UNIVERSIDAD POLITECNICA DE VALENCIA

DEPARTAMENTO DE COMUNICACIONES

TESIS DOCTORAL

Estudio, análisis y desarrollo de una red de distribución de contenido y su algoritmo de redirección de usuarios para servicios web y streaming

Autor: Benjamín Molina Moreno

Director: Carlos E. Palau Salvador

Valencia, 2013

Resumen

Esta tesis se ha creado en el marco de la línea de investigación de Mecanismos de Distribución de Contenidos en Redes IP, que ha desarrollado su actividad en diferentes proyectos de investigación y en la asignatura "Mecanismos de Distribución de Contenidos en Redes IP" del programa de doctorado "Telecomunicaciones" impartido por el Departamento de Comunicaciones de la UPV y, actualmente en el Máster Universitario en Tecnologías, Sistemas y Redes de Comunicación.

El crecimiento de Internet es ampliamente conocido, tanto en número de clientes como en tráfico generado. Esto permite acercar a los clientes una interfaz multimedia, donde pueden concurrir datos, voz, video, música, etc. Si bien esto representa una oportunidad de negocio desde múltiples dimensiones, se debe abordar seriamente el aspecto de la escalabilidad, que pretende que el rendimiento medio de un sistema no se vea afectado conforme aumenta el número de clientes o el volumen de información solicitada.

El estudio y análisis de la distribución de contenido web y streaming empleando CDNs es el objeto de este proyecto. El enfoque se hará desde una perspectiva generalista, ignorando soluciones de capa de red como IP multicast, así como la reserva de recursos, al no estar disponibles de forma nativa en la infraestructura de Internet. Esto conduce a la introducción de la capa de aplicación como marco coordinador en la distribución de contenido. Entre estas redes, también denominadas overlay networks, se ha escogido el empleo de una Red de Distribución de Contenido (CDN, Content Delivery Network).

Este tipo de redes de nivel de aplicación son altamente escalables y permiten un control total sobre los recursos y funcionalidad de todos los elementos de su arquitectura. Esto permite evaluar las prestaciones de una CDN que distribuya contenidos multimedia en términos de: ancho de banda necesario, tiempo de respuesta obtenido por los clientes, calidad percibida, mecanismos de distribución, tiempo de vida al utilizar caching, etc.

Las CDNs nacieron a finales de la década de los noventa y tenían como objetivo principal la eliminación o atenuación del denominado efecto *flash-crowd*, originado por una afluencia masiva de clientes. Actualmente, este tipo de redes está orientando la mayor parte de sus esfuerzos a la capacidad de ofrecer streaming media sobre Internet.

Para un análisis minucioso, esta tesis propone un modelo inicial de CDN simplificado, tanto a nivel teórico como práctico. En el aspecto teórico se expone un modelo matemático que permite evaluar analíticamente una CDN. Este modelo introduce una complejidad considerable conforme se introducen nuevas funcionalidades, por lo que se plantea y desarrolla un modelo de simulación que permite por un lado, comprobar la validez del entorno matemático y, por otro lado, establecer un marco comparativo para la implementación práctica de la CDN, tarea que se realiza en la fase final de la tesis. De esta forma, los resultados obtenidos abarcan el ámbito de la teoría, la simulación y la práctica.

Resum

Esta tesi s'ha creat en el marc de la línia d'investigació de Mecanismes de Distribució de Continguts en Xarxes IP, que ha desenrotllat la seua activitat en diferents projectes d'investigació i en l'assignatura “Mecanismes de Distribució de Continguts en Xarxes IP” del programa de doctorat “Telecomunicacions” impartit pel Departament de Comunicacions de la UPV i, actualment en el Màster Universitari en Tecnologies, Sistemes i Xarxes de Comunicació.

El creixement d'Internet és àmpliament conegut, tant en nombre de clients com en tràfic generat. Açò permet acostar als clients una interfície multimèdia, on poden concórrer dades, veu, vídeo, música, etc. Si bé açò representa una oportunitat de negoci des de múltiples dimensions, s'ha d'abordar seriosament l'aspecte de l'escalabilitat, que pretén que el rendiment mitjà d'un sistema no es veja afectat a mesura que augmenta el nombre de clients o el volum d'informació sol·licitada.

L'estudi i anàlisi de la distribució de contingut web i streaming empleant CDNs es l'objecte d'este projecte. L'enfocament es farà des d'una perspectiva generalista, ignorant les solucions de capa de xarxa com IP multicast, així com la reserva de recursos, en no estar disponibles de forma nativa a l’infraestructura d'Internet. Això condueix a la introducció de la capa d'aplicació com a marc coordinador en la distribució. Entre estes xarxes, denominades overlay networks, s'ha escollit l'ús d'una Xarxa de Distribució de Contingut (CDN, Content Delivery Network).

Aquest tipus de xarxes de nivell d'aplicació són altament escalables i permeten un control total sobre els recursos i funcionalitat de tots els elements de la seua arquitectura. Això permet avaluar les prestacions d'una CDN que distribueix continguts multimèdia en termes de: ample de banda necessari, temps de resposta obtingut pels clients, qualitat percebuda, mecanismes de distribució, temps de vida en utilitzar caching, etc.

Les CDNs van nàixer a finals de la dècada dels noranta i tenien com a objectiu principal l'eliminació o atenuació de l'anomenat efecte *flash-crowd*, originat per una afluència massiva de clients. Actualment, este tipus de xarxes està orientant la major part dels seus esforços a la capacitat d'oferir streaming media sobre Internet.

Per a una anàlisi minuciosa, esta tesi proposa un model inicial de CDN simplificat, tant a nivell teòric com a pràctic. En l'aspecte teòric s'exposa un model matemàtic que permet avaluar analíticament una CDN. Este model introduïx una complexitat considerable conforme s'introduïxen noves funcionalitats, per la qual cosa es planteja i desenrotlla un model de simulació capaç d'una banda, de comprovar la validesa de l'entorn matemàtic i, d'altra, d’establir un marc comparatiu per a l’implementació pràctica de la CDN, tasca que es realitza en la fase final. D'aquesta manera, els resultats obtinguts cobreixen l'àmbit de la teoria, la simulació i la pràctica.

Abstract

This thesis has been developed in the frame of the Content Distribution Mechanisms in IP Networks research line, whose activity ranges from several research projects to the subject “Content Distribution Mechanisms in IP Networks” from the PhD program “Telecommunications” taught by the Communication Department from UPV and, currently in the Master in Technologies, Systems and Communication networks.

The growth of the Internet is widely known, both in number of clients and generated traffic. This allows provisioning customers with multimedia interfaces, where data, voice, video, music, etc. converge. Though this represents a business opportunity from multiple dimensions, the scalability issue must be covered, which intends that the average performance of a system is not significantly affected as the number of clients increases or the volume of information requested and exchanged grows.

The study and analysis of video and web content distribution using CDNs is the main goal of this project. The focus will be from a general perspective, initially ignoring network-layer solutions such as IP multicast, as well as resource reservation mechanisms, as they are not natively available in the Internet infrastructure. This leads to the introduction of the application layer as a coordinating framework for the content distribution. Among these networks, also called overlay networks, a Content Delivery Network (CDN) has been used.

Such application level networks are highly scalable and allow full control over the resources and functionalities of all architecture elements. This allows evaluating the performance of a CDN distributing multimedia content in terms of: required bandwidth, response time obtained by users, perceived quality, delivery mechanisms, lifetime when using caching, etc.

CDNs were born in the late nineties and had as main objective the elimination or minimization of the so-called *flash-crowd* effect, which corresponds to a peak load - on the network or server side - caused by a massive implosion of customers. Currently these networks are shifting most of their efforts to the provisioning of streaming media over the Internet.

For a detailed analysis, this thesis proposes an initial simplified model of a CDN, both at theoretical and practical level. The theoretical aspect exposes a mathematical model which allows to analytically evaluating a CDN. This model introduces considerable complexity as new features are added, so the thesis arises and develops a simulation model that allows on the one hand, checking the validity of the previous mathematical model and, on the other hand, establishing a comparative framework for the practical implementation of the CDN. This task is performed in the final phase of the thesis. In this way, the results of the thesis cover the field of theory, simulation and implementation.

Agradecimientos

Finaliza ante mí una etapa académica y personal con la realización y entrega de esta tesis doctoral. Y es el momento adecuado para agradecer a todas las personas que me han ayudado en la realización de la misma, por todo su apoyo profesional y/o personal.

En primer lugar, quiero agradecer a mi director de tesis, Carlos, por sus muchos consejos, ideas y ánimos a la hora de realizar la tesis. Muchos artículos generados a partir del trabajo elaborado en la misma han visto la luz a raíz de su manifiesta, constante y loable iniciativa por la investigación académica.

En segundo lugar, quiero agradecer al conjunto del grupo de investigación en el que me enmarco laboralmente, SATRD, Grupo de Sistemas y Aplicaciones de Tiempo Real Distribuido, por su ayuda y experiencia mostrada que posibilitaron una mejor confección de la tesis. Quisiera destacar la colaboración de Manuel Esteve, director del SATRD, Israel, Luis, Javier, Flavio, Federico, Ximo, Carlos, Alfonso, Fran, David, Pablo e Isaac, (ex)compañeros y amigos con los que más he convivido, y que más veces me han ofrecido unas manos para teclear o programar, o simplemente una ración sonora y positiva de optimismo que siempre alienta en los momentos más desconcertantes.

Trabajar en un laboratorio de investigación en la UPV también me ha permitido convivir y aprender junto con becarios y estudiantes de último curso que hicieron dentro de nuestro grupo de investigación su proyecto final de carrera o tesina de master, algunos relacionados directa- o indirectamente con la distribución de contenidos. Muchas gracias a José Luis, Antonio, Isidoro, Víctor, Francisco y Mario por sus diferentes o nuevos puntos de vista que me han permitido conocer mejor las CDNs.

Tampoco puedo olvidarme del resto de amigos, valencianos, españoles y europeos, que siempre se han interesado por mi tesis, y me han animado y apoyado de forma continuada en el ámbito personal. Entre ellos quisiera destacar a Jaime, María, Nacho, Amparo, Kike, Juan, Dani, Juanjo, Deborah, Ángela y Robert.

Especialmente quiero agradecer y dedicar esta tesis a mi familia; en términos de Ortega y Gasset, ellos siempre han sido mis circunstancias más positivas que me han permitido desarrollarme como persona. Quisiera empezar por los que ya no están presentes, como mis abuelos, Salvador y Simón, mis abuelas, Salvadora y Benita, y mis tíos, Fernando y Gerda. Entre los presentes, que tengo el honor de disfrutar, quisiera dedicar un muy especial agradecimiento a mis padres, Salvador y María, a mis hermanos, Salvador y Simón, y finalmente a mi sobrina Erika. Ellos me han sufrido y acompañado todos estos años, pero sobre todo me han dado su apoyo y cariño más sincero que me ha permitido llegar a este momento. Sin ellos, querido lector, esta tesis no hubiera sido posible de ninguna manera.

Mi más humilde agradecimiento a todos.

A mis padres, Salvador y María

Índice

Índice de Figuras

Índice de Tablas

1. INTRODUCCIÓN

1.1. Antecedentes

Las redes de distribución de contenido son redes de nivel de aplicación. Ello representa una capa de gestión inteligente superpuesta a la infraestructura de red. La utilización de dos niveles para la implementación de la red supone una clara ineficiencia que debe ser analizada atendiendo a los parámetros de diseño o de funcionamiento, sin embargo los beneficios que puede reportar una infraestructura de estas características puede ayudar a mejorar las prestaciones percibidas por los usuarios.

El entorno de funcionamiento de las CDNs abarca grandes redes privadas o incluso Internet y, pese a que su efectividad ha quedado demostrada por informes de las empresas proveedoras de estos servicios, ciertos aspectos internos de funcionamiento interno son desconocidos, no están especificados, o no son públicos, al representar una ventaja competitiva en el ámbito comercial.

Esta tesis pretende, en primer lugar, describir y proponer una red de distribución de contenido libre, en base a estándares y protocolos abiertos comúnmente utilizados en el entorno telemático. Existen actualmente escasas implementaciones libres y abiertas, como Globule [Pie_01] [Pie_03], CoDeen [Wan_04] u OpenCDN [WWW_Ope]. La primera de ellas, desarrollada en la universidad de Ámsterdam, será estudiada y comparada como parte de la tesis. No obstante, esta CDN (Globule) se centra fundamentalmente en contenido web, por lo que su evaluación desde el punto de vista multimedia no es posible. Los proveedores de servicios de CDNs parecen orientar sus esfuerzos en la oferta de servicios de streaming, lo que justifica el carácter novedoso de esta tesis. OpenCDN, por otro lado, es un proyecto que se centra en contenido multimedia, tanto en directo como prealmacenado, lo que permite establecer comparaciones en este aspecto. Lamentablemente, se trata de un proyecto desactualizado (última modificación en 2007), por lo que el contenido multimedia está orientado a clientes fijos, sin atender a las nuevas demandas, requisitos y formatos de los clientes móviles. CoDeen fue, inicialmente, el proyecto más ambicioso de los tres, al estar desplegado sobre PlanetLab [WWW_Pla] y estar estructurado de una forma modular, lo que permite una cierta continuidad en varios subproyectos. Esta modularidad permitiría comparar la CDN planteada y desarrollada en esta tesis con algún aspecto en concreto de CoDeen (actualmente se trata de un proyecto desactualizado o parado). PlanetLab también dispone de otra CDN, CoralCDN [Fre_04] [Fre_10], también de la Universidad de Princeton (también desactualizado).

En definitiva, se desea crear una red de distribución de contenido, incluyendo la capacidad de distribución de streaming media, y evaluar su rendimiento, en base a medidas objetivas, como el tiempo de respuesta de los clientes, el número de clientes y servidores, la ubicación de estos últimos, etc., como se detallará en las secciones 3 y 4 de la memoria.

1.2. Motivación de la tesis

La justificación y motivación de la presente tesis se basa en cuatro consideraciones principales:

- ***La distribución de contenido multimedia tradicional basada en un servidor central no escala apropiadamente para un elevado número de usuarios.*** Un servidor tradicional de gran capacidad consiste en un computador de gran potencia de cálculo ubicado en un centro de datos con una conexión a Internet de alta velocidad. Aunque su rendimiento es posiblemente aceptable para páginas web que generan poco tráfico (texto e imágenes), pronto se verá seriamente afectado ante un significativo número de usuarios que demanden contenido multimedia. Téngase en cuenta que un contenido multimedia como vídeo, voz o música se ve afectado tanto por el retardo como por la variabilidad de ese retardo (*jitter*), y las técnicas de *buffering* pueden resultar insuficientes. Desde otro punto de vista, un servidor tradicional centralizado posee una única conexión a Internet. Este enlace puede colapsarse o sufrir una caída abrupta, resultando inútil toda la potencia de cálculo del servidor al encontrarse completamente aislado del mundo. En este caso, parece apropiado una arquitectura distribuida (una alternativa parcial sería el empleo de *multihoming*). Un sistema distribuido permite la colocación de servidores de manera dispersa, cuya ubicación puede ser elegida de manera eficaz en las cercanías topológicas de los clientes para obtener un tiempo de respuesta reducido. Este es el caso de una red de distribución de contenido.

- ***Las grandes redes no disponen de forma nativa de mecanismos de distribución a nivel de red ni reserva de recursos anticipada, por lo que el desarrollo más veloz y al alcance de la comunidad científica y empresarial sugiere el empleo de redes de nivel de aplicación (overlay networks).*** Internet no dispone de una gestión centralizada de su infraestructura, sino que se constituye como un elevado número de grandes y pequeñas redes interconectadas. La mayoría de las redes, de forma individual, incorporan tecnología IP multicast en los routers mediante protocolos de enrutamiento intra-AS, o incorporan mecanismos de reserva de recursos (como RSVP) de manera anticipada a la transmisión del contenido. Sin embargo, toda esta capacidad se conserva únicamente en la red que lo incorpora (no existen mecanismos de encaminamiento multicast inter-AS). Dado que la comunicación entre dos entidades de Internet (cliente-cliente, cliente-servidor) atraviesa generalmente varias redes, no es posible asumir las capacidades anteriores a través de toda la ruta, es decir, extremo a extremo. Es por ello que surge la idea de optimizar el nivel de aplicación/servicio, el último nivel de la pila de protocolos que caracteriza una comunicación.

- ***Las redes de distribución de contenido (CDNs) son sistemas distribuidos que permiten un control y monitorización total de los recursos existentes, y cuya tendencia actual es la distribución de streaming media.*** Las CDNs nacieron originalmente vinculadas a entornos de redes donde el tráfico solicitado por los clientes era lo suficientemente elevado como para justificar el despliegue de varias decenas, cientos o incluso miles de servidores con la finalidad de ofrecer un servicio de buena calidad y reducida latencia. En cualquier caso, el funcionamiento de la CDN quedaba gobernado por un ISP o una empresa con propósito comercial (multi-ISP), por lo que varios aspectos de su implementación han permanecido ocultos a la comunidad científica o se han manifestado en forma de patentes. Este proyecto pretende mostrar una visión tanto global como transparente de la arquitectura e implementación de una CDN. Dado el gran control sobre los recursos existentes que permite una CDN, parece un sistema adecuado para variar ciertos parámetros y evaluar el correcto funcionamiento, obteniendo resultados cuantitativos propios de una tesis doctoral. Desde otro punto de vista, el enfoque en el estudio de la CDN debe ser vanguardista, por lo que estará orientado a la distribución de contenido streaming media, puesto que es la clara tendencia mostrada por los principales operadores de CDNs.
- ***El control de recursos ofrecido por una CDN permite no sólo una escalabilidad del servicio sino también una elevada eficiencia energética***. En los últimos años, las técnicas de virtualización y el concepto de nube (*cloud computing*) permiten gestionar recursos de red bajo demanda de tal forma que se puede minimizar el consumo energético necesario. Si bien las CDNs pueden comprender una gran cantidad de servidores dispersos geográficamente, el consumo energético de estos no es constante sino que varía dependiendo de las exigencias del tráfico demandado en cada momento. Esto permite minimizar el consumo energético cuando el número de usuarios es reducido, maximizando la viabilidad económica de estos sistemas. Desde este punto de vista, una CDN bien diseñada se puede considerar como un entorno de aplicación idóneo del concepto generalista de *green computing*. En resumen, la tecnología de CDN es aplicable a la gestión interna de un operador de *cloud* que desee una infraestructura eficiente en recursos computacionales y energéticos.

1.3. Objetivos

Los objetivos específicos son los siguientes:

- Crear un *modelo analítico* que describa la arquitectura y las principales funciones de una CDN, como son la ubicación de los clientes, la redirección global y/o local, el caching y las estrategias de distribución de contenido entre *surrogates*. Este modelo analítico debe permitir analizar de una forma básica el comportamiento de una CDN desde el punto de vista del tiempo de respuesta (medio) percibido por los clientes y la carga (medio) en los servidores. Se propondrá una formulación matemática que permita estimar estos valores, de forma que se pueda representar mediante gráficas ciertos comportamientos básicos de la CDN
- Crear una *plataforma de simulación* que permita caracterizar el comportamiento de la CDN en grandes redes. Como simulador académico se tomará el NS-2 [WWW_NS2] que, aunque carece de un módulo específico de CDN, incorpora la suficiente flexibilidad para modelarlo. El modelo de simulación debe estudiar el comportamiento de la CDN para dos tipos de servicios: (i) web, mediante el empleo del protocolo HTTP y (ii) streaming, mediante el uso de los protocolos RTP, RTCP, RTSP. En ambos casos se debe introducir un mecanismo de redirección basado en el DNS de tal forma que se redirija a los clientes a su servidor más adecuado. Para ambos tipos de tráfico (web y streaming) se debe ofrecer una caracterización estadística (tamaño de los objetos, tiempo entre peticiones, duración de las sesiones, etc.) parametrizable.
- *Implementar y desplegar* un prototipo de una CDN real, capaz de proporcionar contenido web y streaming media. Esto incluye la creación de un banco de pruebas totalmente operativo donde varios usuarios puedan probar los servicios de distribución de contenido (web y streaming). Este banco de pruebas incorporará el servidor origen (punto de inserción de contenido web y streaming), los servidores (punto de acceso de los clientes), los clientes (generadores de carga) y el equipamiento de red (switches y routers). El control total sobre todos los elementos permitirá emular diferentes entornos y escenarios de simulación.
- *Evaluar* las prestaciones de esta CDN real con especial interés en el algoritmo de redirección de usuarios y su efectividad, así como comparar su rendimiento con otras plataformas existentes (Globule, CoDeen). Esta comparación se realizará en la medida en que se disponga de acceso a datos de prestaciones, y sea posible replicar las características de funcionamiento en que se han tomado las medidas. Asimismo, también se establecerá una comparación entre los tres modelos de CDN creados: analítico, simulación e implementación.

1.4. Metodología y plan de trabajo

El estudio de la CDN propuesta se debe abordar desde diferentes perspectivas: en primer lugar, es conveniente partir de un modelo matemático que permita el análisis y la apreciación de ciertos comportamientos de una manera rigurosa. En segundo lugar, y puesto que este modelo puede alcanzar una complejidad inabordable, es necesario establecer modelos de simulación que permitan evaluar el funcionamiento de una CDN. Cabe destacar que la simulación corresponde normalmente a un paso previo a la prueba en un prototipo real. La implementación y puesta en escena de la CDN corresponde al paso final, donde el modelo teórico y simulado es comprobado y verificado. Los resultados obtenidos pretenden mostrar el buen funcionamiento de la CDN. En la medida de lo posible, se intentará caracterizar el tráfico y contrastarlo con los resultados ofrecidos por otras entidades y empresas que trabajan en el ámbito de la CDN.

Son varios los campos en los que puede profundizarse en una CDN:

- mecanismos de caching [Doy_02] [Cra_01] [Gad_00] [WWW_Cha],
- análisis de tráfico [Fuj_12] [Cop_05] [Asa_04] [Sar_02],
- redirección de usuarios [Tur_04] [Siv_04] [Ran_03] [Szy_03] [Wan_02],
- ubicación de *surrogates* [Siv_07] [Kan_02] [Cam_02] [Wie_02] [Jam_01],
- selección de *surrogate* [Wu_06] [Che_03] [Fuj_04] [Say_98] [Car_97],
- balanceo de carga [Hof_05] [Gay_04] [Bar_03] [Ant_97] [Gho_95],
- consistencia y gestión del contenido [Wu_06] [Fuj_04] [Che_03] [Ver_02],
- rendimiento [Vak_03] [Mas_03] [Cal_02] [Kri_01] [Dou_01] [Joh_00] y
- streaming [Zhu_11] [Lu_11] [She_11] [Pad_02] [Gre_01]

A continuación se presenta un listado de las actividades del proyecto agrupadas por áreas, que son fundamentalmente cuatro:

- ***Documentación****: esta área engloba aquellas actividades de búsqueda, selección y recopilación bibliográfica relacionada con las CDNs y sus principales tecnologías relacionadas.*
- ***Programación****: esta área agrupa está relacionada con la implementación e introducción de código en varios lenguajes de programación.*
- ***Simulación****: en esta área se configuran y desarrollan todas las simulaciones*
- ***Análisis****: esta área permite analizar y evaluar los resultados obtenidos en los modelos analíticos, de simulación y prototipos reales*

Área 1. Documentación.

No.	Actividades
1	Búsqueda de información bibliográfica
2	Análisis de información
3	Selección de información bibliográfica
4	Desarrollar y redactar el estado del arte
5	Documentación del código
6	Elaboración de la memoria de tesis

Área 2. Programación

No.	Actividades
1	Desarrollo del código de la CDN
2	Entorno operativo de programación: S.O. libre y herramienta de Control de Versiones
3	Implementación de políticas de gestión y distribución de contenidos
4	Introducción de capacidad de implementar trazas (*logs*)
5	Herramienta de análisis dinámico de trazas (*logs*)
6	Introducción de un sistema de monitorización global

Área 3. Simulación

No.	Actividades
1	Desarrollo de un entorno de simulación de CDN mediante NS-2
2	Introducción de capacidades de streaming en el modelo de CDN creado en NS-2
3	Introducción de esquemas de gestión de contenido y políticas de caching
4	Modelo de recuperación ante fallos con NS-2

Área 4. Análisis y estudios

No.	Actividades
1	Modelo matemático de una CDN
2	Incorporación de un modelo de redirección
3	Incorporación de un modelo de caching
4	Estudio de las limitaciones reales por problemas de CPU, memoria, espacio en disco, etc.
5	Estudio de la inicialización
6	Estudio de cortes, caídas e indisponibilidades. Tiempo de recuperación
7	Efectividad del algoritmo de redirección y balanceo de carga
8	Efectividad de las políticas de gestión de contenido
9	Armonización y comparación de los diferentes modelos teóricos, simulados y experimentales
10	Comparación de resultados con otras plataformas de CDN abiertas

1.5. Organización de la memoria

La memoria de esta tesis está estructurada en varios bloques temáticos:

- El primer bloque presenta un *estado del arte* pormenorizado sobre las redes de distribución de contenido, partiendo de las aproximaciones tecnológicas existentes, como los mecanismos de *caching y mirrors*, hasta la arquitectura básica de las CDNs y los principales aspectos que se deben abordar en su estudio e implementación, desde la ubicación de los *surrogates* hasta los mecanismos de consistencia de datos entre estos.

- El segundo bloque comprende el *diseño e implementación* de la CDN, dividido en tres secciones principales. En primer lugar, se establece un modelo teórico y matemático que permite describir y analizar el comportamiento de una CDN. Aunque se asumen ciertas simplificaciones, es posible justificar las situaciones en las que una CDN resulta beneficiosa. En segundo lugar, se establece un modelo de simulación desarrollado en NS-2. En este caso, el modelo de CDN desarrollado es más complejo y más real, y es posible establecer escenarios de estudio más interesantes. En último lugar, se desarrolla un prototipo de implementación, completamente operativo, para validar los datos anteriormente obtenidos en el modelo matemático y de simulación, o bien identificar las diferencias encontradas. Los resultados obtenidos en todo este bloque se centran en la distribución de contenido web.

- El tercer bloque parte del desarrollo anterior de la CDN y se centra en el *servicio multimedia*. En este caso se modela el servicio multimedia y se introduce en el modelo de simulación (NS-2) y el modelo real. Este apartado requiere una especial atención en el análisis, ya que los servicios multimedia son altamente variables.

- El cuarto y último bloque comprende las conclusiones principales extraídas en la elaboración de la tesis, así como unas líneas futuras de posible investigación.

2. ESTADO DEL ARTE DE LAS REDES DE DISTRIBUCIÓN DE CONTENIDO

2.1. Introducción

La transición hacia la Sociedad de la Información en la década de los 90 convirtió Internet en la infraestructura de comunicación estándar a nivel mundial, introduciéndose en casi cualquier ámbito de la sociedad. Se trata del método más sencillo de acceder a la información, que es presentada bajo un marco de navegación en un formato multimedia con la finalidad de ofrecer una mayor interacción. Esta interfaz representa una oportunidad de mercado para las empresas, pero el incremento de tráfico y los requisitos de ancho de banda suponen un serio inconveniente a la hora de conseguir la confianza de los usuarios, puesto que el tiempo de respuesta de las aplicaciones puede ser impredecible. Este problema en particular es tratado por las redes de distribución de contenido (*CDN, Content Delivery Network*), y en este capítulo se abordará su descripción y análisis en profundidad.

En la actualidad, existe una documentación reducida sobre las CDNs en general, su funcionamiento y arquitectura, siendo bibliografía de referencia [Bru_01], [Ver_01], [Pat_07] y [Pat_08]. Junto a esta bibliografía, existen muchos otros artículos relacionados: como [Sta_00] y [Sta_01], aunque únicamente describen los principios básicos de funcionamiento y operación de las CDNs. Otros artículos posteriores, como [Pen_03] y [Dil_02], no presentan un análisis en profundidad o describen un modelo específico de distribución de contenido.

Por otra parte las contribuciones relacionadas más recientes están orientadas a tendencias actuales de los mercados, como son la incorporación de contenido multimedia [Para_08] y redes móviles [Bah_09] [Lun_10], su uso en sistemas IPTV [Men_09], e incluso la integración con sistemas P2P [Gar_09] [Tak_10]. Muchos otros artículos se centran en aspectos concretos, como la redirección de usuarios, la ubicación de servidores, la gestión de contenido, etc., que constituyen la base de documentación de esta tesis. La intención de este capítulo consiste en resumir toda esta información en un documento único y con un hilo lógico coherente e integrador que describa tanto los aspectos básicos como los avanzados en la distribución de contenido, haciendo especial hincapié en los retos tecnológicos y líneas de investigación en este campo.

2.1.1. El crecimiento de Internet

El propósito de esta sección no es estudiar Internet de una manera detallada, sino justificar que el crecimiento del mismo requiere nuevos mecanismos de distribución de contenidos más eficientes, como son la CDNs.

La forma de estimar o medir el uso de Internet es variable en los estudios realizados. Normalmente se toma como referencia los datos de la Unión Internacional de Telecomunicaciones [WWW_ITU] o de empresas consultoras dedicadas a hacer

estadísticas en este sentido [WWW_Nie]. Esta información se recoge de forma resumida en la Tabla 1. En un entorno local y más controlado, como puede ser España, simplemente se recoge información a modo de entrevista personal y se extrapolan datos en base a la muestra. Este es el método usado por la AIMC (Asociación para la Investigación de Medios de Comunicación) [WWW_AIM].

Región	Población (2012)	Usuarios de Internet (31/12/2000)	Usuarios de Internet (actualmente)	Penetración (% población)	Crecimiento 2000-2012	Usuarios (%)
África	1,073,380,925	4,514,400	**167,335,676**	15.6 %	3,606.7 %	7.0 %
Asia	3,922,066,987	114,304,000	**1,076,681,059**	27.5 %	841.9 %	44.8 %
Europa	820,918,446	105,096,093	**518,512,109**	63.2 %	393.4 %	21.5 %
Oriente Medio	223,608,203	3,284,800	**90,000,455**	40.2 %	2,639.9 %	3.7 %
América del Norte	348,280,154	108,096,800	**273,785,413**	78.6 %	153.3 %	11.4 %
Latinoamérica	593,688,638	18,068,919	**254,915,745**	42.9 %	1,310.8 %	10.6 %
Oceanía	35,903,569	7,620,480	**24,287,919**	67.6 %	218.7 %	1.0 %
TOTAL	**7,017,846,922**	**360,985,492**	**2,405,518,376**	**34.3 %**	**566.4 %**	**100.0 %**

Tabla 1. Uso de Internet y estadística poblacional. (Fuente: Internet World Stats)

La información mostrada en la Tabla 1 permite hacer una comparación del uso de Internet por regiones. Puede apreciarse como EEUU, origen de Internet, sigue ostentando la hegemonía en su penetración, seguido de cerca por Oceanía y Europa. Su crecimiento relativo (menor que en el resto de regiones) respecto al año 2000 está justificado debido a su ya notoria penetración en la población en esa fecha.

Otro criterio para determinar el crecimiento de Internet consiste en estimar el número de hosts disponibles en base a consultas exhaustivas del servicio de nombres (DNS), para estadísticas globales. La Figura 1 muestra esta información de manera gráfica. Puede apreciarse un notable incremento desde principios del año 2000 hasta la actualidad.

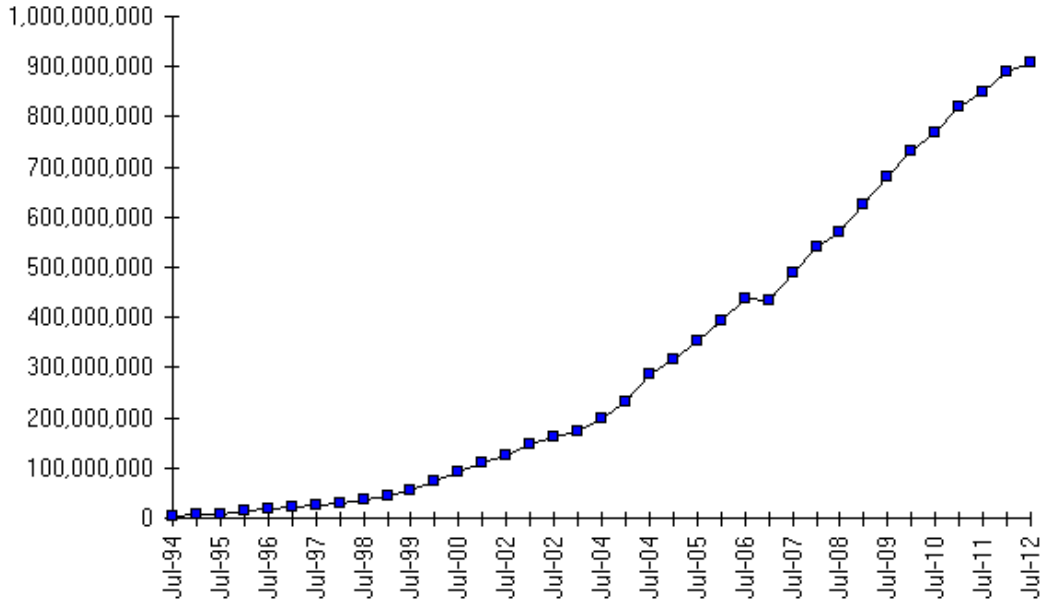


Figura 1. Evolución del número de hosts (1981-2012). (Fuente: ISC)

El crecimiento de Internet también puede ser cuantificado en términos de sitios web (*websites*). Pese a existir muchos servicios distintos, como transferencia de ficheros, e-mail, acceso de bases de datos, etc., el acceso web es posiblemente el más representativo; muchas veces los servicios anteriores se empotran dentro de una interfaz web. Es por ello por lo que, tradicionalmente, se ha venido analizando la evolución de la WWW en paralelo con Internet. Por un lado, se trata de cuantificar el número de sitios web públicos, así como el número de páginas web, tamaño medio de los objetos, etc., mediante el empleo de robots software (*crawlers*) capaces de indexar páginas web y sus respectivos enlaces incluidos.

En este contexto cabe distinguir entre la web superficial (*surface web*) – accesible por crawlers tradicionales – y la web profunda (*deep web*) – no accesible por *crawlers* tradicionales -. Esta última (web profunda), relacionada (entre otras) con la generación dinámica de páginas web, es difícilmente estimable, aunque no cabe duda de su evidente incremento como mecanismo de personalización de contenido.

Respecto al número de sitios web públicos, la Figura 2 muestra la evolución en los últimos once años realizada por Netcraft [WWW_Net]. Puede apreciarse el incremento (cercano al exponencial), así como un ligero estancamiento en los años 2001-2002.

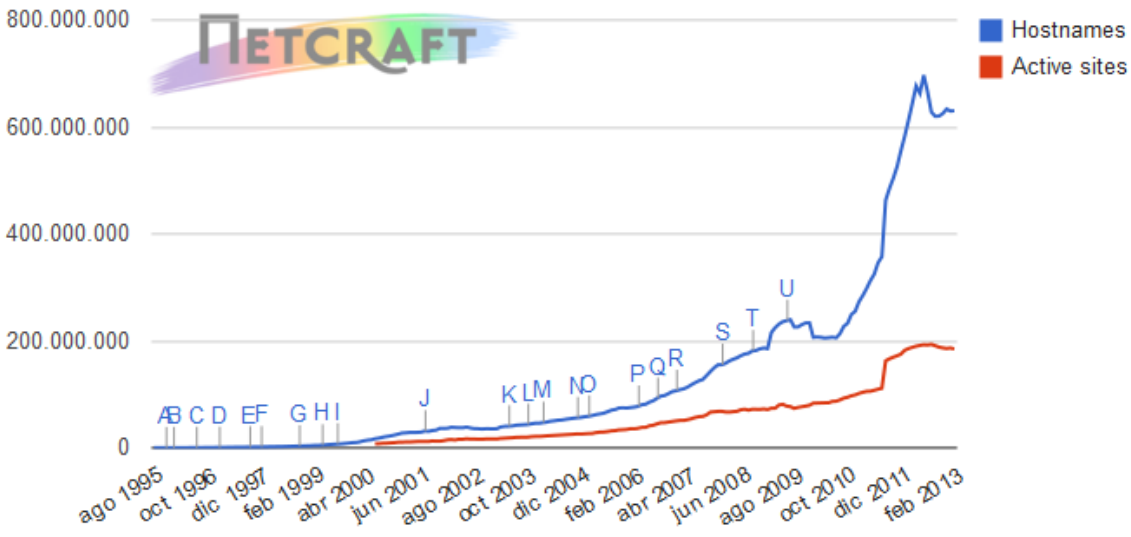


Figura 2. Evolución del número de sitios web (1995-2012). (Fuente: Netcraft)

Existen muchos otros estudios y artículos relacionados con el crecimiento de Internet analizando varios factores, como son: conectividad, número de hosts, número de sitios web, idioma empleado, número de usuarios, etc. [WWW_Zoo].

2.1.2. El tráfico en Internet. Historia y análisis

Además de describir de forma cualitativa y cuantitativa Internet, es necesario también estudiar el tipo y la cantidad de tráfico que se genera sobre la red. Para ello, y de una forma general, son posibles tres formas de acometer la medición:

- ***en el usuario (user-centric)***, mediante un programa instalado en el host cliente, que guarde estadísticas de uso y tráfico. Evidentemente, esto no es viable para todos los usuarios, y habría que recurrir a una muestra significativa.
- ***en los portales (site-centric)***, mediante un proceso que monitorice la interacción de los visitantes. En este caso, la muestra significativa comprendería posiblemente los portales más visitados.
- ***en los proveedores de Internet (network-centric)***, mediante un proceso que analice los *logs* de acceso a los diferentes portales. Este método parece el más eficiente por la capacidad de contar el número de usuarios y su uso de la red bajo un marco homogéneo e imparcial.

K. C. Coffman [Cof_98] indicó en 1998 que la tasa de crecimiento del tráfico en la Internet pública era del 100% cada año, mucho mayor que el tráfico en otro tipo de redes, y estimó que en el año 2002 este tráfico sobrepasaría al tráfico de voz. Más tarde [Cof_01], se introdujo una cierta ley de Moore para el tráfico de datos relacionando la duplicación del tráfico con la tecnología y cómo ésta es absorbida por la sociedad, así como del comportamiento del usuario. Una de las conclusiones más significativas fue la mención a los métodos de *caching* y la distribución de contenido como factores críticos del crecimiento, lo cual está directamente relacionado con las CDNs.

Internet ha soportado a lo largo de su historia distintas aplicaciones y protocolos, cada uno de ellos caracterizado con un perfil de tráfico distinto. En sus inicios (1985), el protocolo más común para el intercambio de información fue el FTP (*File Transfer Protocol*) [RFC 959]. En los sucesivos años, este protocolo evolucionó y se convirtió en el mecanismo principal para obtener contenido, principalmente documentos y software, de tal forma que, a principios de los años 90, el tráfico FTP representaba aproximadamente la mitad del tráfico de Internet [Mer_01]. Sin embargo, este mecanismo para acceder a contenido adolecía de dos serios problemas. En primer lugar, la falta de un directorio centralizado de información dificultaba enormemente la localización de contenido, que se encontraba totalmente distribuido. En segundo lugar, la ausencia de una interfaz amigable e interactiva limitaba la 'navegación' a usuarios no expertos. Numerosas aplicaciones fueron aliviando estos problemas, como *archie* [Emt_92], el proyecto *WAIS* (*Wide Area Information Server*) [Kah_91] y el sistema *Gopher* [RFC 1436]. No obstante, todas estas aplicaciones quedaron obsoletas rápidamente por un nuevo sistema, el *World Wide Web* (WWW), desarrollado en el CERN (Centro Europeo para la investigación Nuclear) por Tim Berners Lee [Ber_92]. Este nuevo sistema empleaba un protocolo diferente, el HTTP (*Hypertext Transport Protocol*) [RFC 1945, RFC 2616], así como una forma de representar la información

basada en HTML (*Hypertext Markup Language*), que no era más que una simplificación del SGML (*Standard Generalized Markup Language*) [ISO 8879-86]. El éxito y la repercusión del HTML fue tal que su estandarización pasó en 1994 del IETF (*Internet Engineering Task Force*) a un nuevo consorcio, el W3C (*World Wide Web Consortium*), encargado de estandarizar el HTML así como un gran número de tecnologías y protocolos derivados o relacionados con el mundo 'web': XML, servicios web, web semántica, etc.

El servicio *Web* ha evolucionado proporcionando una infraestructura y plataforma de desarrollo para un gran número de aplicaciones distribuidas, que se pueden agrupar en cuatro tipos fundamentales:

- ***Intercambio de contenido estático***: se trata de aplicaciones donde el contenido no varía con el tiempo, o lo hace poco frecuentemente (por ejemplo páginas web personales o artículos académicos). No es más que la extensión natural de las aplicaciones anteriores (*archie*, *WAIS* y *Gopher*) dotadas de interactividad web.

- ***Intercambio de contenido dinámico***: consiste en ofrecer contenido personalizado, es decir, dicho contenido se crea en el momento de la solicitud dependiendo de varios criterios (dirección IP del cliente, idioma del usuario, perfil del usuario, etc.). Casos de uso de contenido dinámico incluyen portales que proporcionan noticias, cotizaciones en bolsa, previsión del tiempo, etc., basado en los intereses y la localización del usuario. Ejemplos típicos pueden ser *My Yahoo!* o *My eBay*. Cabe destacar en la definición de contenido dinámico que un servidor web puede entregar un contenido distinto a usuarios individuales solicitando el mismo recurso web. Esto lo hace diferente de un contenido estático que varíe frecuentemente.

- ***Intercambio de contenido streaming***: considerado, en ocasiones, como la reproducción de un flujo continuo de audio o vídeo. Una descripción más acertada distingue entre un streaming real y la simple reproducción de ficheros de audio o video previamente descargados. De hecho, el concepto de streaming puede aplicarse a cualquier contenido, como texto e imágenes, aunque típicamente se asocia a sonido y vídeo. Ejemplos de este tipo de aplicaciones incluyen video bajo demanda y radio sobre Internet. En esta tesis se analizará este tipo de contenido en un entono de CDN, y se tendrán en cuenta las dos categorías principales en el ámbito de streaming:

    - *Streaming bajo demanda (on-demand): consiste en enviar contenido prealmacenado en el servidor a múltiples usuarios en tiempos diferentes.*

    - *Streaming en directo (live):* difunde contenido en vivo a varios usuarios en el mismo instante de tiempo (conforme se va generando el contenido).

- ***Colaboración interactiva**:* consiste en ofrecer un entorno de colaboración con un mayor grado de interactividad a través del web en tiempo real. Típicamente los entornos previamente existentes estaban limitados a actividades asíncronas, sin capacidad de interacción en tiempo real. Aplicaciones web como videoconferencia, juegos en red y mensajería instantánea permiten a los usuarios reaccionar ante acciones de otros usuarios en tiempo real.

Estos cuatro tipos de aplicaciones muestran que el *Web* ha madurado hasta un punto donde éste dispone de un valor mayor que el simple intercambio de contenido estático. Tanto las empresas como los usuarios finales perciben el *Web* como una infraestructura fiable y de calidad capaz de entregar contenido multimedia enriquecido. Sin embargo, recientes desarrollos en los campos de contenido multimedia, aplicaciones interactivas y contenido dinámico ponen de manifiesto una falta de rendimiento del modelo tradicional web y ha conducido al sector académico e industrial a incorporar mejoras en las tecnologías de red sobre Internet, denominadas *content networks*. Este concepto da origen a las CDN y se abordará con detalle en la sección 2.1.3. Pero antes se describirán otros tipos de tráficos también presentes y bien significativos en Internet, y se compararán con el tráfico web.

A principios del año 2000, debido a su gran acogida y popularidad, el tráfico web había sobrepasado al resto de aplicaciones sobre Internet. No obstante, a partir de 2002, las redes P2P adquirieron un éxito considerable, pues permitía a los usuarios intercambiar ficheros y programas mediante un modelo distinto al tradicional cliente servidor. Esto condujo a un elevado número de usuarios a emplear aplicaciones y redes P2P (*Napster*, *FastTrack*, *Gnutella*, *BitTorrent*, etc.) para intercambiar una gran cantidad de contenido, tanto en número como en tamaño de almacenamiento. El tráfico P2P generado entonces desbancó sobradamente a los otros tipos de tráfico, como reflejó un estudio en la Universidad de Washington [Sar_02], que puede apreciarse en la Figura 3.

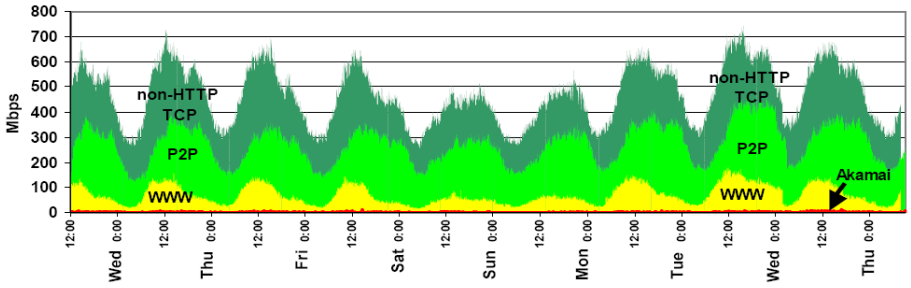


Figura 3. Estudio del tráfico TCP en la Universidad de Washington [Sar_02].

Curiosamente en este estudio aparece mencionado el tráfico servido por una CDN, *Akamai* [WWW_Aka], la más importante de todas, sin duda, en esa época (y en la actualidad). Si bien el estudio indicaba que dicho tráfico no era comparable al resto de tráficos en ese momento, sí ponía de manifiesto el paulatino crecimiento de este tipo de redes de tal forma que cada vez más usuarios accedían a recursos web servidos por una

CDN. En los últimos años algunas CDNs de cobertura mundial han ido incorporando contenido multimedia, con lo que los perfiles de tráfico han variado.

Efectivamente, *Youtube* (www.youtube.com), que emplea una infraestructura de CDN servida por Akamai, es responsable desde 2007 de más del 10% del tráfico total generado en Internet. Según un estudio de la empresa *Ellacoya*, ahora integrada en *Arbor Networks* [WWW_ARB], realizado en 2007, el 46% del tráfico total de Internet es generado por tráfico HTTP, mientras que sólo el 37% procede de P2P, como se observa en la Figura 4. Este dato contrastaba con el 65% de tráfico P2P tan sólo dos años antes. Respecto al tráfico HTTP, un análisis indica que el streaming de audio y video representa el 41%, y la mitad de éste está originado por *Youtube*. Tanto el texto como las imágenes web siguen usando ligeramente un mayor ancho de banda (46%), pero ya se apuntaba que esta tendencia no duraría mucho.

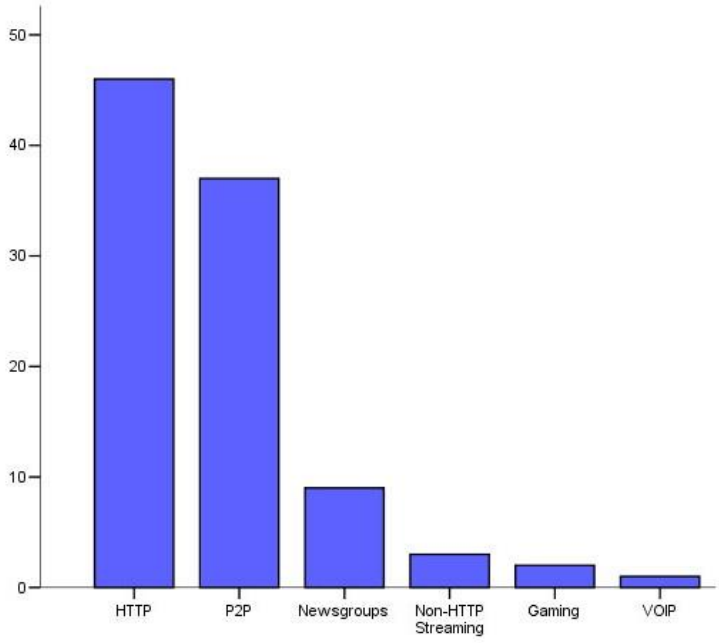


Figura 4. Porcentaje total de tráfico de Internet (2007) (Fuente: Ellacoya)

En la actualidad es complicado precisar si la mayor parte del tráfico en Internet es debido a P2P o a HTTP, ya que ambos incluyen contenido multimedia de gran tamaño. Es más, además de *Youtube*, aparecieron sistemas como *RapidShare* [WWW_Rap] y *MegaUpload* [WWW_Meg] (éste último cerrado en Enero de 2012) que permitían la descarga de contenido multimedia (fundamentalmente películas) mediante HTTP, reduciendo la proporción de tráfico generado por P2P. Sin embargo, nuevas aplicaciones y clientes P2P siguen atrayendo usuarios y consumiendo un elevado tráfico, a veces incluso superior a HTTP. Este análisis también se puede realizar por zonas, como el estudio realizado por la empresa alemana *iPoque* [WWW_Ipo] que aborda diferentes áreas del planeta y tipifica los tipos de protocolos, como se observa en la Figura 5.

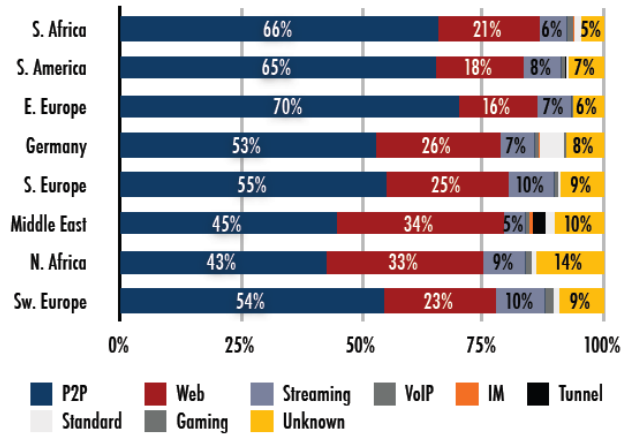


Figura 5. Distribución de clases de protocolos 2008-2009. (Fuente: iPoque)

Si bien el estudio anterior muestra la 'supremacía' del tráfico P2P, también indica, por otro lado, la paulatina reducción de este tipo de tráfico, el incremento del tráfico HTTP y, sobre todo, la creciente demanda por parte de los usuarios de consumir contenido streaming, como se observa en la Figura 6.

Este último dato describe la necesidad actual de las grandes redes de distribuir contenido multimedia en formato streaming de una manera eficiente. El concepto *content networking*, que será descrito en el siguiente capítulo, engloba todas estas tecnologías necesarias a incorporar en la red, y una CDN es un caso particular de implementación. Esta tesis se enmarca dentro del estudio de una CDN para distribuir tanto contenido web como contenido multimedia en la modalidad de streaming, por lo que resulta un tema de actualidad.

