponibles en ningún servidor central, debido a restricciones de licencia.

- *MEJORAR LA PRIVACIDAD Y ASEGURAR EL ANONIMATO*

  Un usuario puede que no quiera que nadie o cualquier proveedor de servicio se entere de su participación en el sistema. Con un servidor central es difícil garantizar el anonimato debido a que el servidor suele estar en condiciones de identificar al cliente, al menos con la dirección IP. Utilizar una estructura P2P donde las actividades se desarrollan localmente, evita que los usuarios envíen información sobre ellos. FreeNet utiliza un sistema de reenvío de mensajes para asegurar que el solicitante original no puede ser rastreado.

- *FACILITAR EL DINAMISMO*

  P2P asume que el entorno es sumamente dinámico. Es decir, que operaciones como el cálculo de los nodos, entran y salen del sistema continuamente. El P2P es recomendable cuando la aplicación está orientada a un entorno ampliamente dinámico. En aplicaciones de comunicaciones, como la mensajería instantánea, las llamadas "listas de amigos" se utilizan para informar a los usuarios cuando están disponibles las personas con las que desea comunicarse. Sin este apoyo, los usuarios se ven obligados a sondear para encontrar amigos para el uso del chat, enviándoles mensajes periódicos.

II.4. *RED DE DISTRIBUCIÓN DE CONTENIDOS (CDN)*

Las redes de distribución de contenido (CDN, Content Delivery Network) son redes de capa de aplicación formadas por un conjunto dedicado de servidores (surrogates), distribuidos en las cercanías de los clientes los cuales proporcionan contenidos con unos valores reducidos de latencia. Su principal objetivo es reducir la latencia percibida por el cliente y la balancear la carga.

Los servicios proporcionados por una CDN mejoran el acceso a contenido especializado al abordar tres áreas básicas en el ámbito del networking: velocidad, fiabilidad y escalabilidad.

Las CDN son redes cuyo propósito general estriba en reducir el tiempo de espera de los usuarios, lo cual implica reducir la latencia de la red y el tiempo de proceso en los servidores. Dada la falta de una gestión globalizada de internet, las empresas han implementado soluciones de escalado *hacia arriba* mediante el uso de servidores cada vez más potentes o, más recientemente, escalado *hacia fuera*, pero de una forma local, con arquitecturas basadas en clústers. Una CDN es un ejemplo de escalado global hacia fuera que trata de reducir la latencia evitando rutas congestionadas, lo que conduce a una reducción del tiempo de respuesta percibido. [11]

Los principales componentes de esta arquitectura son: servidores origen, surrogates, clientes, red de acceso, red de distribución, "content manager" y redirector. [12]

- *SERVIDORES ORIGEN*

  Los servidores origen son servidores de los proveedores de contenidos, que contienen la información a ser distribuida o accedida por los clientes. Esta información puede ser clasificada en contenidos estáticos y dinámicos. Hoy en día, el principal tipo de contenidos

distribuidos con CDN son estáticos pero se están llevando a cabo diversos esfuerzos para distribuir los contenidos dinámicos.

Si la CDN es contratada por un proveedor de contenidos, este delega a la CDN los nombres de los recursos a ser distribuidos. El servidor de origen distribuye los contenidos delegados a los surrogates los cuales la almacenan en caché y esperan las peticiones de los clientes.

- *SURROGATES*

Los surrogates son servidores replica de los servidores origen, actuando como servidores de caché de proxy con la capacidad de almacenar y entregar contenidos. Una CDN generalmente está clasificada de acuerdo a su estructura, al número de servidores surrogate, a la localización de esos servidores y al algoritmo ejecutado para determinar los servidores que deben servir cada petición.

Los surrogates están compuestos por tres componentes:

  - Portal: Es un servidor web basado en HTTP, el cual proporciona acceso a los contenidos almacenados en la CDN.
  - Servidor de Streaming: Un servidor de streaming media se encarga de distribuir contenidos multimedia a los clientes, usando los protocolos estándar como RTP/RTCP para la distribución de streaming media y RTSP para el control del streaming media.
  - Base de Datos: La base de datos del surrogate contiene una lista de todas sesiones streaming disponibles, los objetos almacenados en el surrogate y la información para el control de la CDN.

- *CLIENTES*

Los clientes son individuos con un ordenador que realizan peticiones y descargan una pieza en particular del contenido almacenado en algún lugar de la CDN. La CDN generalmente se encarga de grupos de clientes en lugar de clientes individuales.

- *RED DE ACCESO*

Los clientes acceden al servicio proporcionado por la CDN a través de diferentes redes de acceso, que pueden ser fijas o móviles, de banda ancha o estrecha, dependiendo del proveedor de servicio (ISP). Los surrogates generalmente se ubican en los “puntos de presencia” (POP) de los ISP, para servir de clúster de los clientes que acceden a internet por cada ISP. Con la ayuda de multicast es posible mejorar el proceso de entrega dentro de una red de acceso servida por uno o más surrogates.

- *RED DE DISTRIBUCIÓN*

Esta interconecta los servidores origen con los servidores de streaming multimedia para distribuir los objetos multimedia dentro de la CDN. Hay diferentes tipos de redes de distribución, pero las más usadas son las redes de satélites y los arboles overlay.

En todos los casos, la distribución desde los servidores origen hasta los surrogates se realiza en masa, para optimizar el consumo de ancho de banda y evitar la imprevisibilidad y los problemas de rendimiento del internet.

- *CONTENT MANAGER*

La principal tarea del content manager es el control de los objetos multimedia almacenados en cada surrogate, proporcionándole esta información al Redirector con el fin de servir al cliente con el surrogate más adecuado.

El principal componente del Content Manager es el Localizador de Contenidos, el cual está encargado de determinar: (i) el número de réplicas de un objeto multimedia, (ii) en cual surrogate tiene que ser almacenado un objeto multimedia, (iii) eliminación de objetos no populares de los surrogates, (iv) interacción de la CDN con los servidores origen, (v) actualización de los objetos multimedia en los surrogates cuando una nueva versión está disponible en los servidores origen, y (vi) mover los objetos multimedia hasta los surrogates.

Hay dos tipos de mensajes que se utilizan durante el proceso de gestión de contenidos: mensajes de actualización y mensajes de reporte. Dichos mensajes son enviados por el Content Manager a los surrogates para informarles sobre los cambios en las políticas o actualizar la información de control. Los mensajes de reporte son enviados de vuelta por los surrogates para informar al content manager de situaciones excepcionales, por ejemplo, si se detecta un flash-crowd. La información manejada por el Localizador de Contenidos es almacenada en una base de datos.

- *REDIRECTOR*

Este módulo proporciona inteligencia al sistema, porque estima el surrogate más adecuado para cada petición y cliente. Tiene tres módulos diferentes: CDN_{DNS}, Modulo Monitor y Algoritmo de Redirección.

El CDN_{DNS} acepta peticiones de DNS de los clientes y envía la correspondiente respuesta para enrutar al cliente al surrogate más adecuado. La dirección y el nombre de los surrogates y alguna información adicional esta almacenada en la DB_{DNS}, la cual mantiene todos los registros e información necesaria para responder las peticiones de los clientes cuando solicitan un contenido.

El Modulo Monitor, obtiene periódicamente información estadística de diferentes elementos y lleva a cabo una variedad de medidas para obtener información de la red y los componentes de la CDN.

Finalmente, el algoritmo de redirección, basado en la información recolectada por el Modulo Monitor, selecciona el surrogate óptimo para la petición que ha realizado un cliente.

El proceso que se realiza en una CDN para la distribución de contenidos en el siguiente: [13]

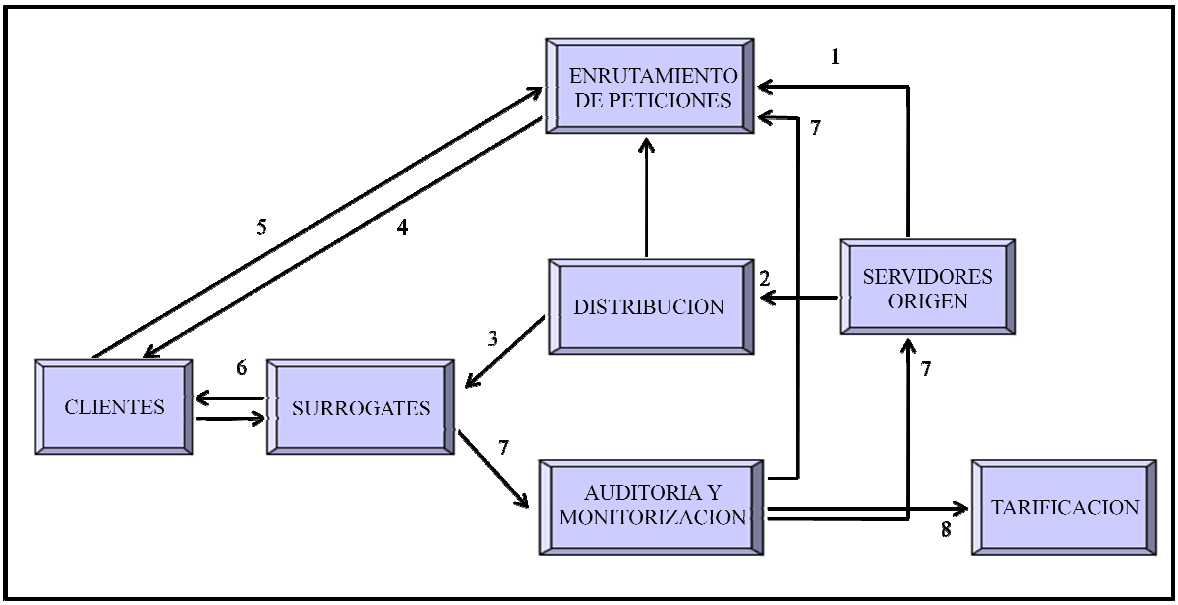

Fig. 9. Subsistemas que componen la CDN.