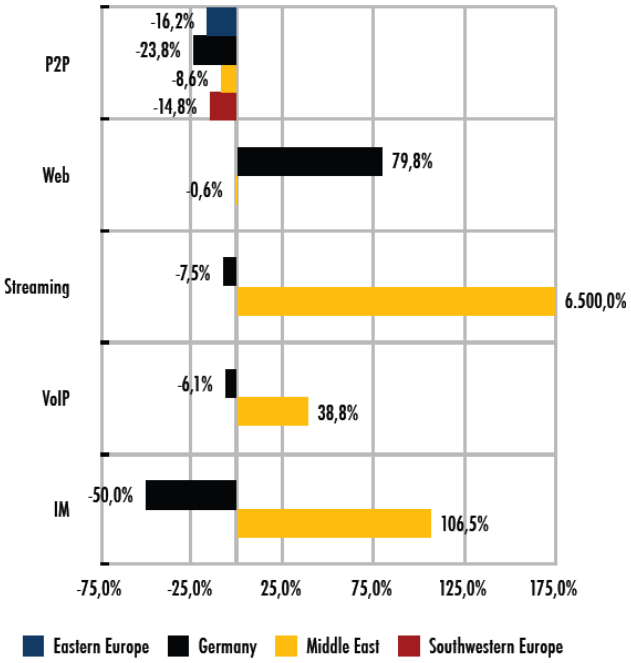


Figura 6. Cambios proporcionales respecto a 2007. (Fuente: iPoque).

2.1.3. Content Networking y su evolución

En los últimos 20 años, el servicio web ha evolucionado desde una simple aplicación de Internet para científicos e investigadores hasta convertirse en un fenómeno comercial en la actualidad del que dependen muchas empresas. Sin embargo, el crecimiento de tráfico de red (descrito anteriormente), la rápida aceptación de la banda ancha, y el incremento en la complejidad de los sistemas y contenidos actuales representan un gran desafío a la hora de gestionar y entregar contenido a los usuarios. Cualquier reducción de la calidad del servicio, o un elevado retardo en el acceso a los contenidos puede conducir a una frustración por parte del usuario y provocar un abandono de un sitio web. Esto es lo que trata de evitarse en los nuevos mercados web, donde se incorporan aplicaciones de red nuevas, herramientas software así como nuevos tipos de servicios de red. Cuando todas estas tecnologías de usan conjuntamente se crea un nuevo tipo de red denominada *content network* [RFC 3466] o red de contenido (CN). Este apartado examina los problemas que condujeron a la emergencia de estas redes y presenta algunas soluciones en términos de tecnologías y servicios que conforman una red de contenido. Es más, estas redes de contenido se pueden basar en servicios de CDNs para distribuir y gestionar el contenido dentro de la red [Plag_06].

El crecimiento de Internet descrito anteriormente conduce a un escenario donde la capacidad de red puede verse desbordada por una demanda excesiva de tráfico en el dominio espacial (algunas subredes dentro de Internet) y temporal (en algunos momentos puntuales). Esto se conoce como congestión espacial y temporal, respectivamente, y algunos protocolos, como TCP, ya incorporan en su diseño mecanismos de funcionamiento para prevenir la congestión. Por otro lado, la incapacidad de los servidores para satisfacer todas las peticiones solicitadas también puede degradar la calidad del servicio obtenido. La Figura 7 muestra un escenario genérico de conexión sobre redes IP.

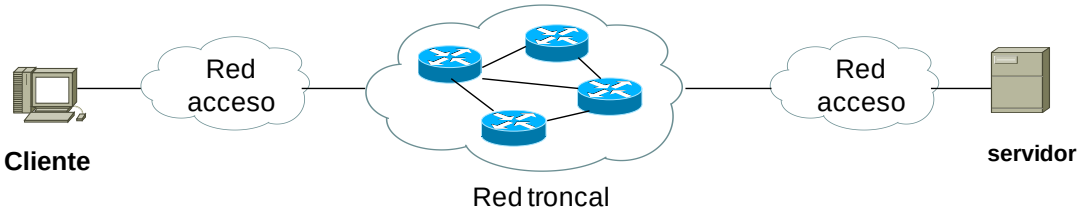


Figura 7. Escenario general de conexión entre un cliente y un servidor.

Puede apreciarse cómo la comunicación entre el cliente y servidor atraviesa varias fases o componentes de la red. El retardo en cada una de estas repercute en el tiempo de respuesta global experimentado por el usuario, que representa una buena métrica del rendimiento ofrecido para un servicio:

- ***red de acceso del usuario***: Inicialmente (década de los 80 y principios de los 90) el retardo de esta red, con módems de hasta 56 Kbps, representaba un serio problema. El uso de la infraestructura de red telefónica para soportar datos representaba una seria limitación debido al teorema de Shannon-Hartley

[Sha_49]. Desde hace ya varios años, con la introducción de las tecnologías de banda ancha (ADSL, cable modem e incluso FTTH), el retardo de la red de acceso parece estar resuelto. Sin embargo, aunque pueda parecer una solución, genera un nuevo problema, y es el hecho de que el resto de redes (red troncal) también debe aumentar su capacidad en la misma proporción para soportar el más que posible incremento de tráfico generado por esta red (debido a las aplicaciones multimedia).

- ***red troncal***: esta red está formada por un conjunto de redes interconectadas entre sí en puntos neutros o NAPs *(Network Access Point*) y acuerdos de *peering*. Normalmente el ancho de banda de las redes troncales está sobredimensionado y no constituye un serio problema, por lo que el cuello de botella lo constituye la interconexión de estas. Téngase en cuenta que el camino atravesado por una comunicación atraviesa típicamente varias subredes dentro de Internet.
- ***red de acceso del servidor***: también llamado problema de la primera milla. Los servidores siempre han gozado de un elevado ancho de banda en su conexión a Internet. Sin embargo, un solo servidor centralizado, para ofrecer un servicio determinado, no podrá absorber todas las solicitudes de cualquier usuario de Internet, ya que no es viable (ni tecnológica- ni económicamente) aumentar el ancho de banda de forma proporcional al número de usuarios: simplemente imagínese 1000 usuarios con tecnología ADSL de 1Mbps tratando de acceder a un servidor de forma masiva y simultánea.
- ***capacidad del servidor*:** además del retardo experimentado en la red de comunicación, el servidor que atiende una petición requiere un tiempo de proceso para poder generar la respuesta, lo que representa un retardo adicional. Este servidor dispone de varios modos de funcionamiento: (i) en una situación de poca carga, el tiempo de respuesta ante una petición es aceptable, (ii) en una situación de media carga, el servidor es capaz de dar servicio, aunque el tiempo de respuesta puede ser significativamente variable, y (iii) en una situación límite, el servidor no es capaz de satisfacer una nueva petición, y ésta será rechazada.

Todos estos (posibles) retardos no suponen un problema reciente, sino que ya existían en las redes de datos de los años 80, aunque evidentemente en una menor proporción debido a un menor número de usuarios y de tráfico generado. Las principales posibilidades que se presentan para abordarlos son varias:

- ***planificación de la capacidad (capacity planning):*** consiste en predecir la carga esperada de la red y asegurar que ningún elemento estará sobrecargado, aumentando el ancho de banda de los enlaces (introducción de fibra óptica) así como de los dispositivos de interconexión (conmutadores, routers) para conmutar o encaminar paquetes. Este método es válido solamente si la predicción es buena, y si la variabilidad de este tráfico es pequeña, a fin de poder soportar los costes asociados. Evidentemente, Internet no cumple ninguna de los requisitos anteriores: no es posible predecir el tráfico con exactitud y su variabilidad es excesiva, por lo que esta solución quedaría relegada a (algunas)

redes empresariales o particulares donde es posible hacer un estudio en profundidad de carga predecible.

- ***Mecanismos de calidad de servicio (QoS, Quality of Service):*** se parte de la idea de que no se va a disponer de recursos suficientes en la red, por lo que no se podrá ofrecer un rendimiento aceptable a todos los flujos de tráfico. Así pues, la calidad de servicio sólo garantiza un buen rendimiento a un determinado conjunto de flujos. Básicamente existen dos formas de ofrecer QoS:
    - *Reserva de recursos dentro de la red para garantizar la capacidad necesaria*: se trata del método empleado en redes ATM y en redes IP de Servicios Integrados (*IP IntServ*) [Bra_94] [RFC 1633]. Las aplicaciones informan a la red de los recursos que van a necesitar mediante un protocolo de señalización previa a la transferencia de datos. La Figura 8 muestra cómo se trata dentro de un *router* con capacidad *IntServ* diferentes tipos de flujos. Aquellos flujos con requisitos de tiempo real (voz) son tratados internamente de tal forma que se garantiza a la salida el mismo perfil de tráfico sin distorsión. El resto de flujos sin requisitos de tiempo real (datos) se adaptan de tal forma que su perfil de tráfico, en media, tampoco varía. Si no es posible adecuar un nuevo flujo, ya sea de voz o de datos, se rechaza la reserva.

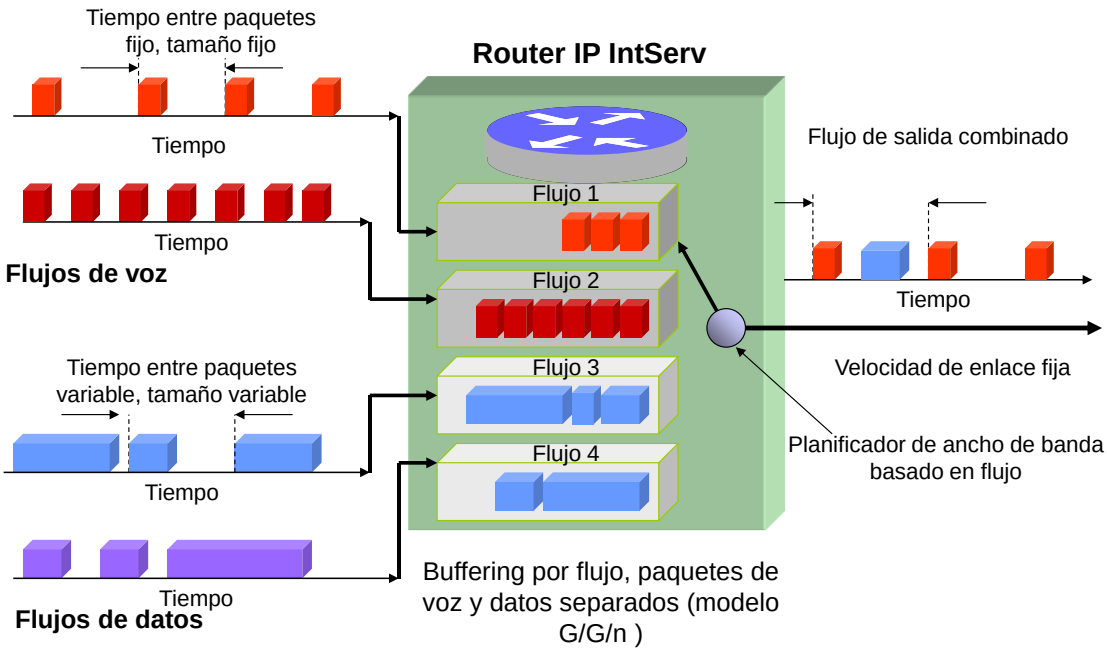


Figura 8. Esquema general en arquitectura IP IntServ [Fuente: CCNY].

Es importante destacar que la reserva de recursos se establece no sólo antes de la transferencia sino también en todos y cada uno de los nodos de la red, como se observa en la Figura 9. Los problemas presentados por este mecanismo son varios:

- Por un lado, el número de flujos de usuarios simultáneos puede ser muy elevado (millones), con lo que la capacidad de gestión y almacenamiento de los routers se puede ver seriamente afectada.
- Por otro lado, esta capacidad de gestión implicaría un hardware muy complejo y costoso, no solo para almacenar un elevado número de flujos, sino también para planificar adecuadamente todos estos.
- La mayoría de los protocolos de señalización (por ej. RSVP [RFC 2205]) son complejos e ineficientes, y sería necesario incorporarlos tanto en los routers como en los clientes de red. Para el caso de dispositivos ligeros, como móviles o UMDs (*Ultra Mobile Devices*), esto puede representar un serio problema.

En términos generales, se trata de una solución que no resulta escalable en Internet. Por este motivo, y con la finalidad de buscar una solución menos compleja, se sugirió el empleo de la diferenciación de servicios.

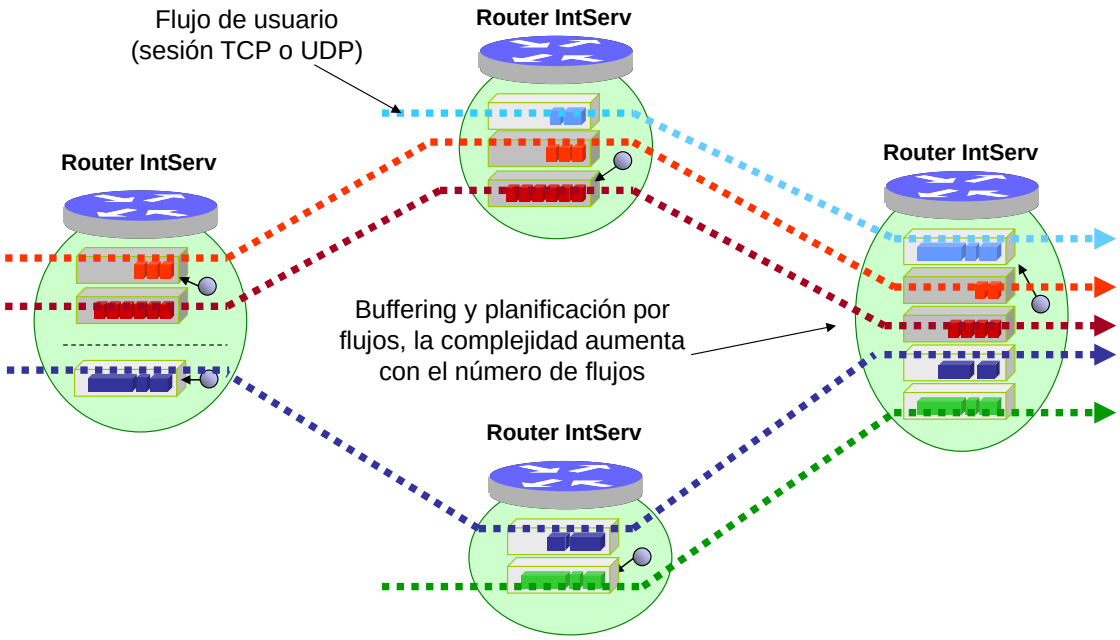


Figura 9. Reserva de recursos en la red mediante IntServ [Fuente: CCNY].

- *Distribución de flujos en varias clases con preferencias*: se trata de un método más simple al no requerir protocolo de señalización. En redes IP, la especificación de referencia es IP de Servicios Diferenciados (*IP DiffServ*) [Bla_98] [RFC 2474]. Los paquetes se clasifican en dos o más clases de servicio, con la información en el propio paquete, como se observa en la Figura 10. Con este mecanismo, se agrupan los flujos en clases de servicios.

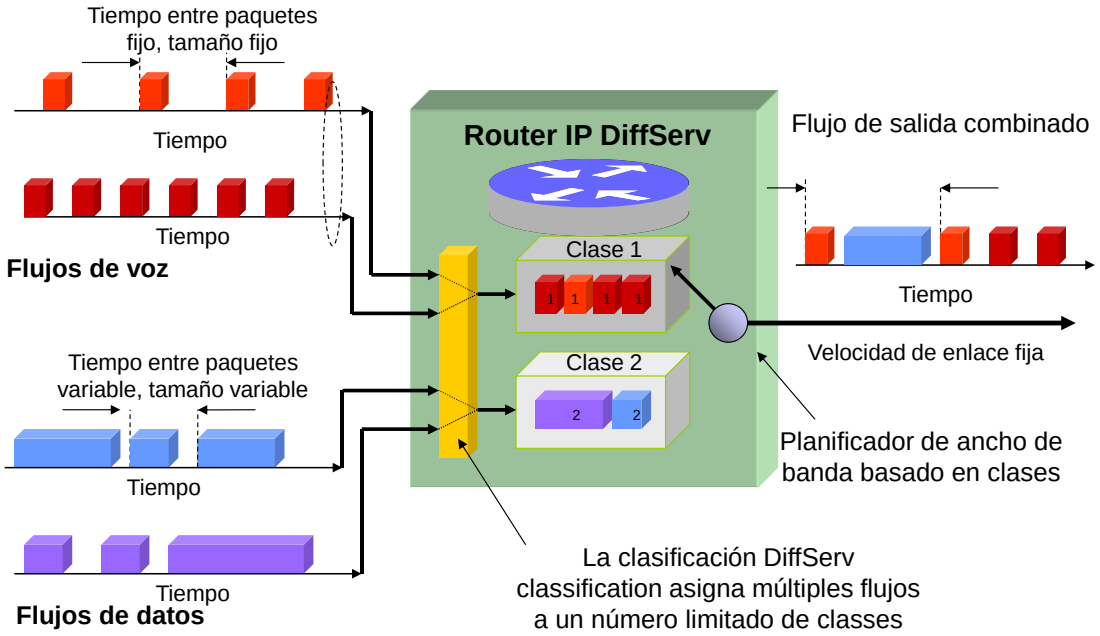


Figura 10. Esquema general en arquitectura IP DiffServ [Fuente: CCNY].

El número de clases de servicio es limitado. *DiffServ* emplea el campo de 6 bits *DSCP* (*Differentiated Services Code Point*) en la cabecera IP para clasificar los paquetes. *DSCP* reemplaza el (desfasado) campo *IP Precedence*, que consistían en 3 bits dentro del byte *ToS* (*Type of Service*) de la cabecera IP empleado originariamente para clasificar y priorizar tipos de tráfico. Aunque en teoría se dispone de un mayor número de clases, en la actualidad sólo se usan un número reducido de éstas (entre 2 y 8). De esta forma, la complejidad de un número potencialmente ilimitado de flujos (como *IP IntServ*) se transforma en un mecanismo mucho más sencillo al tener que gestionar tan sólo unas pocas clases en toda la ruta de comunicación, como se ilustra en la Figura 11.

Este mecanismo adolece del serio inconveniente de ser incapaz de garantizar una calidad de servicio a los flujos individuales. Si bien (en media) se puede ofrecer una cierta calidad de servicio a cada una de las clases, nada puede aventurarse acerca de la calidad de cada flujo de datos de forma independiente. En la práctica, *IP DiffServ* funciona razonablemente bien (es decir, sin apenas percepción por parte del usuario) cuando el tráfico es reducido. En cambio, cuando el tráfico es elevado, el tráfico con requerimientos de tiempo real se ve seriamente afectado (es decir, con una percepción evidente por parte del usuario).

Cuando hay congestión, los paquetes de menor prioridad (las clases típicamente se clasifican o indexan acorde a una prioridad) sufren una mayor degradación.

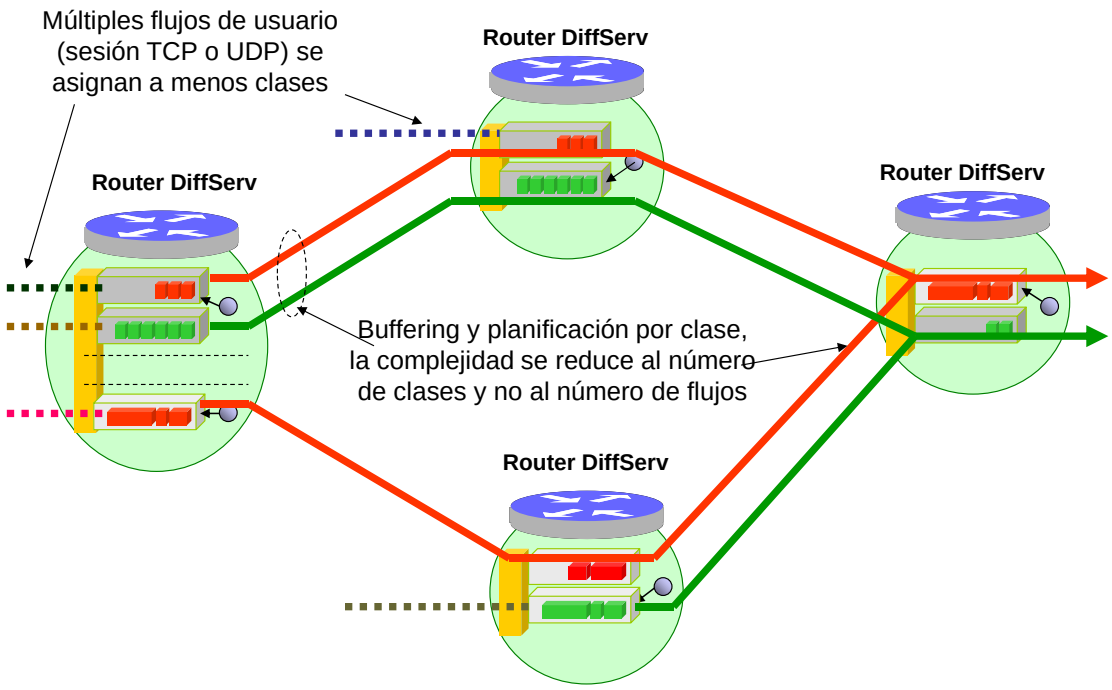


Figura 11. Gestión de recursos en la red mediante DiffServ [Fuente: CCNY].

Desde una perspectiva general, estos dos mecanismos de QoS *(IP IntServ* e *IP DiffServ*) implican una actualización de la infraestructura de red, pues todos los conmutadores y routers deben soportar alguno de ellos, o ambos. Téngase en cuenta, por otro lado, que se trata de mecanismos no excluyentes, sino complementarios. En Internet, sin una gestión globalizada de la misma, resulta imposible introducir cualquiera de estos dos mecanismos en una comunicación extremo a extremo que atraviese más de una subred (o dominio administrativo). Es por ello por lo que esta tecnología queda relegada a redes corporativas u operadores dentro de su propia red. Sin embargo, los ISPs suelen emplear otro tipo de tecnología, MPLS, que se describe a continuación.

- ***MultiProtocol Label Switching (MPLS) [RFC 3031]:*** Se trata de un mecanismo estandarizado basado en la propuesta de *Tag Switching* de Cisco [RFC 2105] que identifica un flujo de datos con una etiqueta (*tag* o *label*), de tal forma que la conmutación se acelera al evitar el encaminamiento basado en la cabecera IP. Esta tecnología permite aplicaciones avanzadas como la ingeniería de tráfico y la capacidad de ofrecer calidad de servicio y la gestión relativamente sencilla de redes privadas virtuales (VPNs, *Virtual Private Networks*). Gestionando adecuadamente estas etiquetas se crean las denominadas rutas de conmutación LSPs (*Label Switched Paths*) que generan una conexión lógica o circuito virtual. De esta forma, los paquetes IP se encapsulan en paquetes MPLS, que son tratados adecuadamente por cada uno de los routers intermedios que conforman la ruta de conmutación LSP. A cada uno de estos routers se les denomina LSN (*Label Switched Node*) y su funcionamiento básico viene ilustrado en la Figura 12.

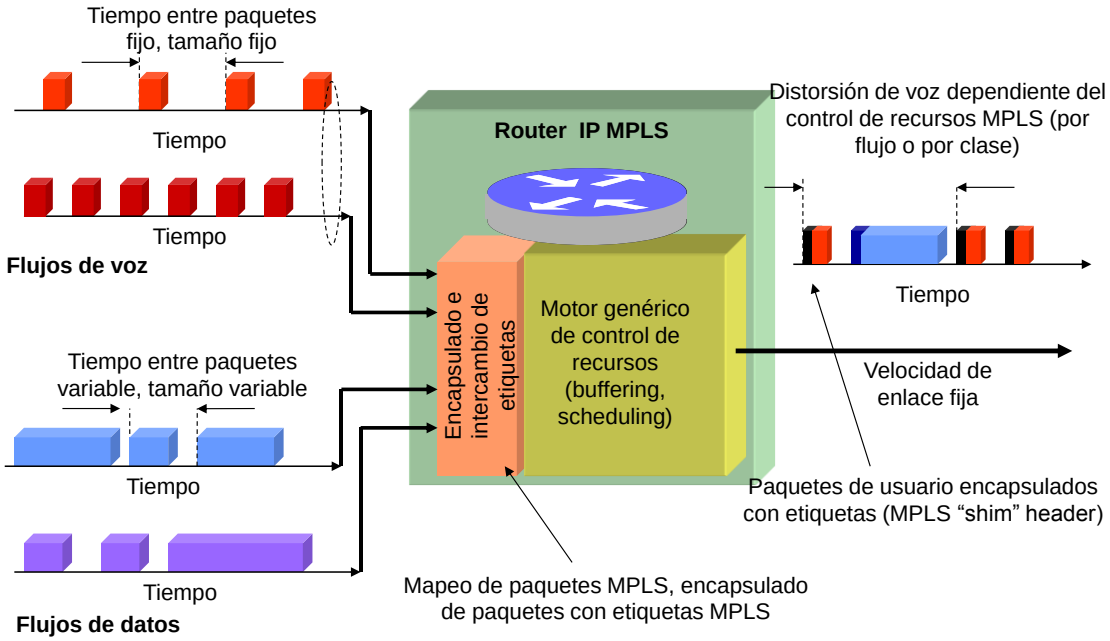


Figura 12. Esquema de un nodo MPLS (LSN) [Fuente: CCNY].

Como se puede apreciar, el funcionamiento es similar a un router IP, pero el encaminamiento viene determinado por las etiquetas MPLS. Adicionalmente, se implementa un control de recursos para tratar de forma distinta los flujos con requisitos de tiempo real (por ej. flujos de voz) de aquellos que no los tienen (por ej. flujos de datos).

Si bien las rutas virtuales tienen sentido en una comunicación extremo a extremo, o bien un segmento de esta, las etiquetas MPLS tienen sentido para cada router, de tal forma que un paquete de un flujo de datos de usuario puede entrar en un router con una etiqueta concreta y salir de éste con otra etiqueta, según una tabla de etiquetas que posee cada router, similar a las tablas de encaminamiento usadas para encaminar paquetes IP. Esto puede apreciarse en la Figura 13.

Nótese que el marco tecnológico ofrecido por MPLS es capaz de permitir el desarrollo de implementaciones operativas de IP *IntServ* e IP *DiffServ*. Para abordar el primer caso (IP *IntServ*), basta con asociar una etiqueta o, mejor dicho, un LSP, a cada conexión de usuario. Para afrontar el segundo caso (IP *DiffServ*) cada LSP agrupa varias conexiones de usuario. Actualmente MPLS está ampliamente soportado por la mayoría de fabricantes de infraestructura de red. El concepto de ingeniería de tráfico, ya existente en redes IP y redes ATM, ha quedado en la actualidad íntimamente ligado a MPLS, de tal forma que es común observar en los equipos de red capacidades de MPLS TE (*Traffic Engineering*). Es ocasiones es común la existencia de soporte RSVP-TE, que típicamente emplea a bajo nivel mecanismos de MPLS.

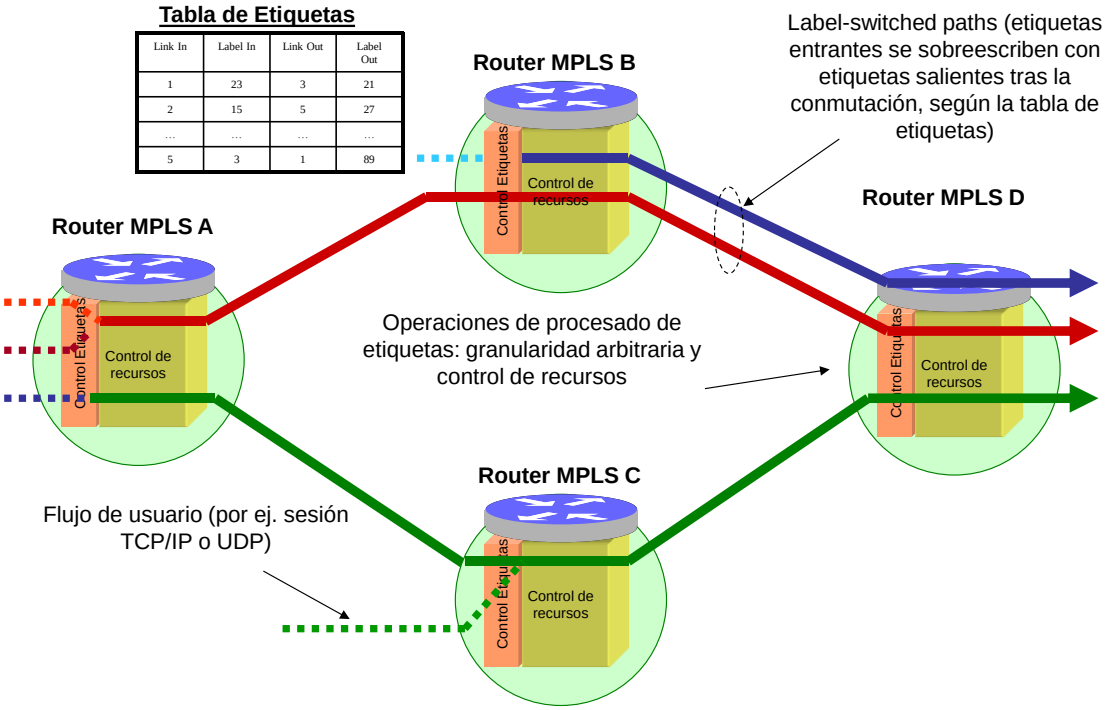


Figura 13. Red MPLS [Fuente: CCNY].

- ***Soluciones basadas en Content Networking****:* Las soluciones anteriores implican el acceso a los recursos de red (o parte de ellos) que, en muchas ocasiones, no se dispone. Adicionalmente, se trata de soluciones de red, por lo que no se aborda el problema fundamental de los servidores sobrecargados; si bien los proveedores de servicios de red (ISPs) están muy comprometidos con escalar adecuadamente para soportar los picos de carga en el tráfico de datos, sólo pueden actuar sobre su propia red, no pueden actuar sobre redes adyacentes ni tampoco sobre los servidores finales, pues desconocen su estado real.
  El aspecto de la escalabilidad (y caching) se abordará con mayor detalle en las secciones 2.2 y 2.3, aunque conviene hacer una introducción general a todas las tecnologías que permiten satisfacer las expectativas presentes y futuras de los usuarios de Internet, englobadas en el concepto de Content Networking.
  Estas aproximaciones y desarrollos deben ser contemplados como una evolución del modelo tradicional Web donde se busca una red de contenido más dinámica. Los pasos de esta evolución se pueden concretar en una serie de acciones fundamentales:
  - *Distribución de carga en un sitio centralizado*: consiste en agrupar lógicamente varios servidores físicos creando una granja de servidores (*server farm*). Un dispositivo frontal balancea la carga entre los servidores. Este dispositivo es especialmente importante y ha dado origen a conceptos relacionados como *Switch Layer 4-7*, *Web Switch* o *Content Switch*, que se describirán en detalle en el apartado siguiente (capítulo 2.2).

- *Distribución de contenido con servicios centralizados*: implica distribuir el contenido a ubicaciones cercanas al cliente de tal forma que se agiliza el acceso. Los dispositivos desplegados en estas cercanías realizan tareas de *replicación* y *proxy caching*. El *caching* de objetos web ha sido estudiado de manera extensiva, comenzando desde el *simple proxy caching* [Luo_97], y las mejoras con el *caching* jerárquico y cooperativo tras los proyectos Harvest [Cha_96] y Squid [WWW_Squ], respectivamente. Un (*proxy*) *caching* efectivo debe integrar métodos de reemplazo, gestión de la validez del contenido, balanceo de carga y replicación. Todos estos aspectos serán abordados con mayor profundidad en el capítulo 2.3.
- *Distribución de contenido y servicios*: en este modelo, no es suficiente distribuir contenido estático a las cercanías del cliente, sino también ciertos servicios, como puede ser el ensamblado de contenido personalizado o la adaptación de contenido para dispositivos inalámbricos. La arquitectura de *Web Services* ofrece un marco de trabajo adecuado para aplicaciones distribuidas; gracias a la interoperabilidad, es posible construir aplicaciones complejas ensamblando pequeñas unidades ofrecidas como servicios web. Por otro lado, es importante destacar que una arquitectura distribuida tiene un coste en términos de mayor complejidad y mayor inversión (económica) inicial, sin embargo escala mejor para un número mayor de usuarios globales y proporciona un mejor rendimiento y fiabilidad.

Las tecnologías descritas en los puntos anteriores han recibido diferentes denominaciones a lo largo de los años. Entre ellas, cabe destacar distribución de contenido (*content delivery*, *content distribution*), *overlays* de caches (*caching overlays*), redes proxy *(proxy networks*) y redes elásticas o con capacidad de recuperación (*RON, Resilient Overlay Network*). En la medida de lo posible, el vocabulario empleado en esta tesis sigue la terminología empleada en la RFC 3466 [Day_03].

El término *contenido* (*content*) se refiere a cualquier tipo de información que se encuentra disponible para otros usuarios en Internet. Esto incluye, entre otros, a páginas web, imágenes, documentos de texto, ficheros de audio y vídeo, así como descargas software, difusiones, mensajería instantánea y formularios.

Como puede apreciarse, el contenido no está sujeto a ningún tipo de medio, de hecho, un contenido puede estar formado a su vez por varios contenidos de distinto tipo de medio, lo que se conoce como contenido *multimedia*.

El término *red de contenido* (*content network*) consiste en una red de comunicación que despliega en su infraestructura una serie de componentes que operan en los protocolos de nivel 4-7. Estos componentes se interconectan entre ellos, de tal forma que conforman una red virtual sobre la infraestructura de red existente.

Los vínculos entre estas redes virtuales y la infraestructura de red subyacente se crean mediante los *intermediarios* (*intermediaries*). Se trata típicamente de

dispositivos de nivel de aplicación que son parte de una transacción web, sin ser ni el origen ni el destino de dicha transacción. Los intermediarios más conocidos son los *proxies* y las *caches web*, que se abordarán en los capítulos 2.2 y 2.3. Las redes de distribución de contenido, que se abordarán en el capítulo 2.5, consisten básicamente en redes virtuales que emplean adecuadamente estos intermediarios para una distribución global de contenido.

De una manera general, una red de contenido requiere una serie de componentes funcionales que colaboran para mejorar la forma de distribuir contenido. Estos componentes son:

- *Distribución de contenido:* se trata de servicios responsables de mover el contenido desde el origen a los usuarios. Estos servicios pueden ser abarcados por caches web u otros dispositivos que almacenen contenido en puntos intermedios en nombre del servidor origen. El componente de distribución también cubre el mecanismo real y los protocolos empleados para la transmisión de datos sobre la red.
- *Enrutamiento de la petición:* se trata de servicios que redirigen las peticiones de los usuarios a la mejor ubicación posible para obtener el contenido solicitado. Dichas peticiones pueden ser servidas tanto por servidores web como por caches web. La decisión de la mejor ubicación en la fase de redirección se toma típicamente en base a parámetros como proximidad de red y disponibilidad de los sistemas y la red.
- *Procesado del contenido:* se trata de servicios para crear o adaptar contenido dependiendo de las preferencias de usuario o las capacidades del dispositivo. Esto incluye la modificación o conversión del contenido e incluso de las peticiones de contenido. Algunos ejemplos lo constituyen la adaptación de contenidos para dispositivos inalámbricos o la adición de privacidad al convertir la información personal embebida en las solicitudes de usuario en anónima.
- *Autorización, autenticación y (AAA):* se trata de servicios que permiten la monitorización, *logging, accounting* y tarificación (*billing*) según el uso del contenido. Esto incluye los mecanismos para asegurar la identidad y los privilegios de todos los participantes en una transacción, así como la gestión de derechos digitales *(DRM, Digital Rights Management).*

Finalmente, cabe destacar que no se requiere que una red de contenido disponga de todos estos componentes funcionales. Por ejemplo, el procesado del contenido es un elemento optativo.

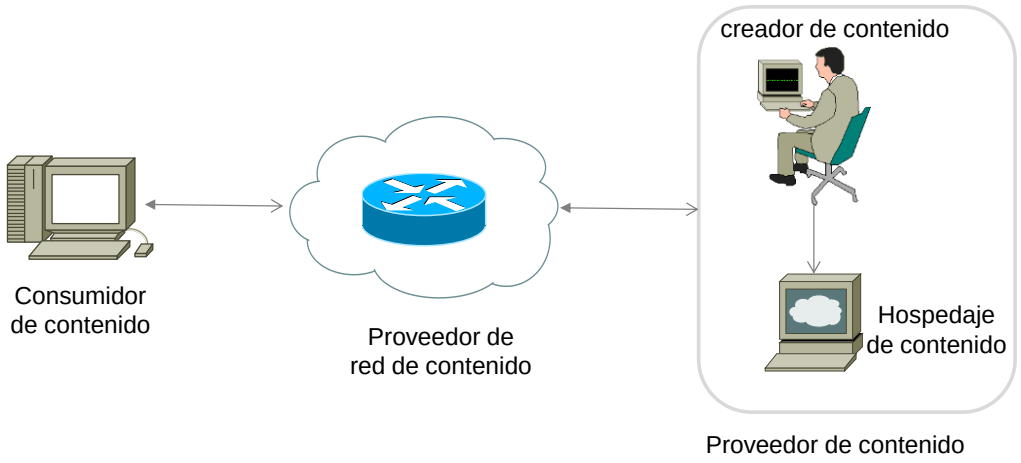


Figura 14. Cadena de valor en Content Networking

Una vez introducido el concepto global de red de contenido, cabe describir brevemente los diferentes participantes y sus intereses en la cadena de valor del *content networking*. Desde un punto de vista de alto nivel, la cadena de valor del *content networking* comienza con el proveedor de contenido y se extiende para incluir al proveedor de red de contenido y al consumidor de contenido, como se observa en la Figura 14:

- ***Proveedores de contenido (content provider)***: Si bien las compañías grandes pueden crear y mantener su propio contenido, normalmente el servidor web que alberga u hospeda el contenido pertenece a un tercero que proporciona espacio de almacenamiento y acceso a Internet. Esta disposición es denominada en ocasiones *server co-location*. Normalmente, las pequeñas empresas y los usuarios domésticos no despliegan su propio servidor web, sino que hacen uso de alguna opción de hospedaje web (*hosting*). Es por ello por lo que cabe distinguir al creador de contenido y a la entidad que alberga el contenido. En el primer caso, la entidad es denominada creador de contenido (*content creator*), mientras que la entidad que ofrece espacio con acceso web se denomina albergador de contenido (*content host*). Si no es necesario hacer esta distinción, entonces la denominación proveedor de contenido (*content provider*) se empleará de forma general en esta tesis para referirse a ambos.

  Los proveedores de contenido se enfrentan continuamente al reto de proporcionar contenido enriquecido con elevados niveles de servicio y de consistencia. Algunas de sus mayores preocupaciones técnicas son los tiempos de respuesta experimentados por los usuarios y la disponibilidad permanente de los sitios web. En el pasado, el aspecto más interesante desde el punto de vista económico para los proveedores de contenido consistía en albergar y gestionar contenido, pero la dificultad creciente de satisfacer las necesidades de los clientes para la distribución de contenido hace que la subcontratación (*outsourcing*) parcial de ciertos servicios adicionales resulte atractiva. En cualquier caso, los proveedores de contenido desean seguir teniendo un control total sobre el contenido y de las máquinas que gobiernan el contenido, los permisos de acceso y las políticas de configuración y gestión. Es más, los proveedores de contenido confían en los análisis que se pueden extraer a partir

de las estadísticas de uso acerca del contenido. Dado que una elevada tasa de acceso se traduce en mayores ingresos por anuncios, los proveedores de contenido suelen estar interesados en atraer tantos usuarios como la infraestructura disponible pueda soportar.

- ***Proveedores de red de contenido (content networking provider)***: El principal objetivo de los proveedores de red de contenido es ayudar a los proveedores de contenido a distribuir contenido a los usuarios. Sus recursos consisten típicamente en proporcionar *caching* y replicación de datos, así como mecanismos de redirección de usuarios y posibles servicios de procesado de contenido. Dado que sus ingresos económicos están mayoritariamente determinados por el volumen de datos saliente de sus redes, estos proveedores tratan de atraer tantas solicitudes de usuario como sea posible. Al mismo tiempo, tratan de reducir la carga agregada en sus recursos y en los enlaces de red que los unen, lo que conduce a servir la mayoría del contenido desde recursos (servidores) ubicados en las cercanías del cliente. En la Figura 14, por ejemplo, el proveedor de red de contenido estaría interesado en servir contenido al usuario final desde una *caché* desplegada entre usuario (consumidor) y el proveedor de contenido. Esto permite generar ingresos y aliviar carga entre la caché y el servidor origen, reduciendo los costes de operación. Sin embargo, en esta situación se perderían las estadísticas de uso y acceso tan valoradas por los proveedores de contenido, como se describía anteriormente. En esta situación, la diversidad de intereses requiere que los proveedores de red de contenido envíen estadísticas detalladas de uso a los proveedores de contenido. En caso contrario, sería poco probable que un proveedor de contenido permitiera a una *caché* distribuir contenido en su nombre. Akamai es un ejemplo de proveedor de red de contenido que se describirá en el capítulo 2.5.
- ***Consumidor de contenido (content consumer)***: el consumidor es el destinatario final del contenido. Estos consumidores son típicamente usuarios de Internet solicitando contenido a través de un navegador web. Con la disponibilidad de la banda ancha, estos consumidores solicitan cada vez más contenido multimedia de alta calidad y exigen retardos reducidos. En términos de dispositivos móviles, los usuarios esperan adaptación de contenido a las capacidades de su dispositivo y de la red. Por otro lado, las expectativas del usuario también van orientadas a recibir contenido personalizado y basado en su contexto.

Los siguientes subcapítulos profundizarán en las tecnologías más comunes de distribución de contenido (*caching*, *proxy*, *mirrors*, etc.) que han propiciado su uso en las grandes redes denominadas redes de distribución de contenido o CDNs (*Content Delivery Network*).

2.2. Escalabilidad

Se dice que un sistema es escalable si su rendimiento medio no se ve significativamente afectado conforme aumenta el número de usuarios. Es importante destacar que significativamente depende del contexto de cada sistema, y es cada administrador o analista el que impone los rangos de rendimiento aceptables.

Existen diferentes mecanismos para dotar de escalabilidad a un sistema, aunque se trata de un factor básico con un fuerte impacto en la arquitectura del sistema a tratar. Las técnicas de escalabilidad en sentido vertical (*scale-up*) no serán consideradas en profundidad en esta tesis, sino las técnicas en sentido horizontal (*scale-out*). Las primeras (véase capítulo 2.2.1) consisten en añadir un hardware o software más potente en un único nodo, mientras que las últimas (véase capítulo 2.2.2) representan una visión modular de todo el sistema, y escalar supone actuar sobre varios nodos, o incluso añadir otros nuevos.

En general, los problemas distribuidos tienen una mejor solución con sistemas distribuidos, por lo que cada vez más se emplean actualmente técnicas en sentido horizontal. Tanto unas como otras técnicas pueden ser subdivididas atendiendo a la forma en que se busca la escalabilidad, como se muestra en la Figura 15.

$$Sistemas\quad Escalables \begin{cases} Verticales \begin{cases} hardware \\ software \end{cases} \\ Horizontales \begin{cases} global \\ local \begin{cases} basado\ en\ cluster \\ distribuido \end{cases} \end{cases} \end{cases}$$

Figura 15. Clasificación de tipos de escalabilidad

Si un servidor reside en una sola ubicación, se puede aumentar su capacidad al aumentar el número de servidores en la misma ubicación (*local scale-out*) o en otra zona (*global scale-out*). La primera opción ha sido tradicionalmente escogida por la mayoría de las compañías en forma de *clústeres* o sistemas localmente distribuidos debido a motivos de seguridad y gestión; la segunda opción representa una aproximación muy empleada actualmente y es donde están localizadas las CDNs.

2.2.1. Escalabilidad vertical

En este tipo de aproximación se sustituye el sistema actual que se está usando por otro completamente nuevo y mucho más eficaz. Esta mayor eficacia se consigue mediante un hardware actual más potente y compatible con el sistema anterior. En términos prácticos, consiste en cambiar un servidor por otro con un procesador mayor, mayor

memoria y más rápida, mayor capacidad de disco duro con menor tiempo de acceso a éste, etc. Dada la compatibilidad, el impacto en la migración de un sistema a otro es mínimo, y consiste en la realización de las copias de seguridad de las tablas y aplicaciones y su posterior volcado sobre el nuevo servidor.

Con la aparición del nuevo hardware, normalmente es recomendable la actualización de software (sistema operativo, bases de datos, servidores web, etc.) para aprovechar toda la potencia ofrecida, aunque no es necesario. En este caso la migración puede resultar más complicada pero, por otro lado, los recursos adquiridos son optimizados.

La escalabilidad en sentido vertical puede parecer la opción más sencilla por su limitado impacto sobre el sistema anterior existente, pero adolece de serios inconvenientes:

- ***Límite máximo***: a pesar del continuo avance tecnológico, la potencia de procesamiento localizada en un único dispositivo (o parte del dispositivo, si tenemos en cuenta componentes como procesador, memoria RAM y disco duro) dispone de un umbral físico que algún día se alcanzará. Por otro lado, también es importante destacar la velocidad a la que la industria consigue estos aumentos de capacidad, puesto que muchas veces es inferior a la velocidad a la que aumenta el tráfico demandado por clientes: según la ley de Moore, la capacidad de los procesadores se duplica cada año y medio, mientras que en el caso de la memoria y disco duro el proceso es aún más lento; si el tráfico de un sitio web se cuadruplica en este período, es necesario recurrir a otra solución para soportar el volumen de tráfico demandado o, al menos, buscar una alternativa complementaria.
- ***Coste***: Los equipos empleados tienden a ser computadores que incorporan procesadores en paralelo, así como un hardware y software que soporta balanceo de carga y consistencia de datos de manera interna. Dada la alta potencia demandada, el coste de estos equipos puede ser fácilmente mayor que la suma de los recursos de manera separada. Por otro lado, el consumo energético de estos grandes servidores es también considerable y requieren unos recintos específicos (salas de servidores) con las condiciones ambientales óptimas para poder operar en buen estado y asegurar una larga durabilidad. Esto implica unos costes adyacentes (energía, espacio, mantenimiento) importantes.
- ***Redundancia***: los sistemas redundantes tienden a duplicar recursos para garantizar una alta disponibilidad en caso de fallo o avería. En caso de equipos de elevadas prestaciones puede implicar duplicar también el coste (si duplicamos exactamente el mismo equipo), lo que puede ser un gasto significativo.

2.2.2. Escalabilidad horizontal

En la escalabilidad horizontal se evita sustituir el sistema actual por otro más potente; en su lugar, se trata de añadir nuevos nodos que coexistan con el sistema actual para dotarlo, en su globalidad, de una mayor potencia. En principio, conforme se van añadiendo nuevos nodos, el sistema global resultante es más potente, aunque hay que tener en cuenta algún mecanismo de coordinación entre nodos. La adición de estos nodos puede tener lugar de forma local (en la misma subred que el sistema actual que se desea escalar), o bien de forma global (en diferentes subredes del sistema actual). Ambos modos de escalabilidad, local y global, se describen en los subcapítulos siguientes.

2.2.2.1. Escalabilidad local

Un sistema distribuido de manera local (Figura 16) consiste en un conjunto de servidores locales cuyas direcciones IP son visibles por los clientes, mientras que un sistema basado en un *clúster*, a veces también denominado granja de servidores (Figura 17), enmascara a todas ellas, ofreciendo un dirección única virtual asociada a un dispositivo delante de los servidores que actúa a modo de interfaz con el mundo exterior. Un completo estado del arte de los sistemas locales puede encontrarse en [Car_01]. Pese a estar basado en aplicaciones web, su enfoque generalista lo hace válido para otros entornos.

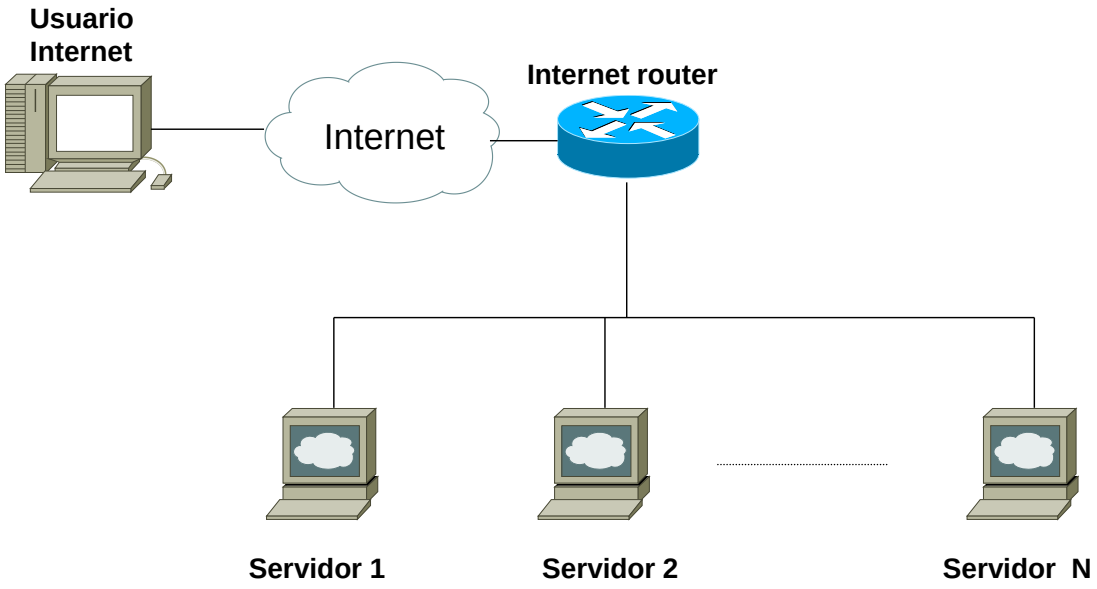


Figura 16. Arquitectura de escalabilidad local (distribuido).

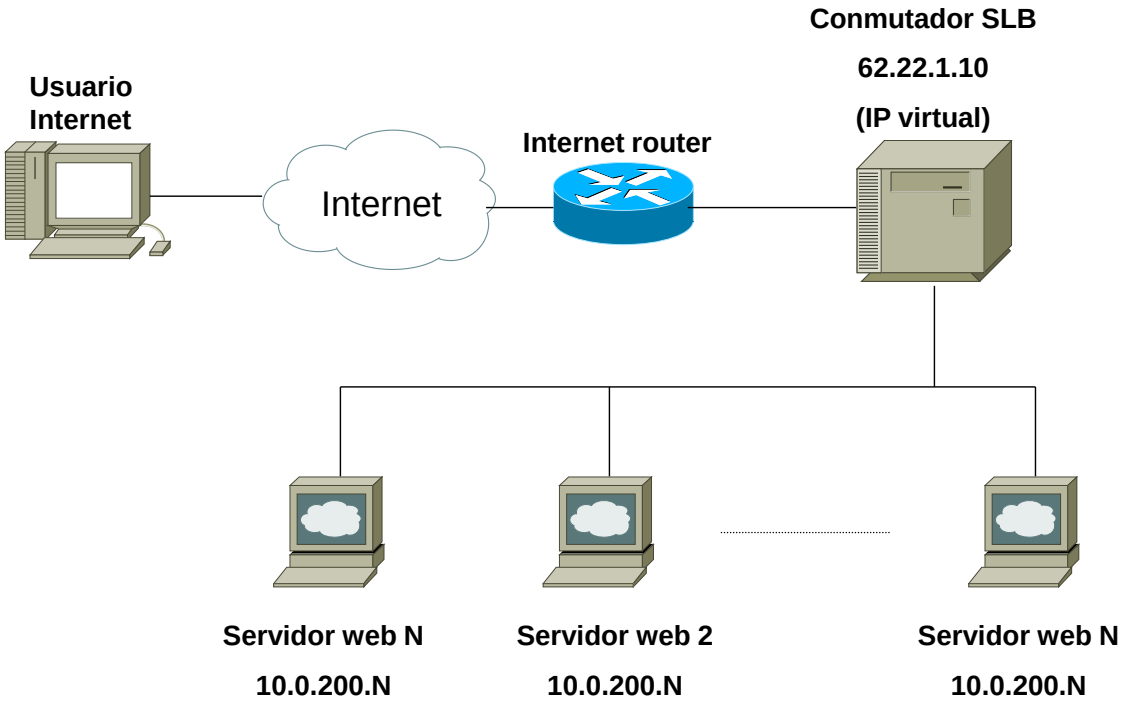
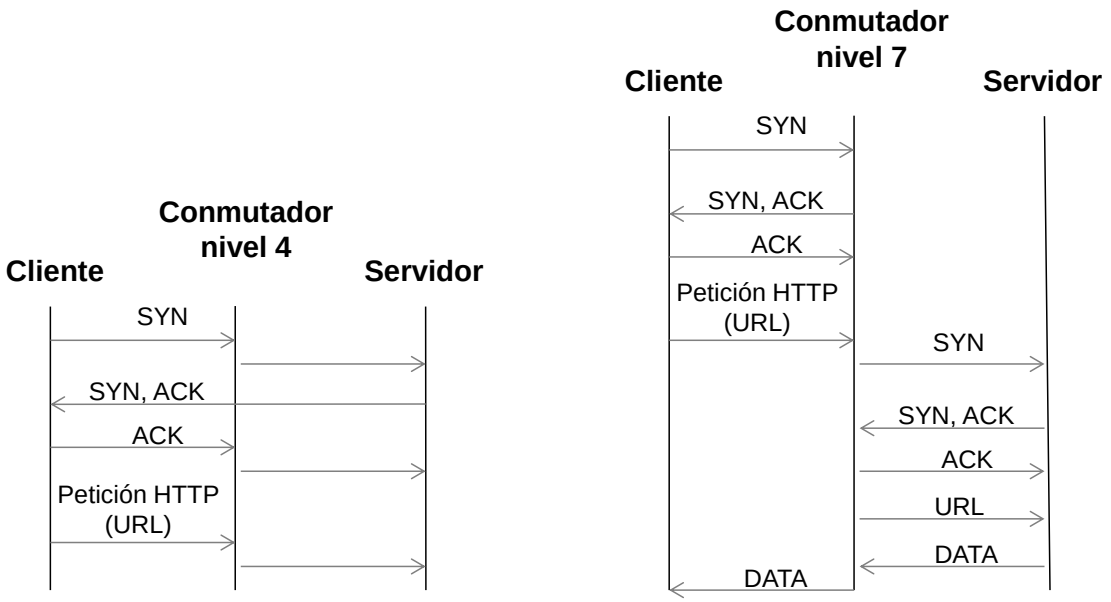


Figura 17. Arquitectura de escalabilidad local (clúster).

En ambas arquitecturas es necesario (i) redirigir al cliente al servidor más apropiado y (ii) conocer cuál es el servidor más adecuado. Pese a tratarse de implementaciones locales, algunas características de diseño pueden emplearse también en técnicas escalables globales en sentido hacia fuera (como es el caso de las redes de distribución de contenido).

En una arquitectura basada en clúster, el *conmutador* (también conocido como balanceador de carga) es responsable de satisfacer las dos condiciones anteriormente citadas. La forma en las que encamina los paquetes entrantes puede tener lugar tanto a nivel 4 como a nivel 7. La Figura 18 representa el protocolo HTTP ejecutándose sobre TCP, que emplea un mecanismo a tres bandas en el establecimiento de la conexión: el cliente envía primeramente un paquete TCP con el campo SYN puesto a “1”, El servidor responde con un paquete con los campos SYN y ACK a “1”, y finalmente, el cliente contesta con un paquete TCP con el campo ACK puesto a “1”. Es sólo a partir de este momento cuando se envía la cabecera HTTP y el servidor o una entidad intermedia puede conocer el contenido deseado a través de su URL. Normalmente, la implementación software del *conmutador* conduciría a la obtención de serios retardos y bajos rendimientos, por lo que, normalmente, se emplea un dispositivo hardware con las rutinas de conexión embebidas en su *kernel*.



a) Conmutación nivel 4 b) Conmutación nivel 7

Figura 18. Flujo de mensajes en las capas 4 y 7.

Los conmutadores de nivel 4 son ciegos al contenido (*content-blind*), puesto que la asignación del servidor tiene lugar en el momento de la petición de conexión sobre TCP. Por el contrario, los conmutadores de nivel 7 son conscientes del contenido (*content-aware*), puesto que retrasan la asignación de servidor hasta que la información acerca del contenido llega en los protocolos superiores y dicha información puede ser examinada. Pese a su menor rendimiento en comparación con soluciones de nivel 4, las arquitecturas de nivel 7 permiten técnicas de *caching*, así como reutilización de sesiones SSL. Además, es posible introducir una cierta persistencia, es decir, la capacidad de asignar a un usuario al mismo servidor en sucesivas peticiones HTTP, dado que muchos *websites* mantienen información de sesión, como una página donde se pueden comprar productos. El método más comúnmente usado es el uso de *cookies* [Kri_00] en el navegador, según se describe en la RFC 2965.

Resulta obvio que todo el tráfico entrante debe ser absorbido por el conmutador; por el contrario, el tráfico saliente puede atravesar el conmutador (arquitectura bidireccional) o partir directamente desde el servidor final que se ha contactado (arquitectura unidireccional).

En estas arquitecturas distribuidas, es conveniente destacar los mecanismos de encaminamiento, es decir, los procesos que suceden desde que un cliente solicita un contenido hasta que finalmente le llega a un servidor que puede satisfacer dicha petición. Normalmente, estos mecanismos de encaminamiento (a veces también denominados de redirección), presentan dos fases, una asociada al sistema DNS [Moc_87] [Moc_87b], y otra asociada a los propios servidores. El encaminamiento a través del DNS representa un primer nivel de redirección, y puede ser empleado tanto

de forma local como en sistemas globales. Por ejemplo, muchos proveedores de CDNs usan un encaminamiento basado en DNS [Bar_01]. El servicio de DNS proporciona la conversión de un nombre de dominio en direcciones IP; durante la fase de consulta, el servidor DNS autorizado responsable del *website* decide el servidor más adecuado y devuelve su correspondiente dirección IP al cliente. Técnicas de *caching* [Bae_97] en la cadena intermediaria de consulta del sistema DNS impiden o dificultan un mecanismo fino de redirección, por lo que técnicas de segundo nivel de redirección se pueden emplear en los propios servidores. Por ejemplo, si el protocolo empleado es HTTP [Fie_99], existen códigos explícitos de redirección (códigos 300).

Tras esta primera fase de redirección, cabe describir la segunda fase, es decir, cómo el sistema determina el servidor en concreto que atenderá la solicitud de un cliente. Los algoritmos utilizados se pueden clasificar de varias formas: estático-dinámico, centralizado-distribuido, etc.

En una arquitectura basada en clúster se emplea típicamente un algoritmo centralizado en el conmutador, que hace balanceo o compartición de carga entre todos los servidores. Mientras que los conmutadores que actúan a nivel 4 suelen emplear algoritmos estáticos (*random*, *round-robin*), los de nivel 7 suelen emplear algoritmos dinámicos, como CAP (*Client Aware Policy*) [Cas_01], su variante más moderna GCAP (*Grouped Client Aware Policy*) [Chi_09] o LARD (*Locality-Aware Request Distribution*) [Viv_98] y su variante más reciente LARD/RC (*Locality-Aware Request Distribution with Replication and Classification*) [Chi_09]. Cuando se emplea un algoritmo dinámico, las decisiones en cuanto al servidor óptimo pueden estar basadas sólo en información de estado del servidor (servidor menos cargado, servidor con menor número de conexiones, etc.), en información del cliente, o en ambas.

En una arquitectura distribuida una de las técnicas más empleadas es el denominado '*broadcast and filter*', donde las peticiones son enviadas a todos los servidores y un protocolo colaborativo es empleado entre todos ellos para garantizar que sólo uno contestará a la solicitud. Por ejemplo, cada servidor puede disponer de una lista de direcciones de clientes a las que contestará, mientras que el resto serán filtradas. La gran ventaja de esta opción distribuida es que se evita un único punto de fallo de la arquitectura centralizada. Por el contrario, la carga puede no estar correctamente balanceada entre los servidores. Además, dado que el servidor dedica parte de su tiempo descartando paquetes, el rendimiento es menor en comparación con un balanceador de carga frontal.

Otra opción distribuida, usada también en esquemas de escalabilidad global, consiste en una adaptación de alto nivel de la técnica de *anycast*. *Anycast* es un esquema de direccionamiento así como una metodología de encaminamiento donde los paquetes que provienen de un emisor son encaminados al nodo topológicamente más cercano dentro de un grupo de receptores potenciales asociados a la misma dirección destino. Actualmente el sistema de *root servers* del servicio DNS emplea anycast [Sar_06].

2.2.2.2. Escalabilidad global

La escalabilidad global consiste en distribuir los equipos que conforman un sistema a través de una gran red, o incluso conectados a través de Internet. Las motivaciones de esta aproximación son variadas y diversas:

- En una red corporativa de ámbito nacional o internacional que interconecta diversos puntos puede resultar conveniente ubicar al menos un servidor en cada uno de ellos, para dar servicio de una forma independiente o, en caso de fallo o avería, disponer de una configuración redundante. Evidentemente esto requiere una sincronización y consistencia de datos entre servidores que hay que garantizar de alguna forma.
- Una red de investigación de ámbito nacional o internacional interconecta típicamente diversas universidades y centros de investigación. En este caso, la gestión tiende a ser descentralizada por motivos de flexibilidad, de tal forma que cada universidad o centro que se adhiere a dicha red puede fácilmente colaborar, ampliar y mejorar la capacidad de la red introduciendo uno o más servidores y configurándolos para formar parte de dicha red de investigación. Un ejemplo claro a nivel nacional lo constituye RedIris [WWW_Red], financiada por el Ministerio de Economía y Competitividad. Otro ejemplo a nivel global en el ámbito de servicios de telecomunicaciones lo constituye PlanetLab [WWW_Pla], una plataforma abierta para desarrollar y desplegar servicios.
- Otra forma de distribuir contenido a nivel mundial es mediante las redes de nivel de aplicación (*overlay networks*), cuyos dos mayores exponentes son las redes P2P y las CDNs. En el caso de las redes P2P, cada usuario es potencialmente un nodo del sistema, y dado su ubicación desconocida (al menos a priori) es necesario configurar el sistema partiendo de un sistema distribuido y global. En estos sistemas no se garantiza la disponibilidad del contenido, pues depende directamente del número de usuarios activos y el tiempo de vida de estos en una sesión P2P. No obstante, las redes P2P de compartición de ficheros son altamente empleadas a nivel mundial y proporcionan un servicio relativamente eficaz con ciertos interrogantes legales. Un completo estudio de este tipo de redes puede encontrarse en [Bu_09], incluso es posible distribuir vídeo con estos sistemas [Set_10], aunque no es objeto de esta tesis. Por otro lado, las redes de distribución de contenido (CDNs), como Akamai o *LimeLight*, cuentan con el interés corporativo de un proveedor, por lo que sí se proporciona alta disponibilidad y se puede hacer un cierto estudio a priori sobre los servicios que se quieren ofrecer para dimensionar el sistema y permitir un crecimiento de una forma económicamente controlada. En apartados posteriores se abordarán en mayor profundidad estas CDNs.

2.3. Web caching

El crecimiento del tráfico HTTP en Internet condujo necesariamente a una demanda de escalabilidad en la infraestructura de Internet. Las técnicas de *caching* cerca de los clientes aparecieron a principios de los 90 como una forma de reducir considerablemente el tráfico de red en las redes troncales. Además, de esta forma, la latencia WAN puede permanecer oculta, de igual forma que momentáneas indisponibilidades de la red. En el caso de aplicaciones web, sin duda alguna el tráfico más relevante cuando aparecieron estos mecanismos, la técnica de *web caching* [Wil_96] puede tener lugar en dos sitios: en los navegadores (*browsers*) y en los *proxies*. La caché asociada a los navegadores almacenan objetos de manera local en cada cliente, mientras que la caché de un *proxy* puede ser simultáneamente accedida por varios usuarios. En este capítulo se describirá básicamente el funcionamiento del *caching* para transacciones HTTP. No obstante, cabe mencionar que las *cachés* (en adelante también *proxy cachés*) actuales permiten almacenar otro tipo de contenido, como audio y vídeo [Pod_02] [Ma_04] [Xie_09]. En algunas ocasiones, es incluso posible embeber este contenido multimedia como objetos web. Sin embargo, las características especiales de este tipo de contenido multimedia (en términos de espacio de almacenamiento) hacen que las políticas de almacenamiento y reemplazo sean distintas al caso de objetos web tradicionales. Este aspecto se abordará con mayor detalle en la sección 2.3.4.

Sin un *proxy caché*, todas las peticiones serían recogidas por el servidor, originándose un efecto de implosión (exceso de peticiones de múltiples clientes en un solo servidor), como se observa en la Figura 19. Retomando el funcionamiento del *proxy caching* ilustrado en la Figura 20, el *proxy caché* recoge las peticiones HTTP de los clientes y las analiza. Si el objeto solicitado se encuentra en la *caché* local (acierto o *hit*), se le devuelve al usuario. Por el contrario, si no se encuentra (fallo o *miss*), el *proxy caché* obtiene primero el objeto del servidor origen, lo almacena en su *caché* local y, posteriormente, se lo envía al cliente. Como puede observarse, la efectividad del caching depende de su tasa de acierto (*hit ratio*).

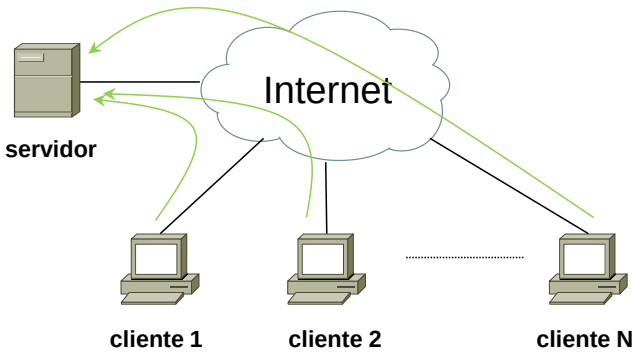


Figura 19. Servidor sobrecargado sin caché.

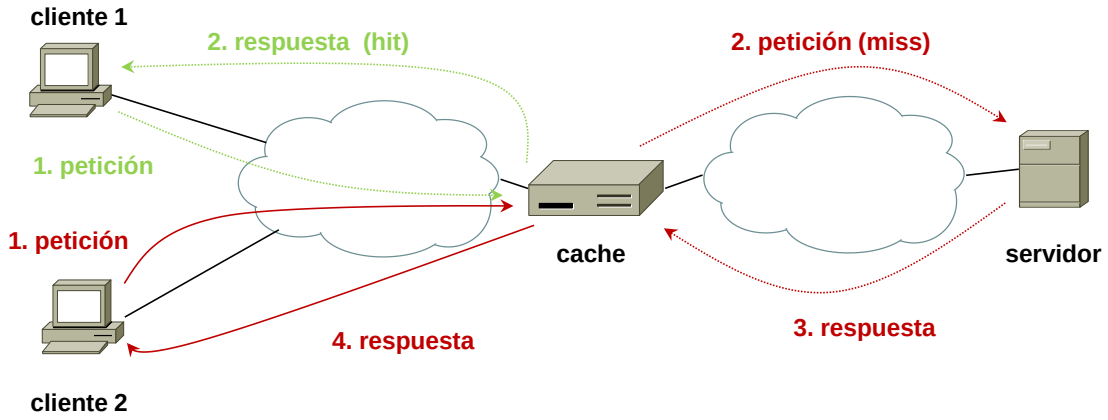


Figura 20. Beneficios del proxy caching.

De una manera más rigurosa, se entiende que se produce un acierto (*hit*) no sólo cuando el contenido ya se encuentra previamente almacenado, sino cuando es también válido o reciente y consistente (*fresh*). Esto se puede determinar examinando la información disponible relativa a la fecha de creación del objeto, la fecha de almacenamiento, la fecha de expiración y las preferencias del cliente y servidor web. El protocolo HTTP 1.1 incluye varios campos y directivas que permiten tomar una decisión precisa. El algoritmo básico para la toma de decisión de un *proxy cache* se puede observar en la Figura 21 cuando un cliente realiza una petición web.

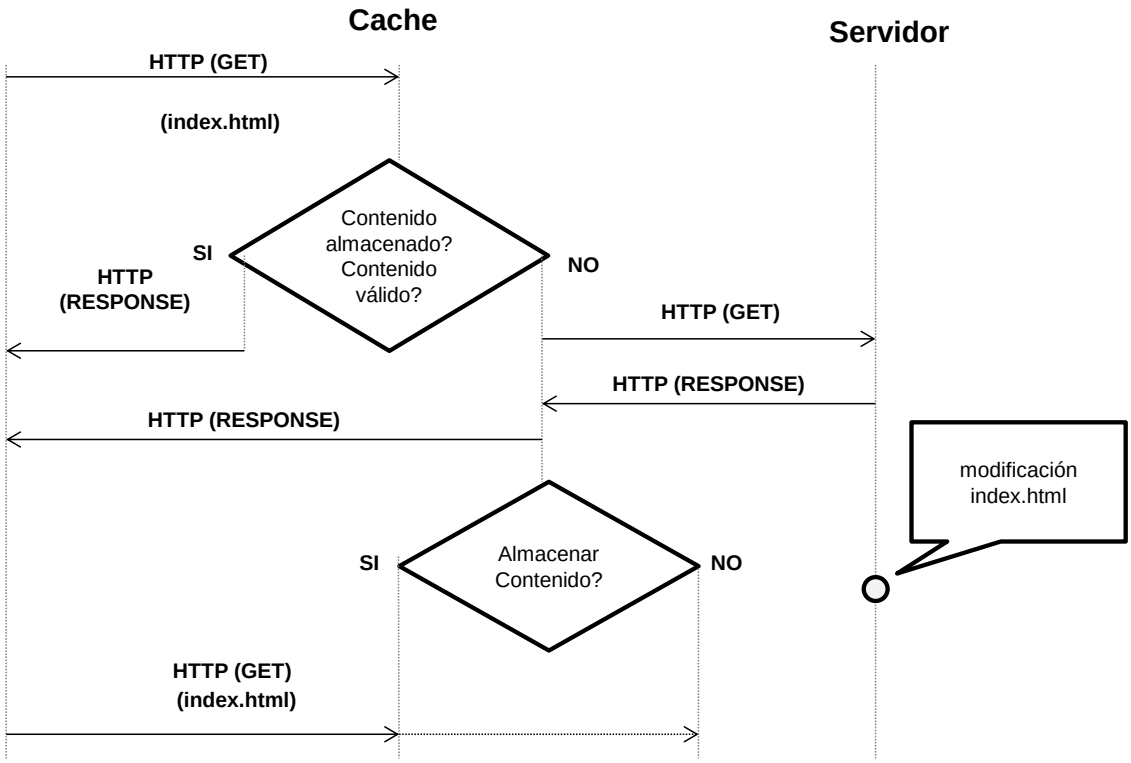


Figura 21. Funcionamiento interno básico de un proxy cache.

La Figura 21 también muestra la problemática asociada una vez el *proxy cache* ha obtenido una copia del servidor, pues debe determinar si almacenar el contenido una vez servido o no; en caso afirmativo, deberá tener en cuenta durante cuánto tiempo debe almacenarlo para que una modificación de un objeto web en el servidor no represente ningún problema, y una nueva petición por parte de un cliente suponga obtener dicho objeto web con la última modificación [Kas_07].

En entornos web, las *caches* suelen emplear la operación HTTP GET condicional para determinar si un objeto almacenado está actualizado. Esta propiedad permite a la *cache* indicar al servidor que únicamente envíe el objeto web si se ha modificado recientemente. Si no ha sido modificado, el servidor retorna únicamente la cabecera HTTP, como se observa en la Figura 22.

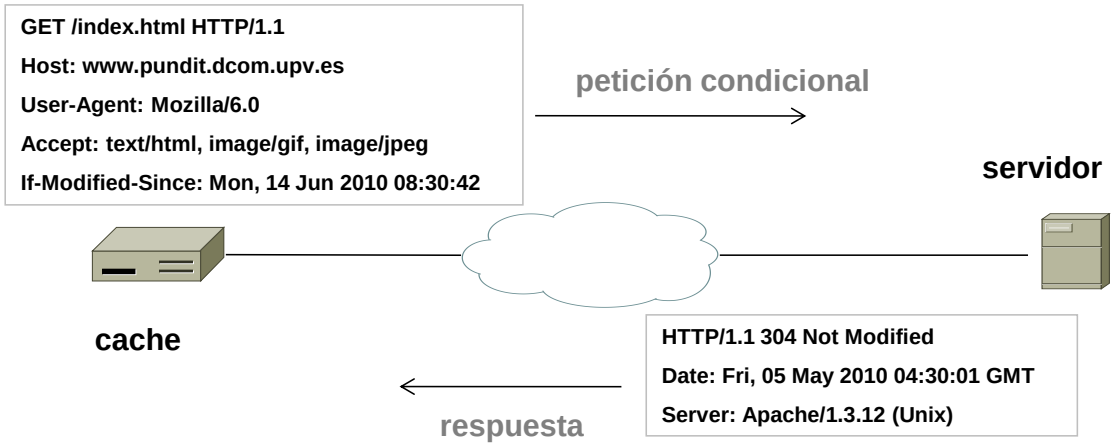


Figura 22. Petición HTTP Condicional

Esto reduce carga en el servidor y en la red, aunque no reduce apenas el tiempo de respuesta, ya que los paquetes de petición y respuesta siguen teniendo que atravesar toda la red.

La gestión de la caché es un componente funcional fundamental de un *proxy* si se desea aumentar el rendimiento, pero este dispositivo también puede cumplir otras funciones, como son:

- ***Filtrado***: un *proxy* puede ser considerado como un cortafuego (*firewall*) de nivel de aplicación, puesto que intercepta los paquetes entrantes y salientes, por lo que puede permitirles el paso o denegárselo. Asimismo, también permite registrar en un fichero de *log* la actividad de la comunicación. Actualmente, un servicio de valor añadido es la posibilidad de comprobar en los paquetes entrantes (ficheros adjuntos en los correos) la presencia o ausencia de virus.
- ***Compartición de conexiones***: un *proxy* puede ser configurado análogamente a un servicio de NAT permitiendo la conexión a Internet de múltiples usuarios.

2.3.1. Conceptos básicos y funcionamiento

El objetivo principal de una *caché* es almacenar algunas de las respuestas que recibe de los servidores origen. En términos generales, se dice que una respuesta es *cacheable* si se puede usar para responder futuras peticiones. De manera más particular, un *proxy cache* determina si una respuesta es *cacheable* dependiendo de varios componentes de la petición y de la respuesta, como son:

- ***Código de respuesta (status codes)***: típicamente se toman los códigos 200, 203, 206, 300, 301, 410 como *cacheables* por defecto, aunque otros códigos también pueden ser *cacheables* si se indica explícitamente mediante otros medios (por ej. mediante una directiva *Expires* en la cabecera HTTP).
- ***Método de petición***: GET es el método más popular y es *cacheable* por defecto. Por otro lado, si bien el método HEAD no incluye cuerpo de mensaje, se pueden emplear las cabeceras de respuesta para actualizar metadatos (por ej. un nuevo tiempo de expiración). La respuesta ante una petición de tipo POST sólo es *cacheable* si incluye un tiempo de expiración o una directiva *Cache-control*.

Tipo de petición	Capacidad de ser cacheable
GET	Si, *cacheable* por defecto
HEAD	Puede ser usado para actualizar contenido anteriormente *cacheado*
POST	No *cacheable* por defecto, puede serlo si la cabecera *Cache-Control* lo permite
PUT	No *cacheable*
DELETE	No *cacheable*
OPTIONS	No *cacheable*
TRACE	No *cacheable*

Tabla 2. Tipo de petición y cacheabilidad.

- ***Expiración y validación***: HTTP 1.1 proporciona dos formas de mantener la consistencia con los servidores origen: tiempo de expiración (*expiration times*) y validadores (*validators*) [Fie_99]. Idealmente, cada respuesta *cacheable* debería incluir una o ambas formas; en la realidad, existe un pequeño pero significativo porcentaje de peticiones que no incluyen ninguna de las dos formas anteriores. Es importante incidir en la diferencia entre expiración y *cacheabilidad*: respuestas que hayan expirado pueden permanecer en la memoria *caché*, simplemente deben validarse nuevamente antes de poderse usar.
- ***Cabecera Cache-control***: Esta cabecera es una característica nueva en HTTP 1.1 e indica a los *proxy caches* cómo deben tratar las peticiones y las respuestas. El valor de dicha cabecera puede constar de varias directivas, algunas de las principales se reflejan en la Tabla 3. La lista completa de directivas de tipo *Cache-Control* se puede encontrar en la RFC 2616 [Fie_99].

Directiva	Función
No-cache	El contenido puede ser almacenado en *caché* pero debe revalidarse antes de un nuevo uso
Private	Permite almacenar contenido en *caché* local del usuario (*browser*), pero no en *proxy cachés* compartidas
public	Permite almacenar contenido en *caché* local del usuario y en *proxy cachés* compartidas

Max-age	Permite especificar un tiempo de expiración. Alude implícitamente la directiva *public*
s-maxage	Similar a la directiva *max-age* pero sólo es aplicable a *cachés* compartidas. Alude implícitamente la directiva *must-revalidate*
Must-revalidate	Permite a las *caches* almacenar respuestas que son típicamente no *cacheables*
Proxy-revalidate	Similar a la directiva *must-revalidate*, pero sólo aplicable a *cachés* compartidas
No-store	Convierte cualquier respuesta en no *cacheable*

Tabla 3 . Tipos de petición y cachabilidad.

- ***Autenticación***: Las peticiones que requieren autenticación no son *cacheables* en términos generales. Solamente el servidor origen puede determinar quién puede acceder a sus recursos. Dado que un *proxy caché* desconoce qué usuarios están autorizados, no puede distribuir aciertos invalidados. Las condiciones bajo las cuales se permite *cachear* respuestas autenticadas son especialmente delicadas y quedan recogidas en la sección 14.8 de la RFC 2616. Básicamente, esto sólo es posible si está presente alguno de las siguientes cabeceras *Cache-control*: *s-maxage, must-revalidate* o *public*.

- ***Cookies***: Una *Cookie* permite mantener información de sesión entre sucesivas peticiones, y queda descrita en la RFC 2965 [Kri_00]. Pese a que dichas *Cookies* habitualmente representaban información privada no *cacheable*, en la actualidad y en la mayoría de ocasiones el objeto web contenido en el cuerpo del mensaje de respuesta puede ser público y *cacheable*. HTTP dispone de un método que permite convertir la información de la *Cookie* en no *cacheable*, mediante la directiva *Cache-control*: *no-cache*.

Un aspecto crucial en el *caching* es cómo medir su efectividad. Para ello, existen dos medidas básicas:

- ***Cache hit ratio***: Se trata del porcentaje de peticiones que se pueden considerar aciertos (*cache hits*). Como se ha descrito anteriormente, se tiende a englobar en los aciertos tanto a los válidos como a los no validados. Nótese que este valor sólo identifica los aciertos, pero no indica cuanto ancho de banda (o latencia) se ha ahorrado. Los valores de *cache hit ratio* oscilan entre el 30% y el 70% dependiendo del tamaño de la caché y las políticas empleadas en la configuración.

- ***Byte hit ratio***: Esta medida indica el ancho de banda ahorrado, aunque existen diferentes maneras de calcular este valor. Una forma sencilla es comparar el tamaño de los objetos asociados a aciertos (*hits*) y fallos (*misses*), aunque a veces no es muy representativa porque no se tiene en cuenta el tráfico en ambos lados del *proxy caché* (red con el cliente y red con el servidor origen). Los valores del *byte hit ratio* típicos suelen ser un 10% inferior al *cache hit ratio*, debido fundamentalmente a la diferencia de tamaño (y función de distribución) entre los objetos almacenados en el servidor y los objetos devueltos a los clientes en los mensajes de respuesta por parte del *proxy caché*.

Básicamente, el *hit ratio* (tanto *cache* como *byte hit ratio*) vienen determinado por tres factores principales: el tamaño de la *caché*, el número de clientes y las heurísticas capaces de estimar cuando un contenido es válido. En primer lugar, si el tamaño de la *caché* aumenta, resulta evidente que aumente la probabilidad de acierto; sin embargo, esta relación no es lineal, sino más bien pseudo-logarítmica [Dus_97]. En segundo lugar, debido a las características de popularidad de las peticiones web, el *hit ratio* aumenta conforme aumenta el número de clientes. Este resultado ha sido demostrado mediante evidencias empíricas y estudios de simulación [Bre_92]. En tercer lugar, las heurísticas de contenido válido pueden ser configuradas para forzar o disminuir el número de validaciones necesarias con el servidor, y con el ello aumentar el *hit ratio*, aunque con ello la probabilidad de entregar contenido obsoleto aumenta. Normalmente las *cachés web* utilizan en su algoritmo un factor LM (*last-modified-factor*). Básicamente, se trata de estimar la fecha de expiración de un objeto a partir del campo *Last-Modified* de la cabecera. Si el servidor web origen no ha proporcionado un tiempo de expiración para un documento web en concreto, entonces el algoritmo estima uno mediante la siguiente regla:

Tiempo de expiración = (Tiempo desde última modificación) * (factor LM) (E 2.1)

Por ejemplo, si un documento web fue modificado por última vez hace 30 horas y el factor LM es 10, el tiempo de expiración sería de 3 horas (siempre que no se sobrepase un umbral máximo común a todos los objetos).

2.3.1.1. Consistencia (cache consistency)

Los datos que se han almacenado en *caché* no deben permanecer cuando sean obsoletos. Para evitar esto, existen varias técnicas [Liu_10] para detectar cuándo un objeto almacenado ya no es equivalente con su correspondiente en el servidor origen. En el denominado *client polling* el *proxy cache* es el encargado de asegurar la consistencia mediante marcación temporal en cada objeto almacenado. Mediante comparaciones periódicas (*modelo de validación*) con el objeto original se asegura dicha consistencia. De cara a mejorar el rendimiento en el proceso, HTTP 1.1 soporta métodos condicionales: de esta forma, cuando un cliente o servidor *caché* solicita una petición condicional por un recurso que previamente ha almacenado en su *caché*, añade el asociado *validador* en la solicitud. Este *validador* es contrastado en el servidor origen con su actual objeto. Si ambos coinciden, el servidor envía un código especial de estado (*304, Not Modified*), sin cuerpo de mensaje. En caso contrario, devuelve una respuesta global con cuerpo de mensaje. Nótese que, tras validar un objeto, no hay forma de conocer si ha cambiado hasta la siguiente comprobación.

Otra solución (*modelo de expiración*) dentro del *client polling* consiste en el uso del parámetro TTL (*time-to-live*) asociado a cada objeto almacenado en *caché*. Tras expirar, el objeto deja de ser válido y se descarta. Si este tiempo no es enviado por el servidor

origen, es responsabilidad del *proxy caché* el asignar de manera heurística un tiempo de expiración, lo que puede comprometer la transparencia semántica. Una variación de esta última propuesta la constituye el método *if-modified-since*, donde los objetos son únicamente invalidados tras ser nuevamente solicitados y su valor TTL asociado ha expirado. Evidentemente, esto supone realizar *polling* de forma continua, por lo que el tráfico adicional puede ser significativo e innecesario si el contenido no cambia frecuentemente. Para evitar un *polling* continuo existen técnicas como TTL adaptativo [Cat_92] basados en la premisa de que objetos recientes son modificados más frecuentemente que objetos antiguos, y que éstos a su vez tienen menos probabilidad de ser modificados. Mediante el uso de TTL adaptativo, la probabilidad de objetos obsoletos está dentro de márgenes aceptables (5%).

Para evitar las comparaciones periódicas, existe la técnica de invalidaciones por parte del servidor origen (*invalidation callbacks*). Pese a que mejora la consistencia y reduce el ancho de banda, existen serios problemas de escalabilidad conforme aumenta el número de objetos que se están *cacheando* en nodos intermedios (el servidor origen debe tener información acerca de cada objeto que algún nodo ha almacenado en su *caché*, así como de todos los nodos intermedios que están actuando como *proxy cachés*). Existen algunas variantes de esta técnica de invalidación por parte del servidor, como son PSI (*Piggyback Server Invalidation*) [Kri_98] y PCV (*Piggyback Cache Validation*) [Kri_97].

Otras técnicas de consistencia implican una interacción cliente-servidor (C/S). Para garantizar la consistencia de caches en sistemas de ficheros distribuidos, Gray y Cheriton propusieron la técnica de *lease* [Gra_89]. La idea principal de esta especie de contrato (*lease*) es que cuando el servidor origen devuelve un objeto tras una petición de un cliente, asigna un valor de *lease* a dicho objeto. El servidor garantiza que no actualizará dicho objeto hasta que el contrato (*lease*) expire. Una vez que el contrato expira, el *proxy caché* enviará un mensaje "*if-modified-since*" al servidor origen, quien responderá con la nueva versión del objeto o, si el objeto no ha sido modificado, extenderá dicho contrato.

Desde otro punto de vista, dependiendo del grado de consistencia en los *proxy cachés*, los algoritmos destinados a esta finalidad pueden ser catalogados de fuertes (*strong*) o débiles (*weak*). Si nunca se devuelve contenido obsoleto (*stale*) a los usuarios, el algoritmo emplea consistencia fuerte; en caso contrario, el algoritmo utiliza consistencia débil. La Tabla 4 muestra los algoritmos de consistencia de cache más conocidos.

Técnica	Consistencia fuerte	Consistencia débil
Validación de cliente	Polling continuo	TTL, PCV en cliente
Invalidación de servidor	Invalidación	PSI
Interacción C/S	Lease	

Tabla 4. Clasificación de algoritmos de consistencia de cache.

2.3.1.2. Sustitución (cache replacement)

Los *proxy cachés* disponen de un tamaño limitado de memoria y disco, por lo que hay que aplicar ciertos mecanismos o políticas de sustitución de contenido almacenado en *caché* dependiendo del tamaño, tiempo de expiración, frecuencia de uso, etc., con el objetivo de maximizar el porcentaje de aciertos. Aunque existen numerosas estrategias para descartar objetos, las políticas tradiciones se siguen empleando bastante en la actualidad por su sencillez. Las más utilizadas son las siguientes:

- ***LRU (Least Recently Used)***: este mecanismo sustituye el objeto cuya última petición es la más antigua de entre todos los objetos. Con esta política el *proxy cache* se llenará con los objetos que son solicitados más recientemente.
- ***LFU (Least Frequently Used)***: este algoritmo descarta el objeto que tiene el menor número de solicitudes desde que se almacenó. Así, el *proxy caché* se llenará con los objetos más frecuentemente usados.
- ***FIFO (First In First Out)***: esta política descarta el objeto más antiguo, basado en cuándo se almacenó por primera vez. Con esta política el *proxy caché* se llenará con los objetos que más recientemente se refresquen.
- ***NTE (Next to Expire)***: este mecanismo sustituye el objeto que previsiblemente va a expirar más pronto. De esta forma, el *proxy caché* se llenará con los objetos más estables que apenas se modifican.
- ***LFF (Largest File First)***: este algoritmo descarta el objeto de mayor tamaño, con la intención de liberar el máximo espacio posible para nuevos objetos. Normalmente, tanto el tamaño como la antigüedad son considerados para determinar el objeto a reemplazar. Con esta política el *proxy caché* tiende a llenarse con muchos objetos pequeños.

Téngase en cuenta que todas las estrategias anteriormente mencionadas tienen ventajas y limitaciones. Dependiendo del tipo de tráfico esperado en la red, pueden ser convenientes unas u otras. Es posible también emplear una política combinada de varias de ellas, con algún parámetro adicional de tipo heurístico que proporcione una solución práctica y eficiente.

La técnica LRU es posiblemente la más empleada por los algoritmos de gestión de *caching* de memoria y disco local, explotando el principio de localidad temporal. Pese a que se trata de un algoritmo simple de implementar, robusto y efectivo, no es adecuado su uso en *web caching* debido a varias limitaciones. LRU se centra en los objetos recientemente usados, mientras que el *web caching* parte de la base de que los objetos web pueden variar sustancialmente en tamaño, y es preferible almacenar cuantos más objetos de menor tamaño mejor (que sean *cacheables*, evidentemente). Es por ello por lo que se han propuesto variantes sobre el algoritmo LRU.

El algoritmo GDS (*Greedy Dual Size*) [Cao_97] combina el principio de localidad temporal con el tamaño del objeto y el coste de obtenerlo. Dependiendo del tamaño y el coste, cada objeto recibe un valor inicial cuando ingresa en la *caché* (u obtiene una nueva referencia o *hit*). El valor decrece con el tiempo si no obtiene una nueva referencia. El objeto de menor valor es el candidato a ser descartado (sustituido). El algoritmo GDS adolece del inconveniente que no considera la diferencia de popularidad. La diferencia de popularidad de contenido web diferente induce a una distribución de peticiones no uniforme, típicamente una distribución de tipo Zipf [Bre_99b], que será el tipo de distribución que se emplee en las simulaciones y pruebas de esta tesis en los capítulos siguientes donde se estudia el funcionamiento de una CDN.

Una variante del algoritmo anterior basado en la popularidad lo constituye GDSP (*Greedy Dual Size Popularity*) [Jin_99], aunque este algoritmo implica disponer de una larga cola de prioridades. Otro algoritmo lo constituye LRV (*Least Relative Value*) [Riz_98], que considera los factores de tamaño, recentud y frecuencia, aunque es fuertemente criticado por su elevada parametrización y sobrecarga en la implementación [Cao_97]. También existen otras extensiones al algoritmo LRU basadas en la frecuencia, como son FBR (*Frequency Based Replacement*) [Rob_90], LRU-K [One_93] y *Segmented* LRU [Kar_94], aunque ninguno de ellos toma en consideración el tamaño de los objetos web. A partir de dos extensiones LRU (*Size Adjusted* LRU y *Segmented* LRU), se ha propuesto el algoritmo LRU-SP [Che_00] que integra las principales ventajas de ambos, como son una reducida carga y una elevada adaptabilidad.

Como se puede apreciar, los principales algoritmos de sustitución de *caché* (así como sus variantes) se realizaron principalmente en la década de los 90, cuando el *web caching* resultaba necesario debido a la liberalización de las telecomunicaciones y la entrada masiva de usuarios de Internet (véase el capítulo 2.1). Tras la especificación de HTTP 1.1 en 2004 con la incorporación de nuevas directivas de *caching* frente a HTTP 1.0, los algoritmos actuales de sustitución de *caching* no han variado sustancialmente con los algoritmos anteriormente descritos. No obstante, el lector puede consultar otros algoritmos y estudios en [Bal_04] y [Sat_12].

Por otro lado, cabe mencionar que la mayor parte de las técnicas y algoritmos descritos en este capítulo son principalmente útiles para *web caching*, y su aplicación es muy limitada para contenido multimedia o streaming. Al tratarse de un tipo de contenido importante para esta tesis, se dedicará un subcapítulo especial a las técnicas de *caching* para este tipo de contenido (véase capítulo 2.3.4).

2.3.2. Arquitecturas y técnicas de diseño

Los *proxy caché* se encuentran normalmente situados en la red en tres tipos de configuraciones:

- Un *proxy* normal (*forward proxy*) actúa en el extremo de una red dando servicio a todos los usuarios de dicha red que consumen contenido de un servidor remoto.
- Un *proxy* remoto (*reverse proxy*), también llamado acelerador web (*server accelerator*), está ubicado junto a un servidor o servidores remotos y agiliza (acelera) las descargas, liberando al servidor de carga innecesaria.
- Un *proxy* de interceptación (*interception proxy*) se encuentra ubicado en un punto intermedio de la red entre cliente y servidor.

La aplicación de las técnicas de *caching* en ambos extremos de la comunicación (en el cliente y en el proveedor de contenidos) está ampliamente recomendada al ser la configuración más efectiva. Otras sugerencias giran en torno a la colocación de servidores caché en puntos estratégicos de la red ([Wie_02] [Ho_07] [Che_07] [Ho_08] [Man_10]) atendiendo a patrones de tráfico y a la topología de la red. Una CDN puede considerarse como una posible implementación.

2.3.2.1. Forward Proxy

La mayoría de los navegadores web soportan el uso de un *proxy normal*, y requieren configurarlo adecuadamente para que encaminen las peticiones al *proxy* y no al servidor web remoto. La principal ventaja de esta configuración es que el *proxy* sirve contenido a toda la LAN y, si los usuarios tienen intereses comunes (entorno corporativo), la probabilidad de acierto aumenta. Es decir, si un usuario se descarga un contenido es muy probable que dicho contenido sea solicitado por otro usuario posteriormente; si se ha almacenado el contenido en el *proxy cache*, todo el tráfico de red permanece en la LAN en sucesivas peticiones.

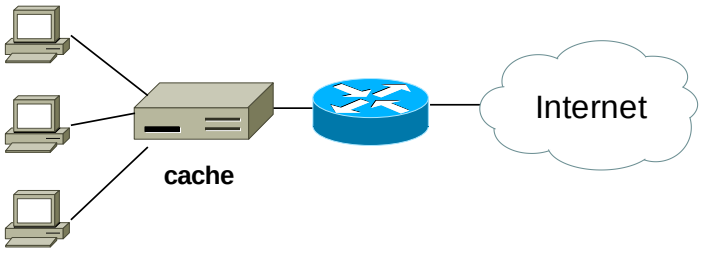


Figura 23. Configuración proxy-caché autónoma.

La única desventaja en esta configuración es que los navegadores de los clientes también deben ser configurados explícitamente para esta función.

El *caching* transparente elimina este contratiempo al interceptar las peticiones de los clientes en un *router* (véase Figura 24) o en un *conmutador* de nivel 4 (véase Figura 25) y redirigirlas a un servidor *caché*, que puede fácilmente escalar en forma de clúster. Un servidor *caché* se considera semánticamente transparente cuando el cliente recibe exactamente el mismo contenido que hubiera recibido al haber contactado con el servidor origen, con la única diferencia de saltos atravesados en la red por los paquetes, así como del tiempo de respuesta.

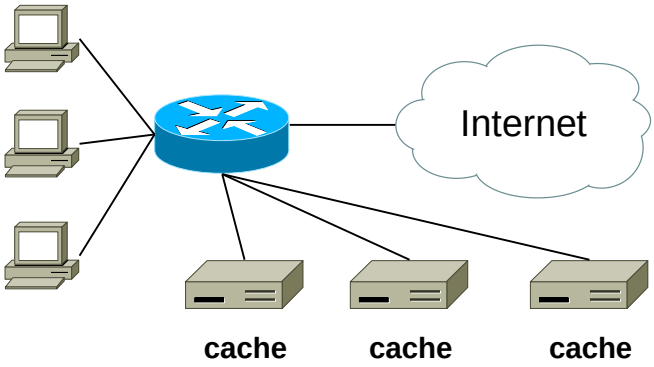


Figura 24. Configuración proxy-caché transparente (con router)

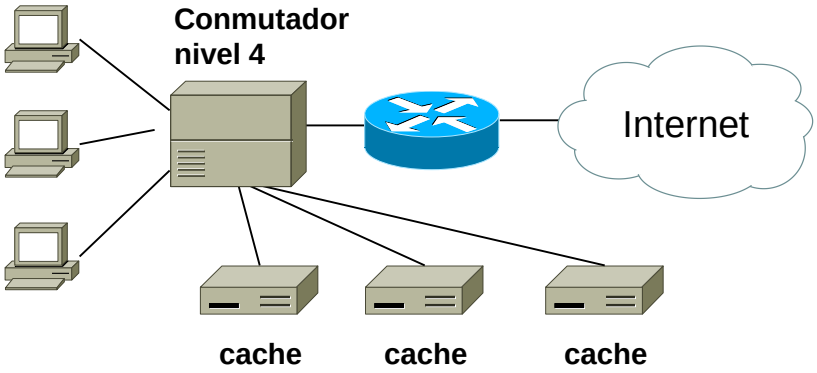


Figura 25. Configuración proxy-caché transparente (con conmutador)

En la actualidad, existen dos aspectos destacados que son objeto de investigación en un *forward proxy*: generación (y gestión) de contenido dinámico y seguridad. El aspecto de la seguridad no se abordará en este capítulo debido a su complejidad y a que no es objeto de estudio en esta tesis. No obstante, se puede consultar abundante bibliografía en este sentido en [Zho_08] [Sun_09] [Fan_11].

En referencia a la generación de contenido dinámico, una técnica relativamente reciente para estos tipos de *proxy* es el *caching* a nivel de fragmento (*fragment level caching*) [Yu_06]. Esta técnica permite un *caching* con un nivel de granularidad más fino que el tradicional *caching* a nivel de página (*page level caching*). Proporcionar este tipo de *caching* dinámico supone ciertos retos, como son:

- Escalabilidad para soportar el aumento de carga.

- Flexibilidad para no limitar la distribución de todo tipo de contenido dinámico.
- Independencia para ser capaz de realizar caching sobre servidores independientes.
- Beneficioso para que su uso mejore el rendimiento en la generación de contenido dinámico.

Una de las primeras técnicas propuestas en este sentido es el *caching* activo (*active caching*) [Tam_98] [Du_02] [Kim_09]. El mayor problema de este mecanismo es que la reutilización de fragmentos impone una solución de compromiso con latencias de acceso de usuario mayores. Según [Dat_01], la carga de computación y el consumo de memoria en el (*forward*) *proxy* conducen a una latencia de usuario más alta. Adicionalmente, se requieren cuatro flujos de datos para obtener la página completa en el peor caso: en primer lugar, el *proxy* envía una petición al servidor, quien le envía el diseño (*layout*) de la página; a continuación, el *proxy* solicita al servidor los fragmentos de los que no dispone; finalmente, el servidor envía dichos fragmentos.

ESI (*Edge Side Includes*) es un lenguaje de marcado relativamente simple empleado para definir componentes de una página web para el ensamblaje dinámico o la distribución de aplicaciones web en los extremos de Internet, por ejemplo es empleado por el operador de CDN Akamai. ESI permite acelerar las aplicaciones web al definir un lenguaje de marcado que describe componentes *cacheables* y no *cacheables* que pueden ser agregados, ensamblados y distribuidos en los extremos de la red. Sin embargo, el esquema ESI adolece de serios inconvenientes. Al emplear ESI, se crea una plantilla (*template*) para cada página web. Esta plantilla emplea etiquetas de marcado para especificar el contenido y el diseño (*layout*) de la página. Tanto la plantilla como los fragmentos son *cacheados* en el *proxy*. Dichos fragmentos se consideran como ficheros HTML independientes. Esto implica que el diseño de la página debe ser conocido de antemano. Es por ello por lo que los sitios web que soportan diseño dinámico no pueden hacer uso de ESI. Adicionalmente, esta propuesta representa un cambio en el paradigma de diseño de páginas web tradicionales. La plantilla debe invocar *scripts* diferentes e independientes para generar fragmentos, lo que implicaría que la mayoría de los sitios web tuvieran que ser rediseñados y desplegados de nuevo. Otro problema que surge con ESI es que si se dan dependencias entre fragmentos dentro de una página, solamente es posible factorizar dicha página en fragmentos al coste de una computación redundante. En conclusión, ESI resulta adecuado cuando las páginas se pueden dividir fácilmente en fragmentos independientes y cuando el diseño de la página no cambia.

2.3.2.2. Reverse proxy

Los *proxy caches* también pueden aparecer cerca de los servidores origen. En dicho caso (véase Figura 26), se les suele denominar *reverse (proxy) caches*, y su propósito no es reducir el tráfico de red, sino la sobrecarga en el servidor. En este caso la configuración es transparente de cara al usuario, y el *reverse proxy* se suele ubicar 'por delante' de una granja de servidores. Todo el tráfico de los clientes es recogido por el *reverse proxy*, quien devuelve una respuesta si el contenido ha sido previamente almacenado y es válido; en caso contrario, selecciona uno de los servidores para obtener dicho contenido.

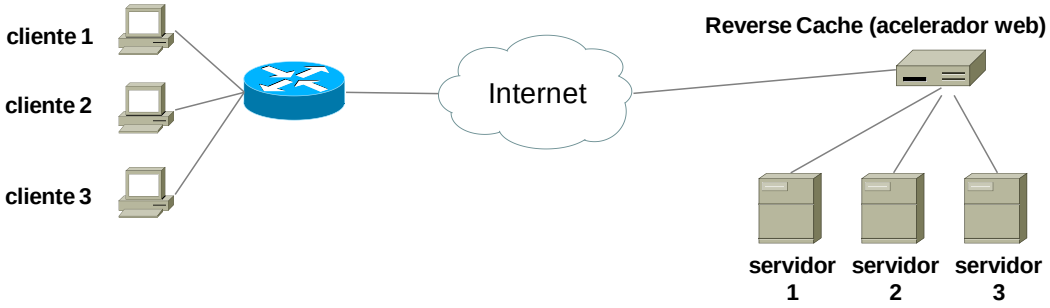


Figura 26. Configuración reverse proxy-caché

Una configuración aún más general estaría constituida por un conjunto de nodos interconectados a través de un balanceador de carga (*load balancer*) conformando un *reverse proxy*, de tal forma que (al ser un esquema más escalable) se puede dar soporte a uno o varios sitios web simultáneamente. De esta forma, dependiendo de la dirección IP del sitio web asociado, el balanceador de carga dirigirá la petición a un nodo en concreto del *reverse proxy*. Cada nodo dispone de dos niveles de almacenamiento (RAM y disco) y se encuentra conectado directamente a Internet. Cada uno de estos nodos *cache* puede ser asignado a un único *sitio web* (*asignación exclusiva*) o a varios sitios web (*asignación compartida*). Por otro lado, la asignación de un nodo a un *sitio web* puede ser tanto *estática* como *dinámica*, dependiendo de las condiciones de carga. De manera análoga, en el caso de asignación compartida, la asignación de objetos en RAM de un sitio web en concreto también puede ser estática o dinámica. Un estudio de estas configuraciones y su rendimiento se puede encontrar en [Cic_03].