- Los servidores origen delegan al sistema de enrutamiento de peticiones la lista de elementos que desean distribuir.
- Los servidores de origen publican los contenidos que serán distribuidos y entregados por la CDN dentro del sistema de distribución.
- El sistema de distribución mueve los contenidos a los surrogates. Además, este sistema interactúa con el sistema de enrutamiento de peticiones a través de una retroalimentación para ayudar a los surrogates en los procesos de selección de las peticiones de los clientes.
- El cliente realiza las peticiones al que él percibe como origen. Sin embargo, debido a que la lista de elementos ha sido delegada, las peticiones son redireccionadas al sistema de enrutamiento de peticiones.
- El sistema de enrutamiento de peticiones enruta las peticiones a los surrogates más adecuados de la CDN.
- El surrogate seleccionado entrega el contenido solicitado por el cliente. Además, el surrogate envía al sistema contable la estadística de los contenidos entregados.
- El sistema contable utiliza la información proporcionada por el surrogate para darle información de actualización al sistema de enrutamiento de peticiones y al sistema de facturación.
- El sistema de facturación usa los registros para generar la facturación del proceso de distribución y entrega.

III. LA CDN APLICADA A MMOG

Como vimos en la sección I, existen diversos tipos de MMOG caracterizados por la forma en que el usuario interactúa con el mundo del juego. Adicionalmente, cada tipo de juego presenta diferencias con respecto a la forma en que acceden los usuarios y la distribución de la información en los servidores.

Por esta razón, analizaremos por separado la implementación de la CDN en los MMOG, ya que las soluciones que presentamos varía dependiendo el tipo de juegos.

III.1. *LA CDN EN MMORPG*

Tradicionalmente, éste tipo de juegos no representan un problema importante ni para los desarrolladores ni para los usuarios por lo que no ha sido un tema de mucho estudio.

Este tipo de juegos están caracterizados por distribuir el mapa del juego en diversos servidores, impidiendo la interacción directa entre jugadores de diferente servidor.

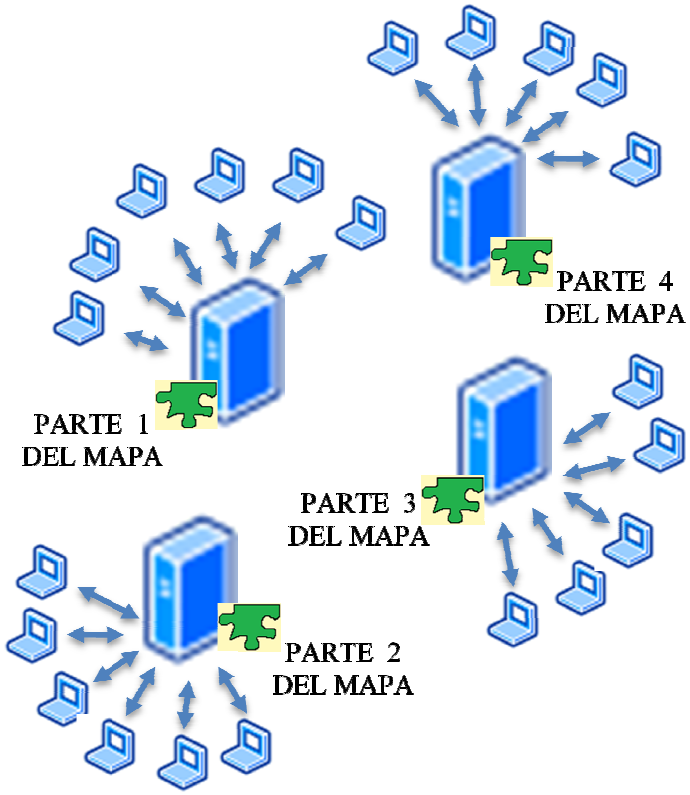

Fig. 10. Implementación habitual de MMORPG

Aunque esto no se considera como un problema, la utilización de un mismo mapa para un gran número de jugadores simultáneos podría mejorar la aceptación de los clientes y la gestión de los recursos en los servidores.

La implementación de la CDN en este tipo de juegos se realiza distribuyendo el mismo “gran” mapa en todos los surrogates, de tal forma que cuando un jugador quiere entrar al juego, el content

manager le asigna el surrogate que le proporciona las mejores prestaciones, y cuando es necesario lo conmuta a otro sin dejar de pertenecer al mismo juego.

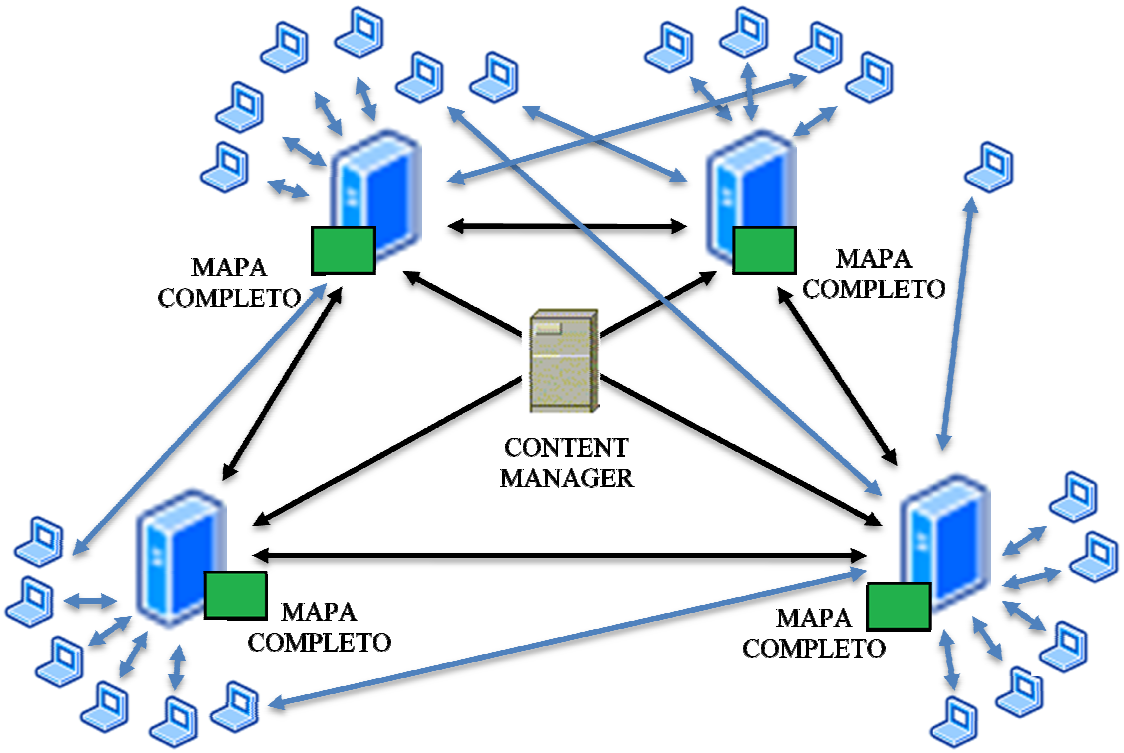

Fig. 11. Implementación de MMORPG en una CDN

La comunicación entre los surrogates se realiza por medio de multicast por lo que todos los surrogates disponen de la misma información. La ubicación y distribución de dichos surrogates depende de un estudio del mercado realizado para cada juego en particular, con el fin de acercarlos a los clientes.

Una ventaja importante de esta aplicación es que se puede garantizar una mínima latencia para el jugador durante todo el tiempo que dure su sesión en el juego.

III.2. *LA CDN EN MMORTS Y MMOFPS*

Este tipo de juegos con elevada tasa de transferencia dependen de la latencia entre el cliente y el servidor como factor fundamental para su calidad de servicio. Cabe recordar que la diferencia entre la latencia de un jugador y otro puede determinar quién gana una partida.

El principal problema planteado por este tipo de juegos es la cantidad de usuarios que puede soportar simultáneamente un servidor, debido al coste computacional y el ancho de banda.

Cuando un cliente de este tipo de juegos desea iniciar una sesión, se conecta al servidor central de su juego (Figura12.1) y éste le muestra una lista de todos los servidores en los que se puede iniciar una partida, con su respectiva latencia y jugadores conectados (Figura 12.2). El jugador puede iniciar una sesión, generalmente, seleccionando un servidor aleatoriamente entre los de menor latencia o uniéndose a un servidor donde ha organizado una cita con un grupo de amigos

(Figura 12.3), manteniéndose conectado a dicho servidor hasta que finalice su sesión o hasta que el administrador del servidor lo decida.