De forma similar al *forward proxy*, el aspecto de la seguridad en el *reverse proxy* también representa un *hot topic*. Se puede consultar algunos estudios relativos en [Hua_11] [Aya_11] [Puj_09], aunque cabe mencionar que los mecanismos de seguridad son esencialmente aplicables tanto a *forward* como *reverse proxy*.

La configuración en *reverse proxy*, o acelerador web, es la más relacionada con las CDNs, ya que suele ser la configuración tradicional en este tipo de redes. Existen numerosos productos en la actualidad relacionados con la aceleración web, como son FastSoft [WWW_Fas], BIG-IP de F5 [WWW_F5] y AppEx LotServer [WWW_Lot], aunque los proveedores de CDNs suelen usar su propio producto.

2.3.2.3. Interception Proxy

Este tipo de configuración de *proxy* en ubicaciones intermedias de la red es típica de los proveedores de acceso a Internet (ISPs), con la finalidad de reducir o minimizar el tráfico WAN y reducir el tiempo de respuesta. Téngase en cuenta que debido a los convenios de *peering* entre ISPs y *carriers* esto también puede redundar en una reducción de costes.

En esta configuración es necesario el empleo de un *conmutador o switch web*, aunque algunos *routers* son capaces de redirigir tráfico a un proxy de interceptación. Este *conmutador web* es el encargado de redirigir el tráfico web al *proxy cache*, y se ubica en la LAN entre el servidor de acceso remoto (*RAS*) y el *router* de salida, como se observa en la Figura 27. El *conmutador web* intercepta el tráfico dirigido al puerto 80 y se lo envía al *proxy caché*.

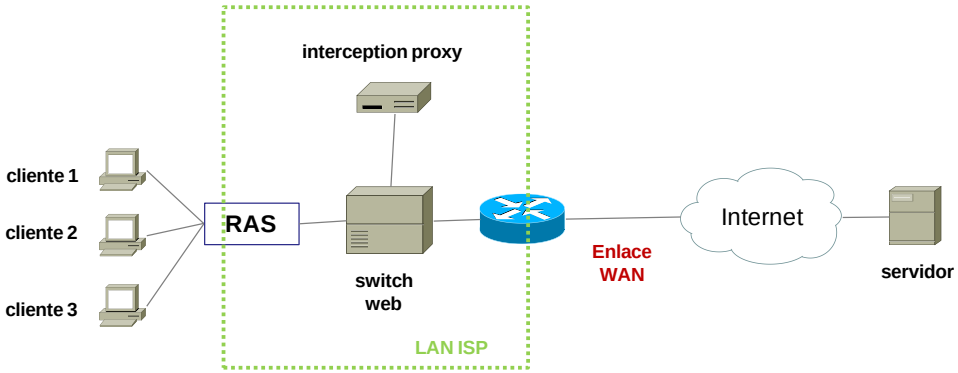


Figura 27. Configuración interception proxy-caché

2.3.3. Evolución del caching. Redes de caches.

El objetivo principal de la interconexión de *cachés* consiste en obtener un contenido por parte de un *proxy caché* (cuando se produce un *cache miss*) desde otro *proxy caché* cercano en lugar de contactar directamente con el servidor origen. Si la asunción es correcta (el contenido se encuentra efectivamente en un *proxy caché* cercano), entonces el retardo es menor. Por el contrario, si la asunción es errónea, el retardo es incluso mayor que si se hubiera contactado directamente con el servidor origen. Es por ello por lo que se hace necesaria la introducción de mecanismos y protocolos de gestión de *caching* eficientes entre *proxy cachés* interconectados en red.

Existen dos configuraciones básicas en la interconexión de *cachés*:

- ***Cachés jerárquicas (hierarchical caching)***: en este tipo de configuración un *proxy caché* solicita un contenido (no disponible localmente) a un *proxy caché* de mayor nivel. Si éste tampoco dispone del contenido lo solicita al *proxy caché* de mayor nivel y así sucesivamente. Por ejemplo, un *proxy caché* de una red

local podría contactar con un *proxy caché* de una red regional y éste con un *proxy caché* de una red nacional. Esta configuración incrementa la probabilidad de obtener un *cache hit* sin contactar con el servidor origen, pero aumenta el tiempo de servir una petición de un cliente. Por otro lado, tampoco se aprovecha la posibilidad de contactar con *proxy cachés* del mismo nivel (*peer caches*), ya que la comunicación sólo se establece con el nivel superior (*parent cache*).

A pesar de demostrarse la mejora del rendimiento en estructuras de *cachés* de dos niveles [Che_02], también se ha podido comprobar que *proxy caché* padres de alto nivel pueden sobrecargarse fácilmente si no se emplean técnicas de escalabilidad locales, como las descritas en el capítulo 2.2. Por otro lado, téngase en cuenta que esta solución no optimiza el almacenamiento de contenido, al haber múltiples copias en diferentes niveles de *caché*.

- ***Cachés en redes parcialmente malladas (intercache communication)***: esta configuración apareció como una alternativa al *caché* jerárquico y parte de un escenario distribuido, permitiendo a las *cachés* preguntarse las unas a las otras mediante diferentes protocolos, como: ICP, cache digests, CRP, CARP (propietario de *Microsoft*) y WCCP (propietario de *Cisco*). Cada servidor caché almacena información de metadatos para decidir quién dispone del contenido. Para una mayor escalabilidad, estos metadatos (pero no el contenido) pueden ser almacenados de forma jerárquica. Actualmente, la comunicación se lleva a cabo mediante HTTP.

El protocolo ICP (Internet Cache Protocol) [Wes_97] [Wes_97b] se diseñó como un mecanismo rápido para descubrir que *proxy caché* (web) disponía de un cierto objeto almacenado. De esta forma, los *proxy cachés* intercambian mensajes, en una primera fase, para conocer si algún *proxy caché* dispone de un objeto. En la segunda fase se envía un mensaje HTTP a dicho *proxy caché* para obtener el objeto. La primera fase se implementa mediante UDP para minimizar el retardo y el tráfico.

Aunque no se indica de manera explícita por el protocolo ICP, la mayoría de los *proxy cachés* que implementan dicho protocolo típicamente incorporan varias funcionalidades adicionales. De esta forma, los *proxy cachés* pueden ser identificados mediante padres (*parents*) o hermanos (*siblings*). En cierto sentido, se establece una estructura jerárquica entre padres y gemelos, aunque es configurable y no guarda necesariamente relación con la topología de la red. En la Figura 28 se ilustra el funcionamiento básico del protocolo ICP. El cliente envía una petición HTTP a la *caché local* (*local web cache*) para obtener un objeto determinado. Mediante ICP, la cache local envía un mensaje a cada uno de los *proxy cachés* configurados como hermanados (*siblings).* Si ninguno dispone del contenido, la cache local encamina la solicitud del objeto a un *proxy caché* configurado como padre (*parent*). Una última configuración posible consiste en hacer la solicitud directamente al servidor origen, sin contactar con los *proxy cachés* padres.

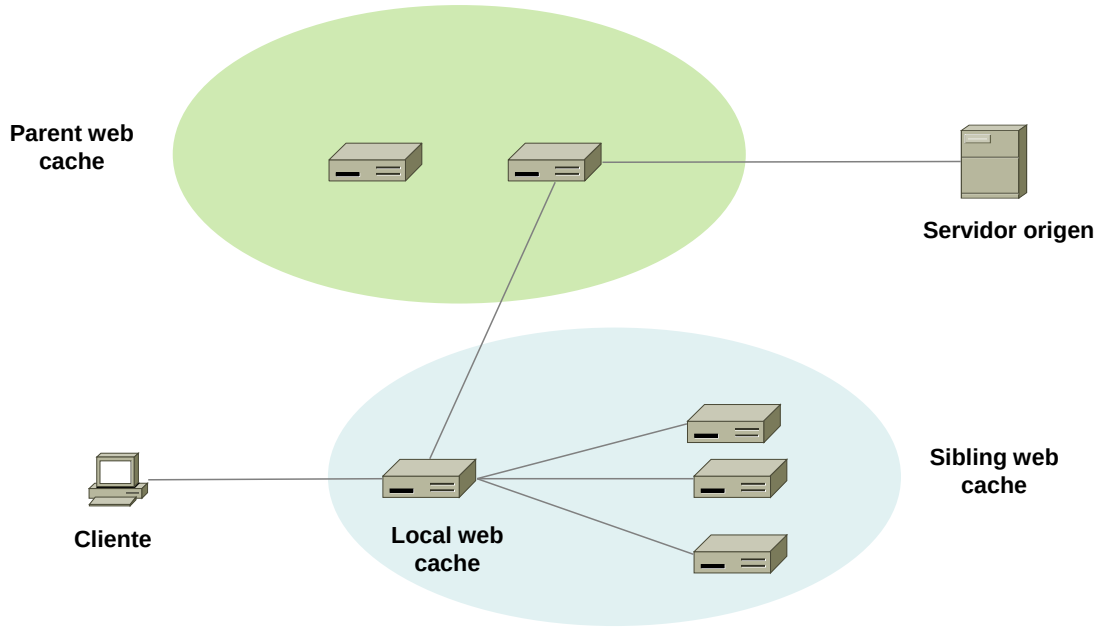


Figura 28. Funcionamiento del protocolo ICP.

No se puede demostrar de manera rotunda que el uso de ICP u otra configuración de *proxy caches* interconectados resulte en unos tiempos de respuesta menores o se reduzca el tráfico de red. En la mayor parte de las situaciones son necesarios intensos estudios de tráfico y experimentación de medidas para obtener la mejor configuración.

El protocolo ICP se desarrolló en el NLANR (*Nacional Laboratory for Applied Network Research*) [WWW_NLA], y se trata de un protocolo sencillo que permite a los servidores caché el intercambio de información acerca de la disponibilidad de objetos con el propósito de disponer del conocimiento acerca de dónde está localizado el contenido. Como el intercambio debe ser rápido, ICP está implementado sobre UDP en vez de TCP. Lamentablemente, usa UDP unicast, lo que supone serios inconvenientes en cuanto a escalabilidad conforme aumenta el número de cachés. No obstante, algunos *proxy caché* como Squid [WWW_Squ] permiten el uso de UDP multicast. ICP proporciona también una cierta indicación de las condiciones de la red: si un mensaje de solicitud ICP no es respondido en un breve intervalo de tiempo (uno o dos segundos), se asume que la red está congestionada o que la *caché* destino no está disponible.

Para solucionar el problema de la escalabilidad de ICP se han propuesto varias alternativas. El protocolo CARP (*Cache Array Routing Protocol*) [Val_98] divide el espacio de URLs entre los servidores *proxy caché*. El beneficio de esta opción es la ausencia de sobrecarga de red durante las peticiones de documentos no existentes (*cache miss*) a los *proxy cachés* vecinos. Obviamente, esta parece una opción adecuada para una granja de *cachés* (*cache farm*) o un clúster de *cachés* homogéneos, pero no es apropiado para un esquema de *caché* compartida en una WAN, que está caracterizada por un clúster de *proxy caches* no uniformes.

Cache Digest [Wes_01] es tanto un protocolo como un formato de datos desarrollado por el NLANR para paliar ciertos problemas de latencia y congestión asociados a ICP y otros protocolos. Como formato de datos, un *cache digest* dispone del contenido de un servidor *caché* en una estructura compacta, de forma que, dada una URL representando al objeto solicitado, se le aplica una función *hash* (en realidad se emplean filtros de *Bloom* [Jin_09] en el algoritmo) que emplea este formato de datos para conocer si el objeto se encuentra en la *caché* o no. Como protocolo sobre UDP, permite el intercambio de estructuras *cache digest* entre servidores *caché* en forma de *peering* sin la necesidad de la interacción del cliente. De esta forma, se reduce la latencia y se mejora la utilización de la red con respecto a ICP. Sin embargo, ICP es un protocolo sin estado (*stateless*) mientras que *cache digest* requiere una cantidad moderada de memoria, además de mayor sobrecarga debido al cálculo de las funciones *hash*.

Prácticamente al mismo tiempo en que apareció *cache digest*, se desarrolló otro protocolo muy parecido denominado *summary cache* [Fan_98]. Este algoritmo representa de manera compacta el directorio de cache, y lo disemina periódicamente entre el resto de *proxy cachés*. En comparación con ICP, se produce un ahorro de ancho de banda en términos de bytes transferidos de un 50% en *summary cache* [Fan_98] y alrededor de un 40% para *cache digest* [Rou_98].

Las técnicas de diseminación de información de *caching* basada en directorios, la representación eficiente, así como el uso de los filtros de Bloom supusieron un gran avance en el rendimiento de los sistemas de caching a finales de la década de los 90. Nótese que fue en esta época cuando surgen las CDNs, que se ven fuertemente influencias por las técnicas de *caching* y viceversa, como se describirá en mayor detalle en el capítulo 2.5 dedicado enteramente a las CDNs. Por ejemplo, se puede mencionar que un *proxy caché* ubicado en una posición intermedia entre cliente y servidor (nótese que sería la situación de una CDN) no es adecuado para filtrar todas las peticiones, ya que puede introducir excesiva carga. Algunas funcionalidades, como identificar si un objeto es *cacheable* o no, o tomar decisiones de cooperación de *caching* se pueden separar y asignar a otro dispositivo de red. En [Niu_09] se sugiere el empleo de *switches* de nivel 5 (L5) para diseminar los directorios de cache proponiendo el protocolo RCD (*Reduced Cache Digest*).

2.3.4. Técnicas de caching para contenido multimedia

Como se ha descrito en el capítulo 2.3.1, las técnicas tradicionales de *caching* no son válidas para contenido que atraviesa la red de comunicación entre cliente y servidor en formato streaming. Ello es debido a que los flujos (*streams*) ocupan un tamaño considerable, pueden parar y reanudarse en cualquier momento, suelen estar caracterizados por una tasa de datos concreta, pueden combinar múltiples orígenes y alcanzar múltiples destinos. Algunos ejemplos de este tipo de contenido lo constituyen programas de audio, de vídeo y streams de datos en tiempo real, como las aplicaciones de monitorización telemétricas.

Existen protocolos diseñados para acometer los requisitos de streaming media, como:

- ***RTP (Real Time Protocol)***: se encuentra documentado en la RFC 1889 [Sch_96] y es un protocolo de transporte que proporciona envío de datos extremo a extremo para aplicaciones que transmitan datos en formato streaming.

- ***RTCP (Real Time Control Protocol)***: tiene como misión principal enviar periódicamente información de control a todos los participantes de una sesión RTP. El objetivo principal es proporcionar *feedback* sobre la calidad de los datos transmitidos.

- ***RTSP (Real Time Streaming Protocol)***: mientras que RTP y RTCP proporcionan un transporte eficiente de streams multimedia, la funcionalidad de comenzar, parar, reanudar, avanzar y retroceder un flujo multimedia es llevada a cabo por el protocolo RTSP, descrito en la RFC 2326 [Sch_98].

- ***SMIL (Synchronized Multimedia Integration Language)***: es un lenguaje de marcas basado en XML (estándar W3C) que proporciona capacidades de distribuir contenido multimedia en pantalla y su secuenciación temporal, especialmente útil en presentaciones multimedia.

- ***Protocolos propietarios***: pese a que el uso de protocolos estándar proporcionaría un mayor desarrollo de la tecnología streaming, las compañías principales en este ámbito empezaron con protocolos propietarios. La empresa *RealNetworks* definió PNA (*Progressive Network Architecture*) en lugar de RTSP, así como RDT (*Real Data Transport Protocol*) en lugar de RTP. Microsoft, por otra parte, hacía uso de MMS (*Microsoft Media Server*). Actualmente la tendencia ha sido migrar hacia protocolos estándar, pero con ciertas extensiones propietarias.

Algunos de estos protocolos serán empleados en el modelo de simulación y en la implementación real de la CDN, como RTP, RTCP y RTSP.

A continuación se describirán las técnicas de caching básicas para contenido multimedia, como son *audio/video smoothing*, *fast prefix transfer*, *object segmentation*, *cache replacement* y *dynamic caching*, pero previamente conviene definir el concepto de streaming y sus formatos.

2.3.4.1. Concepto de streaming

Streaming hace referencia a todo tipo de medio que tiene requisitos temporales y emplea un flujo de datos continuo, como típicamente son las transmisiones de audio y vídeo. La reproducción del flujo empieza mientras dicho flujo está siendo recibido, lo cual es diferente a descargarse un fichero y, posteriormente, reproducirlo. También es distinto de recibir una secuencia de datos de algún sistema de monitorización, puesto que aunque hay un envío periódico de datos, no se requiere una sincronización estricta para la reproducción. En el caso de streaming, existen dos formatos básicos:

- ***Live data***: el contenido no suele estar previamente almacenado. Algunos ejemplos son las aplicaciones como videoconferencia, audioconferencia o juegos en red multi-jugador, que sólo pueden tolerar un retardo mínimo para soportar de manera eficiente el flujo de las sesiones interactivas.
- ***On-demand***: el contenido sí está previamente almacenado. Algunos ejemplos lo constituyen las películas sobre Internet, archivos de noticas, etc. En este caso, si bien se pueden tolerar retardos moderados y un comienzo retardado, no se toleran saltos o un excesivo *jitter* una vez el flujo ha comenzado.

Multicast parece la mejor solución para distribuir contenido en formato streaming a un gran número de usuarios. Sin embargo, IP multicast no está disponible de forma nativa en Internet. Se produjeron algunos intentos de introducir multicast en Internet a principios de los 90, como es MBONE (*Internet Multicast Backbone*) [Sav_98] que trataba de conectar islas de redes con multicast habilitado a través de túneles entre routers unicast. Inicialmente, el proyecto MBONE tuvo un cierto éxito, y en 1996 el sistema creció hasta incluir 3000 subredes [Dee_95]. A partir de ese momento el interés comenzó a decaer debido a varias limitaciones: falta de un modelo de ingresos, capacidad de sesiones limitada, soporte administrativo limitado, así como una organización informal de la administración de la red.

Evidentemente, los modelos de ingresos y modelos de negocio son los que potencian los servicios. Es por ello por lo que, actualmente, el mundo del streaming media está ampliamente dominado por software comercial, incluyendo a *Real Networks RealOne player* y el software de streaming *Helix* [WWW_Real], *Microsoft Windows Media Player* y *Windows Media Services* [WWW_Win], *Apple QuickTime Broadcaster*, *QuickTime Streaming Server* y *QuickTime* [WWW_Quick] y *Flash Player* y su familiar de servidores *Flash Media Server* [WWW_Flash].

2.3.4.2. Suavizado de audio y video (audio/video smoothing)

En cierto sentido, puede decirse que los bits de datos llegan a ráfagas (*burst*), y el contenido multimedia se dice que es inherentemente *bursty*, incluso en la misma fuente. La codificación de tramas (*frames*) de un contenido multimedia revela que el tamaño de las tramas suele ser variable [Leg_91]. Sin embargo, en recepción, las tramas se muestran a una tasa constante, y la variabilidad del tamaño de las tramas se refleja en una variabilidad del ancho de banda requerido [Kru_95].

Los canales de comunicación añaden retardo, pérdidas de paquetes y congestión, lo que hace que el efecto de flujo de ráfaga se vea incrementado. Si las tramas de video se visualizaran conforme llegan, la variabilidad sería perceptible como un *jitter* particularmente molesto entre tramas.

Para ocultar este *jitter* al usuario, los clientes multimedia típicamente incorporan un *buffer* de reproducción, que realiza las funciones de un almacén elástico. Aunque se produce un cierto retardo inicial de algunos segundos, es posible reproducir las tramas a una tasa constante de tal forma que el usuario puede ver el video de una manera suave y fluida (*smoothly*). Esta técnica es la que se denomina *audio/video smoothing*. Si el flujo multimedia que llega sobre la red es particularmente variable, el *buffer* se puede vaciar, lo que implicaría pausar la reproducción hasta que dicho *buffer* vuelva a llenarse.

La Figura 29 muestra los dos tipos de retardo que tienen lugar desde que se solicita un flujo multimedia hasta que finalmente empieza a reproducirse. El retardo de conexión o establecimiento (*connection delay*) es el intervalo de tiempo desde que se solicita el flujo hasta que se recibe la primera respuesta por parte del servidor. El valor de este retardo depende del tiempo de respuesta del servidor y de las características de la red de comunicación (número de saltos, posible congestión, etc.). Tras el retardo de conexión, el *buffer* del cliente empieza a llenarse hasta un momento en que el flujo se puede empezar a reproducir con ciertas garantías de calidad. Este tiempo recibe el nombre de retardo de *buffer* (*buffer delay*) y su valor es típicamente de algunos segundos, aunque depende del reproductor multimedia seleccionado, e incluso si se usa algún tipo de protocolo de control (RTCP) su valor puede variar durante una misma sesión de streaming.

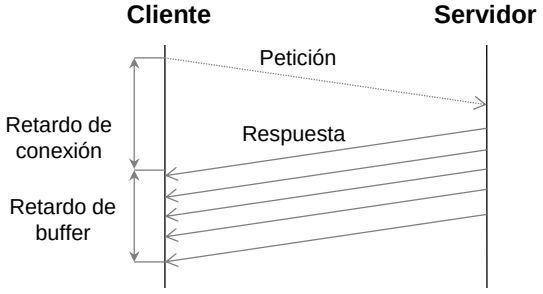


Figura 29. Audio/video smoothing.

Las técnicas de *smoothing* no sólo se han considerado en el lado del cliente, sino también en la propia red de comunicaciones, ya que es más sencillo controlar un flujo con tasa constante (similar a un flujo CBR, *Constant Bit Rate*) que un flujo *bursty* o con tasa variable (similar a un flujo VBR, *Variable Bit Rate*). El consumo de recursos es menor [Zha_97], y los mecanismo de admisión y reserva de recursos se simplifican [Sen_97].

La técnica PCRTT (*Piecewise Constant Rate Transmision and Transport*) ha demostrado reducir la tasa de pico y la desviación típica en un 70-80% si se emplea un *buffer* de cliente de 4 MB [Fen_97]. Una mejora sobre esta técnica la constituye e-PCRTT, propuesta en [Had_98], y básicamente consiste en imponer requisitos locales en el ancho de banda de la comunicación en lugar de requisitos globales. Un análisis sobre esta técnica puede verse en [Had_00].

2.3.4.3. Transferencia anticipada de prefijos (fast prefix transfer)

A diferencia de un canal convencional de TV, donde los usuarios son capaces de conmutar entre canales a una velocidad instantánea, el streaming sobre Internet adolece de retardos significativos, fundamentalmente causados por el retardo de *buffer*, además del retardo de conexión, como se explicó en la sección 2.3.4.2. El problema a abordar es como reducir estos retardos para proporcionar al usuario un sistema más *responsivo*.

Si situamos un *proxy caché* entre cliente y servidor, y suponiendo que dicho *proxy caché* dispone del contenido solicitado (al menos el principio, por cuestiones de tamaño), éste puede enviar el principio del flujo multimedia al cliente, mientras que solicita el resto al servidor. Esta es la esencia de los mecanismos de *prefix transfer* [Sen_99]. Estos mecanismos son capaces de reducir el retardo de conexión, pero no el retardo de *buffer* (véase Figura 29). Es por ello por lo que se desarrollaron los mecanismos de *fast prefix transfer*.

A partir de la Figura 29 se puede observar que si se almacena menos contenido en el *buffer* del cliente el retardo disminuye. Sin embargo, esto no es una buena opción porque se requiere una cierta cantidad de datos para absorber los retardos de los paquetes, así como sus variaciones (*jitter*). Así pues, la única alternativa viable es llenar el *buffer* del cliente más rápidamente, lo que implica en transmitir los datos a una tasa mayor. Esta es la idea básica de *fast prefix transfer*.

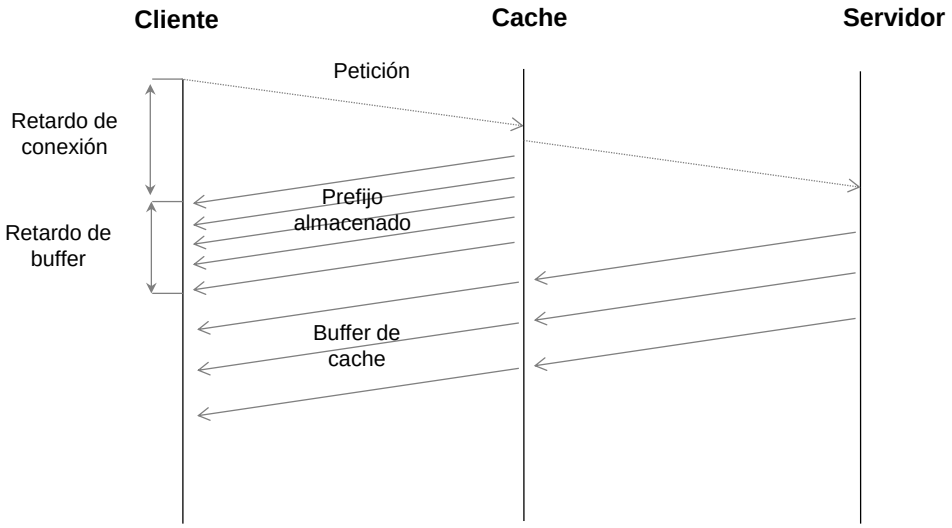


Figura 30. Fast prefix caching

En la Figura 30 se puede apreciar el funcionamiento y cómo se reduce el retardo:

- El cliente contacta con el *proxy caché*, quien inmediatamente empieza a enviarle un prefijo (comienzo) del objeto multimedia que tiene previamente almacenado. Nótese que cuanto más cercano esté el *proxy caché* al cliente, menor será el retardo de conexión. Por otro lado, el *proxy caché* sirve el contenido con una tasa mayor que la tasa de reproducción, por lo que se reduce el retardo de *buffer*.
- Simultáneamente, el *proxy cache* envía una petición al servidor para obtener el resto del objeto multimedia.
- Conforme le llega el objeto multimedia del servidor, el *proxy caché* lo almacena en su *buffer* de *cache* (normalmente distinto del *buffer* donde se guardan los prefijos) y lo envía al cliente.

Los protocolos de streaming incluyen funcionalidades que permiten a un *proxy caché* identificar y almacenar las tramas iniciales conformando el prefijo y, después, solicitar el resto de tramas. El protocolo RTP incluye números de secuencia y marcas temporales que permiten identificar el prefijo. La cabecera *Range request* de HTTP 1.1 también proporciona una funcionalidad similar. El protocolo RTSP soporta además posicionamiento absoluto (*seeking*).

2.3.4.4. Segmentación de objetos y reemplazo

El tamaño del prefijo descrito en la sección anterior debe elegirse adecuadamente. La Figura 31 ilustra la importancia de segmentar un flujo en tamaños que balancean retardo y tráfico del servidor con la capacidad de almacenamiento *caché*. Sin segmentación, las limitaciones de capacidad provocan que el objeto A sea eliminado de la *cache* tras recibir el objeto B. Si se realiza segmentación, una parte del objeto A puede permanecer en *caché* tras recibir el objeto B.

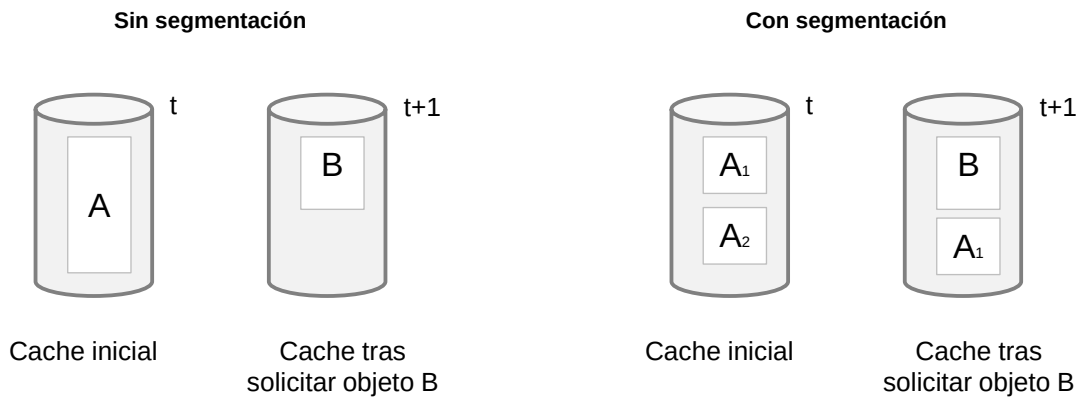


Figura 31. Segmentación de objetos streaming.

Por otro lado, además de segmentar es importante conocer cómo segmentar. El almacenamiento en disco se estructura en torno a bloques de un tamaño particular fijo. Si los flujos multimedia se segmentan en secciones cuyo tamaño es igual a los bloques de disco, entonces el almacenamiento, la obtención y el reemplazo en una *cache local* es muy eficiente. En principio, los segmentos de un objeto multimedia se podrían almacenar en *caché* y remplazar de forma independiente, sin pérdida aparente de eficiencia, además de reducir la contención en el acceso a disco [Hoff_00].

Sin embargo, existe una desventaja en el tratamiento independiente de segmentos. Cuando una petición llega a un *proxy caché,* es muy posible que sólo una porción de los segmentos que conforman el objeto multimedia se encuentren almacenados. Servir todo el flujo supondría pues realizar peticiones al servidor origen (o a otras *cachés* en modo cooperativo). Esto aumenta, por un lado, la señalización y el tráfico. Adicionalmente, se incrementa la probabilidad de perder cierta sincronización en el flujo multimedia. Es por ello por lo que se hace necesario un mecanismo para controlar el número de *gaps* (saltos entre dos segmentos adyacentes). Incrementar el tamaño del segmento reduce el número de *gaps*, pero tiene como contrapartida que aumenta la contención de acceso a disco. Lo que se requiere es una unidad lógica de gran tamaño para el *caching*, mientras se conserva una granularidad fina para la reserva en disco y su reemplazo.

La Figura 32 ilustra una solución efectiva. El objeto multimedia se segmenta en unidades lógicas denominadas *chunks* que son *cacheadas*. Cada uno de estos *chunks* se subdivide a su vez en segmentos correspondientes al tamaño de los bloques de disco. Un *chunk* no es más que un número de segmentos consecutivos correspondientes a un objeto multimedia; siendo éste un concepto ampliamente utilizado en la Ingeniería Telemática y especialmente en la distribución de contenidos, por ejemplo en P2P y en comunicaciones multicast fiables.

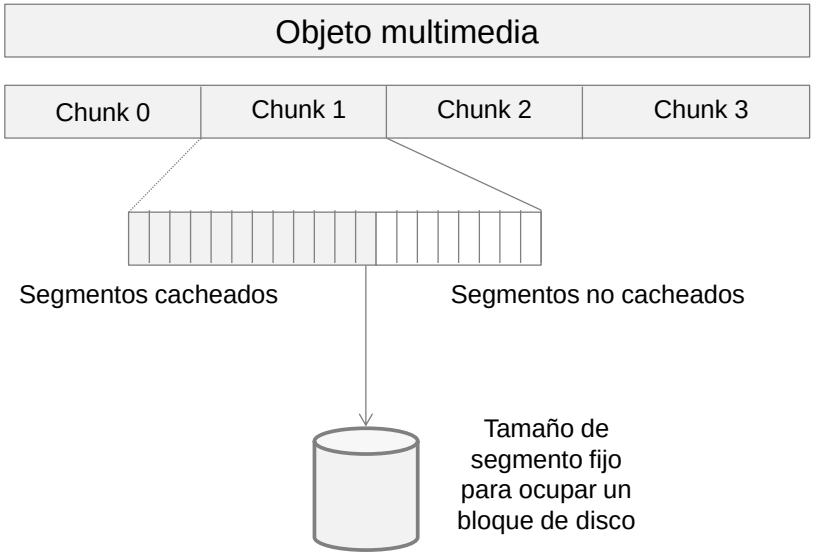


Figura 32. Segmentación y reemplazo en caching

Cada *chunk* es *cacheado* de forma independiente y se utilizan las siguientes reglas como política de reemplazo o sustitución:

- La unidad básica de caching y reemplazo es un segmento, lo que optimiza el almacenamiento en disco.
- Los segmentos reservados (*cacheados*) de un *chunk* siempre forman un prefijo inicial de dicho chunk (véase Figura 32). Esto tiene en cuenta las ventajas del *prefix caching* mencionadas en el subcapítulo anterior.
- Cuando se accede a un segmento de un *chunk*, no se puede descartar ningún otro segmento de dicho *chunk*. Esto obedece al principio de continuidad típico en el acceso a un flujo multimedia.
- Cuando el algoritmo de sustitución selecciona un *chunk* para descartar, entonces se empieza por descartar el último segmento del prefijo *cacheado* de dicho *chunk*. De esta forma, se preserva el prefijo tanto como es posible, aunque se reduzca.

Seleccionar el tamaño adecuado del *chunk* debe ser una solución de compromiso entre reducir el número de *gaps* y aumentar la flexibilidad en el reemplazo de segmentos. Si el tamaño del *chunk* es muy grande, entonces todo el objeto multimedia cabría en un solo *chunk*, y resulta en la técnica de *prefix caching* mencionada anteriormente. Si el tamaño del *chunk* es muy pequeño, entonces contendrá muy pocos segmentos, y la continuidad del flujo puede perderse al aumentar el número de *gaps*. Para más detalles de estas técnicas, incluyendo pruebas de rendimiento, se puede consultar [Hoff_00]. Otros estudios se pueden consultar en [Wu_04] [Par_08] [Dev_12].

2.3.4.5. Caching dinámico (dynamic caching)

La naturaleza de streaming de los objetos multimedia proporciona también otras oportunidades para preservar ancho de banda. Este es el caso de dos clientes (C_1, C_2) solicitando el mismo flujo multimedia pero con una *distancia temporal* Δ. La idea básica del caching dinámico consiste en utilizar un solo flujo entre *proxy caché* y servidor para satisfacer ambas peticiones (o incluso más peticiones que tengan lugar dentro de una *distancia temporal* Δ).

El mecanismo es muy sencillo y consiste en disponer de un buffer en anillo (*ring buffer*) que oculte o absorba esta distancia temporal entre peticiones. Cuando el segundo cliente C_2 solicita el mismo flujo multimedia, el primer cliente C_1 ya ha recibido los primeros Δ segundos (supongamos por sencillez que la tasa de envío es la misma que la de reproducción, de lo contrario el *buffer* tendrá que adaptarse a este hecho), que pueden reutilizarse para servir al segundo cliente. El segundo cliente sólo requiere obtener los primeros Δ segundos de datos que '*ha perdido'* al unirse al flujo más tarde. Este intervalo del flujo se denomina *patch* y puede obtenerse directamente del servidor, o se puede almacenar en el *proxy caché* como prefijo.

El *buffer* en anillo y el mecanismo de *caching* conformando el *caching* dinámico se puede emplear en varias ubicaciones de la red. Algunos estudios pueden encontrarse en [Has_09] [Siv_10].

2.4. Réplicas o mirrors

La carga y descarga de páginas web puede sufrir a menudo un retardo indeseado. Si el contenido es duplicado en múltiples servidores alrededor del mundo, los clientes podrán conectarse a uno cercano para reducir la latencia en el acceso. Un servidor que replica el contenido de otro (servidor origen) se llama espejo o réplica (*mirror*). Los clientes habitualmente interactúan de una forma no transparente con estos servidores, es decir, los clientes deben seleccionar qué servidor contactar de entre una lista que se les ofrece, normalmente eligiendo el más cercano, ya que el cliente es consciente de su propia ubicación geográfica. El uso de espejos distribuidos ha sido ampliamente empleado en aplicaciones de descarga mediante HTTP y FTP, y es también una forma de aumentar la fiabilidad, ya que basta con que, al menos, uno de los servidores esté disponible para poder proporcionar servicio a los clientes [Jam_01].

Existen algunos inconvenientes en cuanto a diseño y escalabilidad en el empleo de esta tecnología (*mirrors*). En primer lugar, una réplica está inicialmente pensada para contenido estático; aunque es posible introducir páginas dinámicas, la gestión de la consistencia conduce a una complejidad en la arquitectura. En segundo lugar, los clientes deben interactuar para decidir con qué servidor contactar. Pese a que la interacción transparente resulta más elegante, esto no supone el mayor problema, sino la falta de capacidad de balanceo de carga debido a que no se gestiona las decisiones de los clientes. Téngase en cuenta que si todos los clientes contactaran con el mismo servidor, éste llegará rápidamente a un estado de congestión, mientras que el resto de réplicas no serán accedidas. El efecto *flash-crowd* tampoco se soluciona con este sistema. Si un gran conjunto de clientes de la misma zona seleccionan el mismo servidor (el más cercano), también se sobrecargará. Dado que los clientes no tienen conocimiento del estado de carga de las réplicas, son incapaces de decidir si una réplica más alejada proporcionará un servicio con una menor latencia.

Una arquitectura reciente, considerada en el ámbito de las CDNs, contempla el caso de una interacción transparente en dos vías posibles:

- ***Redirección por parte de la réplica***: Si un espejo está sobrecargado, puede redirigir la petición entrante a otra réplica. Si, además, son conscientes de la carga del resto de servidores, podrán tomar una decisión más acertada.
- ***Redirección DNS***: Consiste en centralizar la decisión en la fase de resolución de DNS. El servidor DNS devuelve una lista de réplicas adecuadas a los navegadores, quienes podrán seleccionar uno de ellos de manera automática sin intervención del cliente.

Nótese que esta nueva configuración proporciona un cierto conocimiento (centralizado o distribuido) de la carga de los servidores. Información acerca del estado de la red también podría ser introducida, con lo cual sería posible incluir mecanismos de balanceo de carga. Estos mecanismos de selección con conocimiento de la red,

ubicación y estado de los servidores se integran en las CDNs, por lo que se describirán con mayor detalle en el siguiente capítulo.

En conclusión, la primera opción (no transparente) ofrece una arquitectura sencilla, aunque poco escalable, ya que en teoría, cada réplica debería ser capaz de absorber todo el tráfico; la segunda opción (transparente) implica una mayor complejidad que solventa los problemas de escalabilidad derivados.

La consistencia del contenido se consigue mediante sesiones periódicas de actualización con el servidor origen, bajo demanda –cada vez que un cliente solicita un contenido desactualizado, o mediante protocolos específicos de sincronización entre las réplicas y el servidor origen. Nótese que existe cierta similitud con las tecnologías de *caching*. La diferencia radica en que un *proxy caché* normalmente actúa como un cliente cuando solicita contenido del servidor; desde este punto de vista, el servidor es incapaz de determinar si la entidad que hace la solicitud es un navegador o un *proxy caché*. En un escenario de réplicas, los protocolos empleados entre clientes y espejos son habitualmente distintos que los empleados entre espejos y servidor origen, por lo que es posible identificar a un cliente o a una réplica desde el servidor origen.

2.5. Redes de distribución de contenido

2.5.1. Introducción

Tras analizar y describir con anterioridad las tecnologías más relevantes vinculadas a las redes de distribución de contenido, este capítulo se centrará de lleno en las CDNs, su funcionamiento, sus diferentes arquitecturas, su rendimiento y los servicios existentes en este tipo de redes.

En primer lugar, se presentarán unos conceptos básicos que servirán a modo de fundamentos. Tras describir brevemente los orígenes de las CDNs (también descrito en el capítulo 2.1.3) se describen los principales aspectos de interés científico en el campo de las CDNs, como son: la distribución de contenido, la replicación, el almacenamiento en caché, la ubicación de servidores, etc.

A continuación, se describen los modelos existentes sobre CDNs, algunos de ellos matemáticos y otros económicos. Esto permitirá establecer un vínculo con el capítulo 3, donde se presentará un nuevo modelo de CDN que permite analizar otras funcionalidades, aunque se parte de una arquitectura común. Pese a que el modelado económico (y modelos de negocio) no se aborda en esta tesis, sí que se describirá en este capítulo del estado del arte. También se describirá, por otro lado, un análisis de rendimiento de una CDN particularizado en Akamai (la CDN comercial con mayor cuota de mercado), ya que el estudio del rendimiento se abordará posteriormente en los capítulos 3 y 4 y se pueden establecer analogías.

Finalmente, la última sección de este capítulo se centra en plataformas y servicios avanzados en las CDNs, como pueden ser la generación dinámica de CDNs (o de servidores dentro de una CDN), la colaboración entre CDNs, los servicios de streaming media y la distribución de contenido a entornos móviles. De especial interés es la plataforma para servicios de streaming media, pues constituye uno de los aspectos fundamentales de esta tesis, y se abordará en el capítulo 4.

2.5.2. Conceptos básicos de CDNs

Una CDN [Pal_06] [Pen_03] [Rab_02] [Vak_03] [Ver_01] es una agrupación colaborativa de entidades de red, que abarca Internet, donde se replica contenido sobre varios servidores (denominados *surrogates* o réplicas) ubicados en las cercanías de los clientes con el fin de realizar una entrega transparente, eficaz, escalable y rápida. La colaboración entre estos nodos puede tener lugar tanto en entornos homogéneos como heterogéneos, y uno de los objetivos principales es superar las limitaciones inherentes de Internet en términos de QoS percibida por el usuario, y proporcionar servicios que

mejoren el rendimiento de la red al maximizar el ancho de banda y mejorar la accesibilidad. Las funcionalidades básicas de una CDN son:

- ***Mecanismo de redirección***, para dirigir una solicitud al *surrogate* más cercano, utilizando mecanismos para evitar la congestión y evitando de esta forma el efecto *flash-crowd* [Arl_00] o efecto *slashdot* [Adl_99].
- ***Servicios de distribución de contenido***, para replicar o *cachear* contenido desde un servidor origen a los *surrogates* dispersos en Internet.
- ***Servicios de negociación de contenido***, para cubrir los requerimientos concretos de usuarios específicos (o grupos de usuarios).
- ***Servicios de gestión***, para administrar los componentes de red, gestionar la contabilidad y monitorizar y reportar el uso y acceso al contenido.

La Figura 33 muestra un escenario típico de una CDN donde los servidores (*surrogates*) se encuentran repartidos en los extremos de las redes de Internet (dispersas por todo el mundo) a través de los cuales se conectan los usuarios finales. La CDN es la encargada de distribuir el contenido a cada uno de estos *surrogates* remotos, de tal forma que el contenido final llega al usuario de forma fiable y en unos tiempos (retardos) aceptables. El contenido puede ser replicado bajo demanda, cuando los usuarios lo solicitan, o bien con antelación, dependiendo de la estimación del tráfico y contenido demandado. Los usuarios finales, de forma transparente, terminan contactando con un *surrogate* en lugar del servidor origen, pero el contenido es servido con las mismas garantías y con un retardo menor. En el ejemplo de la Figura 33 , el *cliente 1* ubicado en América contactará típicamente con el *surrogate B* en lugar del servidor central, mientras que el *cliente 2* ubicado en Asia contactará con *el surrogate D*.

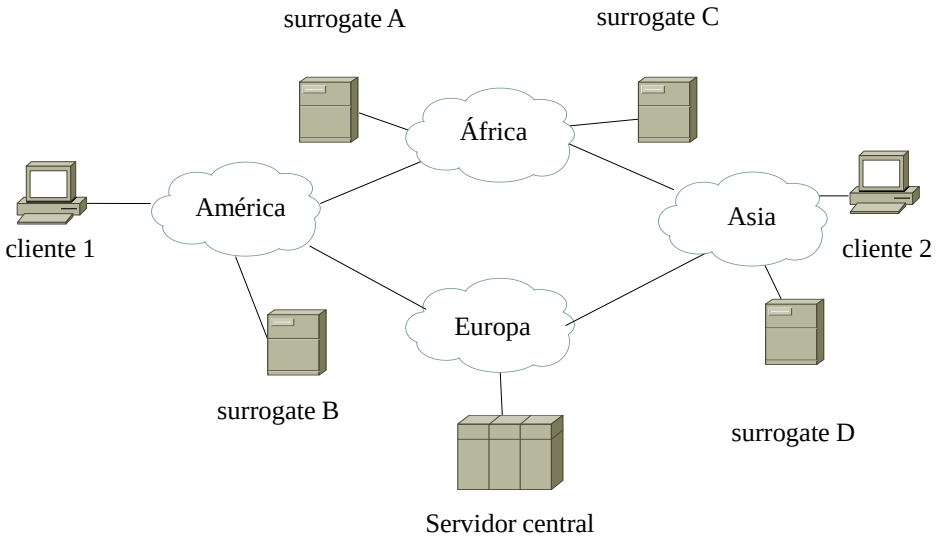


Figura 33. Esquema básico de una CDN.

La infraestructura de una CDN proporciona típicamente los siguientes servicios y funcionalidades: almacenamiento y gestión del contenido, distribución de contenido entre *surrogates* remotos, gestión de *caching*, distribución de contenido estático, dinámico y streaming, soluciones de *backup* y recuperación ante fallos, monitorización, medidas de rendimientos y reporte de uso.

Un proveedor de contenido suele contratar los servicios de un proveedor de CDN de tal forma que su contenido puede ser ubicado en sus servidores. La forma de tarificar suele ser en base al contenido distribuido a los usuarios por los *surrogates*, por lo que la CDN incorpora un mecanismo de contabilidad (*accounting*) que monitoriza el uso de la red durante la redirección, la distribución y el envío de contenido al cliente [Day_03]. Aunque se suele usar de forma indistinta, es conveniente mencionar la diferencia entre distribución y envío. La distribución (*distribution*) hace referencia a la comunicación entre servidor origen y *surrogates*, mientras que el envío (*delivery*) hace referencia a la comunicación entre *surrogate* y cliente.

El coste medio de los servicios proporcionados por un proveedor de CDN es superior comparado con servicios de hosting [Kan_02], por lo que los clientes típicos suelen ser grandes empresas, empresas de anuncios, ISPs y operadores de móviles. Los factores más determinantes que condicionan el precio están asociados al uso del ancho de banda, la variabilidad del tráfico demandado, los *surrogates* involucrados, etc.; una descripción más detallada puede encontrarse en [Pal_06].

Como se mencionó en el capítulo 2.1.3, las CDNs evolucionaron a partir de los sistemas de *caching* y *mirrors* dentro del concepto de *content networking*. Akamai Technologies [WWW_Aka] [Dil_02] surgió como la primera compañía en proveer servicios de CDN a partir de un proyecto de investigación del MIT para solucionar o mitigar el efecto *flash-crowd* [Arl_00] [Jun_02]. La primera generación de CDNs estaba orientada fundamentalmente a contenido web estático y dinámico [Laz_01 [Vak_03], mientras que la segunda y actual generación de CDNs se centra en el video bajo demanda (VoD, *Video On Demand*), noticias bajo demanda y otros servicios de audio y video streaming. Tanto es así que Youtube [WWW_You] emplea Akamai para distribuir sus vídeos. Adicionalmente, además del video streaming, las CDNs también están orientadas a proporcionar contenido a dispositivos móviles. Esta segunda generación se encuentra aún bajo intenso estudio, y es por ello por lo que esta tesis se centra en el estudio de una CDN, en primer lugar, para posteriormente centrarse en su rendimiento cuando se incorpora contenido de streaming.

Desde el punto de vista de la estandarización se han realizado numerosas actividades. El IETF (*Internet Engineering Task Force*) como organismo oficial adoptó algunas iniciativas a través de RFCs en relación a varios proyectos de investigación en el campo de las CDNs [Bar_04] [Bar_04b] [Coo_01] [Day_03]. Además del IETF, otros organismos como el ICAP fórum [WWW_Ica] y la antigua ISMA (Internet Streaming Media Alliance) [WWW_Isma] adoptaron iniciativas para desarrollar estándares con la

finalidad de distribuir contenido de banda ancha y contenido streaming sobre Internet. Todos estos acontecimientos tuvieron lugar desde principios de 2000, cuando aparecen las primeras CDNs y su previsión de boom comercial resulta indiscutible. Sin embargo, unos años más tarde (2002), tras el estallido de la burbuja de Internet [WWW_Bur] algunos proveedores de servicios de CDN desaparecieron, y algunos ISPs aprovecharon la oportunidad para comprarlas a bajo coste para así crear su propia funcionalidad de CDN (aunque dentro de su propia red) para proporcionar servicios personalizados.

Actualmente las CDNs se pueden clasificar básicamente en dos grupos: comerciales y académicas, que se describían en el capítulo 2.5.3, aunque previamente se describirá la arquitectura básica de una CDN.

2.5.2.1. Arquitectura general de una CDN

La arquitectura general de una CDN más ampliamente descrita se encuentra en [Pen_03] y consta de siete componentes: clientes, *surrogates*, servidor origen, sistema de tarificación, sistema de encaminamiento, sistema de distribución de contenidos y sistema de contabilidad, que se ilustra en la Figura 34. Esta arquitectura es el punto de partida para los modelos de simulación e implementación que se llevarán a cabo en esta tesis.

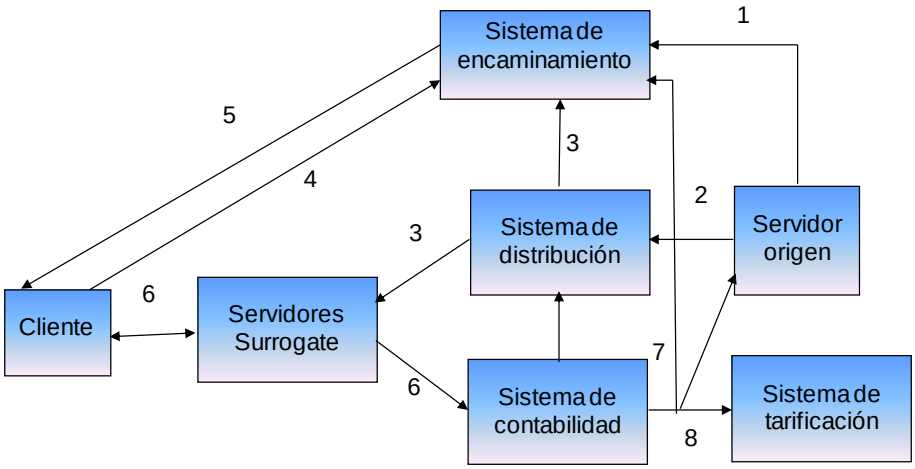


Figura 34. Arquitectura general de una CDN.

La relación entre los componentes de la Figura 34 es la siguiente:

- El servidor origen delega su espacio de nombres (URIs) al sistema de encaminamiento (1).
- El servidor origen publica contenido web que debe ser distribuido por la CDN a través del sistema de distribución (2).

- El sistema de distribución traslada el contenido a los *surrogates*. Adicionalmente, interacciona con el sistema de encaminamiento para colaborar en la selección del *surrogate* más adecuado (3).

- El cliente solicita un documento web de lo que él percibe como servidor origen, pero la solicitud es realmente redirigida al sistema de encaminamiento (4).

- El sistema de encaminamiento encamina la solicitud a un *surrogate* óptimo de la CDN (5).

- El *surrogate* seleccionado sirve el contenido al cliente, e interactúa con el sistema de contabilidad (6).

- El sistema de contabilidad procesa y sintetiza la información obtenida en estadísticas y detalles de uso por contenido, facilitándoselos al servidor origen y al sistema de tarificación o facturación. Las estadísticas también se mandan a modo de *feedback* al sistema de encaminamiento (7).

- El sistema de tarificación emplea los registros detallados de contenido para liquidar cuentas con cada una de las partes involucradas en la distribución del contenido. Téngase en cuenta que el sistema de tarificación tiene sentido en las CDNs comerciales (8).

Para los objetivos de la presente tesis, el componente más importante lo constituye el sistema de encaminamiento, que se pasa a describir a continuación con mayor detalle.

2.5.2.2. Sistema de encaminamiento

Cuando un cliente se conecta a un sitio web (por ejemplo *www.site.com*) dispone de diferentes mecanismos para seleccionar qué servidor debe contactar. En el caso de una arquitectura centralizada, supone una decisión trivial. En una CDN, por el contrario, los usuarios deben ser asociados con el contenido solicitado en un lugar adecuado, que puede ser un conjunto de decenas, centenares o incluso millares de servidores. La decisión acerca de cuál es el servidor más adecuado puede tomarse en base a diferentes métricas:

- ***Proximidad de red***: la distancia entre clientes y *surrogates* se calcula habitualmente en términos de saltos de red con una sencilla aplicación (*traceroute*). Sin embargo, los saltos de red no tienen en cuenta el tráfico de red y las posibles congestiones.

- ***Tiempo de respuesta***: el tiempo de ida y vuelta (*RTT, Round Trip Time*) medido con una aplicación omnipresente (*ping*) aporta información sobre la latencia en la respuesta de cada servidor. Sin embargo, si se toma como única métrica en la

decisión, puede conducir a decisiones de servidores desafortunadas, puesto que el tiempo de ida y vuelta es altamente variable en un entorno WAN (Internet) como es el que suele dar soporte a la infraestructura de una CDN.

- ***Carga del servidor***: existen dos mecanismos básicos de medida de la carga de los servidores; en el primero de ellos, los servidores envían información periódica a ciertos agentes, mientras que en el segundo son los agentes quienes directamente solicitan información. Existe un compromiso entre la frecuencia de solicitud para una mayor precisión y el tráfico de red en que se incurre al mandar paquetes de solicitud de información de estado.

- ***Usuarios***: los usuarios pueden ser clasificados según el contrato establecido con el proveedor. Por ejemplo, clientes especiales que hayan pagado por un mejor servicio pueden ser redirigidos a servidores diferentes que los asociados a clientes convencionales.

Todas o algunas de estas métricas son empleadas por los algoritmos de encaminamiento que se pueden clasificar en algoritmos locales y globales.

2.5.2.2.1. Encaminamiento global

El encaminamiento global asocia a cada usuario con un punto de presencia (PoP, *Point of Presence*) adecuado que contiene un acceso a un conjunto de *surrogates*. En el ejemplo de la Figura 33, un algoritmo global ha decidido encaminar al cliente 1 hacia el *surrogate* B, en base a sus métricas empleadas. Éste se encuentra ubicado en el punto de presencia más cercano al cliente 1, por lo que parece lógico que la mayor parte de las veces el cliente 1 sea encaminado a este *surrogate*, a no ser que el procesado de las métricas indique otro PoP como más adecuado. El mismo proceso tiene lugar con el cliente 2, quien la mayor parte de las veces será encaminado al *surrogate* D, correspondiente con su PoP más cercano.

De manera general, podemos distinguir varios mecanismos de redirección de clientes hacia un *surrogate*:

- ***Redirección HTTP:*** este esquema toma la decisión en el servidor origen mediante el mecanismo de redirección integrado en el protocolo HTTP, de forma que el cliente es redirigido a otra URL ubicada en un *surrogate*. Este mecanismo es considerado como poco eficiente y lento, y supone carga extra de proceso para el servidor origen. La mayor ventaja es la fina granularidad conseguida en comparación con otros mecanismos que contemplan el sitio web (*website*) entero como unidad de redirección.

- ***Redirección DNS***: el mecanismo más usado debido a su sencillez es la redirección DNS [Moc_87] [Moc_87b]. Compañías de infraestructura de red como Cisco y proveedores de CDN como Akamai [WWW_Aka] emplean este mecanismo de redirección global. El servicio DNS convierte nombres de máquinas con direcciones IP, y está basado en un espacio de nombres distribuido, donde los nombres simbólicos se agrupan de forma jerárquica en zonas autorizadas. Cada zona dispone de uno o más servidores, que son responsables de mapear los nombres simbólicos asociados a dicha zona. La resolución DNS sigue un esquema cliente-servidor, donde un proceso en la aplicación cliente (*resolver*), de forma totalmente transparente, contacta con un servidor DNS para obtener una dirección IP. Esta fase implica varios pasos que se encuentran reflejados en la Figura 35.

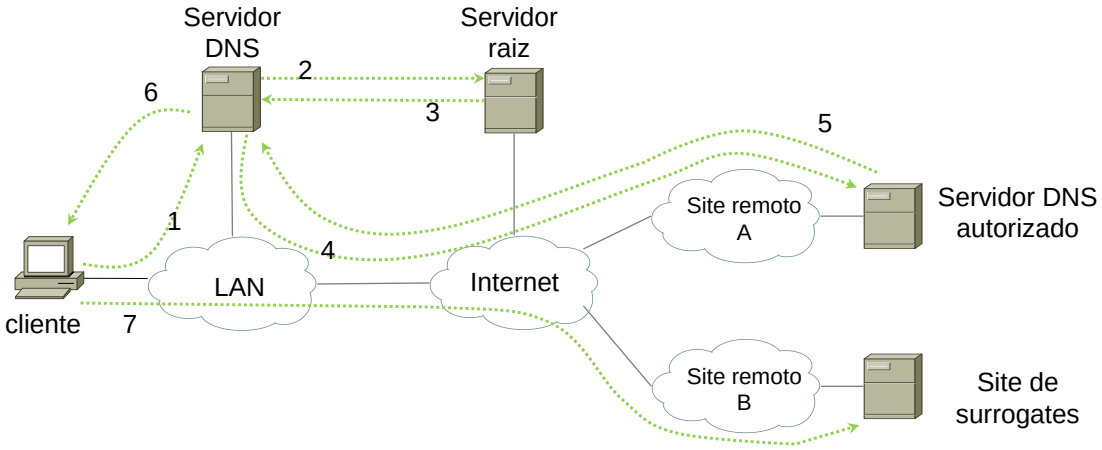


Figura 35. Flujo de datos en la resolución DNS.

Un nombre de dominio puede disponer de múltiples registros IP asociados (varias direcciones IP). Este es el caso de un *site* ofrecido por una CDN. El servidor autorizado de DNS, que pertenece al proveedor de CDN, tiene conocimiento de la disponibilidad de los *sites* de *surrogates* (referenciados por una dirección IP), así como de su idoneidad para servir contenido a cada cliente que lo solicite. Dependiendo de las métricas empleadas, el proceso de resolución en el cliente obtendrá una dirección IP correspondiente al *site* más adecuado.

Pese a su simplicidad, la redirección DNS presenta algunos inconvenientes:

  - La fase de resolución DNS puede suponer un retardo considerable al atravesar un entorno WAN, comparado con una resolución DNS en un entorno local. Para el caso de páginas u objetos web de gran tamaño, un retardo inicial puede no ser importante, pero sí para pequeñas páginas u objetos, pues este retardo puede ser comparable o superior que los tiempos de transmisión. Téngase en cuenta que los tiempos de ida y vuelta (RTT) asociados a las peticiones de DNS atraviesan (o pueden atravesar) numerosas subredes, mientras que el tráfico de datos

atravesará pocas subredes, al estar ubicado en las cercanías del cliente. La conclusión de esto es que se reduce el rendimiento para objetos y páginas web pequeñas.

- La fase de resolución DNS atraviesa el navegador del cliente, el servidor DNS de la propia red de área local o LAN, posiblemente servidores DNS intermedios y, finalmente, los servidores raíz. Algunos o todos estos agentes pueden incorporar mecanismos de *caching* internos, evitando o reduciendo la función de balanceo global para cada petición del servidor DNS autorizado. Una nueva petición solicitando el mismo contenido no llegará al servidor DNS autorizado, sino que sólo alcanzará unos de los servidores DNS intermedios o, incluso, será respondido por el propio cliente a través de la información almacenada previamente en su caché. Pese a que los mensajes de DNS incorporan un campo de tiempo de vida (*TTL, Time To Live*), no existe un valor claro para fijar [Sha_01]. Un valor elevado del TTL reduce la capacidad de responder dinámicamente frente a cambios en servidores y red, mientras que un valor reducido incrementa el tráfico de red.

- El mecanismo DNS es una técnica de mapeo o conversión, y no se diseñó con el propósito de encaminar en base a contenido (es decir, redirigir a un usuario en base al contenido que está solicitando). Un *website* puede ser visto como un lugar donde se encuentra contenido específico que un usuario puede solicitar. En un *site* distribuido, el contenido puede estar disperso dependiendo de la política de la compañía. Los servidores DNS de Internet no comprueban el contenido solicitado, por lo que son incapaces de redirigir en base a éste. Lógicamente, si únicamente se emplean métricas de nivel de red, no existe ninguna posibilidad de responder ante fallos del nivel de aplicación en el sistema de decisión de una CDN. Esto implica que deben emplearse otras estrategias como replicación o *caching* en todos los *sites* de la CDN.

- Como se observa en la Figura 35, el servidor DNS autorizado obtiene en última instancia una solicitud del servidor DNS de la red de área local, encargado de resolver los nombres de dominio de todos los clientes conectados a dicha LAN. Cuando se recibe un paquete de solicitud de resolución de DNS, el servidor DNS autorizado obtiene la dirección IP del servidor DNS de la LAN, con el fin de obtener el punto de presencia (PoP) más cercano. Actualmente, y al contrario que las antiguas redes de área local, las redes corporativas no se encuentran físicamente conectadas en la misma zona geográfica, por lo que es posible que un cliente a miles de kilómetros de su servidor DNS local sea redirigido a un *site* incorrecto, como se muestra en la Figura 36.

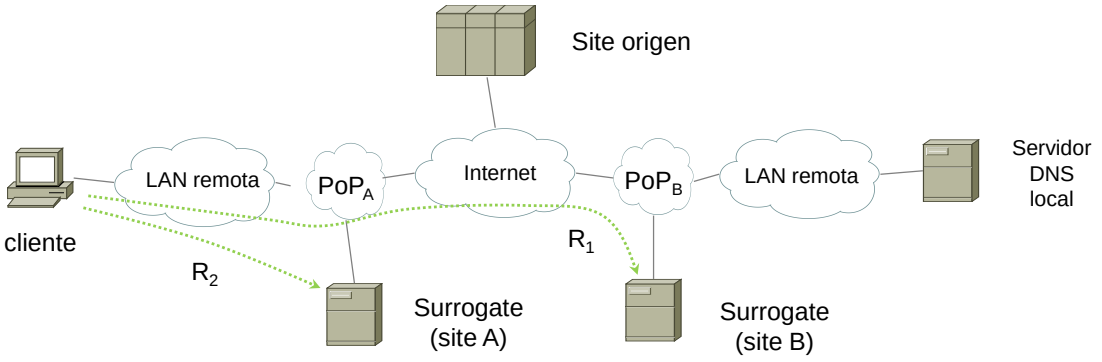


Figura 36. Resolución DNS inadecuada.

Suponiendo un comportamiento normal donde un usuario es redirigido al *site* asociado a su PoP más cercano, el *site* obtenido por el cliente de la Figura 36 sería el *site* B (ruta R_1), ya que el servidor autorizado 'piensa' que el cliente es el servidor DNS local. Sin embargo, el cliente debería haber sido redirigido al *site* A (ruta R_2). Esta situación indeseada se podría reducir si el cliente se encuentra conectado a su red de área local por un enlace propietario de baja latencia, o cada LAN remota dispone de su propio servidor DNS local.

Nótese que el escenario descrito no resulta extraño en Internet, y la situación podría ser incluso más inadecuada si el servidor DNS autorizado obtiene la solicitud de resolución desde un servidor DNS intermedio, cuya localización puede no guardar ninguna relación con la del cliente. Este caso es posible dado que la resolución DNS puede funcionar en modo iterativo o recursivo.

- ***Multiplexación inteligente***: Una alternativa al uso normal del DNS consiste en la selección inteligente del servidor, donde el servidor DNS autorizado devuelve todas las direcciones IP de los *sites* o *surrogates* más adecuados, y corresponde al cliente (a su navegador web) la decisión de contactar entre uno de todos. El cliente toma su decisión interactuando con un router cercano para obtener ciertas métricas de cara a seleccionar el servidor más cercano a contactar. Esta alternativa únicamente reduce carga de proceso en el servidor DNS autorizado al trasladarla a los clientes, pero no soluciona los inconvenientes asociados al DNS anteriormente citados.

- ***Anycast a nivel de red***: La técnica de *anycast* en la capa 3 consiste en un conjunto de servidores que están asociados a una dirección *anycast*, a la que un cliente envía su solicitud. Cada *router* contiene una ruta al servidor *anycast* más cercano en base a una cierta métrica, por lo que encaminarán cada cliente con su *surrogate* o *site* más cercano. Nótese que mientras una comunicación *multicast* supone que un paquete llegará a todos los servidores (realmente a todo el grupo *multicast*), un paquete *anycast* sólo llegará a un servidor. Pese a que esto pueda

parecer un mecanismo de encaminamiento adecuado para una CDN, cuenta con algunos inconvenientes. En primer lugar, no existe un espacio de direcciones para alojar las direcciones *anycast* (al menos en IPv4). Algunos sugieren el uso de direcciones *multicast*, donde sólo un servidor del grupo *multicast* responda a la solicitud. En cualquier caso, IP *anycast* requiere soporte en los *routers*, lo cual resulta imposible en Internet. Además, la decisión de encaminamiento es tomada a nivel de red sin considerar otros parámetros como la carga de la red, el usuario o el tipo de contenido.

- ***Anycast a nivel de aplicación***: Debido a los problemas anteriormente descritos en la redirección DNS, la comunidad científica propuso nuevos mecanismos avanzados de encaminamiento basados en contenido. Nótese que las políticas de redirección mediante DNS son propietarias de cada proveedor de CDN, ya que emplean distintos mecanismos y diferentes métricas en sus decisiones. Aunque una CDN ofrece una configuración distribuida de contenido, no ofrece un método de localización de contenido con un mecanismo global de encaminamiento mediante DNS. Una de las alternativas fue desarrollada por científicos de la universidad de Stanford, quienes proponían un protocolo de encaminamiento basado en el nombre [Gri_01] como parte del proyecto TRIAD [WWW_Tri]. Dado que los clientes realmente desean un cierto contenido y no conectividad a un servidor, se propone un nuevo nivel, denominado nivel de contenido. Éste está basado en una nueva entidad denominada CR (C*ontent Router*) que actúa tanto de *router* IP como de servidor DNS. La resolución de nombres se soporta mediante un nuevo protocolo propuesto denominado INRP (*Internet Name Resolution Protocol*), compatible hacia atrás con el mecanismo de DNS, pero con la futura posibilidad de incorporar la URL entera y no únicamente la porción del servidor, de forma que el CR sea consciente del contenido solicitado. De forma similar a un *router* IP, un CR dispone de una tabla donde se asocia un nombre con el siguiente salto, de forma que se encamina la solicitud INRP hacia el servidor de contenido óptimo.

Por otro lado, e independientemente del mecanismo de redirección empleado, los algoritmos de selección se clasifican en activos y pasivos [Ver_01]:

- ***Esquemas activos***: son aquellos que comienzan la monitorización de la red y de los servidores ante una nueva solicitud, suponiendo la generación adicional de paquetes dependiendo de la finalidad: si sólo se busca escalabilidad, cada petición se encaminará al servidor menos cargado. En este caso se enviará un paquete a todos los *sites*, y un servidor de monitorización contestará con una estimación de la carga de dicho *site*. El encaminamiento se realizará al servidor menos cargado. Si se desea reducir el tiempo percibido por el cliente, se buscará el *surrogate* más cercano, monitorizando la distancia entre clientes y *sites*. Un posible escenario se muestra en la Figura 37.

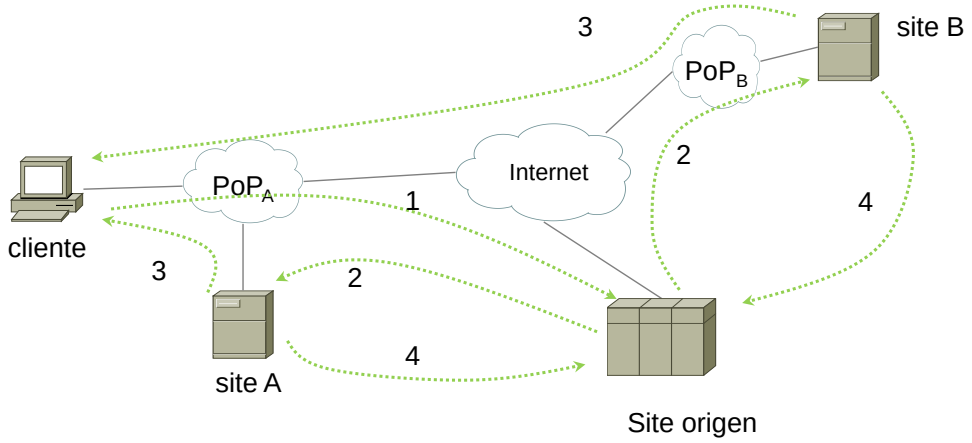


Figura 37. Esquema activo en una selección de site en una CDN.

Cuando se solicita una página, la aplicación cliente debe resolver en primer lugar el mapeo de nombres en direcciones IP consultando a un servidor DNS. El servidor autorizado de DNS es el último en ser consultado, siendo el responsable de devolver una dirección IP adecuada. Por tanto, aquí es donde debe residir la entidad de decisión, que tendrá que elegir un *site* adecuado. En primera instancia, envía un paquete de prueba a todos los *surrogates*, quienes efectúan un *ping* al cliente para obtener valores de tiempo de ida y vuelta (RTT). Este valor se envía de vuelta a la entidad de decisión junto con otros valores posibles, como nivel de carga, número de conexiones, etc. De esta forma, la entidad que tomará la decisión dispondrá de suficiente información. Nótese que esta estrategia supone la generación de una cantidad significativa de tráfico cuando el número de *sites* o *surrogates* es elevado, y también un retardo perceptible cuando la página solicitada es de tamaño reducido. Para paliar esta circunstancia, el servidor DNS puede enviar únicamente un paquete de prueba sólo a *sites* que considere que se encuentran relativamente cerca del cliente. Esta información se puede obtener mediante esquemas pasivos que se describirán a continuación.

- ***Esquemas pasivos***: están basados en el uso de una tabla de encaminamiento que asocia un grupo o *clúster* de clientes con su *site* más cercano dependiendo de la métrica empleada. Esta tabla puede ser rellenada de manera estática o dinámica. También es posible que la entidad de decisión se encuentre distribuida, es decir, que los *sites* intercambien información y cada uno de ellos genere su propia tabla de encaminamiento, convirtiéndose de esta forma en una potencial entidad de decisión global. Dentro de un dominio administrativo (AS, *Administrative Domain*) resulta sencillo crear una topología de red y determinar el *site* más cercano a un cliente. Una tabla de encaminamiento estática consistiría en el producto de asignar un cierto coste a cada enlace dentro del grafo de red. La topología de red completa puede ser conocida si se emplean protocolos del estado del enlace, mientras que los protocolos del tipo vector-distancia sólo proporcionan la ruta más corta entre subredes en términos de una cierta métrica.

Si se toma el coste constante, entonces el número de enlaces atravesados por el cliente se minimizará. En el caso de una CDN, los *sites* de *surrogates* están dispersos en Internet, cuya topología resulta complicada de obtener. Una forma de obtener parte de ésta es mediante el uso de tablas BGP (*Border Gateway Protocol*) proporcionadas por los ISPs, que ofrecen información acerca de (a) las subredes conocidas por los routers de los ISPs y (b) el número de dominios administrativos a atravesar para alcanzar dicha subred. Nótese que BGP no proporciona información acerca de latencia y número de saltos entre routers, por lo que sólo debería tomarse como punto de partida.

2.5.2.2.2. Encaminamiento local

Una vez que una petición de un cliente se ha encaminado a una cierta zona (encaminamiento global) o *site*, debe alcanzar uno de los *surrogates* locales que componen dicho *site*. El mecanismo de encaminar una petición a un *surrogate* adecuado es conocido como encaminamiento local, encaminamiento inteligente (*intelligent switching*) o, simplemente, encaminamiento de contenido (*content switching*). La decisión se toma en base a la información que llega con el paquete solicitante (URL, tipo de contenido, usuario, etc.), así como de la información proporcionada de manera local por los *surrogates* de dicho *site* (carga del servidor, utilización de CPU, etc.). El encaminamiento por contenido puede tener lugar a nivel 4 o a nivel 7, tanto de forma centralizada como distribuida, como se mencionó en el capítulo 2.2 . En una CDN, habitualmente se emplea una configuración de tipo *clúster*, también denominada granja de servidores (*server farm*).

Por ejemplo, el protocolo de Cisco WCCP (*Web Cache Communication Protocol*) [WWW_Wccp] usa este último esquema combinando una estructura de *web caching* transparente a nivel de *router*, como se muestra en la Figura 38.

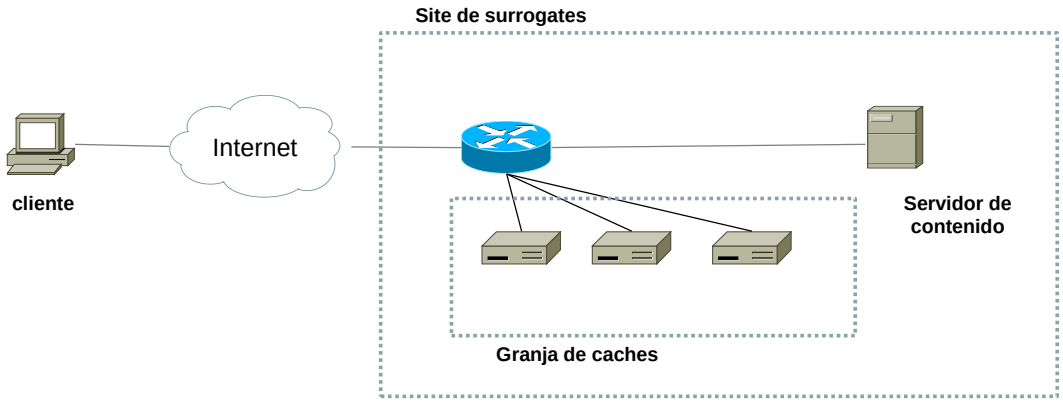


Figura 38. Site de surrogates con una granja de servidores caché.

En este ejemplo, no hay una granja de servidores, sino una granja de *cachés*, quienes almacenan el contenido del servidor de contenido local antes de responder a las solicitudes de los clientes. El router puede balancear entre *las cachés*, dependiendo de ciertos parámetros. El protocolo de Cisco WCCP permite a las *cachés* (es posible disponer de una sola *caché* en vez de un granja de *cachés*) anunciar a los *routers* adyacentes los protocolos a los que dará servicio. Cuando un cliente solicita una petición al servidor de contenido, el *router* intercepta dicha solicitud y la pasa a uno de los servidores *caché* dependiendo de su configuración. Este servidor *caché* será el que sirva al cliente, bien de manera directa (si ya dispone del objeto solicitado en su espacio de almacenamiento *caché*), o de manera indirecta (obteniendo el objeto previamente del servidor de contenido).

Nótese que este escenario sólo incluye un servidor de contenido en el *site* de *surrogates*, y el encaminamiento local se lleva a cabo a través del *router*. Sin embargo, una arquitectura más escalable puede contener más de un servidor de contenido. En una CDN, el servidor de contenido puede estar ubicado en el *site* origen (fuera del *site* de *surrogates*). En una estructura de *caching* jerárquica, podría actuar como nodo padre principal.

2.5.2.3. Mecanismo de distribución de contenido multimedia

En esta sección se abordará brevemente el procesado del contenido multimedia llevado a cabo por los proveedores de contenido y cómo estos lo distribuyen a través de las CDNs. No obstante, cabe indicar que la distribución de contenido digital en general es bastante compleja y requiere de una cierta alianza a nivel empresarial e incluso político-administrativa. Es por ello por lo que han surgido iniciativas como NEM (Networked Media). La iniciativa NEM tiene por misión ser el órgano de consulta de la Comisión Europea para desarrollar políticas de liderazgo y desarrollo sostenible europeo en Contenidos Digitales y la convergencia de las redes de Telecomunicaciones para mejorar la vida de los ciudadanos europeos a través de una enriquecida experiencia multimedia. Se puede encontrar más información en [WWW_Nem].

La distribución de contenido abarca las etapas desde la codificación hasta la distribución al usuario final. El contenido puede ser imágenes, audio, video, documentos, etc., que son susceptibles de ser transformados de un formato fuente a otro formato destino. Esto es debido a que los datos originales pueden ser creados por dispositivos de captura que consideran únicamente parámetros hardware y capacidad de almacenamiento, así como un formato de datos neutro. El formato destino resulta de un procesado con el fin de adaptar los datos originales a la infraestructura de distribución de contenidos y a las necesidades de los clientes. Actualmente, este proceso sólo se realiza con la distribución de contenido en vivo, que incluye streaming de vídeo y audio.

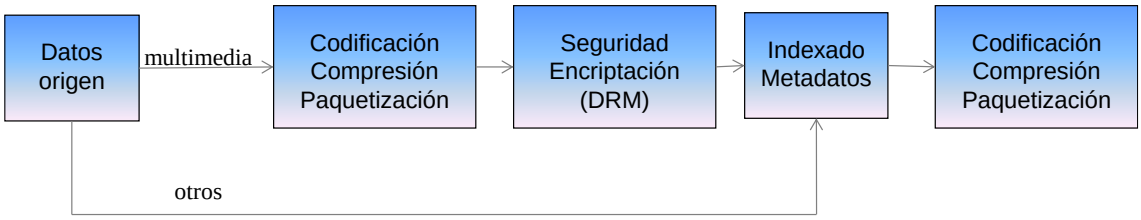


Figura 39. Procesado de datos en el proveedor de contenido.

La Figura 39 muestra el esquema de procesado de los datos antes de proceder a su distribución. Este proceso es llevado a cabo por el proveedor de contenido e implica varias etapas:

- Los datos multimedia requieren un ancho de banda excesivo en formato bruto, por lo que deben comprimirse aplicando codificadores de audio y video, que suelen empaquetar la información para obtener un formato de streaming.

- Tras la fase de codificación, el contenido debe ser seguro mediante la inclusión de mecanismos de encriptación con el propósito de impedir el uso no autorizado. Los esquemas de protección son conocidos como sistemas DRM (*Digital Rights Management*) [Yen_10].

- Es relativamente sencillo indexar contenido en formato texto, pero no en formato multimedia, que requiere software específico. A veces, este proceso se lleva a cabo en paralelo con la codificación, generando los metadatos asociados. Los metadatos se publican en un servidor web, y actúan como punto de referencia del contenido multimedia.

- Una vez se ha formateado apropiadamente el contenido para los clientes, deben ser incorporados en el servidor web y en el servidor multimedia. Ambos servidores pertenecen al proveedor de contenidos. El servidor web contiene los metadatos y enlaces que hacen referencia al contenido multimedia, así como contenido web; el servidor multimedia sirve el contenido multimedia a través de un software específico. Nótese que existen diferentes plataformas para el contenido multimedia, por lo que los clientes pueden necesitar el uso de un software cliente propietario (*plugin)* capaz de entender el formato multimedia.

Una vez se ha formateado el contenido y está disponible para poder ser distribuido a los clientes, existen varias formas de llevarlo a cabo. Consideremos el caso de una página web pública ofrecida por una CDN, por ejemplo, un diario o periódico. Esta página contendrá imágenes, texto y posiblemente clips. Supongamos también el escenario considerado en la Figura 40, donde podemos distinguir dos *sites* origen:

- El *site* de la compañía proveedora del contenido, que pertenece a dicha empresa.

- El *site* origen de la CDN, que pertenece al proveedor de la CDN.

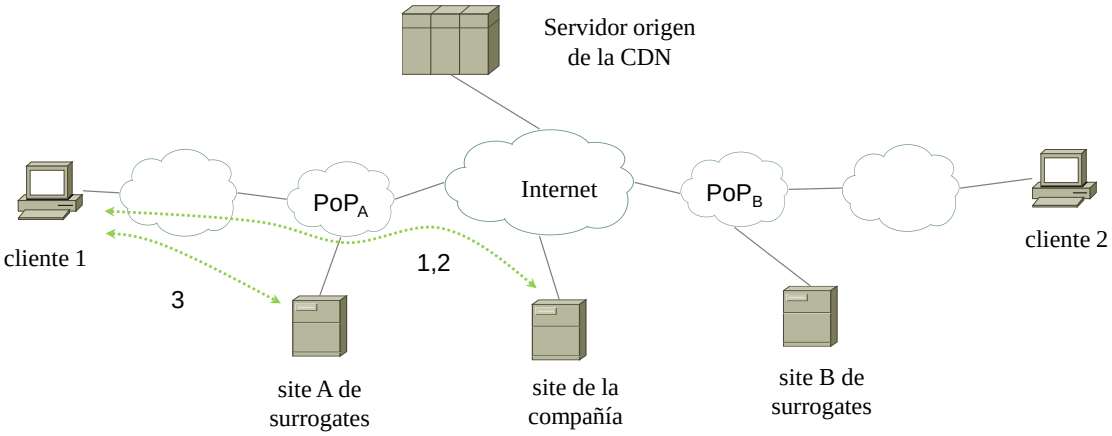


Figura 40. Escenario de distribución de contenido.

Los *surrogates*, en sus respectivos *sites*, pueden obtener el contenido que todavía no se ha almacenado en *caché* (o replicado) tanto del *site* de la compañía como del *site* origen de la CDN, dependiendo del contrato establecido entre el proveedor de contenido (compañía) y el proveedor de la CDN. Si éste último actúa como proveedor de *hosting*, entonces los *surrogates* sólo 'verán' un *site* origen bajo el control del proveedor de CDN. Si, por el contrario, la compañía desea tener su *site* bajo su propio control, usará el *site* origen de la CDN como un mero nodo de distribución. La forma en la se lleva a cabo esto es propietaria de cada proveedor de CDN.

La página que contiene datos multimedia está ubicada en el *site* de la compañía distribuidora de contenidos, pero estos datos pueden ser suficientemente considerables como para distribuirlos de manera eficiente desde este sitio si el número de clientes que demanda estos contenidos es elevado (de otra forma el proveedor de contenido quizás no emplearía una CDN). Un método para solventar esta circunstancia consiste en obtener la página principal junto con los enlaces a las imágenes y clips desde el *site* de la compañía. El proceso sería el siguiente, representado en la Figura 40:

- El cliente contacta con el *site* de la compañía y solicita la página.
- El servidor de la compañía estima la ubicación del cliente y rescribe los enlaces embebidos en la página asociados a imágenes y clips con el correspondiente *site* del *surrogates* más cercano.
- El navegador muestra la página al cliente, obteniendo las imágenes y los clips desde el *site* de *surrogates* especificado en los enlaces.

Esta opción tiene ciertos inconvenientes. En primer lugar, no existe ningún encaminamiento global basado en DNS al contactar el *site* de la compañía. Como sólo hay una única ubicación, todos los clientes contactarán el mismo *site*. Pero para estimar el *site* de *surrogates* más adecuado para cada cliente, el *site* de la compañía debe interactuar de alguita forma con el proveedor de CDN para obtener información, lo que

representa un retardo en la comunicación percibida por el cliente. Además, el servidor debe rescribir los enlaces embebidos para cada cliente, lo que supone una carga y retardo extra.

Una alternativa al caso anterior es empleada por Akamai [WWW_Aka] en lo que se denomina *page rewriting*. El servidor en el *site* de la compañía usa la misma URL para los enlaces embebidos. El cliente debe resolver dicha URL contra un servidor DNS bajo control del proveedor de CDN, y posteriormente, obtendrá el contenido de un *site* de *surrogates* cercano. Este mecanismo tiene tres características principales: (i) permite servir siempre la misma página a todos los clientes, mejorando el rendimiento por *caching*, (ii) reduce carga en el servidor, al no tener que estimar éste la ubicación de los clientes y (iii) evita que el proveedor de CDN intercambie información topológica de sus *sites* de *surrogates*, que podrá permanecer oculta.

La comunicación entre los *sites* de *surrogates* y el *site* origen puede tener lugar a través de Internet o a través de un canal *broadcast* mediante satélite. La primera opción es la más comúnmente empleada por proveedores de CDN como Akamai, y consiste típicamente en una red de capa de aplicación en forma de árbol comenzando en el *site* origen. Un canal satélite proporciona la deseada calidad y predicción requerida en aplicaciones de streaming, eventos en vivo y aplicaciones de tiempo real.

2.5.3. Principales CDNs

2.5.3.1. CDN comerciales

La mayor parte de las CDNs operativas están desplegadas y controladas por empresas comerciales; algunas de ellas se han consolidado con el tiempo mientras que otras han sido adquiridas o se han fusionado con otras (el fenómeno de concentración también tiene lugar en este sector). La Tabla 5 enumera las principales empresas que ofrecen servicios de CDN, y una lista actualizada puede encontrarse en [WWW_Web].

Akamai Technologies [WWW_Aka] [Dil_02] se fundó en 1998 en Massachusetts, EEUU, y es desde su nacimiento el líder del mercado es proporcionar servicios de distribución de contenido. Dispone de un gran número de servidores *cache* distribuidos por todo el planeta, capaces de distribuir contenido estático (páginas HTML, imágenes, documentos PDF, etc.), contenido dinámico (animaciones, scripts, DHTML) y streaming media. La infraestructura de Akamai redirige las solicitudes de los clientes a *surrogates* cercanos con alta probabilidad de disponer del contenido, y emplea como mecanismo de redirección un sistema DNS dinámico y tolerante a fallos.

Nombre	Servicios y soluciones	Cobertura
Akamai	Servicios básicos de CDN incluyendo streaming (Edge Platform, Edge Control, NOCC)	80% del mercado. Dispone de más de 25000 servidores distribuidos en 900 redes ubicados en 69 países. Cerca del 20% del tráfico de Internet pasa por estos servidores
EdgeStream	Orientado fundamentalmente a streaming e IPTV (EdgeStream platform)	Proporciona video streaming sobre cable o ADSL por todo el planeta
LimeLight Networks	Live video y VoD, música, juegos y descarga de ficheros (Limelight ContentEdge, Limelight, MediaEdge, Limelight Custom CDN)	Dispone de servidores ubicados en 72 redes alrededor del mundo
Mirror Image	Distribución de contenido, streaming media, web computing (Global Content Caching, Extensible Rules Engine)	Dispone de servidores ubicados en 22 países alrededor del mundo

Tabla 5. Principales CDNs comerciales

El sistema de mapeo resuelve un nombre de host dependiendo del servicio solicitado, de la ubicación del cliente y del estado de la red. El sistema DNS también se emplea para hacer balanceo de carga. El sistema DNS de Akamai es realmente complejo y emplea agentes software que se comunican con algunos routers extremos (*border routers*), de tal forma que se obtiene información a nivel de red BGP (*Border Gateway Protocol*) [Rek_95] para determinar la topología de red. El sistema de mapeo de Akamai combina la información de la topología de red obtenida con estadísticas de red en tiempo real (por ejemplo datos de *traceroute* [Mal_93]) para hacer una estimación dinámica y ajustada de la estructura de la red, y determinar los caminos o rutas (*paths*) con mejor calidad. En este sistema es común la denominación ARL (*Akamai Resource Locator*).

El sistema de balanceo de carga basado en DNS de Akamai monitoriza de forma continua el estado de los servicios, de los servidores y de las redes. Para realizar esta monitorización extremo a extremo, Akamai emplea agentes software que simulan el comportamiento de un cliente remoto al descargarse objetos web y medir tanto el tiempo de descarga como la tasa de fallo. Akamai emplea toda esta información para monitorizar el rendimiento global de su sistema, así como para, de manera automática, detectar y suspender aquellos *surrogates* problemáticos. Cada uno de los *surrogates* de la red de Akamai envía reportes periódicos sobre su carga a una aplicación de monitorización, quien la combina y publica un reporte generalizado de carga al servidor DNS local. Este servidor DNS local determina que direcciones IP (de su *clúster* o centro de datos correspondiente) debe devolver cuando se resuelve un nombre. Si la carga de un *surrogate* supera un umbral determinado U_1, entonces el servidor DNS asigna de forma simultánea contenido de ese servidor a otros servidores. Si, por otro lado, la carga del *surrogate* excede un umbral aun mayor U_2 ($U_2 > U_1$), entonces la dirección IP de dicho *surrogate* quedará temporalmente indisponible para los clientes. De esta forma, el

surrogate podrá descargar parte de su carga cuando experiencia una carga alta o moderada. El sistema de monitorización de Akamai también transmite datos de carga del centro de datos al DNS *resolver* de alto nivel, para evitar cargar todo el centro de datos. Además del balanceo de carga, el sistema de monitorización de Akamai también proporciona un reporte centralizado sobre los servicios para cada cliente y por cada servidor. Esta información es útil para un correcto mantenimiento operativo y diagnóstico.

Akamai distribuye contenido estático y dinámico sobre HTTP y HTTPS. Los servidores de contenido de Akamai emplean tiempos de vida (*lifetime*) y otros atributos sobre el contenido estático dependiendo del tipo de dato. Dependiendo de estos atributos, los *surrogates* remotos son capaces de garantizar la consistencia del contenido. Por otro lado, Akamai gestiona el contenido dinámico a través de ESI (*Edge Side Includes*) [WWW_Esi], que ya se mencionó en el capítulo 2.3.2. El empleo de ESI permite a los proveedores de contenido segmentar su contenido en piezas con capacidades de *caching* independientes, y por ello pueden gestionarse como objetos separados en los *surrogates* de Akamai y ensamblados dinámicamente con posterioridad. En referencia a servicios de streaming, Akamai soporta los formatos de Microsoft Windows Media y Apple QuickTime tanto para streaming en vivo (*live*) como bajo demanda (*on demand*). En la modalidad de *live streaming*, el video es capturado y codificado por el proveedor de contenido y, posteriormente, es enviado a un servidor de Akamai que ejerce las funciones de punto de entrada al sistema (*entry point*), quien a su vez sirve el contenido a los usuarios. Con la finalidad de evitar un punto único de fallo, se realizan *backups* del servidor que ejerce de punto de entrada. Es más, este *surrogate* envía el contenido a través de rutas redundantes al resto de *surrogates*.

EdgeStream [WWW_Edge] fue fundada en el año 2000 en California, EEUU. Se trata de un proveedor de aplicaciones de video streaming sobre Internet. Proporciona software de video bajo demanda y streaming IPTV para permitir el transporte de video a una elevada tasa sobre Internet. Básicamente, emplea HTTP streaming para el envío de contenido. EdgeStream soporta diferentes formatos de compresión, y ha desarrollado las herramientas CROS (*Continuous Route Optimization Software*), ICTT (*Internet congestion Tunnel Through*) y RPMS (*Real Time Performance Monitoring Service*), que están orientados a reducir la latencia, la pérdida de paquetes y los cuellos de botella o congestiones en la red.

La plataforma de EdgeStream está compuesta por dos componentes de software, uno en la parte cliente y otro en la parte servidora. La parte servidora consta, a su vez, de los siguientes módulos: CMOR (*Content Management and Online Reporting*), *Server Software Module, EdgeStream Control Server Software Module, EdgeStream Database System Module, and EdgeStream Streaming Server Module*. Estos módulos pueden ejecutarse en un solo servidor físico o en varios. El módulo CMOR gestiona las cuentas,

el contenido, y el resto de servidores en el sistema. También genera reportes web en tiempo real para la visualización de estadísticas y transacciones desde una base de datos SQL. El módulo de control proporciona la autoridad necesaria para obtener la información de localización de contenido, así como las funciones de distribución de contenido streaming y *logging*. El módulo de base de datos almacena los logs para la contabilidad y tarificación, y emplea Microsoft SQL Server. El módulo de streaming está diseñado para balancear carga y para ejecutarse incluso sobre servidores de bajo coste. El componente software en la parte del cliente consiste en un plugin para los reproductores Windows Media y RealPlayer. También se emplea para medir la calidad de la conexión a Internet segundo a segundo. El software se puede ejecutar sobre plataforma Windows y Windows CE.

Limelight Networks [WWW_Lime] fue fundada en 2001 en Tempe, Arizona, EEUU. Sus servicios de distribución de contenido incluyen distribución web (HTTP) de ficheros digitales como vídeo, música, juegos, software y medios sociales (*social media*). Proporciona servicios básicamente a empresas de medios, como las que trabajan en TVE, música, radio, revistas, películas, videojuegos y software.

Los proveedores de contenido pueden subir el contenido directamente a un servidor de la CDN de Limelight o a un servidor propio, que está conectado con la red de Limelight. Tras la solicitud de un usuario final, Limelight distribuye el contenido a uno o varios *clústeres* de servidores quienes a su vez alimentan los *surrogates* remotos que servirán el contenido a los usuarios. Como otras CDNS, Limelight emplea redirección DNS para redirigir a los usuarios a *clústeres* locales, una vez se han creado mapas detallados de Internet a través de una combinación de medidas propias (*traceroutes*) y otras que provienen del protocolo BGP.

Limelight soporta Adobe Flash, audio MP3, Microsoft Windows Media, RealPlayer, y el formato Apple QuickTime para distribuir servicios de streaming bajo demanda. El software propietario de Limelight incluye *Limelight ContentEdge* para la distribución de contenido vía HTTP, Limelight MediaEdge Streaming para distribuir contenido de video y música vía streaming, Limelight StorageEdge para almacenar el contenido del cliente dentro de la arquitectura de la CDN y Limelight Custom CDN para soluciones personalizadas en la distribución de contenido. Los proveedores de contenido que emplean los servicios de streaming usan *Limelight User Exchange* (LUX), que es una consola web para la gestión y el reporte que permite monitorizar la actividad del usuario final en tiempo real.

Mirror Image [WWW_Mir] se fundó en 1999 en Massachusetts, EEUU. Se trata de un proveedor de distribución de servicios de contenido online, aplicaciones, streaming media y web computing hacia usuarios finales. Al contrario que Akamai, con muchos *clústeres* de servidores distribuidos a lo largo del mundo, el esquema de la arquitectura

de CDN se basa en la concentración, con pocos *clústeres* pero de gran tamaño (es decir, con un elevado número de servidores) ubicados en zonas densamente pobladas. A estos *mega clústeres* se les suele denominar CAP (*Content Access Point*). Cuando una petición de un cliente solicita un contenido gestionado por Mirror Image, se encamina en primer lugar a un balanceador de carga global de la red CAP. Este balanceador emplea DNS para determinar la ubicación del CAP que proporcionará el menor tiempo de respuesta. Una vez la petición alcanza el CAP seleccionado, las *caches* consultan si disponen de dicho contenido. Si se encuentra el contenido, se envía al usuario final. Si se produce un fallo (*cache miss*), se devuelve de forma automática un código de redirección (*status code 302*) al servidor origen. Entonces el contenido solicitado es distribuido por el servidor origen y la red CAP obtiene el contenido (*pull*) desde el servidor origen y lo almacena para futuras peticiones. Las soluciones que proporciona Mirror Image son: (i) *Global Content Caching* para descargar tráfico cuando se sirve contenido estático,(ii) *Digital Asset Download* para gestionar el almacenamiento y descarga de contenido digital, (iii) *Video-on-Demand* para soluciones de streaming, (iv) *Extensible Rules Engine* para proporcionar a los clientes control sobre el proceso de distribución y (v) *Webcasting* para permitir a los usuarios el envío de mensajes *one-to-many* para aplicaciones de tele enseñanza, marketing, *training*, etc.

Al margen de estas (breves) descripciones de alguna de las CDNs comerciales más destacadas actualmente, sería conveniente tratar de hacer una valoración o comparación entre ellas para poder establecer posiciones dominantes en el mercado. Una de las comparativas más recientes data de 2009 y fue elaborada por la empresa de investigación y consultoría Yankee Group [WWW_Yan]. De manera gráfica, se puede observar en el informe [WWW_Yan_SCR] la posición dominante de Akamai, ilustrada en la Figura 41.

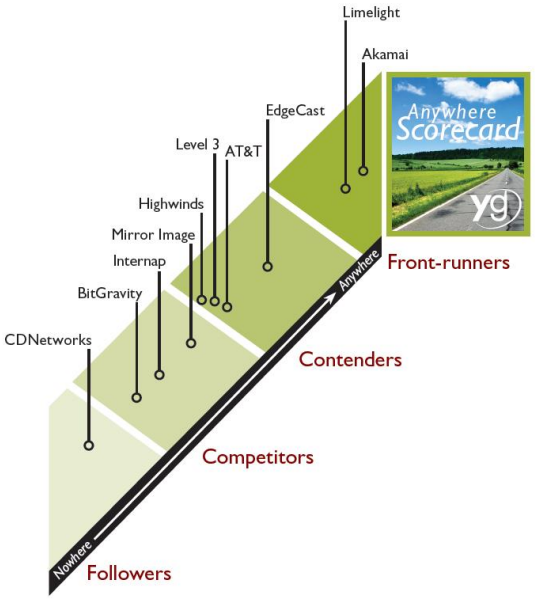


Figura 41. Ranking CDNs comerciales (Fuente: Yankee Group, 2009).

La metodología que se empleó establecía una evaluación basada en dos criterios principales:

- ***Visión***. Este criterio tiene en cuenta los siguientes parámetros: capacidad, ubiquidad, apertura, innovación, *context-awareness*, etc.
- ***Capacidad de transformación***. Este criterio tiene en cuenta los siguientes parámetros: I+D, gestión, reputación de marca, precio, etc.

A partir de todos estos parámetros se obtiene un valor para cada proveedor de CDN que determina su posición en el *ranking*. Cabe destacar que, si bien aparecen las principales CDNs comerciales en el *ranking*, su posición ha sido criticada por ciertos analistas del sector [WWW_Ray]. En primer lugar, la definición de los parámetros no aparece claramente definida en el documento; por ejemplo, el parámetro capacidad no indica que tipo de contenido se está considerando (páginas web, audio, vídeo, juegos, descargas, etc.). En segundo lugar, el documento no aclara su utilidad a la hora de seleccionar un proveedor de CDN u otro, pues hay muchos otros criterios o parámetros que se podría considerar; por ejemplo, el documento no indica que compañía dispone de la mejor herramienta de reporte y uso, o cuál es la mejor compañía para proporcionar servicios en Asia. Este tipo de clasificaciones por zonas o sectores no están disponibles en dicho informe.

Como ejemplo nacional y más reciente (2012), la empresa Telefónica ha lanzado al mercado su propia CDN [WWW_Tel_Cdn], aunque incorpora también tecnologías P2P. Esta CDN ofrece servicios de aceleración web (servicio *Web Dynamics Services*), descarga de ficheros (servicio *Smart Download*) y video streaming (servicio *Video Services*).

2.5.3.2. CDN académicas

Al contrario que en las CDNs comerciales, el uso de las tecnologías P2P está muy extendido en las CDNs académicas. Teniendo en cuenta este hecho, la distribución de contenido sigue un esquema descentralizado y la carga se distribuye entre todos los hosts participantes; de esta forma, el sistema puede abordar fallos en los nodos o picos altos de carga. Las CDNs académicas que emplean tecnologías P2P son particularmente efectivas para contenido estático únicamente, y son incapaces de tratar con contenido dinámico ya que por su naturaleza no es *cacheable*. En esta sección se describirán tres CDNs académicas representativas, como son CoDeeN, Coral y Globule (véase Tabla 6), así como FCAN y COMODIN, esta última particularmente relevante por estar orientada a servicios de streaming.

CDN	Descripción	Servicios	Disponibilidad
CoDeeN	CDN para pruebas montada sobre PlanetLab. Implementada en C/C++ y testada en Linux(2.4/2.6) y MacOS (10.2/10.3)	Proporciona *caching* de contenido y redirección de peticiones HTTP	No
Coral	CDN híbrida y gratuita que combina tecnologías P2P. Está ubicada en PlanetLab. Implementada en C++ y testada en Linux, OpenBSD, FreeBSD, y Mac OS X	Proporciona replicación de contenido dependiendo de la popularidad de dicho contenido	Aunque no hay una versión oficial, Coral es gratuito y se licencia bajo GPLv2
Globule	CDN de código abierto (*open source*) colaborativa. Implementada mediante scripting PHP, C/C++ y testada en Unix/Linux y Windows	Proporciona replicación de contenido, monitorización de servidores y redirección de clientes a replicas disponibles	Globule es de código abierto, licenciado bajo BSD y bajo la licencia del servidor HTTP Apache para el módulo correspondiente.

Tabla 6. Principales CDNs académicas

CodeeN [WWW_Cod] [Wan_04] es esencialmente un *servidor proxy* basado en tecnologías P2P desarrollado por la universidad de Princeton, EEUU. La comunicación dentro de la CDN entre *proxies* está basada en HTTP. CoDeeN proporciona tanto caching de contenido web como redirección de peticiones HTTP. Se encuentra desplegada sobre PlanetLab [WWW_Pla], que a su vez consiste en una red de servidores *proxy* de alto rendimiento. Los nodos que conforman CoDeeN están desplegados como *proxies* 'abiertos', en el sentido en que permiten el acceso desde el exterior de la plataforma. Cada uno de los nodos CoDeeN es capaz de actuar como un *forward proxy*, un *reverse proxy* y un *redirector*. El funcionamiento es el siguiente:

- Los usuarios deben configurar su *proxy* de Internet con un *proxy* cercano que participe en el sistema CoDeeN como nodo.
- Este nodo CoDeeN actúa como *forward proxy* y trata de satisfacer las peticiones de manera local. Si se produce un fallo (*cache miss*), la lógica del redirector dentro del mismo nodo determina donde se debe redirigir la petición.

Para la mayoría de las peticiones el redirector tiene en cuenta la ubicación de la petición, la carga del sistema y la fiabilidad para encaminar las peticiones al resto de nodos CoDeeN, quienes actúan como *reverse proxy* para estas peticiones. Las peticiones que tampoco pueden ser satisfechas en este segundo nivel son redirigidas directamente al servidor origen.