Cuando se llega al máximo de usuarios conectados a un servidor simultáneamente, el servidor se cierra impidiendo el inicio de nuevas sesiones.

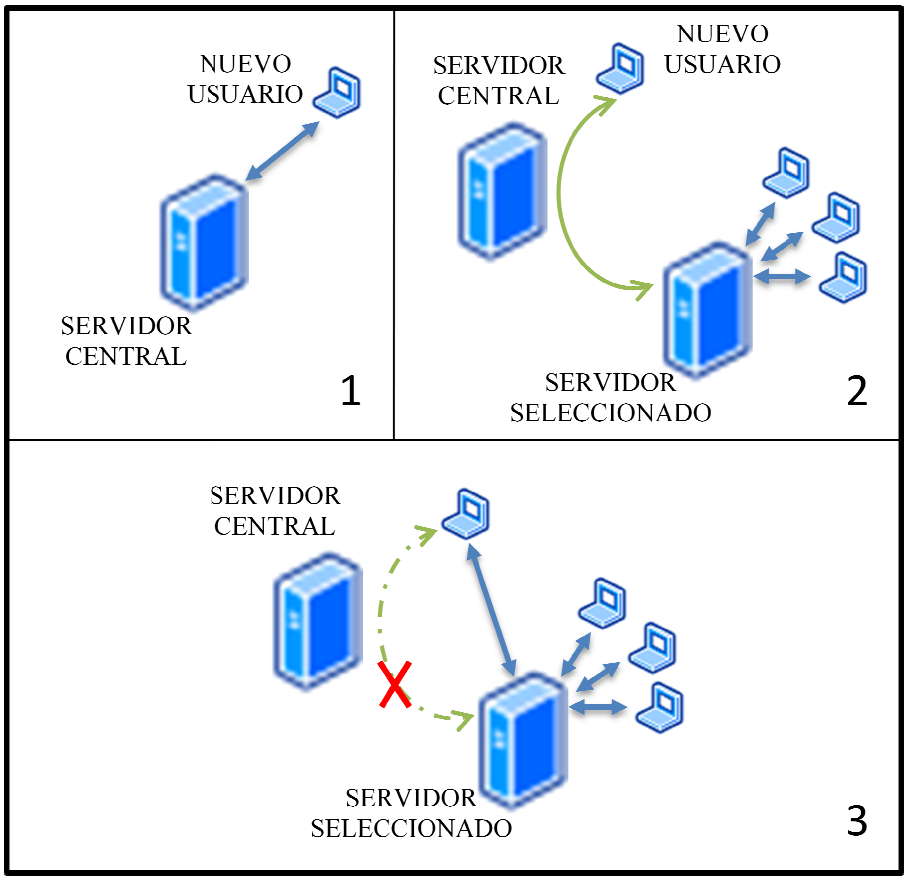

Fig. 12. Inicio de una sesión de MMOFPS o MMORTS

La implementación de la CDN en esta arquitectura depende de lo que desee el usuario:

- *INICIAR UNA SESIÓN SIN PREFERENCIAS DE JUGADORES*

  Si el usuario no tiene ninguna preferencia sobre los jugadores con los que desea compartir una sesión de juego, la CDN lo enruta al surrogate que mejor latencia le proporciona, y lo conmuta cada vez que sea necesario para mantenerlo en el mejor de los surrogates.

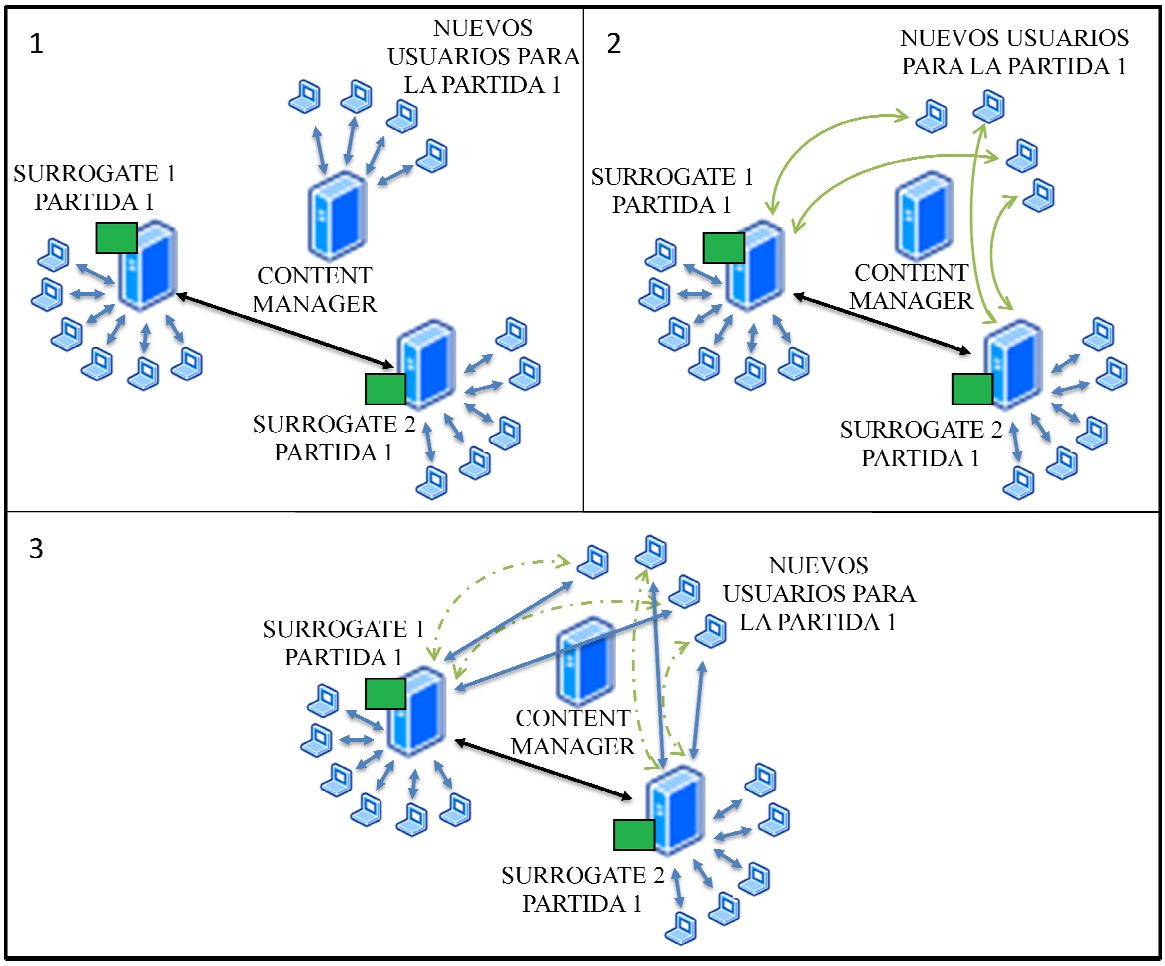

Fig. 13. Inicio de una partida sin preferencia de jugadores

Como se aprecia en la gráfica, los clientes le manifiestan al content manager su intención de participar en la Partida 1 (Figura 13) y éste los redirecciona a cada cliente al surrogate que mejor latencia le proporciona, dentro de la misma Partida.

Cada surrogate dispone de una copia del juego y se une a los demás de manera multicast para mantener actualizado el estado de la sesión.

Según el algoritmo utilizado por la CDN, se pueden generar surrogates finitos en las proximidades de los clientes, ya que no tiene ningún sentido generar un surrogate que no va a ser utilizado por ningún usuario.

- *INICIAR UNA SESIÓN CON PREFERENCIA DE JUGADORES*

Si un usuario ha hecho una cita con un grupo de amigos para participar simultáneamente en un juego, es posible que la CDN realice una “gestión de grupo”

La gestión de grupo consiste en que una vez esté conformado el grupo y que la CDN conoce la ubicación de cada participante, ésta “engancha” el grupo al surrogate que les proporcione una latencia promedio a todos participantes. Si en algún momento es necesario

realizar la conmutación a un surrogate que proporciona mejor latencia, el proceso se hace con todo el grupo.