CoDeeN dispone de una funcionalidad de monitorización local que examina los recursos más básicos de un servicio, como son el número de *sockets* libres, la carga de CPU y el servicio del *resolver* DNS. También almacena información sobre el estado de la instancia de CoDeeN que se está ejecutando en dicho nodo, y todo esto le sirve para

valorar la disponibilidad de los recursos y del propio servicio. Para monitorizar el estado del resto de nodos, cada nodo CoDeeN emplea dos mecanismos: un mecanismo ligero basado en UDP que hace las funciones de *heartbeat* y otro mecanismo más pesado basado en HTTP [Wan_04]. En el primer caso, cada proxy envía un mensaje de *heartbeat* cada segundo a uno de sus *peers*, quien responde a modo de ACK (*piggyback acknowledgement*) con información adicional sobre su estado: carga media, tiempo del sistema, disponibilidad de sockets, tiempo de funcionamiento del nodo y del proxy, hora media con más carga y estadísticas de tiempos y fallos DNS. De esta forma, un nodo CoDeeN es capaz de obtener información del resto de nodos. En el segundo caso (HTTP) cada nodo debe especificar en un mensaje que falla cuando no es capaz de obtener una página (recurso web) dentro del tiempo permitido. A partir del histórico de estos mensajes junto con los mensajes UDP, cada nodo es capaz de determinar si un nodo es fiable (viable) para la redirección de una solicitud.

Coral [WWW_Cor] es una red de distribución de contenido P2P libre. Fue desarrollada por el grupo de sistemas seguros de computación de la universidad de Nueva York, EEUU. Su diseño original estaba orientado a replicar contenido web. Esta CDN emplea el ancho de banda de los participantes (voluntarios) para evitar el efecto *flash-crowd* y reducir la carga de los sitios web. CoralCDN se encuentra desplegada sobre PlanetLab, en lugar de otros sistemas (voluntarios) posibles. Para usar CoralCDN, un publicador debe añadir la terminación *".nyud.net:8090*" al nombre de host en la URL. De esta forma, los clientes son redirigidos a una *cache* Coral cercana de forma transparente mediante redirección DNS. Las caches de Coral cooperan entre sí para transferir datos entre *peers* cercanos cuando es posible, minimizando tanto el tráfico en el servidor origen como la latencia percibida por el usuario. CoralCDN está montada sobre el nivel de Coral de indexación de pares <clave-valor>. Esto permite a los nodos acceder a objetos *cacheados* cercanos sin interrogar de forma redundante nodos distantes. También previene la creación de *hotspots* en la infraestructura de indexación.

CoralCDN está formada por tres grandes partes: (i) una red de proxies HTTP colaborativa para gestionar las peticiones de los clientes, (ii) una red de servidores de nombres DNS para el dominio *".nyud.net*" que mapean las peticiones de los clientes a proxies cercanos, y (iii) una infraestructura subyacente de indexación y *clustering* sobre la que se asientan las dos aplicaciones anteriores. Coral emplea una abstracción de indexación denominado DSHT (*Distributed Sloppy Hash Table*) [Fre_03], que es una variante de la conocida DHT (*Distributed Hash Table*) para generar índices <clave-valor>.

Los *proxies* de Coral satisfacen las peticiones HTTP para URL '*coralizadas*'. Para minimizar la carga en los servidores origen, los *proxies* obtienen páginas web de otros *proxies* cuando es posible. Cada *proxy* dispone de una *cache* local para satisfacer las peticiones de manera inmediata. Si un *proxy* descubre un cierto contenido en uno o más *proxies*, establece conexiones paralelas entre todos ellos y solicita el contenido al primer

proxy con el que consigue conectarse. Una vez que el *proxy* vecino comienza a enviar contenido válido, el resto de conexiones TCP se cierran. Si el contenido no se puede localizar en ningún otro *proxy*, entonces se obtiene del servidor origen. En el caso de un efecto *flash-crowd*, todos los *proxies* Coral forman de manera natural una especie de árbol multicast para obtener la página web, en lugar de realizar peticiones simultáneas al servidor origen, y los datos fluyen desde el *proxy* que primero obtiene los datos del servidor origen.

Globule [WWW_Glob] es una CDN de código abierto colaborativa desarrollada por la Vrije Universiteit de Amsterdam, Holanda. Su objetivo es permitir a los proveedores de contenido web auto-organizarse y operar su propia plataforma de web *hosting*. En particular, Globule es básicamente una red overlay compuesta por nodos que operan en modo P2P a través de una WAN (*Wide Area Network*), donde los miembros participantes ofrecen sus recursos como son capacidad de almacenamiento, ancho de banda y capacidad de proceso. Se proporciona replicación de contenido, monitorización de servidores y redirección de peticiones de usuarios sobre replicas disponibles. En Globule, un sitio web queda definido como una colección de documentos que pertenecen a un usuario específico (el dueño del sitio web) y un servidor se define como un proceso que se ejecuta en una máquina conectada a una red, que ejecuta una instancia del software de Globule. Cada sitio web está compuesto por varios tipos de servidores: origen, backup, réplicas y redirector. El servidor origen tiene la autoridad de disponer de todos los documentos del sitio web y tiene la responsabilidad de distribuir el contenido entre los servidores implicados (replicas). Los servidores de backup mantienen una copia actualizada de todo el sitio web, aunque también pueden emplearse las réplicas para este cometido. Mientras que los servidores backup simplemente mantienen una copia, las réplicas tratan de maximizar el rendimiento basándose en la carga demandada y los requisitos de QoS. Un servidor réplica de un sitio web está operado típicamente por un usuario diferente que el del servidor origen, y normalmente, sólo dispone de una copia parcial de todo el contenido. El servidor réplica se puede percibir como un *proxy cache* que obtiene el contenido desde el servidor origen cuando se produce un fallo (*cache miss*). El servidor de redirección está encargado de redirigir las peticiones de los clientes a la réplica óptima. Los redirectores en Globule pueden emplear tanto redirección DNS como HTTP. El redirector también implementa una política para la redirección del cliente. La política por defecto redirige al cliente a la réplica más cercana en términos de latencia estimada. Los redirectores también monitorizan la disponibilidad de otros servidores para asegurar una redirección fiable y efectiva. Globule toma la latencia inter-nodo como medida de proximidad. Esta métrica es empleada para ubicar óptimamente replicas con los clientes cercanos más próximos. Globule se ha implementado como un módulo de terceros del servidor HTTP de Apache y permite a otro servidor replicar su contenido a otro servidor de Globule. Por ello, para replicar contenido, los proveedores de contenido tan sólo deben compilar este módulo extra para el servidor HTTP de Apache y editar un fichero de configuración.

FCAN (Flash Crowd Alleviation Network) [Ata_07] [Pan_06] es una solución de capa intermedia, que emplea básicamente una estructura similar a una CDN compuesta por *proxy caches* capaces de almacenar objetos y distribuirlos a los clientes, del mismo modo que los *surrogates* en una CDN. Teniendo en cuenta la corta duración y la impredictibilidad de un *flash-crowd*, FCAN invoca la red overlay sólo cuando un servidor está sobrecargado debido a un gran número de peticiones, y es capaz de reorganizar la red *overlay* si es necesario. Este dinamismo es la característica más importante de FCAN.

FCAN intenta complementar una infraestructura de servidor web para que sea capaz de gestionar eficazmente los picos de carga de corta duración, pero no está diseñado para soportar una carga que sea constantemente superior a la capacidad inicial planificada para el sitio web. En principio, FCAN está orientada para sitios web pequeños, aunque es posible su uso en sitios web grandes.

FCAN usa redirección DNS como las CDN desplegadas en una WAN, pero emplea un servidor DNS específico (estrictamente se trata de una *wrapper* DNS) denominado TENBIN [Shi_00] [Shi_06] que permite modificar de manera dinámica las entradas de la tabla DNS.

COMODIN [For_07] es una CDN orientada a servicios de streaming. La arquitectura está organizada en dos planos: (i) el *plano básico*, que consiste en una CDN de streaming (SCDN) proporcionando servicios de streaming bajo demanda, y (ii) el *plano colaborativo*, que proporciona un servicio de reproducción colaborativa [For_03]. Este último concepto permite a un grupo formado de manera explícita por clientes reproducir y controlar una sesión multimedia de una forma compartida, y se consigue básicamente a través del protocolo HCOCOP (*Hierarchical COoperative COntrol Protocol*) [For_09]. Al tratarse, en parte, de una SCDN, se ha estimado oportuno describir con un poco más de detalle la arquitectura, pues luego se empleará en el capítulo de implementación (capítulo 4), ya que el trabajo realizado en esta tesis ha servido para la implementación del plano básico.

El plano básico está formado por los siguientes componentes:

- El servidor origen, quien almacena los objetos multimedia para ser distribuidos por la CDN.
- Los *surrogates*, que son réplicas parciales del servidor origen con la capacidad de almacenar temporalmente contenido y distribuirlo a los clientes a través de la red de acceso usando un MSS (*Media Streaming Server*).
- El cliente, que es una aplicación multimedia que solicita contenido multimedia

- El redirector, quien selecciona el *surrogate* más adecuado para cada cliente basado en un algoritmo específico fruto de esta tesis que se describirá en los sucesivos capítulos.

- El gestor de contenido, quien coordina el almacenamiento de contenido entre el servidor origen y los *surrogates*.

El plano colaborativo consiste en los siguientes componentes:

- El CPSM (*Collaborative Playback Session Manager*), que proporciona el servicio básico de formación y gestión de grupos que está basado en un protocolo de gestión de sesión.

- El CPCS (*Collaborative Playback Control Server*), que está integrado dentro del MSS del plano básico y soporta el control remoto del streaming multimedia compartido entre los miembros de una CPS (*Collaborative playback sesión*).

- El CCC (*CPCS Coordination Channel*), que coordina a los diferentes CPCSs sirviendo la misma CPS a través del protocolo CCP (*Coordination Channel Protocol*).

- El CC (*Collaborative Client*), que es una mejora del cliente del plano básico que le proporciona al usuario una interfaz con el servicio colaborativo de reproducción.

Más información acerca del plano colaborativo puede encontrarse en [Raj_08].

2.5.4. Taxonomía de las CDNs

Como se ha descrito en capítulos anteriores, existen varios aspectos de interés científico en las CDNs como son la distribución de contenido, replicación, *caching* y ubicación de servidores. Sin embargo, en ninguna de las referencias bibliográficas encontradas para estos aspectos se realiza una categorización completa de CDNs considerando aspectos tanto *funcionales* como *no funcionales*. Los aspectos funcionales cubren el empleo de la tecnología y las operaciones llevadas a cabo dentro de una CDN, mientras que los aspectos no funcionales se centran en otras características de las CDNs, como pueden ser la organización, la gestión, así como aspectos de rendimiento. En este capítulo se tendrán en cuenta ambos aspectos para describir una taxonomía detallada de CDNs. Además, en este capítulo se realizará un mapeo de taxonomías con algunas CDNs representativas en la actualidad, con la finalidad de demostrar su aplicabilidad para categorizar y analizar las CDNs actuales.

La taxonomía de CDNs se presentará en base a cuatro aspectos:

- ***Composición***: hace referencia a los aspectos de organización y arquitectura, clasificando las CDNs en base a sus atributos estructurales.
- ***Distribución de contenido y gestión***: se refiere a la ubicación de servidores, selección y distribución de contenido, externalización de contenido y organización de cachés y réplicas.
- ***Encaminamiento y redirección***: hace referencia a los mecanismos de redirección de clientes para encaminar las peticiones hacia los *surrogates* más cercanos.
- ***Rendimiento***: se refiere a las metodologías de evaluación de prestaciones de una CDN.

2.5.4.1. Composición de CDNs

El análisis de los atributos estructurales de una CDN revela que los componentes de la infraestructura de una CDN están fuertemente relacionados. Es más, la estructura de una CDN varía dependiendo del contenido y de los servicios que proporcionan a los usuarios. Dentro de la estructura de una CDN, se emplea a un conjunto de *surrogates* para conformar el componente encargado de distribuir contenido, se emplea una combinación de relaciones y mecanismos para redirigir peticiones de clientes y se emplean protocolos de interacción para la comunicación entre elementos de la CDN.

La Figura 42 muestra una taxonomía basada en las características estructurales de las CDNs, considerando aspectos de organización, servidores, interacciones, etc.

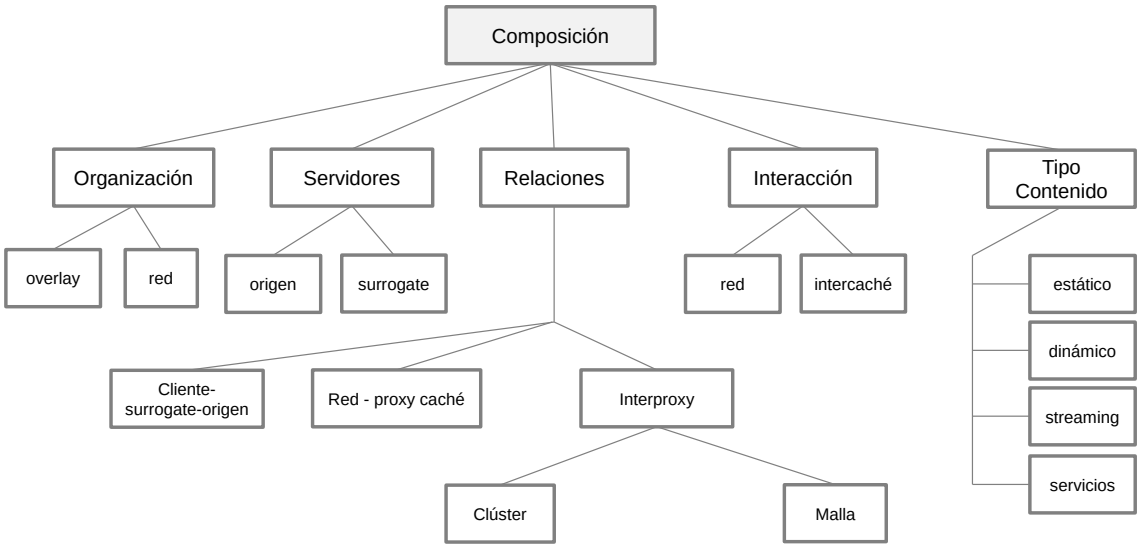


Figura 42. Taxonomía de CDNs según composición.

En el aspecto de la organización, existen dos formas de crear una CDN [Laz_01]. La primera forma, que es la de interés en esta tesis, es mediante una red *overlay* (o red de capa de aplicación) formada por un conjunto de servidores y *proxy cachés* que atienden las peticiones de ciertos tipos de contenido (web, media streaming, vídeo en tiempo real, etc.). Los componentes de la red troncal (*routers* y *switches*) no desempeñan ningún papel activo en la distribución de contenido, sino simplemente proporcionar conectividad básica de red y, en algunos casos, un cierto QoS para tráfico específico. La mayoría de las CDNs comerciales como Akamai y Limelight siguen este esquema overlay en la organización de la CDN. La segunda forma de crear una CDN consiste en involucrar de manera directa a los componentes de red (*routers, switches*) de tal forma que mediante políticas configuradas en estos componentes se identifica ciertos tipos de aplicación y se encaminan las peticiones adecuadamente. Los ejemplos más claros de este tipo de CDNs serían aquellas que incluyen dispositivos que redirigen las peticiones a servidores caché o específicos, optimizados para servir un tipo de contenido concreto. Este aspecto ya se mencionó en el capítulo 2.3.2. Algunas CDNs comerciales, como Akamai y Mirror Image emplean en realidad los dos tipos de aproximación (overlay y red): la parte overlay corresponde al encaminamiento global, mientras que la parte de red corresponde a un encaminamiento local, como se describió en el capítulo 2.5.2.2.

En relación a los ***servidores***, se pueden distinguir dos tipos: el servidor origen y las réplicas o *surrogates*. El servidor origen es el que alberga el contenido, que es introducido o actualizado por el proveedor de contenido. Por otro lado, el *surrogate* replica total o parcialmente el contenido del servidor origen, y puede funcionar como servidor web, servidor cache o servidor multimedia.

En cuanto a las ***relaciones***, puede decirse que la arquitectura compleja y distribuida de una CDN puede llevar asociada varias relaciones entre sus componentes (clientes, *surrogates*, servidor origen, *proxy cachés* y otros elementos de red). La mayoría de

estos componentes se comunican entre sí para replicar o *cachear* contenido dentro de la CDN. En la replicación, básicamente consiste en hacer '*pushing'* de contenido desde el servidor origen a los *surrogates* [Bru_01]. Por otro lado, *caching* implica almacenar respuestas cacheables para futuras peticiones [Coo_01] [WWW_Web] [Wan_99]. En un entorno CDN, la relación más básica en la distribución de contenido tiene lugar entre el cliente, los *surrogates* y el servidor origen. Cada cliente se comunica con uno o más *surrogates* que satisfacen (de forma transparente) las peticiones en lugar de uno o más servidores origen. Si no se emplean *surrogates*, el cliente se comunica directamente con el servidor origen.

Como se ha descrito anteriormente, las CDNs se pueden formar siguiendo una aproximación de red, donde la lógica se despliega en los elementos de red (*routers, switches*) para encaminar tráfico a los servidores *proxy cachés* que son capaces de atender las peticiones. La relación, en este caso, es entre los clientes, la red y los servidores *proxy-caché*. Por otro lado, los *proxy-cachés* dentro de una CDN se pueden comunicar entre sí. Dentro de esta comunicación *inter-proxy* [Coo_01], los *proxy cachés* se pueden agrupar en estructuras en *clúster* (*array*) y en malla. Un *clúster* o *array* es una agrupación fuertemente acoplada de *proxy cachés*, donde existe un *proxy* autoritativo que actúa a modo de maestro (*master*) en la comunicación con el resto de *proxy cachés*. Por el contrario, una malla es una agrupación débilmente acoplada de *proxy cachés*, y cada servidor dispone de una relación *one-to-one* con otros *proxies*; a su vez, la comunicación puede tener lugar al mismo nivel (*peers* o *siblings*) o a un nivel superior (*parents*) [Coo_01], como se describió en el capítulo 2.3.3.

En el aspecto de la ***interacción***, se pueden clasificar los protocolos en dos tipos: protocolos de interacción entre elementos de red e interacción entre *cachés*. Entre los primeros, se pueden mencionar el NECP (*Network Element Control Protocol*) [Cie_00] y el WCCP (*Web Cache Control Protocol*) [Cie_00]. Por otro lado, como ejemplos de protocolos *intercaché* están CARP (*Cache Array Routing Protocol*) [Val_98], ICP (*Internet Cache Protocol*) [Wes_97], HTCP (*Hypertext Caching Protocol*) [Vix_00] y *Cache Digest* [Ham_98]. Algunos de estos protocolos ya se describieron brevemente en el capítulo 2.3.3.

Finalmente, en relación con el ***tipo de contenido***, las CDNs realizan *hosting* de diferentes tipos de contenido (estático, dinámico, *streaming*), así como de diferentes servicios (servicio de directorio, servicio de transferencia de ficheros, etc.). Cada tipo de contenido o servicio tiene implicaciones en la arquitectura, por lo que algunas CDNs pueden centrarse sólo en algún tipo particular de contenido o servicio. El contenido estático hace referencia a aquel que no cambia frecuentemente, como pueden ser páginas HTML estáticas, imágenes, documentos PDF, ficheros, etc. Actualmente, todos los proveedores de CDNs soportan este tipo de contenido, ya que éste puede ser *cacheado* fácilmente y la consistencia se puede gestionar empleando mecanismos tradicionales de *caching*, ya *descritos* en el capítulo 2.3. El contenido dinámico hace

referencia a aquel que está personalizado para el usuario o se genera bajo demanda, y suele cambiar frecuentemente, como pueden ser animaciones, *scripts* y DHTML. Dada su naturaleza cambiante, este tipo de contenido no es *cacheable* (o muy poco, como se mencionó en la sección 2.3). El contenido de *streaming* puede ser en vivo (*live*) o bajo demanda (*on-demand*), y algunos ejemplos son retransmisiones deportivas, películas a la carta, etc. Los servidores de streaming adoptan protocolos especializados (*RTP*, *RTCP*, *RTSP*, etc.) para la distribución de este tipo de contenido, como se describió en el capítulo 2.3.4.

Adicionalmente, una CDN puede ofrecer sus recursos para ser empleados como un canal de distribución de servicios y, de esta forma, dar cabida a los proveedores de servicios de valor añadido. Algunos servicios son el de directorio (acceso a bases de datos), almacenamiento web (similar al contenido estático), transferencia de ficheros (distribución de software, películas, imágenes médicas), y servicios de comercio electrónico (gestión del carrito de la compra). Este último servicio implica típicamente disponer de *caching* de contenido dinámico (véase sección 2.3).

Nota: En la actualidad muchos de estos servicios se están ofreciendo al público como servicios en la nube (*cloud services*). Ello es debido a que un operador de tipo *cloud* puede emplear perfectamente una CDN como infraestructura. La relación entre CDNs y *cloud computing* se abordará con mayor detalle en la sección 2.5.5.1.

CDN	Organización	Servidores	Relaciones	Interacción	Tipo contenido
Akamai	Red y overlay	Origen y surrogates	Cliente-surrogate-origen Red-proxy- cache Interproxy	Red e intercaché	Estático, dinámico, streaming, servicios
Edge Stream	Red	Origen (varios)	No disponible	Red	streaming
LimeLight Networks	Overlay	Origen y surrogates	Cliente-surrogate-origen Red-proxy- cache	Red	Estático, streaming
Mirror Image	Red y overlay	Origen y surrogates	Cliente-surrogate-origen Red-proxy- cache	Red	Estático, streaming, servicios
CoDeeN	Overlay con proxies abiertos	Origen y surrogates	Cliente-surrogate-origen Red-proxy- cache Interproxy	Red e intercaché	Estático
Coral	Overlay	Origen y surrogates	Cliente-surrogate-origen Red-proxy- cache Interproxy	Red e intercaché	Estático
Globule	Overlay	Origen y surrogates	Cliente-surrogate-origen Red-proxy- cache	Red e intercaché	Estático, servicios

Tabla 7. Composición de CDNs. Correspondencia a las principales CDNs.

2.5.4.2. Distribución de contenido y gestión

La distribución de contenido y la gestión constituyen un aspecto vital para una CDN eficiente. En la Figura 43 se pueden apreciar los diferentes aspectos que se pueden considerar en este sentido: selección de contenido y distribución, ubicación de *surrogates*, externalización y organización de cache, que se pasan a describir a continuación.

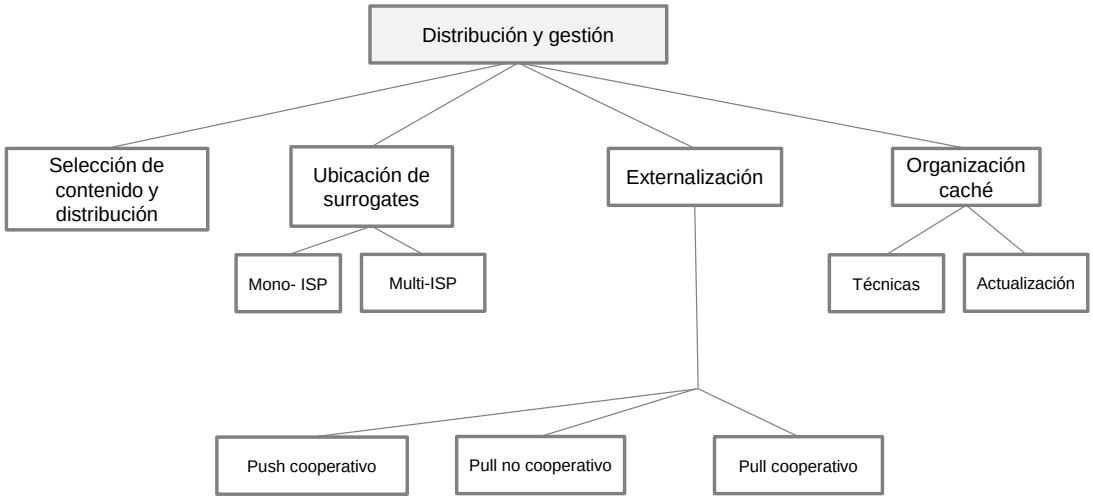


Figura 43. Taxonomía de distribución y gestión de contenido.

2.5.4.2.1. Selección de contenido y distribución

Este aspecto representa uno de los parámetros más importantes que determinan la eficiencia en la distribución de contenido, ya que implica la reducción en el tiempo de respuesta percibido por los clientes. El contenido que se debe seleccionar para su replicación, presente en el servidor origen, se puede distribuir hacia los *surrogates* de forma total o parcial, como se muestra en la Figura 44.

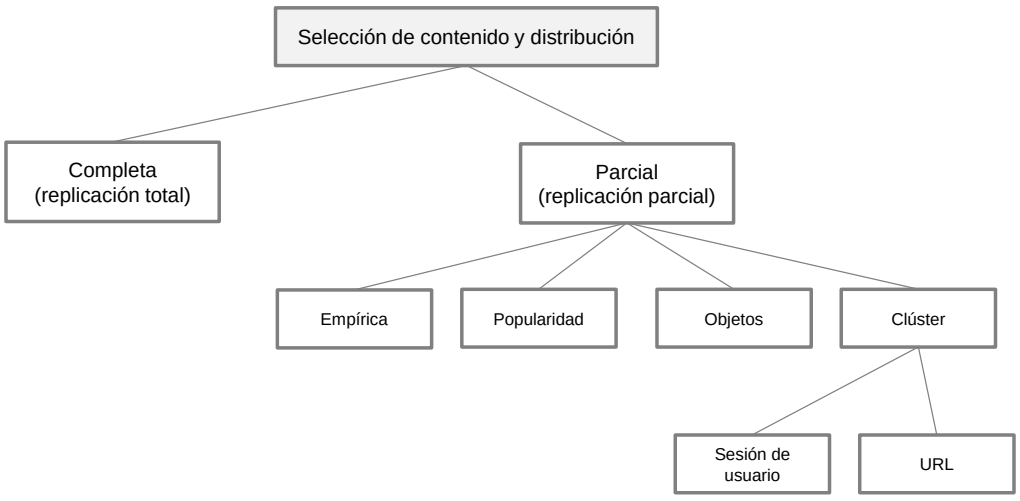


Figura 44. Taxonomía de selección de contenido y distribución.

La selección y distribución total es la aproximación más sencilla donde los *surrogates* replican todo el contenido del servidor origen. En este esquema el proveedor de contenido configura su servicio DNS de tal forma que todas las peticiones son resueltas por un servidor de la CDN (*surrogate*) quien distribuye el contenido. La gran ventaja de este esquema es su sencillez. Sin embargo, no es una solución práctica considerando el crecimiento de Internet y los objetos web (véase sección 2.1), por mucho que el hardware de almacenamiento se reduzca, ya que la gestión de las actualizaciones resultaría en todo caso inabordable.

Por otro lado, en la selección y distribución parcial, los *surrogates* realizan una replicación parcial para distribuir únicamente objetos embebidos (por ej. imágenes) desde la CDN. De esta forma, un proveedor de contenido modifica su contenido de tal forma que los enlaces a objetos específicos embebidos referencian nombres de *host* de los cuales el proveedor de CDN es autoritativo. Así, la página web básica se obtiene del servidor origen, mientras que los objetos embebidos se obtienen de *proxy cachés* de la CDN.

La selección de contenido a replicar también requiere de una estrategia de gestión adecuada para replicar dicho contenido web. Dependiendo de la forma en que se seleccionan los objetos web embebidos para su replicación, la replicación parcial se puede subdividir en *empírica*, basada en *popularidad*, *objetos* o *clúster* (véase Figura 44). En la modalidad ***empírica*** [Che_03], el administrador del *sitio web* selecciona de manera empírica el contenido que debe ser replicado en los servidores extremos (*surrogates*). Aunque se hace uso de heurísticas para tomar esta decisión empírica, existe un cierto grado de inseguridad sobre si los parámetros heurísticos son siempre los más adecuados. En un esquema basado en ***popularidad***, los objetos más populares son replicados en los *surrogates*. Esto introduce una cierta carga en el sistema, y tampoco se puede asegurar siempre unas estadísticas fiables de la popularidad de los objetos web ya que pueden variar considerablemente (por ejemplo, para objetos web nuevos estos datos simplemente no están disponibles). En un esquema basado en ***objetos***, el contenido se replica hacia los servidores en unidades de objetos. Se trata de un esquema voraz (*greedy*) porque cada objeto se replica en el *surrogate* (bajo ciertas limitaciones de almacenamiento) de tal forma que se obtiene la mayor ganancia en el rendimiento [Che_03] [Wu_06]. Pese a que el rendimiento pueda ser óptimo (o subóptimo), se requiere una elevada complejidad a la hora de implementarlo en aplicaciones reales. Finalmente, en un esquema basado en ***clúster***, el contenido web se agrupa atendiendo a algún criterio (por ej. frecuencia de acceso) y se replica en unidades de *clústeres* hacia los *surrogates*. El procedimiento de *clústering* se realiza o bien fijando el número de *clústeres* o fijando el diámetro máximo del *clúster*, ya que no pueden ser conocidos a priori. Por otro lado, el *clústering* de contenido puede estar basado en la sesión de usuario o en la URL (véase Figura 44). En la modalidad basada en la ***sesión de usuario*** [Fuj_04], se emplean los *weblogs* para conformar un *clúster* a partir de las sesiones de

los usuarios que muestran características similares. Esto resulta útil porque ayuda a determinar tanto los grupos de usuarios con patrones similares de navegación así como los grupos de páginas que tienen contenido relacionado. En la modalidad basada en URL, el *clústering* de contenido web se realiza dependiendo de la topología del sitio web [Che_03] [Coo_01]. Los objetos web más populares son identificados para un sitio web y se replican en unidades de *clústeres* donde la distancia de correlación entre cada par de URLs está basada en una métrica de correlación determinada. Aunque este tipo de esquemas de *clústering* reducen el tiempo de descarga y la carga en el servidor, su despliegue resulta altamente complejo.

2.5.4.2.2. Ubicación de surrogates

Dado que la ubicación de los *surrogates* está íntimamente relacionada con el proceso de distribución de contenido, es conveniente prestar especial atención a la hora de seleccionar la mejor ubicación para cada *surrogate*.

El objetivo en la ubicación óptima de *surrogates* es reducir la latencia percibida por el usuario al acceder a contenido, así como reducir el ancho de banda global en la transferencia de datos desde los *surrogates* a los clientes. La optimización de estas dos métricas conlleva una reducción en los costes de infraestructura y comunicación del proveedor de CDN, lo que permite ofrecer servicios de mejor calidad a un coste menor [Siv_07]. En la Figura 45 se pueden apreciar las diferentes estrategias que se pueden emplear en la ubicación de servidores.

Las estrategias ***teóricas***, como el problema de la métrica *k* mínima (*minimum k-center problem*) o los k-HST (*k-Hierarchically well-Separated Trees*) modelan el problema de la ubicación de servidores como un problema de ***ubicación centrada*** (*center placement problem*) que se define de la siguiente forma: para la ubicación de un número concreto de servidores centrados, minimizar la máxima distancia entre un nodo y el centro servidor más cercano. El algoritmo k-HST [Bar_96] [Jam_00b] resuelve el problema empleando teoría de grafos. En este método, la red se representa como un grafo *G(V,E)* donde *V* es el conjunto de nodos y $E \subseteq V \, x \, V$ es el conjunto de enlaces. El algoritmo tiene dos fases. En la primera fase, se selecciona un nodo arbitrario del grafo completo (partición padre) y todos los nodos que están dentro de un radio aleatorio desde ese nodo arbitrario forman una nueva partición (partición hija). El radio de la partición hija es *k* veces más pequeño que el diámetro de la partición padre. Este proceso continúa hasta que cada uno de los nodos se encuentre en una partición propia. De esta forma, el grafo se divide de forma recursiva y se obtiene un árbol de particiones siendo el nodo raíz la red entera, mientras que los nodos hoja serían los nodos individuales dentro de la red.

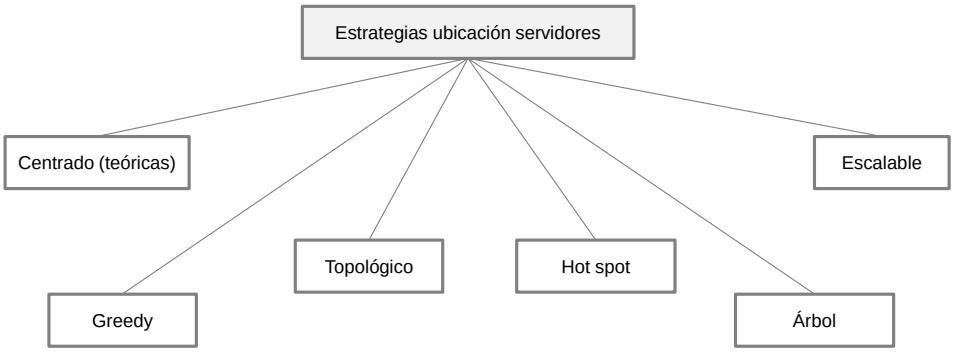


Figura 45. Estrategias de ubicación de servidores.

En la segunda fase, se asigna un nodo virtual a cada una de las particiones en todos los niveles del árbol. Cada nodo virtual en una partición padre desempeña la función de 'padre' de los nodos virtuales en las particiones hija; de esta forma, los nodos virtuales también forman un árbol. A continuación, se aplica una estrategia de tipo voraz (*greedy*) para encontrar el número de centros servidores necesarios para el árbol k-HST obtenido cuando la distancia máxima entre un centro y un nodo está acotada por un valor *D*. Por otro lado, la segunda estrategia teórica, asociada al problema de la métrica *k-mínima* se plantea como un problema basado en grafos bi-dimensionales. [Jam_00]

Debido a la complejidad de los algoritmos teóricos y a la carga computacional que ello implica, se han introducido métodos heurísticos, como los métodos ***greedy*** (*greedy replica placement*) y ***topológicos*** *(topology-informed placement)*. Estos algoritmos subóptimos tienen en cuenta la información disponible en la CDN, como los patrones de carga y la topología de la red, proporcionando una solución adecuada con una carga computacional reducida. Por un lado, el algoritmo ***greedy*** [Kri_00b] selecciona *M* servidores entre *N* ubicaciones (*sites*) disponibles. En la primera iteración, se calcula el coste asociado con cada ubicación, teniendo en cuenta el acceso de todos los clientes a dicha ubicación. En la segunda iteración, el algoritmo *greedy* busca una segunda ubicación (la siguiente con menor coste) en conjunción con la primera ubicación seleccionada. La iteración continúa hasta que se hayan seleccionado *M* servidores. El algoritmo *greedy* es efectivo incluso con datos imprecisos, pero requiere el conocimiento de las ubicaciones de los clientes en la red y todas las distancias entre nodos. En el método ***topológico*** [Jam_01], los servidores se ubican en *hosts* candidatos en orden descendiente de grado (el grado viene determinado por el número de enlaces que conectan un nodo con otro). En este método se asume que los nodos de mayor grado pueden alcanzar otros nodos con una menor latencia. El algoritmo emplea las topologías de sistemas autónomos (*AS, Autonomous System*) donde cada nodo representa un AS y un enlace corresponde con una relación de peering en el protocolo BGP (*Border Gateway Protocol*). Si se desea mejorar la estrategia basada en la topología [Rad_01], se puede emplear la topología a nivel de *router* en lugar de la topología a nivel de AS; en este caso, cada red de área local (*LAN, Local Area Network)* asociada con un *router* corresponde con una ubicación potencial, en lugar de un *AS*.

Otros algoritmos de ubicación son ***Hot spot*** [Qiu_01] y los basados en ***árbol*** [Li_99]. El algoritmo ***Hot spot*** ubica a los *surrogates* cerca de los clientes que generan mayor carga de tráfico. Básicamente, lista las *N* ubicaciones potenciales en base a la cantidad de tráfico generada por los clientes, y luego ubica los *surrogates* en las *M* ubicaciones que generan mayor tráfico. El algoritmo basado en ***árbol*** asume que la topología subyacente son árboles, y modela la estrategia de ubicación como un problema de programación dinámico. De esta forma, un árbol *T* se divide en varios árboles T_i y la ubicación de *t* proxies se consigue al colocar t'_i proxies de la mejor manera en cada árbol T_i, siendo $t = \sum_i t_i^{'}$.

En cuanto a los ***métodos escalables*** en la ubicación de servidores, un ejemplo lo constituye SCAN (*Scalable Content Access Network*) [Che_02b], que es un marco de gestión de réplicas (*surrogates*) escalable que genera réplicas bajo demanda y las organiza dentro de un árbol *multicast* de nivel de aplicación. Esta aproximación minimiza el número de réplicas a la vez que se cumplen los requisitos de latencia de los clientes los requisitos de capacidad de los servidores.

Una vez vistos los algoritmos o estrategias de ubicación de servidores, y retomando la Figura 43, los administradores de CDNs también determinan el número óptimo de *surrogates* empleando una aproximación ***mono-ISP*** o ***multi-ISP*** [Vak_03]. En el caso ***mono-ISP***, el proveedor de CDN típicamente despliega al menos 40 *surrogates* en el extremo de la red del ISP para dar soporte a la distribución de contenido [Dou_01]. La política empleada en este caso es ubicar uno o dos *surrogates* en cada ciudad ampliamente poblada donde de servicio dicho ISP. La desventaja de este método es que los *surrogates* pueden estar ubicados lejos de los clientes a los que el proveedor de CDN desee dar servicio. En el método multi-ISP, el proveedor de CDN ubica varios *surrogates* en el mayor número de puntos de presencia (*PoP, Point of Presence*) de ISPs como sea posible, con la finalidad de estar lo más cerca posible de los clientes. Las grandes CDNs comerciales como Akamai disponen de más de 25000 servidores repartidos por todo el mundo de esta forma [WWW_Aka] [Dil_02]. Más que el coste y la complejidad en la configuración, el mayor inconveniente de este método multi-ISP es que cada *surrogate* pueda recibir un número reducido de peticiones (o incluso ninguna petición), lo que conlleva a una infrautilización de la infraestructura y a una posible baja valoración del rendimiento de la CDN en su conjunto [Pal_06]. Algunas estimaciones de rendimiento [Dou_01] indican que el método mono-ISP funciona mejor para tráficos de red bajos o moderados, mientras que el método multi-ISP funciona mejor para tráficos de red elevados, por lo que son adecuados para sitios web con gran carga.

2.5.4.2.3. Externalización

La externalización de contenido (*content outsourcing*) hace referencia al mecanismo de distribución desde el servidor origen hacia los *surrogates*. Como se mostraba en la Figura 43 existen tres mecanismos de distribución:

- ***Push cooperativo***: en este sistema el contenido se envía desde el servidor origen hacia los *surrogates* en un primer momento y, posteriormente, estos *surrogates* cooperan entre ellos para reducir costes de replicación y actualizaciones. En este esquema, la CDN mantiene un mapeo entre el contenido y los *surrogates*, y cada una de las peticiones de clientes se encaminan al *surrogate* más cercano o (en última instancia) al servidor origen. Desde esta perspectiva, resulta adecuado emplear un algoritmo de tipo *greedy* para realizar las decisiones de replicación entre *surrogates* cooperativos [Kan_02]. En cualquier caso se trata de un esquema teórico puesto que no se emplea por ninguna CDN comercial [Che_03] [Fuj_04].

- ***Pull no cooperativo***: en este caso, las peticiones de los clientes son dirigidas hacia sus *surrogates* más cercanos. Si se produce un fallo (*cache miss*), los *surrogates* obtienen el contenido del servidor origen. La mayor parte de las CDNs comerciales (Akamai, Mirror Image) emplean este mecanismo. El principal inconveniente de este esquema es que no siempre se toma un *surrogate* óptimo para servir contenido [Joh_00]. Sin embargo, la mayoría de CDNs comerciales estiman que es una solución acertada mientras el esquema *push cooperativo* se encuentre en una etapa experimental [Pal_06].

- ***Pull cooperativo***: la diferencia con el esquema anterior es que, en este caso, los *surrogates* cooperan entre ellos para obtener el contenido en caso de un fallo (*cache miss*). Mediante el empleo de un índice distribuido, los *surrogates* son capaces de localizar contenido en *surrogates* cercanos y almacenarlos en caché. Este esquema es reactivo desde el punto de vista de que un objeto es cacheado sólo cuando un cliente lo solicita. La CDN académica Coral, descrita en el capítulo 2.5.3.2, emplea este esquema de externalización de contenido.

En el contexto de la externalización de contenido, resulta extremadamente importante determinar en qué *surrogates* se debe replicar qué contenido. Se pueden encontrar varios trabajos en la literatura científica demostrando la efectividad de diversas estrategias de replicación. En [Kan_02] se describen cuatro métodos heurísticos, como son aleatorio (*random*), basado en popularidad, *greedy* simple y *greedy* global. En [Tse_05] se presentan otras estrategias de tipo *greedy* donde el contenido se distribuye balanceando la carga y la capacidad de los *surrogates*. Pallis et al. [Pal_05] describen un algoritmo autoajustable sin parámetros (de entrada) denominado *lat-cdn* para ubicar de forma óptima contenido en una CDN. Este algoritmo emplea la latencia de los objetos

para tomar las decisiones de replicación. La latencia de un objeto se define como el retardo experimentado desde que se solicita un objeto hasta que se obtiene completamente. Una mejora sobre este algoritmo la constituye *il2p* [Pal_06b], que ubica el contenido en los *surrogates* dependiendo no sólo de la latencia sino también de la carga de dichos objetos.

2.5.4.2.4. Organización de la caché y gestión

La correcta gestión del contenido es fundamental para un buen rendimiento en una CDN, que está principalmente basado en el esquema de organización de *caché* empleado, que abarca tanto las técnicas de *caching* como la frecuencia de las actualizaciones para asegurar recentud, disponibilidad y fiabilidad del contenido. Además, es posible el uso de forma integrada en la infraestructura de la CDN de *caching* y replicación con la finalidad de introducir mejoras potenciales en la latencia percibida, *hit ratio* y *byte hit ratio* [Sta_06]. Es más, el uso conjunto de ambas técnicas permite aumentar la robustez del sistema frente a eventos de tipo *flash-crowd*. En este contexto, en [Sto_03] se describe un método heurístico no paramétrico que integra ambas técnicas y evalúa el nivel de integración. Otro ejemplo, denominado Hybrid [Bak_05], combina replicación estática con *web caching* empleando un modelo analítico basado en *LRU*.

Una taxonomía de las técnicas de *caching* se muestra en la Figura 46, donde se aprecian las dos opciones básicas, inter-clúster e intra-clúster, así como los subtipos soportados.

El caching en una CDN puede ser intra-clúster o inter-clúster.

En un esquema basado en petición (*query*) [Wes_97], un *surrogate* de la CDN realiza una operación de *broadcast* a otros *surrogates* (con los que coopera) si se produce un fallo (*cache miss*). El problema principal de este esquema es el exceso de tráfico en la red durante la petición, así como el retardo incurrido, ya que el *surrogate* debe esperar hasta la última contestación de fallo para determinar cuándo ninguno de sus *peers* tiene el contenido solicitado. En un esquema de tipo *digest*, cada uno de los *surrogates* dispone de un resumen (también denominado *digest*) con la información disponible en el resto de *surrogates* cooperativos, que se actualiza periódicamente. Consultando este resumen (*digest*), un *surrogate* sabe a qué otro *surrogate* cooperativo debe solicitar el contenido. La principal desventaja de este método es precisamente la sobrecarga de tráfico en las actualizaciones para garantizar que los *surrogates* cooperativos disponen de la información correcta. El esquema basado en directorio (*directory*) [Gad_97] no es más que una versión centralizada del esquema *digest*, donde un servidor centralizado almacena la información de localización de contenido de todos los *surrogates* cooperativos.

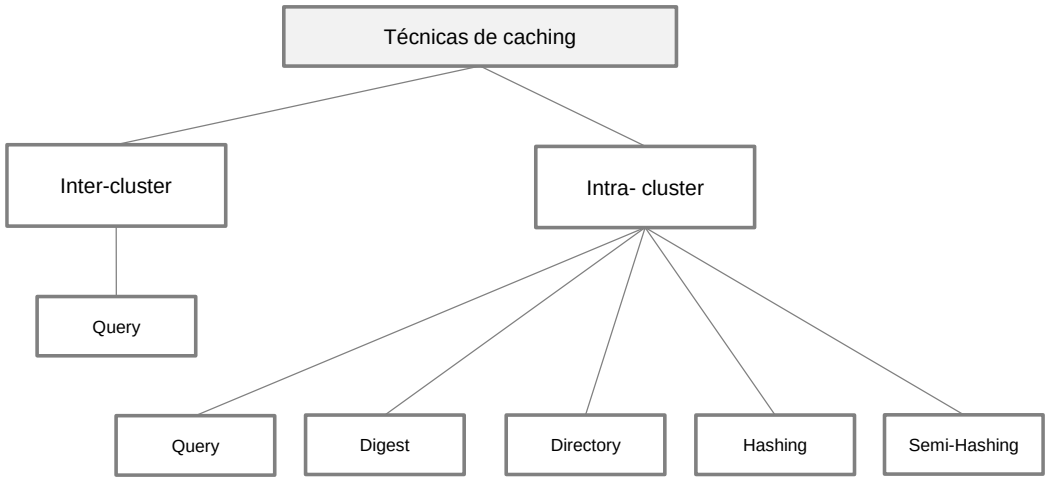


Figura 46. Taxonomía de técnicas de caching

Los *surrogates* sólo notifican al servidor de directorio cuando se producen actualizaciones locales, y lo consultan siempre que se produce un fallo local (*cache miss*). Este mecanismo representa un *cuello de botella* potencial y un punto centralizado de fallo. En un esquema basado en *hashing* [Kar_99] [Kar_98], los *surrogates* cooperativos emplean la misma función *hash*. Cada uno de los *surrogates* almacena un cierto contenido en base a la URL del contenido, las direcciones IP de los *surrogates* y la función *hash*. Los esquemas basados en funciones *hash* son los más eficientes al tener menor sobrecarga de implementación, pero no escala bien con peticiones locales y contenido multimedia ya que las peticiones pueden ser encaminadas a un *surrogate* lejano (en lugar de uno local y cercano). Para solventar este problema, se puede emplear un esquema *semi-hashing* [WWW_Wccp] [Jia_05]. En este caso, un *surrogate* divide su espacio de almacenamiento en dos zonas: una primera zona donde se aloja el contenido más popular para sus usuarios locales (incrementando el *hit rate* local), y una segunda zona donde se aloja el contenido indicado por la función *hash* en el modo cooperativo.

Nótese en la Figura 46 que las técnicas *inter-clúster* sólo disponen de esquemas basados en peticiones (*query*). Un sistema *hashing* no resultaría apropiado para el *caching* cooperativo *inter-clúster* ya que los *surrogates* están típicamente distribuidos geográficamente y el impacto en la latencia sería considerable. Los esquemas de tipo *digest* y directorio tampoco son adecuados porque el tamaño de la información a almacenar puede ser considerable entre tantos *surrogates* dispersos en diferentes *clústeres*. Por ello se suele emplear un esquema basado en peticiones para un *caching inter-clúster* [Jia_03]. En este caso, cuando un *clúster* no es capaz de servir un cierto contenido, realiza una petición a un *clúster* cercano. Si no consigue el contenido de este *clúster*, puede preguntar a otro *clúster* vecino. Los *surrogates* dentro del *clúster* suelen emplear un esquema de tipo *hashing*, y el servidor representativo de ese clúster (*front end*) solamente contacta con el *surrogate* designado por la función *hash*.

En referencia al proceso de actualizaciones de caché para garantizar la consistencia, una taxonomía general se representa en la Figura 47.

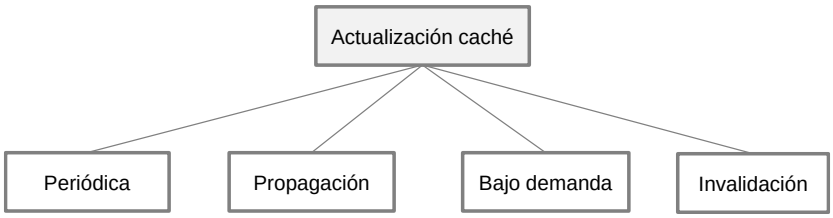


Figura 47. Taxonomía de actualización de caché.

El método más común para actualizar la caché son las ***actualizaciones periódicas***. Para garantizar la consistencia del contenido, el proveedor de contenido configura su servidor origen y proporciona instrucciones a las *cachés* acerca de qué contenido es *cacheable*, durante cuánto tiempo un contenido se considera válido, cuándo se debe interrogar al servidor origen por contenido actualizado, etc. [Gay_04]. De esta forma, las *cachés* se actualizan de forma periódica. Sin embargo, este mecanismo incurre en niveles significativos de tráfico generado en cada intervalo de actualización. En un esquema de ***propagación*** la actualización sólo se produce cuando hay un cambio en el contenido, y se notifica a los *surrogates* (*pushing*) desde el servidor origen. El problema en este caso es que en una situación de cambio frecuente de contenido en el servidor origen, se genera demasiado tráfico de actualización de datos hacia los *surrogates*. En una actualización ***bajo demanda*** sólo se actualiza el contenido en los *surrogates* cuando es solicitado explícitamente por éste. La desventaja de este mecanismo es el tráfico generado entre *surrogate* y servidor origen para garantizar que el objeto solicitado por el cliente es el último actualizado. Finalmente, otro método de actualización lo constituye la ***invalidación***, donde se envía un mensaje de invalidación a todas las *cachés* cuando se realiza un cambio en el servidor origen, y los *surrogates* bloquean dicho contenido hasta que no se obtiene nuevamente del servidor. Generalmente, las CDNs comerciales ofrecen a los proveedores de contenido el control sobre el grado de actualidad (*freshness*) y garantizan la consistencia entre todos los múltiples sites de la CDN. Por otro lado, los proveedores de contenido pueden crear sus propias políticas de caching específicas. En este caso, el proveedor de contenido debe especificar sus políticas de caching en un formato único de cada proveedor de CDN, quien propaga el conjunto de reglas a sus *surrogates*.

CDN	Selección de contenido y distribución	Ubicación de surrogates	Externalización	Organización de la caché y gestión
Akamai	Completa, Parcial (clúster basado en sesión de usuario)	Multi-ISP	Pull no cooperativo	Inter-clúster Intra-clúster Actualización caché: propagación y bajo demanda
Edge Stream	Parcial (clúster)	Mono-ISP	Pull no cooperativo	Inter-clúster
LimeLight Networks	Parcial (clúster)	Multi-ISP	Pull no cooperativo	Intra-clúster
Mirror Image	Parcial (clúster basado en URL)	Multi-ISP	Pull no cooperativo	Intra-clúster Actualización caché: bajo demanda
CoDeeN	Parcial (clúster)	Multi-ISP	Pull cooperativo	Inter-clúster Intra-clúster

				Actualización caché: bajo demanda
Coral	Completa, Parcial (clúster basado en sesión de usuario)	Multi-ISP	Pull cooperativo	Inter-clúster Intra-clúster Actualización caché: invalidación
Globule	Completa, Parcial (clúster)	Mono-ISP	Pull cooperativo	Inter-clúster Intra-clúster Actualización caché: adaptativa

Tabla 8. Distribución de contenido y gestión. Correspondencia a las principales CDNs.

2.5.4.3. Encaminamiento y redirección

El sistema de encaminamiento es responsable de encaminar las peticiones de los clientes a un *surrogate* adecuado que le sirva el contenido. La aproximación más sencilla consiste en redirigir la petición al *surrogate* más cercano al cliente. Sin embargo, el *surrogate* más cercano no siempre es el mejor servidor para satisfacer al cliente [Che_05], por lo que es necesario considerar una serie de métricas como son la proximidad de red, la latencia percibida por el cliente, la distancia y la carga de los *surrogates*. La selección de contenido y distribución en una CDN, mencionada en el capítulo 2.5.4.2.1, tiene un impacto directo en el sistema de encaminamiento.

Este sistema de encaminamiento y redirección consta básicamente de dos partes: el algoritmo de redirección y el mecanismo de redirección [Siv_04]. El algoritmo de redirección se invoca cada vez que se recibe una petición e indica cómo seleccionar un *surrogate* remoto adecuado para ese cliente. Por otro lado, el mecanismo de redirección es una forma de notificar al cliente el *surrogate* que debe contactar.

El proceso de encaminamiento y redirección (interacción cliente-servidor) se puede describir en varios pasos básicos:

- El cliente solicita un contenido introduciendo una URL en su navegador, contactando en primera instancia con un servidor origen.
- El servidor origen proporciona únicamente información básica (objetos de poco tamaño), mientras que los objetos de mayor tamaño y frecuentes, susceptibles de consumir un elevado ancho de banda, son redirigidos por parte del servidor origen a un espacio de nombres de una CDN.
- La CDN emplea su algoritmo de redirección para decidir el *surrogate* más adecuado al cliente y servirle los objetos solicitados.
- El *surrogate* seleccionado obtiene los objetos del servidor origen, sirve el contenido y lo *cachea* para siguientes peticiones.

2.5.4.3.1. Algoritmos de redirección

Los algoritmos de redirección se pueden clasificar básicamente en dos categorías:

- ***Adaptativos:*** consideran las condiciones en tiempo real de la red y/o los servidores para decidir el *surrogate* más adecuado a partir de ciertas métricas (congestión en los enlaces, carga de los servidores, etc.)

- ***No adaptativos:*** emplean ciertas heurísticas para tomar la decisión en lugar de considerar las condiciones del sistema en tiempo real.

Los algoritmos no adaptativos son más sencillos de implementar, ya que los adaptativos suelen ser más complejos al tener que adaptarse a las condiciones cambiantes de la red y/o de los servidores en tiempo real. Esto los hace más robustos [Wan_02] que los no adaptativos, especialmente en los eventos de tipo *flash-crowd*, ya que estos últimos (no adaptativos) sólo son eficientes cuando se cumplen las condiciones o asunciones de las heurísticas.

El algoritmo de encaminamiento no adaptativo más común y sencillo es uno de tipo *round robin*, que distribuye todas las peticiones a los *surrogates* de la CDN balanceando la carga entre ellos [Szy_03], asumiendo la misma capacidad para todos los servidores. Estos algoritmos sencillos resultan eficientes para clústeres, donde los servidores están ubicados físicamente en una misma localización [Pal_98]. Sin embargo, no son adecuados en un entorno WAN donde los servidores están ubicados en lugares distintos, y un cliente podría ser redirigido a un *surrogate* lejano, con una latencia percibida claramente mejorable. Por otro lado, el objetivo principal del balanceo de carga puede no quedar satisfecho ya que cada petición puede ser diferente y requerir cargas computacionales distintas.

En otro ejemplo de algoritmo no adaptativo, los *surrogates* se ordenan dependiendo de la carga esperada en cada uno de ellos. Dicha predicción está basada en el número de peticiones que cada servidor ha satisfecho, y considera también la distancia cliente-servidor. De esta forma, se balancea la carga entre los servidores en base a estos dos parámetros [Siv_04]. Si bien el algoritmo resulta eficiente [Agg_98], la latencia percibida por el cliente es mejorable.

Sin embargo, en la mayoría de estos algoritmos se puede producir una redirección a un servidor sobrecargado, lo que degrada el rendimiento percibido por el usuario. En [Kar_99] se propone un algoritmo que se adapta a efectos *flash-crowd*, basado en el cálculo de una función *hash* considerando la URL del contenido. Este cálculo determina un encaminamiento eficiente a un anillo lógico de servidores caché. Algunas variaciones de este algoritmo se han empleado en contextos de caching intra-clúster [Ni_05] [Ni_03] y sistemas P2P de compartición de ficheros [Bal_03].

Globule [Pie_06] emplea un algoritmo de encaminamiento adaptativo que selecciona la réplica más cercana a los clientes en términos de proximidad de red [Szy_03]. La estimación de la métrica en Globule se basa en la longitud del trayecto (*path*) que se actualiza de manera continua. El servicio de estimación de métrica en Globule es pasivo, con lo cual no se introduce tráfico adicional en la red. Sin embargo, en [Huf_02] se muestra que dicha estimación de la métrica no es muy exacta.

Otros algoritmos de encaminamiento adaptativos [And_02] [Ard_01] están basados en la distancia cliente servidor. En estos, se tiene en cuenta o bien los *logs* de acceso de clientes o medidas pasivas de latencia en los servidores. De esta forma, se encamina una petición al *surrogate* que reporta la menor la latencia. Si bien estos algoritmos resultan eficientes, se requiere el mantenimiento de una base de datos central de medidas, lo que limita la escalabilidad del sistema [Siv_04].

Akamai [WWW_Aka] [Dil_02] emplea un algoritmo complejo de encaminamiento adaptativo muy enfocado a evitar *flash-crowds*. El algoritmo considera una serie de métricas, como son la carga de los *surrogates*, la fiabilidad de las cargas entre cliente y *surrogates* y el ancho de banda disponible en cada *surrogate*. El algoritmo es propietario de Akamai y los detalles tecnológicos no están disponibles.

2.5.4.3.2. Mecanismos de redirección

Los mecanismos de redirección de clientes indican al cliente la selección de *surrogate* llevada a cabo por el algoritmo de encaminamiento. Estos mecanismos pueden ser clasificados dependiendo de varios criterios, aunque típicamente se tiene en cuenta la forma de procesar la petición. La Figura 48 muestra esta clasificación.

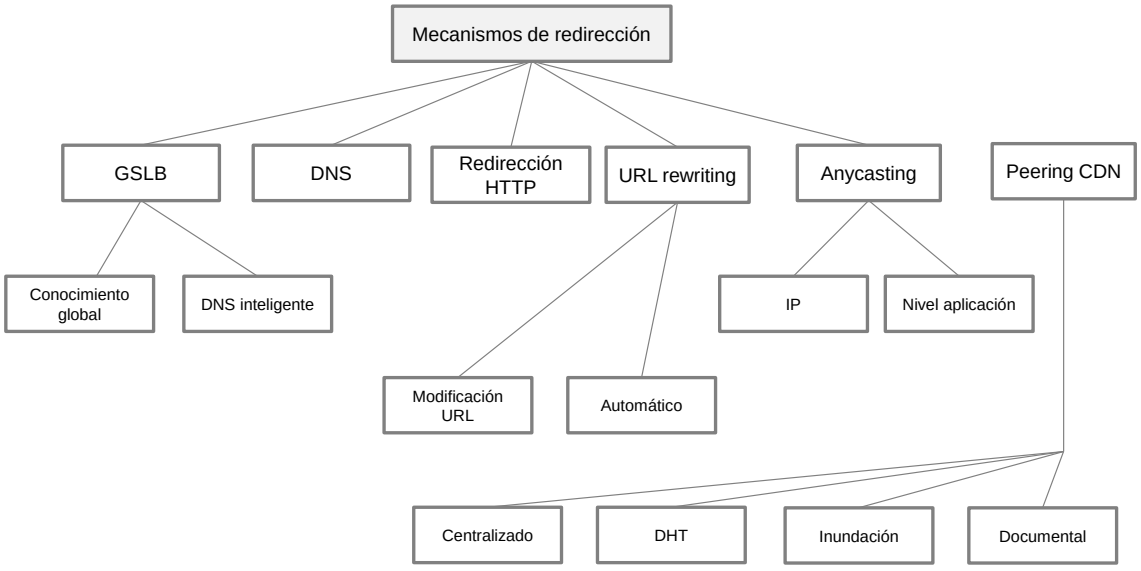


Figura 48. Taxonomía de mecanismos de redirección

En el ***balanceo global de carga (GSLB, Global Server Load Balancing)*** se dispone de un *switch* global y un conjunto de *surrogates* desplegados por todo el mundo [Hof_05]. Existen dos capacidades especiales de los *surrogates* que les permiten soportar balanceo de carga global. La primera de ellas es el conocimiento global (*global awareness*) y la segunda un DNS autoritativo inteligente [Hof_05]. En el balanceo de carga local, cada nodo (servidor) es consciente del estado y rendimiento de los servidores directamente conectados. En GSLB, existe un nodo (*switch*) que es consciente de la información disponible en los otros nodos e incluye sus IPs virtuales en su lista de servidores conocidos. En casa *site* de la CDN se dispone de un switch que tiene un conocimiento global del resto de switches así como de su información de rendimiento. Para hacer uso de este conocimiento global, los switches GSLB actúan a modo de servidores DNS autoritativos para ciertos dominios. La gran ventaja del mecanismo GSLB es que dado que cada nodo (*switch*) dispone de un conocimiento global del resto de nodos (*switches*), es capaz de seleccionar el *surrogate* más adecuado para cada petición, ya sea local (conectados directamente al *switch*) o global (conectados a un *switch* remoto). Otra ventaja significativa de GSLB es que el administrador de red puede añadir capacidades GSLB sin necesidad de introducir dispositivos adicionales en la red. El mayor inconveniente de GSLB es que típicamente implica una configuración manual de los *switches*.

En el mecanismo de ***redirección basado en DNS***, los servicios de distribución de contenido se basan en servidores DNS modificados capaces de realizar un mapeo entre el nombre de un *surrogate* y su correspondiente dirección IP. Normalmente, un dominio dispone de múltiples direcciones IP asociadas. Ante la llegada de una petición de un cliente, el servidor DNS devuelve una lista de IPs de servidores (*surrogates*) que disponen de una réplica del contenido solicitado. El *resolver* DNS en la parte del cliente debe seleccionar un *surrogate* de dicha lista. Para tomar esta decisión, el *resolver* DNS puede mandar mensajes de prueba hacia los *surrogates* y seleccionar aquel cuyo tiempo de respuesta es menor. También podría tomar información histórica de los clientes basada en el acceso previo a dichos *surrogates*. Los proveedores de servicios CDN, tanto globales (*full-site*) como parciales (*partial-site*) emplean redirección DNS. El rendimiento y la efectividad del encaminamiento basado en DNS se han examinado en varios estudios [Bar_03] [Gay_04] [Mao_02] [Sha_01]. La ventaja de este mecanismo es la transparencia, ya que los servicios y contenidos quedan referenciados por sus nombres DNS, y no por sus direcciones IP.

El esquema DNS es extremadamente popular debido a su simplicidad, independencia y ubicuidad, puesto que está incorporado en el servicio de resolución de nombres, y puede ser empleado por cualquier aplicación de Internet [Siv_04]. La gran desventaja de la redirección DNS es que se incrementa la latencia de red porque se requiere un tiempo adicional en la resolución de nombres. Los administradores de CDNs abordan este problema dividiendo el servicio DNS en dos niveles (alto y bajo nivel) para la distribución de carga [Kri_01]. Otra limitación del sistema DNS es que en la resolución

se dispone de la dirección IP del servidor DNS local, y no de la dirección IP del cliente, por lo que se puede producir una mala resolución si el cliente y el servidor DNS local no están próximos. Finalmente, otro aspecto negativo en la redirección DNS es que no pueden controlar todas las peticiones debido al sistema de *caching* interno, tanto en el ISP como en el cliente. Desgraciadamente, en ocasiones el control puede reducirse únicamente a un 5% de las peticiones [Car_01], y resulta muy conveniente evitar el *caching* en el cliente ya que en caso de error el sistema puede no ser responsivo.

La ***redirección HTTP*** propaga la información acerca de los *surrogates* en las cabeceras HTTP. El protocolo HTTP permite a un servidor indicar a un cliente que contacte con otro servidor. La gran ventaja de este mecanismo es su flexibilidad y simplicidad, además de poder gestionar la redirección con una granularidad fina [Pen_03]. Sin embargo, la redirección HTTP ofrece poca transparencia, al tener que introducir cambios en cada página web; adicionalmente, la sobrecarga en el servidor al gestionar las redirecciones es elevada.

Pese a que la mayoría de las CDNs actuales usan DNS como esquema de encaminamiento, algunos sistemas emplean la ***modificación de URLs*** (*URL rewritting* o *navigation hyperlink*). Normalmente se emplea en la modalidad de replicación parcial donde los objetos embebidos se envían con una URL modificada. En este esquema, el servidor origen redirige a los clientes a diferentes *surrogates* al modificar de manera dinámica la URL de los enlaces de los objetos embebidos (la página principal la sirve el propio servidor origen). Para automatizar el proceso, las CDNs actuales proporcionan scripts especiales que *parsean* de forma transparente contenido web y modifican las URLs embebidas [Kri_01]. El mecanismo de modificación de URLs puede ser proactivo o reactivo. En la modalidad proactiva, se crean las URLs de los objetos embebidos de la página principal antes de cargar el contenido en el servidor origen. En la modalidad reactiva, las URLs se modifican cuando la petición del cliente llega al servidor origen. La gran ventaja de la modificación de URLs es que los clientes no están asociados a un *surrogate* únicamente, ya que las URLs contienen nombres DNS que apuntan a un grupo de *surrogates*. Además, se puede conseguir un elevado nivel de granularidad, ya que cada objeto embebido se puede gestionar de forma independiente. La gran desventaja de este mecanismo es el retardo adicional debido al *parsing* de URLs, así como el posible cuello de botella que puede experimentar alguno de estos elementos embebidos.

El mecanismo de ***anycasting*** puede ser dividido en anycast IP y anycast de nivel de aplicación. En el primer caso, propuesto en [Par_93], se asume que la misma IP se asigna a un conjunto de hosts (servidores) y cada router almacena una ruta en su tabla de encaminamiento al host más cercano a dicho router. De esta forma, routers diferentes disponen de rutas distintas a la misma dirección IP. La desventaja de este método es que tocaría asignar algún rango de direcciones anycast dentro del espacio de direcciones IP, lo cual no se llevó a cabo en IPv4 e implicaría modificar los routers. Por otro lado,

en [Fei_98] se propone un anycast de nivel de aplicación donde el servicio consiste en un conjunto de *resolvers anycast* que realizan el mapeo de nombres de dominio anycast con direcciones IP. Los clientes interaccionan con un *resolver anycast* generando una petición anycast. El *resolver* procesa la petición y genera una respuesta. Una base de datos de métricas, asociadas con los resolvers anycast contiene información de rendimiento sobre cada *surrogate*. Este rendimiento se estima en base a la carga y la capacidad de proceso de cada *surrogate*. Una ventaja del mecanismo anycast a nivel de aplicación es su gran flexibilidad, puesto que es altamente configurable y no implica modificación alguna en los routers de la red. Por el contrario, la mayor desventaja de este mecanismo es que implica cambios tanto en los clientes como en los servidores, lo que repercute en un mayor coste unitario, si el número de clientes y servidores es elevado.

Las ***redes de contenido basadas en tecnología peer-to-peer*** están formadas por conexiones simétricas entre hosts (servidores), y distribuyen contenido en nombre de sus *peers*. En estos casos, una CDN puede alcanzar a un mayor número de usuarios si emplea servidores de otra CDN con la que tiene acuerdos de interconexión. En cierto sentido, la idea es similar a los puntos neutros entre ISPs. Típicamente un proveedor de contenido establece un contrato con un único proveedor de CDN, y este proveedor es el que establece convenios con otras CDNs (*peering CDNs*) en nombre del proveedor de contenido [Pat_07b]. Este tipo de CDNs son más tolerantes a fallos, dado que la red de obtención de información necesaria se puede desarrollar en los propios *peers* en lugar de emplear una infraestructura dedicada. Para localizar el contenido en una CDN de tipo *peering*, se puede emplear un modelo de directorio centralizado, un esquema DHT (Distributed Hash Table), un modelo de inundación o un modelo de encaminamiento documental [Hof_05] [Mil_02].

En un modelo centralizado, los *peers* contactan un directorio centralizado donde todos los *peers* publican el contenido que quiere compartir con el resto de *peers*. Cuando se solicita una petición, se responde con el peer que dispone de dicho contenido. Si hay más de un peer, se selecciona aquel peer con mejores métricas basadas en proximidad de red, mayor ancho de banda, menor congestion o mayor capacidad de almacenamiento. Al obtener la respuesta, el peer cliente contacto el peer seleccionado para obtener el contenido. El mayor problema de este mecanismo es que un directorio centralizado representa un único punto de fallo. Adicionalmente, la escalabilidad de un sistema basado en un directorio centralizado está limitada a la capacidad de dicho directorio.

En los sistemas que emplean DHT, los peers se indexan con claves hash dentro de un sistema distribuido. De esta forma, un peer que albergue el contenido deseado puede ser localizado empleando funciones de descubrimiento [Har_02]. Un ejemplo de protocolo que emplea DHT lo constituye Chord [Sto_03]. La ventaja de este mecanismo es que

permite realizar balanceo de carga al descargar servidores excesivamente cargados hacia peers menos cargados [Bye_03].

En los sistemas que emplean un mecanismo de inundación, la petición de un *peer* se difunde a todos los *peers* conectados con este, quienes a su vez difunden los mensajes a los *peers* directamente conectados. Este proceso continúa hasta que se recibe una respuesta o se alcanza un límite broadcast establecido. Evidentemente, el inconveniente de este mecanismo es que genera tráfico de red innecesario y requiere un excesivo ancho de banda. Es por ello por lo que adolece de problemas de escalabilidad y limita el tamaño de la red [Hof_05]. Un ejemplo de este mecanismo lo constituye el protocolo Gnutella [Abe_02] [Wan_07].

En los sistemas que emplean un modelo de encaminamiento documental se emplea un peer autoritativo para obtener el contenido. Cada peer resulta útil en el modelo, ya que todos contienen, al menos parcialmente, información de referencia [Hof_05]. En este esquema, cada peer es responsable de un rango de identificadores de ficheros. Cuando un peer desea obtener un fichero, envía una petición insertando el identificador de fichero. La petición se reenvía al peer cuyo identificador es más similar al identificador de fichero. Una vez se localiza el fichero, se transfiere al peer solicitante. La gran ventaja de este mecanismo es que puede realizar una búsqueda dentro de un número acotado de pasos O (logn), además de mostrar un buen rendimiento y escalabilidad en grandes redes.

CDN	Técnica de redirección
Akamai	Algoritmos de redirección adaptativos que tienen en cuenta la carga del servidor y otras métricas Combinación de redirección basada en DNS y URL rewriting
Edge Stream	Redirección HTTP
LimeLight Networks	Redirección basada en DNS
Mirror Image	Redirección con balanceo global de carga (GSLB), empleando tanto conocimiento global como DNS inteligente
CoDeeN	Algoritmo de redirección que tiene en cuenta ubicación de la petición, carga sistema, fiabilidad e información de proximidad Redirección HTTP
Coral	Algoritmos de redirección con estimación de la ubicación al explotar medidas de red y almacenar detalles de red Redirección basada en DNS
Globule	Algoritmos de redirección considerando proximidad proporcionada por AS Redirección basada en DNS

Tabla 9. Encaminamiento y redirección. Correspondencia a las principales CDNs.

2.5.4.4. Medidas de rendimiento

Las medidas de rendimiento en una CDN tienen en cuenta la capacidad de ésta para proporcionar a los clientes el contenido o servicio deseado. Generalmente se consideran cinco tipos de métricas [Dou_01] [Gad_00] [Kri_01] por parte de los proveedores de contenido para evaluar el rendimiento de una CDN:

- ***Cache hit ratio***: se define como el ratio entre el número de objetos *cacheados* frente al total de objetos solicitados. Un valor elevado de esta métrica indica que la CDN está empleando una técnica efectiva de caching para gestionar sus *proxy caches*.
- ***Ancho de banda reservado***: hace referencia al ancho de banda empleado por el servidor origen.
- ***Latencia***: indica el tiempo de respuesta experimentado por el usuario. Un tiempo de latencia reducido indica que el servidor origen requiere menor ancho de banda.
- ***Utilización de surrogates***: indica la fracción de tiempo que el *surrogate* en cuestión está ocupado. Esta métrica permite al administrador calcular la carga de CPU, el número de peticiones servidas y el uso de almacenamiento E/S.
- ***Fiabilidad***: las medidas de pérdidas de paquetes se emplean para determinar la fiabilidad de una CDN. Una fiabilidad alta indica que apenas hay pérdidas de paquetes y la CDN siempre está disponible para los clientes.

Las medidas de rendimiento se pueden obtener desde dos puntos de vista:

- ***Medidas internas***: los servidores de una CDN se equipan con capacidad de recoger estadísticas de uso con la finalidad de obtener una medida de rendimiento extremo a extremo. Adicionalmente, se pueden desplegar sondas (hardware o software) a través de toda la red y correlar esta información con los logs de los servidores.
- ***Medidas externas***: adicionalmente a las medidas internas, las medidas externas llevadas a cabo por terceros, como Keynote Systems [WWW_Key] permiten verificar con mayor fiabilidad para los clientes el rendimiento de una CDN. Este proceso es eficiente y fiable puesto que estos terceros soportan el *benchmarking* de redes dispersas, ya que ellos también disponen de servidores de prueba distribuidos por toda la red de Internet. Estos servidores miden el rendimiento de desde la perspectiva del cliente, considerando varias métricas [Vak_03].

Para ambos tipos de medidas, se pueden emplear diferentes técnicas de adquisición de estadísticas de red basadas en diferentes parámetros. Estas técnicas pueden considerar pruebas de red, monitorización de tráfico y realimentación por parte de los *surrogates* (véase Figura 49). Los parámetros más comunes en las estadísticas de red son la proximidad geográfica, la proximidad de red, la latencia, la carga de los servidores y el rendimiento de un servidor en general (agregado de varias métricas internas).

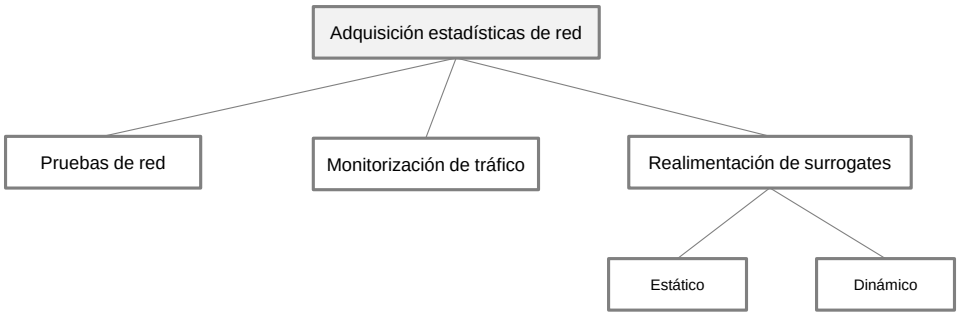


Figura 49. Técnicas de adquisición de estadísticas de red.

Las pruebas de red (*network probing*) son una técnica de medición donde se prueban entidades solicitantes (potenciales clientes) para determinar una o más métricas de uno o varios *surrogates*. Esta técnica se puede emplear en CDNs cooperativas basadas en P2P donde los *surrogates* no están controlados por un único proveedor de CDN. Un ejemplo básico de prueba es el envío periódico de un mensaje ICMP ECHO desde uno o varios *surrogates* a la entidad solicitante. En ocasiones, las pruebas activas no son adecuadas o están limitadas por alguna razón. Téngase en cuenta que introduce latencia adicional en la red que puede resultar significativa para peticiones de objetos web pequeños. En otras ocasiones, la realización de múltiples pruebas hacia una entidad dispara una alerta de detección de intrusos por parte de los denominados IDSs (*Intrusion Detection System*), lo que puede falsear los resultados [Fre_06]. Las pruebas también pueden resultar imprecisas si se emplean métricas derivadas de mensajes ICMP, ya que estos pueden ser ignorados o re-priorizados debido a una detección de un posible ataque de denegación de servicio distribuido (*DDoS, Distributed Denial of Service*). En [35] se muestra un sistema anycasting donde las pruebas ICMP y TCP en puertos elevados son generalmente bloqueadas por cortafuegos.

La monitorización de tráfico es una técnica de medida donde se monitoriza el tráfico entre cliente y servidor para conocer las métricas actuales de rendimiento. Una vez el cliente se conecta, se mide el rendimiento actual de la transferencia de datos, y se envía de nuevo al sistema de encaminamiento a modo de realimentación. Varias medidas posibles son la pérdida de paquetes entre cliente y *surrogate* o la latencia percibida por el cliente al observar el comportamiento del protocolo TCP. Normalmente, la latencia es la métrica más empleada, que se puede estimar sin más que monitorizar el número de paquetes que circulan entre cliente y servidor. Un sistema de estimación de métricas lo constituye IDMaps [WWW_Idm] que mide y disemina información de distancias en todo Internet en términos de latencia y ancho de banda.

La realimentación por parte de los *surrogates* se puede obtener interrogando periódicamente a un *surrogate* mediante una petición específica (por ej. HTTP) y obteniendo medidas relacionadas. También es posible obtener información de realimentación por parte de agentes que estén desplegados en los propios *surrogates*. Estos agentes pueden reportar una gran variedad de métricas sobre sus nodos. Los métodos para obtener esta realimentación pueden ser clasificados como estáticos o

dinámicos. Los métodos estáticos seleccionan una ruta para minimizar el número de saltos, aunque también pueden optimizar otros parámetros estáticos. Por el contrario, las pruebas dinámicas permiten calcular los tiempos de ida y vuelta (RTT, Round Trip Time) y parámetros de QoS en tiempo real [Fran_01].

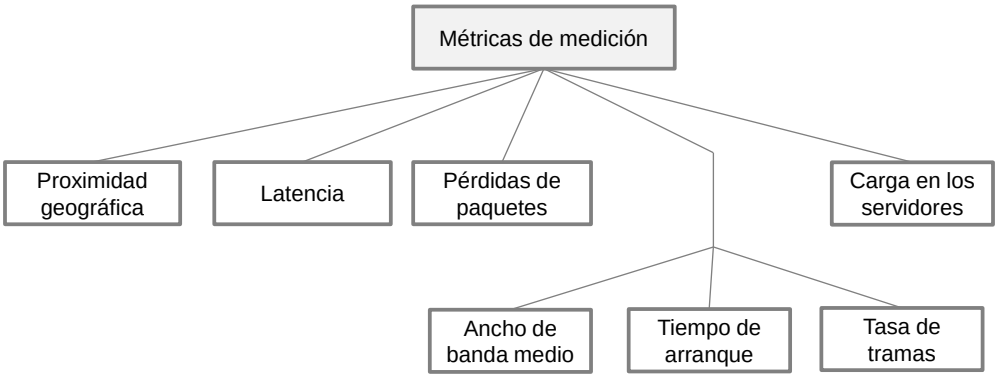


Figura 50. Métricas empleadas para medir el rendimiento del sistema y de la red.

La Figura 50 muestra las diferentes métricas que se usan en las CDNs para medir el rendimiento del sistema (*surrogate*s y servidor origen) y de la red:

- ***Proximidad geográfica***: implica identificar a un usuario dentro de una región. Se emplea para redirigir a todos los usuario de una cierta región al mismo punto de presencia (PoP, *Point of Presence*). La medida de proximidad de red se deriva típicamente a partir de las tablas de encaminamiento del protocolo BGP (*Border Gateway Protocol*).

- ***Latencia***: la latencia percibida por el usuario es una métrica muy útil para seleccionar el *surrogate* más adecuado para dicho usuario.

- ***Pérdida de paquetes***: la contabilidad de pérdidas de paquetes a través de varias rutas de red permite seleccionar la ruta con menor tasa de pérdidas.

- ***Ancho de banda medio, tiempo de arranque, tasa de tramas***: se trata de métricas empleadas para seleccionar la mejor ruta para la distribución de contenido multimedia.

- ***Carga del servidor***: la carga de un servidor se puede estimar en base a métricas internas como carga de CPU, carga del interfaz de red, conexiones activas, y carga de almacenamiento. Esta métrica agregada permite seleccionar el *surrogate* con menor carga.

Además del uso de medidas internas y externas, los investigadores también utilizan herramientas de simulación para medir el rendimiento de una CDN, ya que el acceso a trazas reales de CDNs comerciales no es posible debido a su naturaleza propietaria. Por otro lado, se trata de un método flexible ya que permite modificar fácilmente los parámetros de red (disponibilidad y ancho de banda de un enlace, tamaño de las colas en

los routers, gestión de las colas en los routers, etc.). En la actualidad existe una gran cantidad de simuladores de propósito general [WWW_Sim] que se pueden utilizar para evaluar el rendimiento de varios sistemas de comunicaciones, como puede ser una CDN, aunque requiere configuraciones y adaptaciones sustanciales. Por otro lado, existen también ciertos simuladores orientados a la simulación de CDNs [WWW_CDN2] [WWW_NS2] [Che_03] [Kan_02] [Wan_02], aunque hay que ser riguroso con los resultados, ya que muchos de estos simuladores no consideran aspectos críticos como pueden ser los cuellos de botella que se pueden ocurrir en una red, el número de nodos atravesados, etc.

CDN	Medida de rendimiento
Akamai	Medidas internas: pruebas de red, monitorización de tráfico (proactive) Medidas externas: realizadas por terceros (Giga Information Group)
Edge Stream	Medidas internas: monitorización de tráfico a través de su servicio RPMS (Real Time Performance Monitoring Service)
LimeLight Networks	No disponible
Mirror Image	Medidas internas: pruebas de red, monitorización de tráfico
CoDeeN	Medidas internas: monitorización local de tráfico y sistema
Coral	Medidas internas: monitorización de tráfico y de disponibilidad vía UDP RPC
Globule	Medidas internas: monitorización de tráfico y de disponibilidad de los servidores por parte de los redirectores

Tabla 10. Medidas de rendimiento. Correspondencia a las principales CDNs.

2.5.5. Evolución tecnológica de las CDNs. Integración con cloud computing y CDIs.

Como se ha descrito en secciones anteriores, las CDNs evolucionan desde diversos puntos de vista. Si tenemos en cuenta el ámbito del tipo de contenido ofrecido a los usuarios, las CDNs han evolucionado desde el tradicional servicio web (1ª generación) hacia la capacidad de ofrecer contenido multimedia en formato streaming (2ª generación). Desde el punto de vista de los terminales objetivo, las CDNs se han adaptado para ofrecer todo tipo de contenido a otros terminales diferentes de los tradicionales ordenadores de sobremesa y portátiles. Con la aparición de los *smartphones*, los *tablet PC*, así como del mayor ancho de banda de las comunicaciones móviles, las CDNs son capaces de adaptar y ofrecer contenido (incluso multimedia) a prácticamente cualquier usuario a través de un terminal con conectividad a la red de datos (Internet).

En esta sección se va a abordar la evolución tecnológica de las CDNs desde la perspectiva de la infraestructura considerando dos aspectos aun no tratados en esta tesis,

pero de gran interés en la actualidad. Por un lado, la oferta, prestación y provisión de servicios se está llevando a cabo cada vez más a través de lo que se denomina *cloud computing*, y es importante destacar las diferencias y las convergencias de ambos tipos de tecnologías.

Por otro lado, la interoperabilidad y la globalización de servicios requieren infraestructuras y modelos de negocio versátiles y heterogéneos capaces de satisfacer los requisitos actuales y futuros de los clientes. En este sentido, es conveniente y necesario estudiar y analizar un escenario de interconexión entre CDNs para identificar y especificar los requisitos que permitan una adecuada integración entre ellas.

2.5.5.1. Cloud computing vs. CDNs

La computación en nube (*cloud computing*) es un paradigma de computación que ofrece varios tipos de servicios a través de Internet. Si bien no existe una definición unívoca de este término, se suele tomar la ofrecida por el NIST [Nist_11]. El objetivo principal es permitir a las organizaciones TIC alcanzar una mejora sustancial en la provisión elástica de servicios, tanto en términos de coste como en la propia gestión del servicio. Esta mejora sustancial cobra un mayor significado en los servicios críticos (gestión de emergencias, por ejemplo), donde la disponibilidad debe ser prácticamente del 100%.

Las características principales del *cloud computing* son las siguientes:

- Centralización de aplicaciones, servidores, datos y recursos de almacenamiento.
- Virtualización extensiva de cada componente, incluyendo servidores, aplicaciones, conmutadores, *routers*, *firewalls*, etc.
- Automatización y orquestación de tantas tareas como sea posible (configuración, aprovisionamiento, *troubleshooting*, etc.)
- Elevada confiabilidad de la red.
- Tarificación típicamente basada en el pago por uso (*pay-as-you-go*).
- Desarrollo de estándares que permiten la federación de infraestructuras *cloud*.

De forma general, se pueden distinguir tres clases de soluciones de *cloud computing*:

- ***Privada***: consiste en la implementación de alguna de las características anteriores dentro de un entorno privado (AS o red corporativa), con la finalidad de obtener servicios como recuperación ante desastres, centros de datos privados y virtuales o incluso una computación de alto rendimiento.

- ***Pública***: esta clase a, su vez, se subdivide en los denominados XaaS (*Anything as a Service*):

    - ***SaaS (Software as a Service)****:* representa la capa más alta y caracteriza una aplicación completa ofrecida como un servicio. Ejemplos típicos son *Salesforce.com*, *WebEx* y *Google Docs*. Según un estudio de Gartner [WWW_Gar], el mercado SaaS proporcionará unos ingresos de 22.1 billones de dólares en 2015.

    - ***PaaS (Platform as a Service)***: consiste en empaquetar una serie de módulos que proporcionen una funcionalidad transversal. Algunos ejemplos son *Google App Engine* o *Windows Azure*.

    - ***IaaS (Infrastructure as a Service)****:* representa la capa inferior y es una forma de ofrecer almacenamiento básico y capacidades computacionales como servicios estandarizados en la red. Algunos ejemplos son *Dropbox*, *Amazon*, *IBM*, *Rackspace* y *Verizon*. Según un estudio de Gartner [WWW_Gar], el mercado IaaS crecerá desde los 3.7 billones de dólares en 2011 a 10.5 billones de dólares en 2014.

- ***Híbrida:*** se trata de una composición de dos o más infraestructuras *cloud* (privadas o públicas), que si bien operan de forma autónoma disponen de una conexión entre ambas. Este tipo de combinaciones pueden resultar útil en el contexto del denominado *cloud balancing*, que consiste en encaminar la petición de un servicio a diferentes centros de datos (*nubes*) dependiendo de algún criterio concreto (técnico y/o económico), y se puede considerar como una extensión lógica del concepto de GSLB descrito en la sección 2.5.4.3.

En la mayoría de los casos, los denominados acuerdos de servicio prestado (SLA, *Service Level Agreement*) asociados con servicios de *cloud computing* públicos, como *SalesForce.com* [WWW_Sal], *Amazon Simple Storage System* (S3) [WWW_AmS3] o *Amazon Web Services* (AWS) [WWW_AmAWS] son en cierta medida "débiles" desde el punto de vista del empresario; aunque se garantiza una disponibilidad superior al 99%, hay una tendencia a considerar este tipo de servicios *best effort*. Ello es debido a que la gran mayoría de los SLAs asociados a servicios de *cloud computing* públicos no contienen una cláusula relativa al rendimiento extremo a extremo de dicho servicio. El motivo de esto es precisamente la forma en que dicho servicio se distribuye a los usuarios, que típicamente es a través de un centro de datos de un ISV (*Independent Software Vendor*) ubicado en Internet y, por ello, no es posible proporcionar calidad de servicio (QoS). Esto conduce a la imposibilidad de introducir métricas en los SLAs como retardos, *jitter* o pérdidas de paquetes.

El hecho de que los proveedores de servicios de *cloud computing* (CCSPs, *Cloud Computing Service Providers*) no proporcionen estas métricas en sus respectivos SLAs

no va a cambiar en el futuro cercano. Sin embargo, se están intentando mejoras en este sentido, como puede ser la introducción de técnicas de MPLS o el uso de CDNs que mejoren el rendimiento y la calidad del servicio sobre Internet. En este sentido, la empresa Akamai ofrece una oferta de productos más allá de las meras tecnologías de caching de una CDN, solventando las ineficiencias en los protocolos de aplicación, transporte y encaminamiento.

Lo cierto es que, si bien inicialmente las diferencias entre *cloud computing* (muy básicamente, un conjunto de servidores centralizados) y CDNs (muy básicamente, un conjunto de servidores distribuidos) era obvia, las ineficiencias de Internet (falta de QoS) obligan a una paulatina integración de ambas tecnologías. En 2011, IBM alcanzó un acuerdo con Akamai para integrar su middleware *IBM WebSpehere* sobre algunos servidores de los primeros, de tal forma que se accede una instancia de una aplicación en estos servidores de Akamai, sin necesidad de recurrir a una aplicación centralizada en un centro de datos. Estos acuerdos entre IBM y Akamai están lejos de ser exclusivos. Pero pese a que Akamai es un proveedor dominante de servicios de CDN, existían aquellos que creían que Akamai disponía de una arquitectura propietaria que no propiciaba el despliegue de aplicaciones en la *nube*. Este fue el caso de Cotendo, que en 2011 ofreció un nuevo servicio de CDN denominado *cloudlet*, que permitía llevar la lógica de negocio (*business logic*) a los propios servidores de Internet sin necesidad de tener que ajustar el código; téngase en cuenta que, en los entornos propietarios de CDNs, se requiere que los clientes cumplan una serie de protocolos propietarios para distribuir contenido (en otros términos, la lógica de negocio está centralizada y no distribuida). Poco tiempo después, Cotendo fue adquirido por Akamai.

La sinergia entre *cloud computing* y CDNs resulta especialmente interesante en aplicaciones de video streaming por múltiples razones. En primer lugar, *cloud computing* permite disponer de un espacio de almacenamiento prácticamente ilimitado para almacenar una gran cantidad de vídeos, así como una elevada capacidad de procesamiento para transcodificar dichos vídeos dependiendo del terminal del usuario. Por ejemplo, hace un par de años, Netflix [WWW_Netf] tuvo que transcodificar más de 17.000 películas y shows de televisión para soportar la nueva consola PlayStation 3, enviando más de 80 TB de datos a Amazon. Dado que el material estaba en 19 formatos diferentes y Netflix necesitaba 6 tasas de video distintas además de una tasa de audio, Amazon empleó 1200 instancias EC2 para realizar el trabajo en unos pocos días.

Una vez se dispone del almacenamiento y capacidad de procesamiento para el vídeo, el siguiente paso es el mecanismo de distribución, donde típicamente se emplea una CDN que permite minimizar las latencias, así como los flujos necesarios, ya que los *proxy caché* son capaces de agregar múltiples flujos desde el centro de datos, minimizando de esta forma el consumo de ancho de banda.

2.5.5.2. Content Delivery Interconnection (CDI)

La interconexión de las CDNs viene originada por diferentes motivos, si bien en última instancia se busca conseguir una mejor calidad de experiencia (QoE, *Quality of Experience*). Dado el tamaño de Internet, no parece viable que cualquier CDSP (*Content Delivery Service Provider*) despliegue una gran cantidad de servidores en múltiples redes, al menos inicialmente, ya que las barreras de entrada (costes económicos) serían inasumibles. Akamai es el mayor CDSP y dispone de más de 105000 servidores [WWW_Data] desplegados por todo el mundo, pero este despliegue ha sido más bien gradual desde que se originó la empresa hace algo más de 10 años. Por otro lado, y como se mencionó en esta sección, muchos operadores de red (NSP, *Network Service Providers*) desplegaron su propia infraestructura de CDN, en algunas ocasiones absorbiendo la tecnología de antiguos CDSPs que quebraron tras la burbuja de las .com. Una utilidad actual y a la vez tendencia en los NSPs es la introducción de servicios de valor añadido sobre dicha CDN, siendo el más destacado el video bajo demanda para sus propios clientes. Evidentemente, si gestionan su propia infraestructura pueden proporcionar una distribución muy eficiente con unos tiempos de latencia muy reducidos, pero esta eficiencia está acotada por el dominio geográfico de actuación de dicho NSP. Es por ello por lo que se estudia analizar la interconexión de CDNs (a veces denominadas islas) para incrementar el ámbito geográfico de actuación. Existen también otros motivos que favorecen la interconexión, como pueda ser el balanceo de carga o la prevención del efecto *flash-crowd*.

Sin embargo, pese a su claro potencial en la mejora global del servicio, no existe ningún método estándar para interconectar CDNs hasta la fecha, sino simplemente algunos intentos experimentales de interconexión con la finalidad de especificar métodos e interfaces para dicha interconexión.

La problemática inicial que surge es la referida a los interfaces existentes. Si tenemos en cuenta los diferentes módulos de la arquitectura estándar de una CDN (véase sección 2.5.2.1), los interfaces de cada uno de ellos son específicos de cada fabricante o desarrollador.

2.5.5.2.1. Escenario de operación e interfaces. Requerimientos.

El escenario básico de operación en la interconexión de CDNs se ilustra en la Figura 51, donde dos CDSPs cooperan entre sí para distribuir el contenido proporcionado por los proveedores de contenido a los usuarios finales. Por simplicidad, se asume que la inicialización de los interfaces (CDNI, *Content Delivery Network Interconnection*) ya está establecida. En este escenario, la CDN_1 representa el punto de entrada o ascendente

(*authoritative upstream CDN*), mientras que la CDN_2 representa el punto de salida o descendente (*downstream CDN*). Es decir, la CDN_2 actúa a petición de la CDN_1, quien a su vez lo hace en base al proveedor de contenido. Dado que el usuario final está ubicado próximo a la CDN_2, resulta lógico que sea servido a través de dicha CDN. A continuación se describen los pasos que tienen lugar a través de la interfaz CDNI para permitir la distribución de contenido a un usuario:

- El proveedor de contenido ubica su contenido en la CDN_1.
- La CDN_1 monitoriza en cierta medida la CDN_2. Esto le permite disponer del conocimiento de aquellos usuarios que la CDN_2 puede servir, además de conocer si en el momento de la petición la CDN_2 esta sobrecargada y no puede gestionarla adecuadamente (en ese caso lo hará la CDN_1).
- El usuario solicita un contenido, y la petición es gestionada por el sistema de encaminamiento de la CDN_1, quien determina que la CDN_2 debe servir el contenido (no está sobrecargada). En este caso, se redirige al cliente al sistema de encaminamiento de la CDN_2.
- El sistema de encaminamiento de la CDN_2 determina el *surrogate* más adecuado para servir el contenido y redirige al cliente a éste.
- El cliente contacta con este *surrogate* y obtiene el contenido de éste.

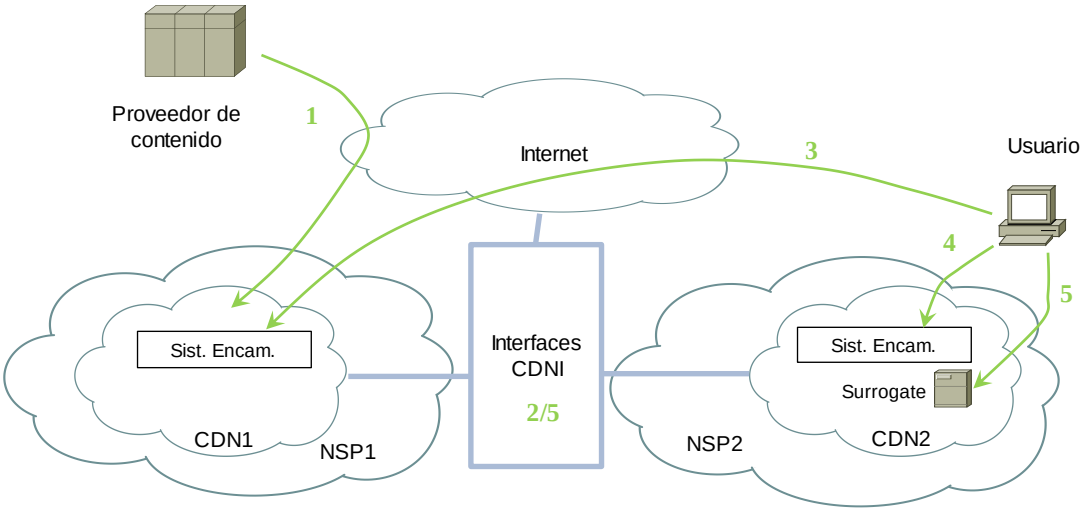


Figura 51. Escenario de operación en CDNI.

Si se analiza la interconexión de las CDNs, uno puede observar que se requiere interconectar los subsistemas de cada CDN, ya que deben intercambiarse información para cooperar. La Figura 52, extraída de un *Internet Draft* [Niv_11], muestra con mayor detalle la interconexión entre CDNs. Las interfaces relativas entre los subsistemas de control, contabilidad, encaminamiento y distribución fueron objeto de estudio del grupo

de trabajo de CDI dentro del IETF (*Internet Enginnering Task Force*), mientras que el resto de interfaces (por ej. adquisición de datos) quedaron fuera de su ámbito de trabajo. Es importante destacar que el grupo IETF CDI estuvo activo del 2001 al 2003 y, una vez consolidado el mercado de CDNs, cesó su actividad. Sin embargo, las especificaciones de interconexión (RFC 3466, RFC 3568, RFC 3570) siguen siendo válidas, dada su generalidad. Recientemente, France Telecom Orange Labs realizó un experimento de interconexión entre dos CDNs diferentes [Ber_11b] donde se consiguió la interconexión parcial pese a los protocolos propietarios de cada CDN. Los resultados de este experimento, junto con las especificaciones anteriores, sirven de punto de partida para el nuevo grupo de CDNI creado en 2011. Existen contribuciones aún más recientes en el contexto de la interconexión, sobre todo para garantizar la distribución de contenido multimedia. Tal es el caso de un documento técnico de la ETSI [ETSI_12]. En este se contemplan casos de uso de federación y *peering* entre CDNs. Otro modelo de CDN *peering* menos reciente puede consultarse en [Buy_06], aunque indica que, desde que se configuró el grupo CDI siempre ha habido intentos de especificar una arquitectura de interconexión de CDNs.

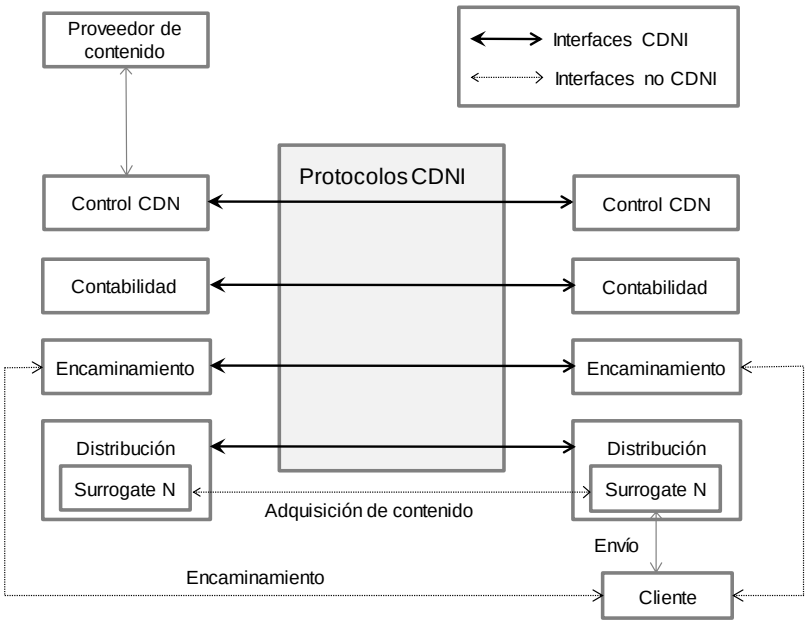


Figura 52. Interfaces CDNI (Fuente: [Niv_11])

A continuación se indican una serie de requerimientos especificados para la interconexión CDNI, aunque para una mayor información se puede consultar directamente en [Niv_11].

- ***Interfaz de control***: esta interfaz permite a los sistemas de control de las CDNs intercambiar información para establecer, actualizar o terminar una interconexión CDN. También es responsabilidad de esta interfaz la configuración y puesta a punto del resto de interfaces.

- ***Interfaz de contabilidad***: permite a las CDNs interconectadas intercambiar información sobre la cantidad y los datos transmitidos. Este tipo de información puede ser usado para tarificar o como fuente de datos para los sistemas de análisis y monitorización. En este último caso es conveniente también proporcionar información adicional, como direcciones IP del cliente y agente de usuario.

- ***Interfaz de encaminamiento***: facilita el intercambio de información entre los subsistemas de encaminamiento de las CDNs ascendente y descendente (recursos disponibles, carga, etc.). Esta interfaz debe ser capaz de soportar los protocolos empleados para redirigir usuarios, como pueden ser DNS, HTTP y RTSP.

- ***Interfaz de distribución***: esta interfaz permite a los subsistemas de distribución de las CDNs interconectadas intercambiar metadatos para así satisfacer adecuadamente las peticiones de los clientes. De esta forma, por ejemplo, la CDN ascendente es capaz de distribuir, actualizar y eliminar metadatos asociados a su contenido que están disponibles en la CDN descendente.

El diseño de estas interfaces, especialmente la última, debe ser flexible y fácilmente extensible para futuras mejoras. Por ejemplo, en el caso de la distribución de vídeo, debería ser capaz de soportar técnicas de video streaming adaptativo como DASH (*Dynamic Adaptive Streaming over HTTP*) [Pu_12] [Loh_11]. Esta técnica permite incrementar la calidad de la experiencia (QoE) cuando se consume contenido de vídeo y es, por ello, una funcionalidad deseable de ser soportada en la interconexión CDNI. Básicamente, el contenido del vídeo se encuentra disponible en varias tasas de bit (*bit rates*). Dependiendo del ancho de banda disponible, se selecciona aquel fichero de video cuya tasa de bit no excede dicho ancho de banda, de tal forma que el usuario siempre experimenta aquel contenido con mejor calidad disponible y sin interrupciones.

Por otro lado, existen ciertos requerimientos propuestos que la interconexión CDNI debe considerar [Ber_11] [Lou_11]. Por un lado, se requiere un pre-posicionamiento de contenido o metadatos dentro de la CDN descendente con la finalidad de reducir el retardo en la distribución y proporcionar el nivel de servicio solicitado por el proveedor de contenido. Por otro lado, el contenido también puede ser capaz de ser solicitado bajo demanda, de tal forma que sólo sea necesaria la información de dónde localizar el contenido actual en la CDN adyacente. Adicionalmente, las CDNs deben proporcionar métodos para asegurar la coherencia de las caches. Por ejemplo, las operaciones de invalidación de contenido también deben alcanzar la CDN descendente, de tal forma que el contenido no esté disponible para los clientes sin una nueva petición al servidor origen (o a la CDN ascendente).