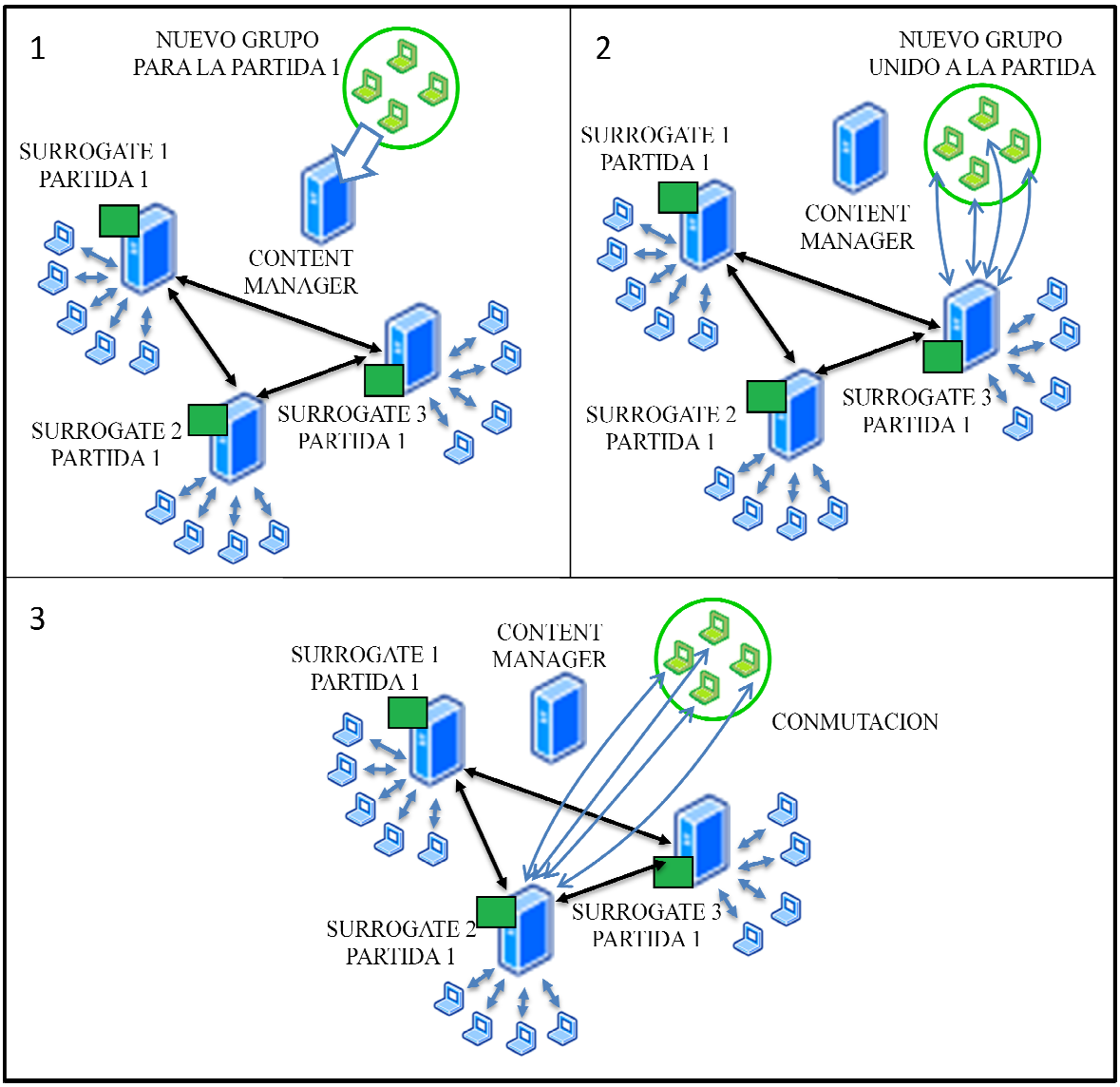

Fig. 14. Inicio y conmutación de una partida de grupo

Si un grupo de usuarios desea participar de una Partida, se lo manifiestan al content manager (Figura 14.1) el cual calcula el surrogate que proporciona la latencia media a los miembros del grupo. Una vez identificado, une el grupo a dicho surrogate (Figura 14.2), y los conmuta cuando otro surrogate ofrezca mejores prestaciones (Figura 14.3).

De esta forma, no existiría ninguna ventaja entre los usuarios del grupo, ya que todos tendrían una latencia media y los clientes con poco ancho de banda podrían aprovechar las capacidades de las redes de los usuarios con banda ancha.

La implementación de una CDN en los tipos de juegos MMOFPS y MMORTS permite acercar el servidor de juegos (surrogate) a la subred del usuario, reduciendo la latencia y permitiendo la participación de usuarios con poco ancho de banda.

IV. IMPLEMENTACIÓN DE UNA CDN PARA MMOG

Como hemos explicado en las secciones anteriores, existen diversos mecanismos que pueden ser implementados para mejorar la calidad del servicio en los juegos en red y llegar a más y más usuarios.

Para la realización de esta tesina, hemos seleccionado la red de distribución de contenidos (CDN) debido a que nos permite acercar los servidores al cliente en una comunicación que tiene que ser constante. Es de especial interés la CDN porque el "content manager" asume la responsabilidad de conmutar al cliente, de un servidor a otro, cuando la calidad del servicio de su servidor actual ha disminuido. Hay que aclarar, que si un servidor se encuentra en la misma subred del usuario, no necesariamente es el servidor que mejores prestaciones ofrece al jugador, ya que éste, además de soportar el servicio de juegos, soporta muchas otras aplicaciones que consumen sus recursos.

Molina Moreno et al [12] implementaron un algoritmo de redirección basados en el algoritmo de balanceo de carga de Castro et al [14] y realizaron diversas pruebas reales en una red pequeña con tres clientes y tres surrogates para comprobar su eficiencia.

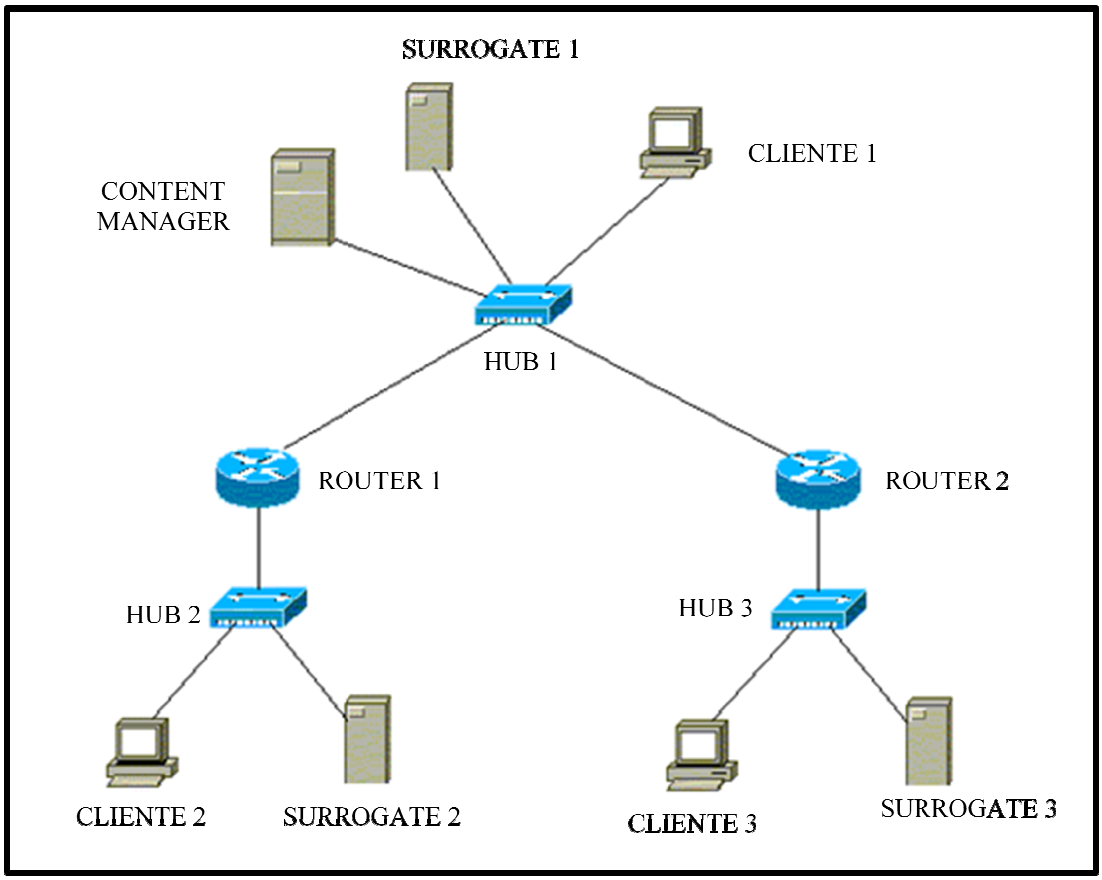

Fig. 15. CDN implementada en las simulaciones

En dicho estudio, debido a la naturaleza de la carga de los servidores, es difícil apreciar gráficamente la eficiencia de su algoritmo, por lo que en esta sección, explicaremos los principios de éste, pero para las simulaciones estableceremos la carga de los servidores de tal forma que tengan naturaleza lineal (para mejorar la apreciación gráfica) y simularemos dicho algoritmo, con algunas modificaciones que se explicarán posteriormente, para apreciar los resultados y analizar diversas variables por separado. Las simulaciones aquí presentadas fueron hechas con MATLAB.

IV.1. *ALGORITMO DE REDIRECCIÓN*

En un entorno de una CDN, un cliente debe ser redireccionado al surrogate más apropiado, lo cual requiere cierto grado de inteligencia. El modulo redirector tiene tres inconvenientes que debe prevenir: (i) realizar el enrutamiento a contenidos que no existen, (ii) diferentes cantidades de carga provenientes de diferentes áreas podrían sobrecargar un servidor y (iii) los mecanismos de cacheo del servicio DNS pueden provocar que no se redireccione cada petición al servidor adecuado.

Cuando la información a transmitir entre el cliente y los surrogates es pequeña, como la información contenida en una página web, no es innecesario llevar a cabo tareas extras para determinar cuál es el surrogate más cercano ya que cualquiera puede proporcionar la información en un periodo de tiempo muy pequeño. Para nuestra aplicación, los juegos MMOG, los paquetes intercambiados entre el cliente (jugador) y el surrogate no son grandes, pero la transmisión es constante por lo que puede ser comparado con tamaños de paquetes muy grandes.

La ecuación general de redireccionamiento de Molina Moreno et al, además de considerar la carga de los surrogates, considera la distancia temporal y espacial con respecto al cliente. La carga de los surrogates puede ser medida con la ayuda de variables como la utilización de la CPU, el consumo de memoria y el número de conexiones activas, por lo que los surrogates deben disponer de elementos que proporcionen esta información al content manager para la ejecución del algoritmo. La distancia temporal es posible medirla con el tiempo de ping entre el cliente y el surrogate, y la distancia espacial se puede calcular por el número de saltos entre subredes.

Para facilitar la comprensión de las ecuaciones, la siguiente tabla presenta un resumen de las variables usadas:

$x(i,j)$	Carga del servidor en el *ith* surrogate durante el *jth* intervalo de tiempo
$m(i,j)$	Memoria usada en el *ith* surrogate durante el *jth* intervalo de tiempo
$l(i,j)$	Utilización de la CPU en el *ith* surrogate durante el *jth* intervalo de tiempo
$L(i)$	Máxima carga de CPU deseada en el surrogate *i*
$c(i,j)$	Número de conexiones establecidas con el *ith* surrogate en el *jth* intervalo de tiempo
$C(i)$	Máximo número deseado de conexiones en el surrogate *i*
$f(i,j)$	Fracción de nuevas sesiones que serán enrutadas al surrogate *i* en el *jth* intervalo de tiempo
$k_1( )$	Función dependiente de $x(i,j-1)$, seleccionada para que los sevidores poco congestionados obtengan un valor alto y los servidores muy cargados obtengan un valor bajo.
$y(i,j)$	Mide la proximidad del cliente al surrogate *ith* durante el jth intervalo de tiempo
$d(i,j)$	Número de saltos de la red entre el cliente y el *ith* surrogate durante el *jth* intervalo de tiempo

p(i,j)	Latencia de la red entre el cliente y el *ith* surrogate durante el *jth* intervalo de tiempo
$k_2(\ )$	Función inversamente proporcional a *y(i,j)*, la cual puede tomar diferentes formas, donde la más simple es la lineal
g(i,j)	Función de ganancia usada para evaluar la adecuación del surrogate *ith* para un cliente, en el *jth* periodo de tiempo en términos geográficos.
h(i,j)	Función que determina la adecuación del *ith* surrogate para una petición de un cliente en el periodo de tiempo *jth*

Tabla 1: Funciones y variables para el algoritmo de redirección

Las siguientes son las funciones que compone el algoritmo de redirección:

- *CARGA DEL SERVIDOR:*

La función *x(i,j)* representa la carga del servidor en el *i*th surrogate durante el intervalo *j*th de tiempo, de la siguiente forma:

$$x(i,j) = \alpha_1 \cdot m(i,j) + \alpha_2 \cdot \frac{l(i,j)}{L(i)} + \alpha_3 \cdot \frac{c(i,j)}{C(i)} \quad (1)$$

Donde:

- *m(i,j)* representa la cantidad de memoria usada por el *ith* surrogate durante el *jth* intervalo de tiempo.

- *l(i,j)* es la utilización de la CPU en el ith surrogate durante el jth intervalo de tiempo.

- *L(i,j)* es la máxima carga de CPU deseada en el *ith* surrogate.

- *c(i,j)* representa el numero de conecciones establecidas con el *ith* surrogate en el *jth* intervalo de tiempo.

- *C(i)* es el máximo número de conexiones deseadas para el ith surrogate.

- α_1, α_2, α_3 son factores entre 0 y 1 donde (α_1,+ α_2,+ α_3= 1), los cuales permiten la asignación de diferentes pesos en la ecuación a cada variable según su relevancia (Carga de CPU, memoria y conexiones).

- *NUEVAS SECIONES QUE ACEPTA EL SURROGATE*

La función *f(i,j)* representa el número de nuevas sesiones que serán enrutadas al *ith* surrogate durante el *jth* intervalo de tiempo, considerando la carga que manejan los servidores. Para generar esta función, introducimos el factor k_1 que es una función dependiente el valor anterior de *x*, el cual tiene un valor alto cuando el servidor esta poco congestionado, y un valor bajo cuando el servidor está muy congestionado.

$$f(i,j) = k_1 \cdot [x(i,j-1)] \cdot f(i,j-1) \quad (2)$$

Hay varias implementaciones para el valor k_1 las cuales se expondrán durante las simulaciones. Molina Moreno et al, utilizan la siguiente función:

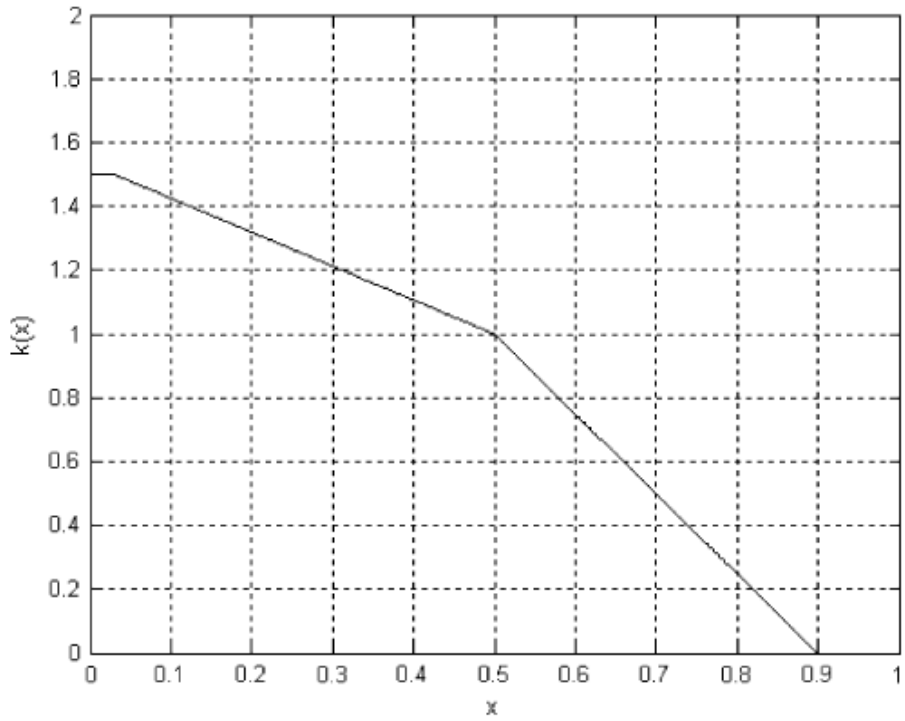

Fig. 16. Función k_1.

Como se verá posteriormente, esta función k_1 determina el tiempo de respuesta para la conmutación de un surrogate a otro.

La simulación e implementación de la CDN debe considerar las siguientes situaciones:

- Si *x(i,j)* = 0 implica que el servidor ha fallado y no serán enviada ninguna petición a este surrogate.
- Al iniciar, o reiniciar la función *f(i,j)*, se debe fijar en un valor f_{min}.
- Si *k* = 0, significa que el servidor esta congestionado, pero un instante de tiempo después puede no estarlo, por lo que se debe reiniciar *f(i,j)* para evitar continuas multiplicaciones por 0.
- *f(i,j)* debe estar normalizado de 0 a 1 para todo *j*.

$$\sum_{i=1}^{N_S} f(i,j) = 1 \qquad \forall j \tag{3}$$

- El valor máximo de *k* se presenta cuando el servidor tiene un mínimo de carga.

- *PROXIMIDAD ENTRE EL SURROGATE Y EL CLIENTE*

La función *y(i,j)* representa la proximidad entre el *ith* surrogate y el *jth* cliente, de la siguiente forma:

$$y(i,j) = \beta_1 \cdot \frac{d(i,j)}{D} + \beta_2 \cdot \frac{p(i,j)}{P} \quad (4)$$

Donde:

- *d(i,j)* representa los saltos de red entre el *ith* surrogate y el *jth* cliente.

- *D* es el máximo número de saltos de red deseados.

- *p(i,j)* representa la latencia de la red (pings) entre el *ith* surrogate y el *jth* cliente.

- *P* es la latencia máxima de la red.

- β_1 y β_2 son factores entre 0 y 1, donde $\beta_1 + \beta_2 = 1$. Esto permite asignar diferentes pesos a la distancia temporal y la distancia espacial.

Para la aplicación de los juego MMOG asignaremos a β_1 el valor de 0.3 y a β_2 el valor de 0.7 debido a que en esta aplicación en especial nos importa más la latencia entre el cliente y el surrogate que los saltos de la red.

Con referencia a los parámetros *D* y *P* de la ecuación, es extraño encontrar en internet o en cualquier red privada un camino del cliente al servidor con más de 30 saltos, y un ping promedio para aplicaciones de juegos en internet mayor de 200 ms.

Una vez más se incluye una función que permite generar una política de asignación de surrogate para la parte del algoritmo de redirección correspondiente a la proximidad del surrogate con el cliente. (k_2)

La función g(i,j) representa la adecuación del ith surrogate para el cliente jth donde:

$$g(i,j) = k_2[y(i,j)] \quad (5)$$

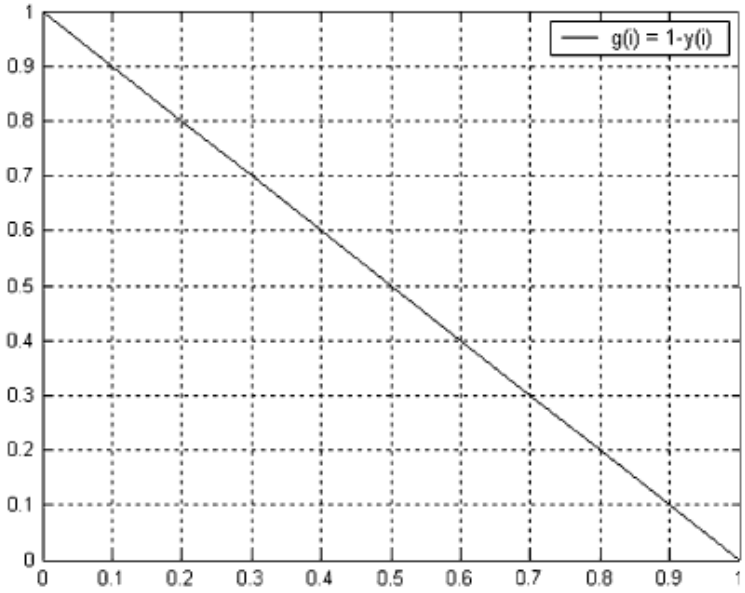

Fig. 17. Función k_2.

- *ALGORITMO DEFINITIVO*

  Una vez ha sido procesada la información sobre el estado de la red por medio de g(i,j), se combina la información de los recursos de los servidores por medio de f(i,j) para obtener el algoritmo de redirección resultante:

$$h(i,j) = \gamma_1 \cdot f(i,j) + \gamma_2 \cdot g(i,j) \quad (6)$$

  Donde γ_1 y γ_2 son factores entre 0 y 1 y ($\gamma_1 + \gamma_2 = 1$), los cuales determinan el peso a los recursos de los servidores o el estado de la red.

IV.2 *SIMULACIONES Y RESULTADOS*

Se desarrollaron y analizaron diferentes simulaciones para verificar la robusteza y rendimiento del algoritmo de redireccionamiento descrito anteriormente.

Como dijimos anteriormente, la carga de los servidores será fijada lineal con variaciones instantáneas, para apreciar gráficamente los resultados y analizarlos más fácilmente. Por esta misma razón, hemos dividido las simulaciones en intervalos de acontecimientos para facilitar los análisis.

La topología utilizada es la de la *Figura 15*.

- *SIMULACION 1*

  La primera simulación que haremos será para analizar la primera parte del algoritmo (consideración de la carga de los servidores).

  Para implementarlo, hacemos $\gamma_1 = 1$ y $\gamma_2 = 0$. La segunda parte del algoritmo (distancia temporal y espacial de los surrogates con respecto a los clientes) no la implementamos por separado porque sabemos que si el surrogate no está sobrecargado, los clientes serán direccionados al surrogate de la misma subred, pero si se sobrecarga, las peticiones serán redireccionadas a los surrogates de otras subredes.

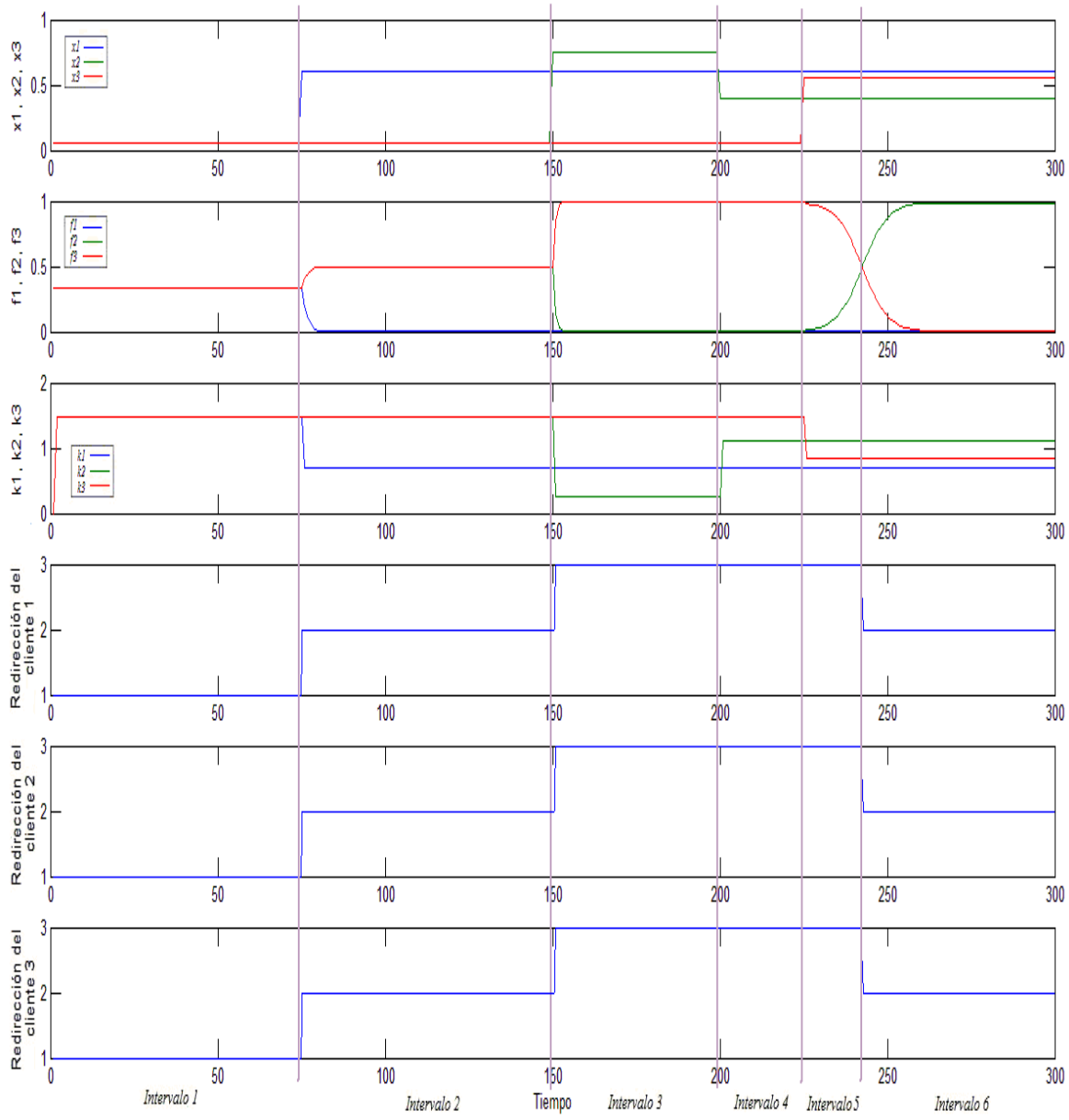

Fig. 18. Simulación de la parte del algoritmo correspondiente al estado de los servidores

Al no considerar la distancia espacial entre el cliente y los surrogates, todos los clientes son direccionados al surrogate que mejor latencia proporciona. En el *intervalo 1* (Figura 18), todos los clientes iniciaron conectados al surrogate 1 debido a que al hacer los cálculos matemáticos se seleccionó el primer surrogate debido a que todos los servidores se encontraban igual de cargados.

En el *intervalo 2*, el servidor x1 se carga por lo que los clientes son conmutados a otro surrogate. Al igual que en el caso anterior, debido a los cálculos matemáticos se redireccionan al surrogate siguiente (surrogate 2).

En el *intervalo 3*, el servidor x2 se sobrecarga dejando al servidor x3 como el de menor carga y por lo tanto todos los clientes se direccionan al surrogate 3.

En el *intervalo 5*, cambian las cargas de cada servidor siendo el servidor x2 el de menos carga, pero solo hasta el *intervalo 6* no se hace la conmutación al surrogate 2. La longitud del *intervalo 5* es de 18 unidades de tiempo y como se puede apreciar en la gráfica de la función *f*, esto se debe al tiempo de respuesta que proporciona la función *k*.

- *SIMULACION 2*

Esta simulación es la implementación de todo el algoritmo (consideración de la carga de los servidores y la distancia espacial y temporal de los surrogates con respecto a los clientes).

Para realizarla, establecimos los valores de $\gamma 1 = 0.5$ y $\gamma 2 = 0.5$ de la *Ecuación 6*, con el fin de dar igual importancia a la carga de los servidores y a la distancia espacial y temporal entre los surrogates y los clientes.

El objetivo de esta simulación es compararla con la *Simulación 1* para apreciar el efecto de la consideración espacial y temporal dentro del algoritmo de redireccionamiento.

Usando los mismos parámetros de la *Simulación 1* obtenemos los siguientes resultados:

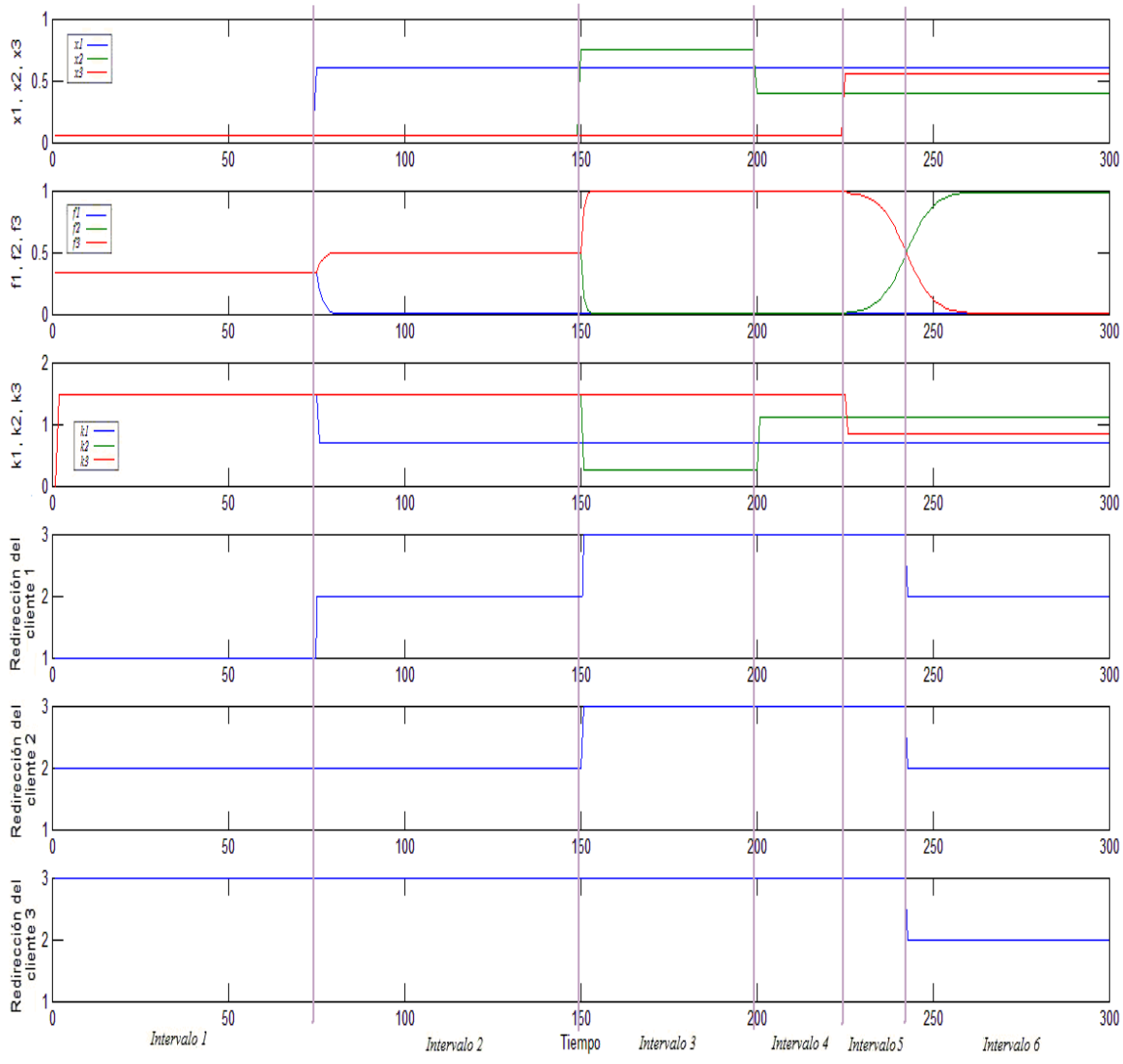

Fig. 19. Simulación del algoritmo de redireccionamiento de la CDN.

De esta simulación cabe resaltar que, como se aprecia en los intervalos de tiempo *1* y *2* (Figura 19), los clientes se mantienen en el surrogate de su subred mientras la carga de los servidores es igual en todos.

En el *intervalo 2*, el servidor *x1* se carga y solo es el cliente 1 el que se sale de su subred en busca de un surrogate menos cargado.

- *SIMULACION 3*

Con el fin de apreciar la incidencia de la función k_1 en el tiempo de respuesta en la conmutación, hemos implementado dicha función con los parámetros variables para apreciar su implicación.

Los parámetros considerados son los siguientes: x_{max}, x_{mid}, x_{min}, k_{max}, k_{mid} y, k_{min}.

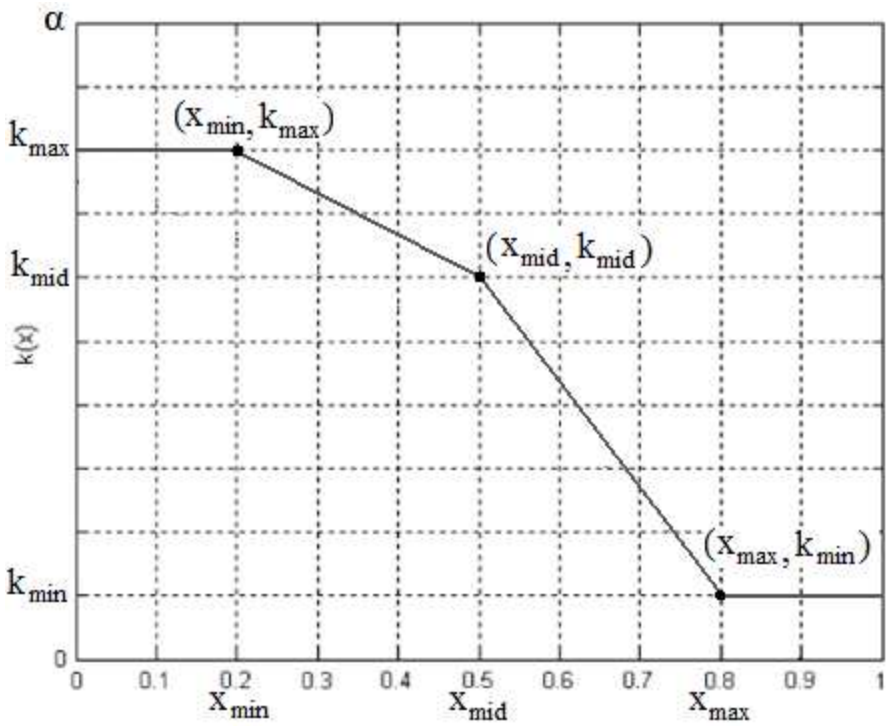

Fig. 20. Modificación de la función k_1 para alterar el tiempo de respuesta.

Si acercamos los valores de x_{min}, x_{mid} y x_{max} a 1 obtenemos lo siguiente:

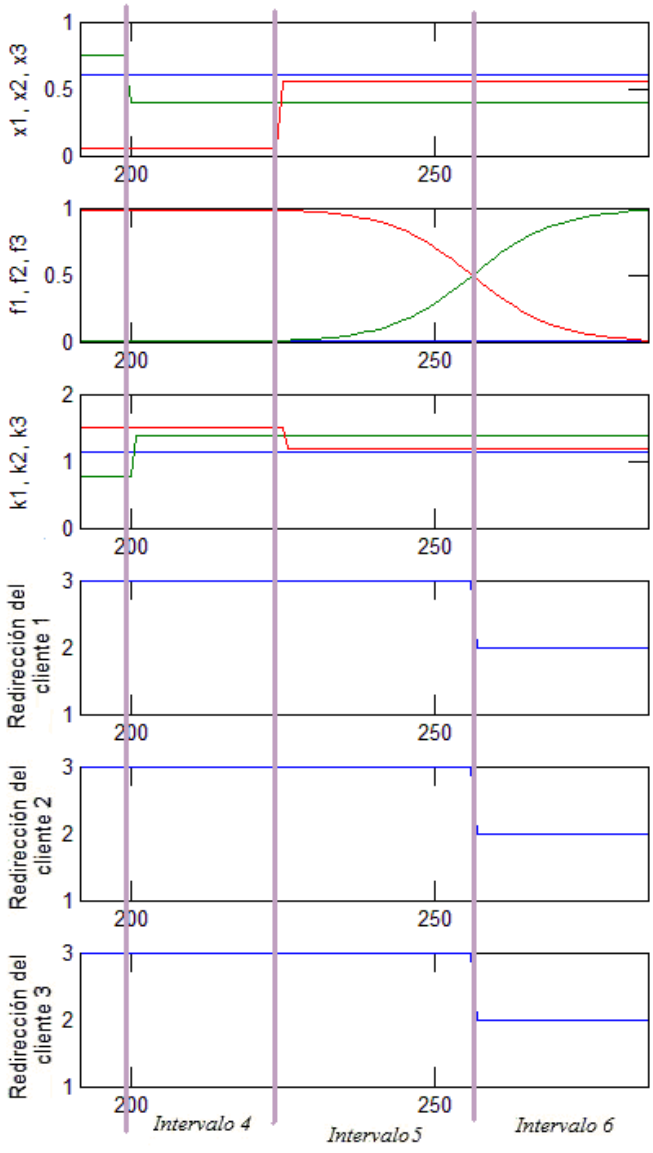

Fig. 21. Simulación desplazando los valores de *x* hacia 1, en la función k_1

Como se aprecia en la Figura 21, en el *intervalo 5*, el tiempo de respuesta aumento, pasando de 18 unidades de tiempo a 32 unidades de tiempo.

Si acercamos los valores de x_{min}, x_{mid} y x_{max} a 0 obtenemos lo siguiente:

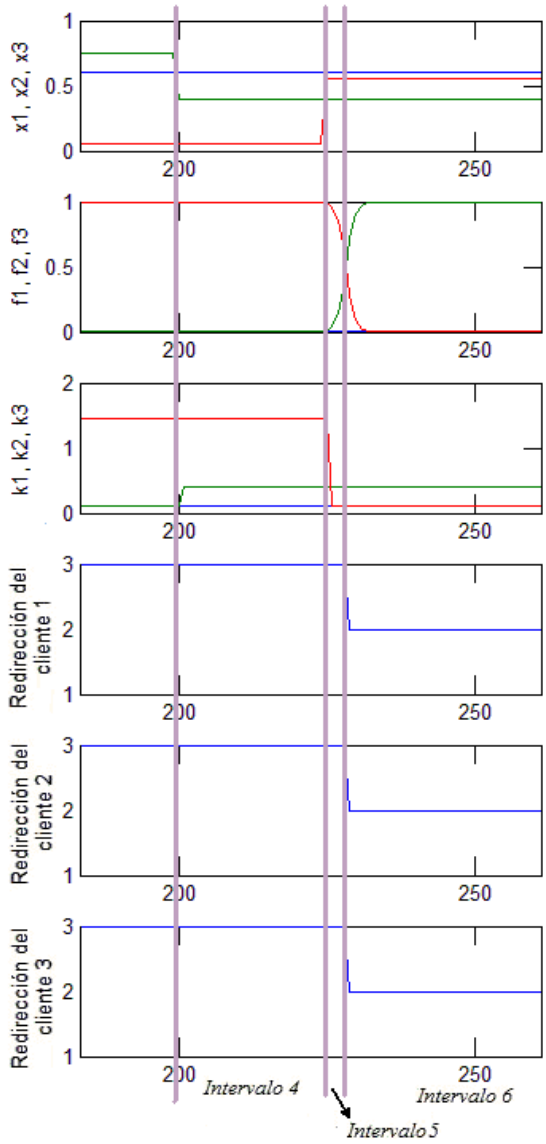

Fig. 22. Simulación desplazando los valores de *x* hacia 0, en la función k_1

Como se aprecia en la Figura 22, en el *intervalo 5*, el tiempo de respuesta disminuyó significativamente pasando de 18 unidades de tiempo a solo 4 unidades de tiempo.

Como comentamos anteriormente, la función k_1 nos permite determinar el momento en el que deseamos hacer la conmutación, basados en la carga de los servidores. Los resultados demuestran que con este algoritmo de redireccionamiento la conmutación es más rápida si la forzamos a realizarse cuando los servidores están aproximadamente por debajo del 50% de su carga.

CONCLUSIONES

Utilizar una Red de Distribución de Contenidos (CDN) para los juegos MMOG nos permite mejorar o evitar los problemas que se presentan con las arquitecturas usadas actualmente. Dichos problemas dependen del tipo de juego:

MMORPG

Los juegos MMORPG disponen de un mapa muy grande el cual es particionado y distribuido entre los servidores, donde los clientes solo pueden interactuar con los otros jugadores de su mismo servidor, y si quiere interactuar con otra parte del mapa, debe cambiar de servidor. En este tipo de juegos, el usuario selecciona el servidor donde quiere jugar por lo que es posible encontrar servidores muy cargados mientras otros se encuentran vacíos.

Implementar una CDN para este tipo de juegos permite disponer del mismo mapa en todos los servidores, donde la asignación del servidor (surrogate) para cada cliente la realiza el Content Manager, favoreciendo el balanceo de carga y ubicando al usuario en el surrogate que mejor latencia le proporciona.

MMOFPS y MMORPS

Los juegos MMOFPS y MMORPS disponen de múltiples mapas de tamaño pequeño ubicados en los servidores donde los usuarios acceden para iniciar una partida que suele ser de una duración finita. Para unirse al juego, el usuario selecciona el servidor en el que desea jugar basado en sus preferencias de mapa o usuarios.

Los principales problemas identificados en este tipo de juegos son las limitaciones de usuarios por servidor debido al coste computacional que representa para los servidores y el ancho de banda que necesitan los clientes para tener la latencia adecuada para una buena calidad de servicio.
Implementar una CDN para este tipo de juegos permite unir más jugadores a la misma partida, mediante la distribución del mismo mapa entre varios servidores, con el fin de distribuir el coste computacional necesario para este tipo de juegos.

Además, la CDN acerca el servidor al cliente, haciendo una réplica en un servidor adyacente de una partida a la que se encuentra unido, favoreciendo la participación de usuarios con escaso ancho de banda.

Como comentan algunos autores, en este tipo de juegos, la diferencia de la latencia de red entre dos jugadores puede definir el ganador y el perdedor de una partida. La CDN puede ser implementada para el control de grupo, asignando un surrogate que proporcione una latencia de red media a un grupo conformado previamente. De esta manera, ningún jugador tendría “ventaja” por su ancho de banda.

Con el objetivo de validar las pruebas realizadas en una CDN [12], simulamos en MATLAB el algoritmo de redireccionador bajo el mismo escenario y establecimos la carga de los servidores para facilitar la apreciación gráfica.

Además de comprobar la eficiencia del algoritmo, modificamos la función de balanceo de carga con el fin de apreciar su implicación en el algoritmo. Modificando algunos parámetros, comprobamos que si forzamos el sistema a conmutar de un servidor más cargado a uno menos cargado antes de que supere el 50% de la carga máxima que puede soportar, el tiempo de conmutación es más rápido.

AGRADECIMIENTOS

Quiero expresar mi más sincero agradecimiento al grupo de Sistemas en Tiempo Real Distribuidos del departamento de Comunicaciones de la Universidad Politécnica de Valencia, por la ayuda brindada para la realización de esta tesina, como apoyo al proyecto EXPESHARE de la convocatoria ITEA 2006.

BIBLIOGRAFÍA

[1] http://www.mmogchart.com/.

[2] T. Henderson, S. Bhatti, *Modelling user behavior in networked games*, ACM Multimedia (2001) 212-220.

[3] W. Feng, F. Chang, W. Feng, J. Walpole, *Provisioning on-line games: a traffic analysis of a busy Counter-Strike server*, in ACM SIGCOMM Internet Measurement Workshop, 2002, pp.151-156.

[4] L. Pantel, L. Wolf, On the impact of delay on real-time multiplayer games, in: Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), 2002, pp. 23-29.

[5] N. Sheldon, E. Girard, S. Borg, M. Claypool, E. Agu, The effect of latency on user performance in Warcraft 3, in: Proceedings of the 2nd Workshop on Network and System Support for Games, 2003, pp. 3-14.

[6] M. Borella, Source models of network game traffic, Computers Communications 23 (4), 2000, pp. 403-410.

[7] T. Lang, G. Armitage, P. Branch, H. Choo, A synthetic traffic model for Half-Life, Australian telecommunications, network and applications conference (ATNAC), 2003.

[8] Cisco, Internetworking Technology Handbook, Chapter 43: Internet Protocol Multicast, pp. 43-1,43-16

[9] D. Wessels, Web Caching, Reducing Network Traffic, O'Reilly, (2001)

[10] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, Z. Xu, Peer-to-Peer Computing, 2002.

[11] B. Molina, C. E. Palau, M. Esteve, Estudio y modelado de una red de distribución de contenidos, IEEE América Latina, Volume 3, Edición 2, 2005, pp. 44-48.

[12] B. Molina Moreno, C. E. Palau Salvador, M. Esteve Domingo, I. Alonso Peña, V. Ruiz Extremera, On content delivery network implementation, in: Computer Communications 29, (2006), pp. 2396-2412.

[13] G. Peng, CDN: Content Distribution Network, Technical Report TR-125, Experimental Computer Science, State University of New York, Stony Brook, NY, 2003.

[14] M. Castro, M Dwyer, M Rumsewicz, Load balancing and control for distribution Worl Wide Web servers, in: Proceedings of the 1999 IEEE International Conference on Control Applications, 1999, pp. 1614-1619.