Otro aspecto que se debe considerar en la interconexión CDNI es la gestión de los derechos digitales (DRM, *Digital Rights Management*). Para ello se deben definir

funcionalidades que permitan un bloqueo geográfico que formen parte de la propia especificación CDNI. Mediante estas funcionalidades se evita que ciertas regiones con sus correspondientes rangos de direcciones IP puedan obtener un contenido en particular. De esta forma se puede asegurar que un contenido en concreto esté disponible únicamente dentro de un país concreto.

Una última restricción requerida en CDNs y, por ello, en CDNIs, es la restricción temporal en la disponibilidad de contenido. Esto se emplea para prevenir que los niños pueden acceder a aquel contenido que no se corresponde con su edad, por lo que cierto contenido puede estar sólo disponible por las noches.

2.6. Conclusiones del estado del arte de las CDNs

En este capítulo 2, relativo al estado del arte, se ha partido del crecimiento del tráfico (significativamente multimedia) en Internet para justificar la necesidad de escalabilidad de las redes y la mejora de los mecanismos de distribución de contenido en general, con la finalidad de poder ofrecer mejores servicios a los usuarios de Internet dentro de unos márgenes de calidad aceptables. Desafortunadamente, si bien existen mecanismos de QoS especificados a nivel de red, solamente se encuentran desplegados en entornos intra-AS y no en Internet (inter-AS). Es por ello por lo que la calidad en Internet está típicamente relacionada con la capacidad de ofrecer un tiempo reducido de espera (retardo y *jitter*) a los clientes cuando contactan con un servidor, lo que no es suficiente para algunos tipos de aplicaciones actualmente desplegados.

Para abordar este aspecto, se han propuesto numerosas tecnologías desde prácticamente el comienzo de Internet. Así, el *caching*, aplicado al web y, más recientemente, a objetos multimedia, es capaz de reducir el tiempo de latencia percibido por los usuarios. Estos sistemas de *caching* disponen de varias configuraciones (*forward*, *reverse* e *interception proxy*) e incluso pueden ser desplegados en redes malladas o jerárquicas, especificándose varios protocolos de comunicación (ICP, cache digest, etc.). Por otro lado, los sistemas de réplicas, *mirrors* o espejos duplican la información de un servidor origen en otros servidores remotos con la finalidad de descargar a éste de un exceso de peticiones, además de aumentar la disponibilidad de los datos.

Las redes de distribución de contenidos (CDNs) representan una evolución dentro de las redes de caché y los espejos, integrando ambas tecnologías de una forma eficiente dentro del concepto de *content networking*. El objetivo principal es reducir el tiempo de espera del usuario (latencia percibida) mediante la redirección a un servidor cercano, denominado *surrogate*, que implementa tanto la funcionalidad de *proxy cache* como de *mirror*, aunque parcialmente. Evidentemente, en un entorno de Internet, el número de *surrogates* a desplegar debe ser significativo para minimizar el efecto de la latencia WAN, que resulta impredecible.

Existen diversas formas de clasificar una CDN atendiendo a aspectos de composición, distribución de contenido y gestión, encaminamiento y redirección, etc., como se ha descrito en la sección de taxonomía. El algoritmo de redirección de usuarios es el aspecto más relevante para caracterizar la efectividad de una CDN, pero para ello se deben considerar todos y cada uno de los aspectos relativos a la CDN: cómo está formada, cómo está monitorizada, cómo se externaliza el contenido, etc. En definitiva, es necesario un correcto análisis de la CDN para poder especificar e implementar un algoritmo de redirección efectivo y eficiente.

En esta tesis, se realiza un estudio del funcionamiento de las CDNs para posteriormente poder profundizar en el análisis y evaluación del rendimiento de un algoritmo de redirección. Las aportaciones de esta tesis en los sucesivos capítulos son:

- ***Estudio del funcionamiento de una CDN desde un punto de vista analítico***: de esta forma, se puede caracterizar mediante un modelo matemático una CDN y ver las situaciones de mejora con respecto a un servidor único. Este modelo, aunque sencillo, es relevante, y no se dispone en la literatura relacionada ningún otro modelo analítico general, sino que se centran en aspectos concretos.

- ***Estudio del funcionamiento de una CDN en un entorno de simulación***: en este caso, se refuerzan las conclusiones del modelo matemático, por un lado, y se amplía la caracterización de una CDN al ser capaz de modelar más variables y mecanismos de funcionamiento que tienen lugar en el entorno de una CDN, por otro lado. Adicionalmente, el modelo de simulación se compara con otros modelos propuestos en la literatura (CDNsim, CDN Simulator).

- ***Estudio del funcionamiento de una CDN en un entorno operativo mediante la implementación de un prototipo desarrollado por el doctorando***: en última instancia, los resultados analíticos y simulados se validarán frente a un prototipo de implementación. La implementación también permite evaluar adecuadamente la efectividad del algoritmo de redirección de usuarios. Adicionalmente, la implementación de un sistema real también permite detectar los principales problemas de un sistema completo o, en otros términos, donde reside la mayor complejidad.

- ***Estudio del algoritmo de redirección para contenido web y streaming***: evidentemente, el algoritmo de redirección no sólo depende de cómo está formada una CDN, sino también de la aplicación final que se desea ofrecer al usuario. En esta tesis se analiza el algoritmo de redirección para dos tipos de servicios: (i) por un lado, el servicio web, que era la aplicación tradicional de la primera generación de CDNs y (ii) por otro lado, el servicio de streaming, que corresponde al servicio ofrecido por la actual segunda generación de CDNs.

3. DISEÑO, SIMULACIÓN E IMPLEMENTACIÓN DE UNA CDN

3.1. Planteamiento general y estructura

El objetivo principal de la tesis es el estudio y análisis comparativo de los principales parámetros de funcionamiento de una CDN (retardo medio, carga del sistema, mecanismo de redirección, etc.) en entornos web y en servicios de streaming. Este estudio se realiza de una forma progresiva, de menor a mayor complejidad. En el capítulo 3 se hace un análisis y estudio de una CDN con tráfico web, mientras que en el capítulo 4 el objeto de estudio sigue siendo la CDN, pero con tráfico multimedia, ya que tiene implicaciones en el despliegue y configuración de la arquitectura y el sistema de redirección.

En la primera parte (sección 3.2) se presenta un modelo analítico de una CDN, donde se describe un modelo básico. Este modelo, si bien está bastante simplificado, permite comprender y evaluar el funcionamiento básico de una CDN, y observar el comportamiento de los parámetros de rendimiento (tiempos de respuesta) que definen una CDN conforme aumenta el número de usuarios, la carga de los servidores o la congestión de red.

En la segunda parte (sección 3.3) se presenta un modelo de simulación que permite describir o emular de una manera más fiel el comportamiento de una CDN, al poder introducir más funcionalidades. Para ello, se ha empleado un simulador de redes de comunicaciones (NS-2) de propósito general ampliamente usado en la comunidad científica, y se ha adaptado y configurado para poder simular una CDN. A partir de este modelo, se han realizado numerosas simulaciones para evaluar parámetros de interés como son la carga media y los tiempos medios de respuestas, dependiendo del número de clientes, número de servidores, distribución de acceso, patrón de tráfico, etc.

En la tercera parte (sección 3.4) se presenta un prototipo de implementación de CDN. De esta forma, se dispone de un entorno real controlado y se puede, por un lado, verificar que los resultados de las simulaciones anteriores son correctos y, por otro, evaluar la efectividad del algoritmo de redirección implementado. El algoritmo de redirección es el componente de la CDN donde más se ha profundizado en esta tesis (excepto en el modelo analítico), pues tiene un impacto directo y medible en el rendimiento de una CDN.

Para el modelo analítico, de simulación y de implementación se ha partido de una arquitectura genérica de una CDN, descrita en la sección 2.5.2.1 y se ha adaptado dependiendo de los componentes que se han modelado.

3.2. Modelo analítico

3.2.1. Introducción

El objetivo principal de esta sección consiste en establecer un modelo sencillo mediante el cual se pueda describir el funcionamiento básico de una CDN, así como las ventajas que representa en términos de retardo este esquema de comunicación distribuido.

El principal parámetro de análisis en este modelo analítico será el retardo medio experimentado por los clientes miembros de la CDN, ya que éste constituye un parámetro fundamental en el diseño de cualquier CDN real, como se ha descrito en la sección 2.5. Este retardo será desglosado en varios componentes en la cadena de comunicación con la finalidad de identificar los nodos o enlaces más 'saturados'. De esta forma, se puede analizar el retardo desde dos puntos de vista:

- Retardo medio percibido por el cliente accediendo al servidor origen y a los *surrogates*. Desde esta perspectiva, se puede observar si el retardo mayor se está produciendo en el servidor origen (esto indicaría que el *hit ratio* es reducido y las políticas de *caching* no están acertando demasiado), o si los *surrogates* están generando un retardo considerable (esto indicaría que el mecanismo de redirección no está enviando a un cliente a un *surrogate* cercano o que el *surrogate* está sobrecargado)

- Retardo medio en los enlaces y en el procesado. Desde esta perspectiva, se puede analizar si el retardo experimentado por el cliente se debe a una congestión en la red (retardo elevado en los enlaces) o en los servidores (retardo elevado en el procesado de la petición).

No existe ningún modelo analítico que describa y permita analizar una CDN a un nivel básico (sin excesiva complejidad) y en términos generales (considerando toda la dimensión de la CDN). Los modelos teóricos existentes se restringen a aspectos concretos, como problemas de gestión y asignación de recursos [Bek_08], ubicación de *surrogates* [Qui_01], externalización de contenido [Kan_02], evaluación de modelos de precios [Hos_06] y mecanismos de redirección [Oli_05] [Bek_08]. En ocasiones, los modelos matemáticos también se emplean para estudiar soluciones concretas sobre problemas reales que aparecen en las CDNs reales. Así, en [Ngu_05] se propone una solución basada en un modelo matemático para evaluar el efecto del clustering de datos en el beneficio de un proveedor de CDN. En otras ocasiones, los modelos matemáticos se pueden emplear como punto de referencia para evaluar una serie de métodos heurísticos [Lao_05]. Sin embargo, todos estos modelos tratan con los problemas individuales de forma separada, sin considerar la posible interacción entre ellos. Es por ello por lo que, si bien proporcionan una información de interés, es necesario disponer de un modelo general donde se puedan estudiar todas estas interacciones.

3.2.2. Modelo básico

Una CDN está formada por tres componentes principales desde el punto de vista de distribución, como se muestra en la Figura 53:

- Un *servidor origen*, situado en una posición central,
- P *surrogates* (servidores sustitutos), ubicados en algún lugar entre los clientes y el servidor origen.
- M *clústeres de clientes*, dispersados alrededor del mundo. Un clúster de clientes es una forma de asociar múltiples usuarios situados en una cierta zona. En esta tesis se estudiará este tipo de escenario, de manera que los resultados obtenidos serán para todo el grupo y no para usuarios unitarios. Sin embargo, dada la generalidad del enfoque no resulta complicado extraer un modelo basado en clientes aislados.

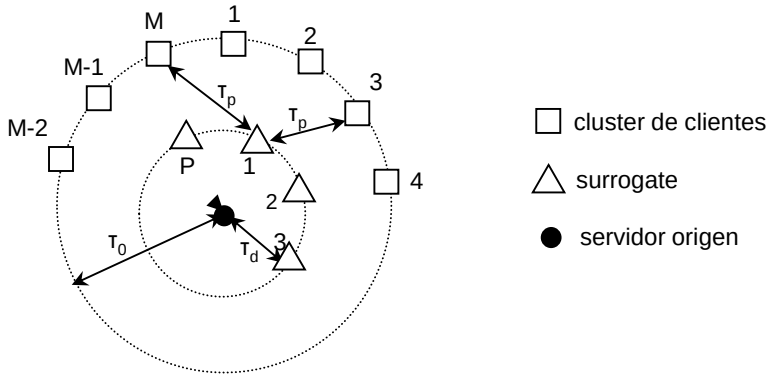


Figura 53. Escenario general de una red de distribución de contenidos.

En una primera aproximación podemos asumir que todos los clústeres de clientes (en lo sucesivo también denominados clientes) se encuentran simétricamente distribuidos alrededor del servidor origen a una cierta distancia temporal o RTT (Round-Trip Time) τ_0. Los *surrogates* se encuentran situados entre el servidor origen y los clientes, a un RTT τ_d y τ_p respectivamente. Un cliente generará peticiones, que serán atendidas por el servidor origen o por los *surrogates*. El mecanismo mediante el cual el cliente conoce el servidor que debe contactar no será considerado; en su lugar, se asumirá que el cliente será encaminado hacia los *surrogates* con una cierta probabilidad *p*. Por el contrario, el cliente contactará el servidor origen con una probabilidad opuesta (*1-p*).

Normalmente, la medida de rendimiento más usada en cualquier análisis de CDN es el tiempo medio de respuesta experimentado por los usuarios. Para ello, se partirá de una fórmula inicial para dicho tiempo de respuesta:

$$R = p * R_{srrgt} + (1-p) * R_{origin} \qquad \text{(E 3.1)}$$

donde R_{srrgt} corresponde al tiempo medio de respuesta asociado al contactar a un *surrogate* (y ser servido por éste), mientras que R_{origin} es el tiempo medio de respuesta

asociado al contactar el servidor origen (y ser servido por éste). Es más, según [Agra_01], el tiempo medio de respuesta se puede representar de una forma lineal mediante la expresión:

$$R = N * \tau + S \qquad \text{(E 3.2)}$$

donde *N* es un factor de escala que incorpora el efecto de las pérdidas de paquetes en la red, retransmisiones y, en general, la cantidad de información intercambiada necesaria para una solicitud; τ representa el tiempo de ida y vuelta en la red (RTT) y *S* es el tiempo de proceso requerido, que por simplicidad será modelado como un sistema M/M/1.

Un sistema M/M/1 [Kle_96] consiste en un buffer FIFO (First-In-First-Out) donde los paquetes llegan de manera aleatoria en base a un proceso de Poisson, mientras que un procesador, denominado servidor, obtiene y sirve a dichos paquetes con una determinada tasa de servicio, como se muestra en la Figura 54.

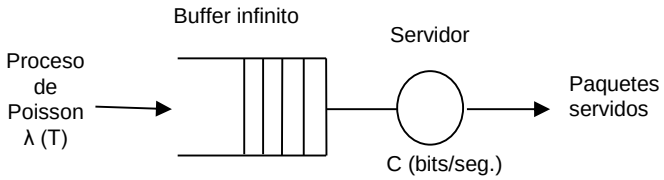


Figura 54. Modelo de una cola M/M/1.

En un sistema M/M/1 los siguientes parámetros y expresiones representados en la tabla inferior son de interés para nuestro modelo de CDN.

Denominación	Parámetro/Expresión
Tasa de llegada Tiempo medio de llegada	$\lambda = \frac{1}{T}\left(\frac{paquetes}{segundo}\right)$
Tamaño medio de los paquetes	$\overline{p}$ (bits/paquete)
Capacidad de servicio	C (bits/ segundo)
Tasa de servicio	$\mu = \frac{C}{\overline{p}}\left(\frac{paquetes}{segundo}\right)$
Retardo medio	$\overline{W} = \frac{1}{\mu - \lambda}\left(\frac{segundos}{paquete}\right)$ con $\mu > \lambda$

Tabla 11. Expresiones matemáticas para un modelo M/M/1

Con estos valores, podemos obtener una nueva expresión para el tiempo medio de respuesta:

$$R = p \cdot \left[N \cdot \tau_p + \frac{1}{\mu_p - \lambda_p} \right] + (1-p) \cdot \left[N \cdot \tau_s + \frac{1}{\mu_s - \lambda_s} \right] \quad \text{(E 3.3)}$$

Las variables que aparecen en la fórmula superior se describirán de forma separada para una mejor comprensión.

La variable *p* denota la probabilidad de que una petición sea satisfecha por un *surrogate*. Téngase en cuenta que un cliente no será siempre redirigido al mismo *surrogate*, sino a varios de ellos. Pese a que en teoría pudiera resultar deseable que un cliente fuese redirigido al *surrogate* más cercano, los mecanismos actuales de redirección empleados en una CDN (redirección mediante DNS) no siempre aciertan de manera precisa la localización del cliente, redirigiéndolo así a otro *surrogate*. No obstante, incluso suponiendo una correcta estimación de la localización, resulta interesante establecer un balanceo de carga entre *surrogates* próximos para prevenir condiciones de sobrecarga en estos servidores. Teniendo en cuenta todo esto, podemos representar el valor de la probabilidad *p* para el clúster *i-ésimo* como la suma de probabilidades en contactar los *P surrogates*.

$$p_i = \sum_{j=1}^{P} p_i^j \quad \text{(E 3.4)}$$

La variable τ_p representa el tiempo de ida y vuelta (RTT) asociado a contactar a los *surrogates*. Nótese que este valor es variable y dependiente del cliente y *surrogate* en cuestión. De esta forma, el valor de RTT para el i-ésimo clúster y hacia el j-ésimo *surrogate* se expresará como $\tau_p^{i,j}$.

Las variables μ_p y λ_p corresponden a la tasa de servicio y la tasa de llegada percibida por cada uno de los *surrogates*. Una vez más, estos valores son distintos para cada *surrogate*. Suponiendo que el clúster *i-ésimo* genera paquetes con una tasa λ_i, la tasa de llegada que percibirá el *j-ésimo surrogate* desde el cliente *i* será:

$$\lambda_i^j = p_i^j \cdot \lambda_i \quad \text{(E 3.5)}$$

Debido a la condición de estabilidad de un sistema M/M/1 ($\mu_p > \lambda_p$) se podría introducir un factor *k* (k>1) de tal forma que la tasa de servicio quedase determinada por este factor y el tiempo de llegada:

$$\mu_p = k \cdot \lambda_p \quad \text{(E 3.6)}$$

El valor del factor k es importante: un valor reducido puede conducir a situaciones límite de carga donde el tiempo de proceso sea remarcable sobre la latencia, mientras

que un valor elevado puede suponer un tiempo de proceso prácticamente instantáneo, donde sólo la latencia (o RTT) afecta significativamente el tiempo de respuesta global.

Las mismas consideraciones pueden aplicarse al servidor origen para las variables *p,* τ_s, μ_s y λ_s. Sin embargo, en esta ocasión el servidor origen está absorbiendo M distribuciones exponenciales (debido a la caracterización como proceso de Poisson) desde cada uno de los clientes. Desde el punto de vista estadístico, la suma de M distribuciones exponenciales se puede contemplar como una nueva distribución exponencial cuya tasa de llegada es la suma de cada una de las tasas de llegada individuales.

Finalmente, y antes de llegar a la expresión final, es importante destacar que la fórmula anterior permite obtener el tiempo medio de respuesta tanto para un clúster de clientes como para todo el sistema global. En este trabajo de investigación se considerará el último caso, que consiste en calcular la media aritmética de cada uno de los tiempos de respuesta obtenidos en cada clúster:

$$\overline{R} = \frac{1}{M} \sum_{i=1}^{M} R_i \qquad \text{(E 3.7)}$$

Tras todo esto ya podemos plantear una expresión general para el tiempo de respuesta medio global:

$$\overline{R} = \frac{1}{M} \cdot \sum_{i=1}^{M} \left\{ \sum_{j=1}^{P} \left[p_i^j \cdot \left(N \cdot \tau_p^{i,j} + \frac{1}{\mu_p^j - \sum_{l=1}^{M} \lambda_l^j} \right) \right] + \left(1 - \sum_{j=1}^{P} p_i^j \right) \cdot \left(N \cdot \tau_0^i + \frac{1}{\mu_s - \sum_{l=1}^{M} (1 - p_l) \cdot \lambda_l} \right) \right\} \qquad \text{(E 3.8)}$$

Variable	Significado
M	número de clústeres de clientes
P	número de *surrogates*
p_i^j	probabilidad del i-ésimo cliente en contactar el j-ésimo *surrogate*
N	número de paquetes requeridos en la transacción cliente-servidor
$\tau_p^{i,j}$	tiempo de ida y vuelta (RTT medio) entre el i-ésimo clúster y el j-ésimo *surrogate*
μ_p^j	tasa media de servicio para el j-ésimo *surrogate*
λ_l^j	tasa media de llegada que el l-ésimo clúster envía al j-ésimo *surrogate*
τ_o^i	tiempo de ida y vuelta (RTT medio) entre el i-ésimo clúster y el servidor origen
μ_s	tasa media de servicio del servidor origen
$(1-p_l)\lambda_l$	tasa media de llegada que el l-ésimo clúster envía al servidor origen

Tabla 12. Variables de la expresión analítica del modelo.

3.2.3. Descripción y análisis del modelo

Previamente al traslado de la expresión general a escenarios de simulación, es conveniente aclararlo mediante algunas consideraciones de carácter gráfico o matemático.

En primer lugar, puesto que hay M clientes y *P surrogates* para servirlos, puede parecer lógico suponer ambos en la misma cuantía (*P=M*) con la finalidad de asociar un *surrogate* óptimo a cada uno de los clústeres de clientes. Sin embargo, esto puede resultar complicado e incluso imposible, ya que en la fase de diseño de una CDN resulta complejo definir clústeres de clientes. Incluso así, es posible asignar varios *surrogates* para el mismo clúster de clientes (*P>M*), si existe una necesidad de servir contenido dependiendo del perfil del cliente, como podría ser el caso de una situación comercial donde existieran clientes normales y 'premium'. Por el contrario, si un proveedor de CDN está desplegando su red, resulta económicamente más atractivo comenzar con unos cuantos *surrogates* (*P<M*) y, posteriormente, aumentar el tamaño de la red sin más que agregar más *surrogates*. Independientemente de la estrategia comercial, razones técnicas como la congestión de los servidores debido a efectos *flash-crowd* sugieren la casi necesidad de servir contenido desde diferentes *surrogates*.

Supongamos un escenario como el representado en la Figura 55, donde existen dos clientes y 3 *surrogates*. Aunque se trata de un entorno poco real, la sencillez de este ejemplo permite comprender mejor el modelo.

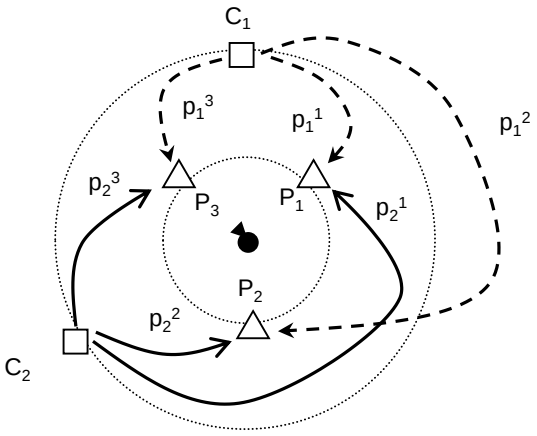


Figura 55. Escenario de CDN para 2 clientes y 3 surrogates.

El cliente C_1 genera peticiones con una tasa λ_1. Como este cliente es servido por varios *surrogates*, la visión más general es suponer que un cierto porcentaje de peticiones (p_1^1) serán encaminadas al *surrogate* P_1, y otro porcentaje (p_1^2, p_1^3) a P_2 y P_3. La suma de estas tres probabilidades representa la tasa de acierto (p_1), mientras que la diferencia hasta la unidad ($1-p_1^1-p_1^2-p_1^3$) representa la probabilidad con la que cliente C_1 enviará peticiones al servidor.

Como puede deducirse lógicamente de la figura anterior, las probabilidades en contactar con un *surrogate* no son iguales, sino diferentes. Cada cliente contactará la mayoría de las veces con el *surrogate* que tenga más próximo, de forma que se establecerá una jerarquía en la asignación de pesos (o probabilidades) en base a la distancia temporal. De esta forma, si el cliente C_2 está más próximo del *surrogate* P_1, luego del *surrogate* P_2 y finalmente de P_3, entonces se cumplirá que $p_1^{\,1} > p_1^{\,2} > p_1^{\,3}$. De esta forma, podríamos construir una matriz que contenga todas las probabilidades, donde la fila indicaría el cliente en cuestión y la columna estaría asociada a la probabilidad de ese cliente en contactar con el *surrogate* correspondiente. En el caso de nuestro ejemplo, un ejemplo de dicha matriz podría ser:

$$P_p = \begin{bmatrix} 0.4 & 0.1 & 0.2 \\ 0.0 & 0.3 & 0.2 \end{bmatrix} \qquad \text{(E 3.9)}$$

Nótese que la suma de cada fila es distinta de la unidad. Esto es así porque un cierto número de peticiones son encaminadas al servidor origen (aquellas cuya respuesta no se haya podido almacenar en la memoria *caché* de algún *surrogate*). Sería posible de esta forma construir un vector a partir de las tasas de acierto (*caching hit ratio*) que indicaría las probabilidades de cada cliente en ser encaminado al servidor origen. En nuestro ejemplo didáctico, tendríamos:

$$\vec{p}_s = \begin{bmatrix} 1-(0.4+0.1+0.2) \\ 1-(0.0+0.3+0.2) \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.5 \end{bmatrix} \qquad \text{(E 3.10)}$$

Generalizando nuestro ejemplo a una situación más real con *M* clientes y *P surrogate*s, obtendríamos una matriz con *M* filas y *P* columnas y un vector de *M* filas:

$$P_p = \begin{bmatrix} p_1^1 & p_1^2 & p_1^3 & \cdots & p_1^P \\ p_2^1 & p_2^2 & p_2^3 & \cdots & p_2^P \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ p_M^1 & p_M^2 & p_M^3 & \cdots & p_M^P \end{bmatrix}_{M \times P} \qquad \vec{p}_s = \begin{bmatrix} 1-\sum_{j=1}^{P} p_1^j \\ 1-\sum_{j=1}^{P} p_2^j \\ \cdots\cdots \\ 1-\sum_{j=1}^{P} p_M^j \end{bmatrix} \qquad \text{(E 3.11)}$$

Otro aspecto a considerar es que un cliente puede no contactar nunca un cierto *surrogate*, es decir, $p_i^j = 0$ para un cierto valor de i,j .En nuestro ejemplo anterior, el cliente 2 nunca contactará ni recibirá contenido desde el *surrogate* P_1. Esto es debido al hecho de que (en términos de latencia) el servidor origen está más cercano que el *surrogate* P_1 ($\tau_p^{\,2,1} > \tau_0^{\,2}$). Esta propiedad, vista de una manera general, puede ser empleada como un criterio para asignar los pesos o probabilidades, de tal forma que:

$$\begin{cases} si \quad \tau_p^{i,j} < \tau_p^{k,l} \quad \rightarrow \quad p_i^j > p_k^l \quad ,\forall\, i,k \in [1..M], \forall\ j,l \in [1..P] \\ si \quad \tau_p^{i,j} > \tau_0^i \quad \rightarrow \quad p_i^j = 0 \quad ,\forall \quad i \in [1..M], \forall \quad j \in [1..P] \end{cases} \qquad \text{(E 3.12)}$$

Este comportamiento puede apreciarse en la Figura 56. Normalmente, un cliente contacta con pocos *surrogates*, cercanos a su ubicación geográfica. Un umbral puede ser asignado mediante la introducción de un factor α (α<1) de tal forma que $\tau_p^{i,j} < \alpha \cdot \tau_0^i$. Si el número de *surrogates* es elevado, entonces reducir el factor α también reduce el número de *surrogates* asignados. En la Figura 56, una reducción de α=1 a α=0.8 supone contactar con tres servidores en lugar de cinco. Por el contrario, si el número de *surrogates* es reducido, debería ser posible obtener contenido desde *surrogates* más alejados (α>1); se trataría más bien de una situación bajo severas condiciones de carga en los *surrogates* cercanos y en el servidor origen.

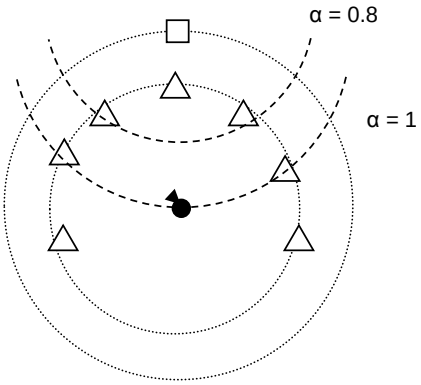


Figura 56. Umbral de latencia para un clúster cliente.

3.2.4. Evaluación del modelo

Como puede apreciarse, la expresión para el tiempo medio de respuesta corresponde a una función *n-dimensional* representado por un valor de *n* variables. Por ejemplo, la probabilidad de que un cliente contacte con un *surrogate*, representado por p_i^j, es realmente un conjunto de variables cuya longitud depende del número de clientes (*M*) y *surrogates* (*P*). Un comportamiento similar tiene lugar con los tiempos de ida y vuelta ($\tau_p^{i,j}$, τ_0^i), las tasas de servicio (μ_p^j,μ_s) y las tasas de llegada (λ_i^j). Por tanto, se simulará una CDN para varios valores de *M* y *P*, con la intención de obtener algunas conclusiones generales. Durante la simulación, las variables que se considerarán como parámetros de entrada serán:

- El número de clientes (M). Los clientes estarán uniformemente distribuidos alrededor del servidor origen separados cada uno (2π/M) radianes. Para introducir una cierta aleatoriedad, el primer cliente se desplazará un ángulo aleatorio, uniformemente distribuido entre [0, 2π/M] radianes; el resto de

clientes se situarán de manera equidistante. De esta forma tenemos para los ángulos de situación las siguientes expresiones:

cliente 0	$\alpha_{c0} = rand\left(0, \frac{2\pi}{M}\right)$
cliente i-ésimo	$\alpha_{ci} = \frac{2\pi}{M} \cdot i + \alpha_{c0}$ i = 0..M

Nota: El índice a partir del cual se numera a los clientes es 0, mientras que en anteriores ocasiones, como en las anteriores figuras, este índice era 1. Esto es debido a que el índice 0 permite una mejor simplicidad en la formulación matemática.

- El número de *surrogates (P*). El mismo procedimiento de aleatorización tiene lugar en este caso, de forma que obtenemos expresiones análogas para los ángulos asignados:

surrogate 0	$\alpha_{s0} = rand\left(0, \frac{2\pi}{P}\right)$
surrogate j-ésimo	$\alpha_{sj} = \frac{2\pi}{P} \cdot j + \alpha_{s0}$ j = 0..P

- El número necesario de retransmisiones (*N*)
- Una cierta tasa de acierto para cada cliente (p_i ,*i=1..M*)
- Un valor mínimo y máximo para las latencias entre cada cliente y el servidor origen (τ_0^{min}, τ_0^{max}). Como cada cliente no se encuentra a la misma 'distancia' temporal del servidor origen, se tomará un valor arbitrario (τ_0^{i}) entre el intervalo indicado.
- Un valor mínimo y máximo para las latencias entre cada *surrogate* y el servidor origen (τ_d^{min}, τ_d^{max}). En este caso, análogo a la situación anterior, se les asignará a los *surrogates* un valor aleatorio entre estos dos valores indicados.
- Un valor mínimo y máximo para las tasas medias de tráfico de cada cliente (λ_{min}, λ_{max}), ya que cada uno de los clientes puede tener una caracterización distinta en cuanto al tráfico generado. Una vez más, se tomará un valor aleatorio entre los dos valores indicados.
- Un factor umbral (α) indicando el área o zona para cada cliente de cara a contactar a los *surrogates*, tal como se describió previamente en la Figura 3.1.4.

Aquellos *surrogates* situados fuera de esta área de contacto no servirán contenido al cliente correspondiente.

- Un factor de capacidad (*k*) representando el incremento necesario con respecto a la tasa media de llegada, de tal forma que la condición de estabilidad para una cola M/M/1 quede satisfecha ($\mu > \lambda$). Pese a poderse fijar de manera distinta para cada *surrogate*, se tomará en un primer momento el mismo valor para todos los *surrogates*.

La Figura 57 muestra un ejemplo para 8 clientes y 4 *surrogates*. Nótese que tanto los clientes como los *surrogates* no tienen la misma distancia (en términos de latencia) con respecto al servidor origen. Cada uno de ellos se encuentra en el interior de un anillo cuyo grosor está determinado por [τ_0^{min}, τ_0^{max}] y [τ_d^{min}, τ_d^{max}] respectivamente. Sin embargo, la distancia en radianes entre ellos es determinista, ya que ambos se encuentran uniformemente distribuidos alrededor del anillo dependiendo del valor de *M* y *P*.

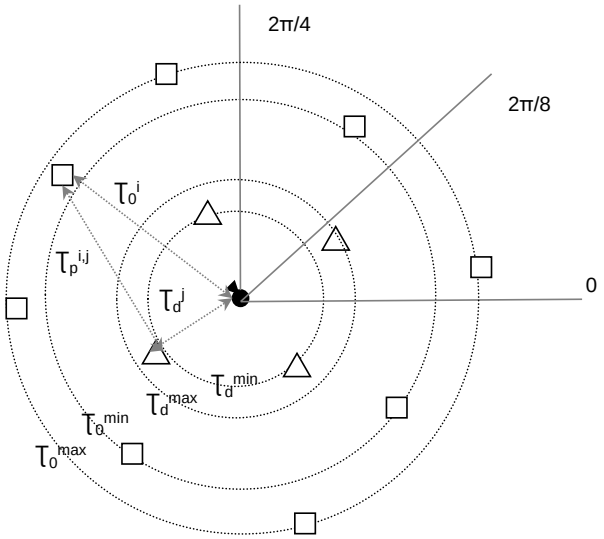


Figura 57. Estructura de una CDN para M=8 y P=4.

Con esta distribución geométrica presentada la distancia (RTT) entre un cliente i y un *surrogate* j puede ser calculada de manera automática mediante el teorema del coseno, sin necesidad de ser requerida como parámetro de entrada. Por tanto, tenemos que:

$$\tau_p^{i,j} = \left(\tau_o^i\right)^2 + \left(\tau_d^j\right)^2 - 2 \cdot \tau_o^i \cdot \tau_d^j \cdot \cos\beta^{i,j} \qquad \text{(E 3.13)}$$

$$\beta^{i,j} = \left|\alpha_{ci} - \alpha_{sj}\right| = \left| \left(\frac{2\pi}{M} \cdot i + \alpha_{c0}\right) - \left(\frac{2\pi}{P} \cdot j + \alpha_{s0}\right) \right| \qquad \text{(E 3.14)}$$

Pese a que el valor absoluto debe ser usado debido a la posición relativa que puede adoptar un cliente y un *surrogate*, la propiedad par del coseno lo convierte en innecesario. De esta forma podemos crear una matriz y un vector de latencias:

$$T_p = \begin{bmatrix} \tau_1^1 & \tau_1^2 & \tau_1^3 & \dots & \tau_1^P \\ \tau_2^1 & \tau_2^2 & \tau_2^3 & \dots & \tau_2^P \\ \dots & \dots & \dots & \dots & \dots \\ \tau_M^1 & \tau_M^2 & \tau_M^3 & \dots & \tau_M^P \end{bmatrix}_{M \times P} \qquad \vec{\tau}_S = \begin{bmatrix} \tau_0^1 \\ \tau_0^2 \\ \dots \\ \tau_0^M \end{bmatrix} \qquad \text{(E 3.15)}$$

La matriz de latencias T_p está relacionada con la matriz de probabilidad anteriormente mencionada (P_p). Nótese que como parámetro de entrada sólo se solicita una tasa de acierto para cada cliente, y no todos los pesos o probabilidades asociadas a la hora de contactar cada uno de los *surrogates*. Esto es debido a que la matriz P_p se puede obtener automáticamente a partir de la matriz T_p, el vector $\vec{\tau}_S$, el factor umbral α y la tasa de acierto p_i.

La forma de obtener la matriz P_p es relativamente simple. Para cada fila i de la matriz T_p existe un umbral dado por $\alpha \cdot \overrightarrow{\tau_S}(i)$, descrito gráficamente en la Figura 56, que discrimina entre los *surrogates* que el i-ésimo cliente contactará. De esta forma es posible construir a modo de máscara una matriz M_p (de dimensión MxP) cuyos elementos $m_p^{i,j}$ satisfacen la siguiente condición

$$m_p^{i,j} \begin{cases} 1 & , si\ T_p(i,j) \le \alpha \cdot \vec{\tau}_S(i) \\ 0 & , si\ T_p(i,j) > \alpha \cdot \vec{\tau}_S(i) \end{cases} \qquad \text{(E 3.16)}$$

La matriz M_p actúa como punto de partida (conjuntamente con T_p) para construir otra matriz intermedia P_w que asigna los pesos para un cliente *i* que contacta con un *surrogate j*. De esta forma, la suma de cada columna en la matriz P_w debe ser 1. Nótese que se trata de los pesos asignados, no de la probabilidad de contacto final, aunque fácilmente puede deducirse en este instante que para obtener la probabilidad de contacto no habrá más que multiplicar esta matriz de pesos por la correspondiente probabilidad de acierto. Los pasos para construir la matriz P_w son los siguientes:

1) En primer lugar, se crea una matriz auxiliar multiplicando la matriz de latencias T_p con la matriz máscara, componente a componente. Esto es, no se trata de una multiplicación de matrices (nótese que la dimensión de ambas no lo permite), sino que se multiplica cada componente de una matriz con su correspondiente componente de la otra matriz, de tal forma que se obtiene una nueva matriz de dimensión MxP similar a T_p, pero discriminando aquellas latencias que son mayores que el umbral impuesto. De una forma más matemática, los elementos de esta matriz auxiliar satisfarán la siguiente expresión:

$$aux^{i,j} = \tau_p{}^{i,j} \cdot m_p^{i,j} \qquad \text{(E 3.17)}$$

2) Como cada fila de la matriz Pw debe ser 1, uno podría pensar en dividir cada elemento de la fila por la suma de todos los elementos:

$$p_w{}^{i,j} = \frac{aux^{i,j}}{\sum_{j=1}^{P} aux^{i,j}} \qquad \text{(E 3.18)}$$

Sin embargo, este proceso supondría la obtención de un mayor peso cuando la distancia cliente-*surrogate* es mayor, que no es el propósito perseguido; al contrario, el enlace entre un cliente y un *surrogate* cercano debe representar un mayor peso que otro que no lo está tanto (téngase en cuenta que los *surrogates* alejados ya se han eliminado en el paso anterior. Una forma de obtener una correcta asignación es sopesar la diferencia de latencias con respecto al máximo sumando teórico impuesto por el umbral, conservando la condición de que cada fila de la matriz debe ser la unidad:

$$p_w{}^{i,j} = \frac{\left(\alpha \cdot \vec{\tau}_s(i)\right) - aux^{i,j}}{\sum_{j=1}^{P}\left(\alpha \cdot \vec{\tau}_s(i) - aux^{i,j}\right)} \qquad \text{(E 3.19)}$$

A continuación se mostrará esta última explicación con un ejemplo. Supongamos una situación con 4 clientes y 6 *surrogates*, como se indica en la Figura 58.

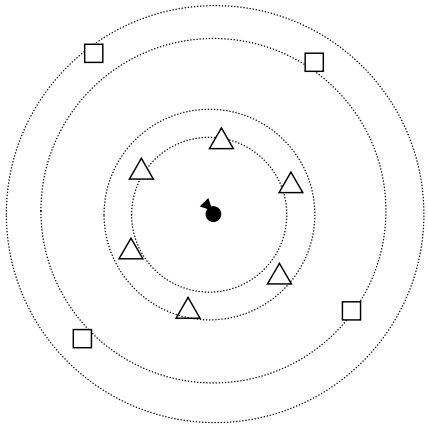


Figura 58. Ejemplo de una CDN para M=4 y P=6.

Para esta situación, supongamos los siguientes valores iniciales de la matriz T_p, el vector $\vec{\tau}_s$, el factor umbral α (para una mejor comprensión, se ha representado como vector) y la tasa de acierto p_i.

$$T_p = \begin{bmatrix} 0.3 & 0.9 & 1.3 & 1.8 & 0.8 & 0.5 \\ 0.7 & 0.2 & 0.5 & 1.1 & 1.7 & 1.0 \\ 1.4 & 0.5 & 0.3 & 0.3 & 1.2 & 1.5 \\ 0.8 & 1.9 & 1.1 & 0.6 & 0.2 & 0.4 \end{bmatrix}_{4\times 6} \quad \vec{\tau}_s = \begin{bmatrix} 0.9 \\ 0.9 \\ 0.8 \\ 0.9 \end{bmatrix} \quad \vec{\alpha} = \begin{bmatrix} 0.9 \\ 0.8 \\ 0.7 \\ 0.7 \end{bmatrix} \quad \vec{p}_i = \begin{bmatrix} 0.5 \\ 0.6 \\ 0.5 \\ 0.4 \end{bmatrix} \quad \text{(E 3.20)}$$

A partir de estos valores indicados, podemos obtener cada uno de los umbrales (por filas). Si lo representamos por un vector, tendremos:

$$\vec{th} = \vec{\alpha}(i) \cdot \vec{\tau}_s(i) \quad , i = 1..4 \qquad \vec{th} = \begin{bmatrix} 0.9 \cdot 0.9 \\ 0.8 \cdot 0.9 \\ 0.7 \cdot 0.8 \\ 0.7 \cdot 0.9 \end{bmatrix} = \begin{bmatrix} 0.81 \\ 0.72 \\ 0.56 \\ 0.63 \end{bmatrix} \quad \text{(E 3.21)}$$

Por lo que la matriz que actúa a modo de mascara M_p y la matriz auxiliar *Aux* serán:

$$M_p = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}_{4\times 6} \qquad Aux = \begin{bmatrix} 0.3 & 0 & 0 & 0 & 0.8 & 0.5 \\ 0.7 & 0.2 & 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0.3 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0.6 & 0.2 & 0.4 \end{bmatrix}_{4\times 6} \quad \text{(E 3.22)}$$

A partir de aquí, ya podemos obtener una primera aproximación a la matriz de pesos calculando solamente el numerador de la expresión (E 3.19):

$$P_w^{(1)} = \begin{bmatrix} 0.81-0.3 & 0 & 0 & 0 & 0.81-0.8 & 0.81-0.5 \\ 0.72-0.7 & 0.72-0.2 & 0.72-0.5 & 0 & 0 & 0 \\ 0 & 0.56-0.5 & 0.56-0.3 & 0.56-0.3 & 0 & 0 \\ 0 & 0 & 0 & 0.63-0.6 & 0.63-0.2 & 0.63-0.4 \end{bmatrix}$$

$$= \begin{bmatrix} 0.51 & 0 & 0 & 0 & 0.01 & 0.31 \\ 0.02 & 0.52 & 0.22 & 0 & 0 & 0 \\ 0 & 0.06 & 0.26 & 0.26 & 0 & 0 \\ 0 & 0 & 0 & 0.03 & 0.43 & 0.23 \end{bmatrix}$$

(E 3.23)

Lo único que nos queda para obtener la matriz de pesos P_w es cumplir la condición de que cada una de sus filas suma la unidad. La suma acumulada de cada elemento de la fila viene dado por el siguiente vector:

$$\overrightarrow{sum}_{Pw(1)} = \begin{bmatrix} 0.51+0.01+0.31 \\ 0.02+0.52+0.22 \\ 0.06+0.26+0.26 \\ 0.03+0.43+0.23 \end{bmatrix} = \begin{bmatrix} 0.83 \\ 0.76 \\ 0.58 \\ 0.69 \end{bmatrix} \quad \text{(E 3.24)}$$

Por tanto, la matriz de pesos P_w será:

$$P_w = \begin{bmatrix} 0.51/0.83 & 0 & 0 & 0 & 0.01/0.83 & 0.31/0.83 \\ 0.02/0.76 & 0.52/0.76 & 0.22/0.76 & 0 & 0 & 0 \\ 0 & 0.06/0.58 & 0.26/0.58 & 0.26/0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.03/0.69 & 0.43/0.69 & 0.23/0.69 \end{bmatrix} \quad \text{(E 3.25)}$$

Y la matriz de probabilidades P_p será la multiplicación de cada fila por el índice asociado de $\vec{p}_i$

$$P_p = \begin{bmatrix} 0.5\cdot(0.51/0.83) & 0 & 0 & 0 & 0.5\cdot(0.01/0.83) & 0.5\cdot(0.31/0.83) \\ 0.6\cdot(0.02/0.76) & 0.6\cdot(0.52/0.76) & 0.6\cdot(0.22/0.76) & 0 & 0 & 0 \\ 0 & 0.5\cdot(0.06/0.58) & 0.5\cdot(0.26/0.58) & 0.5\cdot(0.26/0.58) & 0 & 0 \\ 0 & 0 & 0 & 0.4\cdot(0.03/0.69) & 0.4\cdot(0.43/0.69) & 0.4\cdot(0.23/0.69) \end{bmatrix}$$

(E 3.26)

Una vez se ha descrito el proceso de simulación, procedemos a la representación y análisis de los resultados. La Figura 59 muestra tiempos de respuesta para varios valores de probabilidad de acierto y para un número reducido de clientes y *surrogates*. Además, se han establecido unas relaciones sencillas entre variables para poder validar analíticamente los resultados. Por ejemplo, la probabilidad de acierto se ha fijado la misma para todos los clientes, quienes además envían tráfico de red con la misma tasa. El parámetro α se ha fijado a la unidad para asegurar de esta forma que cada cliente contactará al menos con un *surrogate*. Por simplicidad, no se ha establecido un anillo de latencia ni para los clientes ni para los *surrogates*.

Nótese que el tiempo de respuesta puede ser contemplado desde dos puntos de vista:

- Como el tiempo de respuesta de los *surrogates* y el servidor origen
- Como el tiempo de respuesta asociado a tiempo de ida y vuelta (RTT) y tiempo de proceso en los servidores (tanto *surrogates* como servidor origen).

El primer caso permite analizar las situaciones de trabajo donde es preferible o no descargar contenido a los *surrogates*, mientras que el segundo caso indica si el tiempo de respuesta está principalmente condicionado por la congestión en la red o por sobrecarga en los servidores.

Como puede apreciarse en la Figura 59 todos los tiempos de respuesta siguen un comportamiento lineal. Nótese también que el valor más pequeño de tiempo de respuesta se obtiene para el valor más elevado de probabilidad de acierto (*caching hit ratio*), esto es, cuando todo el contenido es servido desde los *surrogates* y nada es servido por el servidor origen. Aunque la reducción en el tiempo de respuesta pueda no ser significativa en valores absolutos, sí lo es en términos relativos (alrededor de un 40%).

Además, el tiempo de respuesta asociado al procesado es muy reducido, prácticamente imperceptible, por lo que sólo el tiempo de ida y vuelta afecta al tiempo de respuesta medio global. Esto puede ser debido a que el valor del parámetro *k* elegido haya conllevado a situaciones de descongestión en todos los servidores. En cualquier caso, el valor de latencia para este escenario es considerable para una CDN real. Nótese que un valor de RTT de hasta 10 segundos para obtener un contenido total de una CDN es significativo; esto es debido a que el modelo, simplificado, no está considerando el efecto *pipeline* de TCP, donde se pueden enviar múltiples peticiones de forma concurrente. En cualquier caso todos los resultados están escalados por un factor N, por lo que sí se puede hacer un análisis cualitativo para comparar el efecto de una CDN cuando se dispone de un mayor número de *surrogates*.

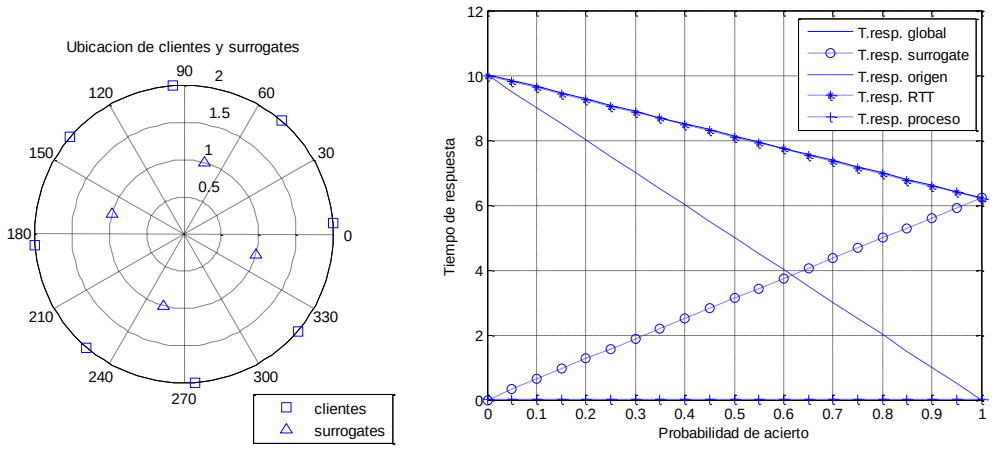


Figura 59. Tiempos de respuesta (M =8, P =4, N=5, α=1, k=1.2, hit_ratio=0..1, τ0min =τ0max = 2 seg, τdmin =τdmax =1 seg, λmin=λmax = 100)

Procedemos a analizarlas a partir de la formulación previa desarrollada. Supongamos para ello (y en primer lugar) el tiempo de respuesta asociado al tiempo de ida y vuelta ($\overline{R}_t$).

$$\overline{R}_t = \frac{1}{M}\sum_{i=1}^{M}\left\{\sum_{j=1}^{P} p_i^j \cdot N \cdot \tau_p^{i,j} \quad + \left(1-\sum_{j=1}^{P} p_i^j\right)\cdot N \cdot \tau_0^i\right\} \qquad \text{(E 3.27)}$$

Dado que todos los clientes cuentan con la misma probabilidad de acierto y que se encuentran a la misma distancia con respecto al origen en términos de RTT, se desprende que:

$$\left(1-\sum_{j=1}^{P} p_i^j\right) = (1-p) \qquad \tau_0^i = \tau_0 \qquad p_i^j = p \cdot p_w^{i,j} \quad \forall i \qquad \text{(E 3.28)}$$

Por lo que $\overline{R}_t$ queda simplificado a la siguiente expresión:

$$\overline{R}_t = (1-p)\cdot N\cdot \tau_0 + \frac{p\cdot N}{M}\cdot \sum_{i=1}^{M}\sum_{j=1}^{P} p_w^{i,j}\cdot \tau_p^{i,j} \qquad \text{(E 3.29)}$$

Nótese que el modelo considera solamente *surrogates* que se encuentren cercanos, por lo que se puede asumir un valor similar de RTT para todos los *surrogates* que un cliente contactará ($\tau_p^{i,j} \cong \tau_p^*$), por lo que:

$$\overline{R}_t = (1-p)\cdot N\cdot \tau_0 + p\cdot N\cdot \tau_p^* \qquad \text{(E 3.30)}$$

Que corresponde con la ecuación asociada a una función lineal de variable *p*. Dado que se cumple que

$$\overline{R}_t(p=0) = N\cdot\tau_0 > N\cdot\tau_p^* = \overline{R}_t(p=1) \qquad \text{(E 3.31)}$$

Podemos concluir que desde el punto de vista del parámetro RTT es deseable descargar todo el contenido a los *surrogates*.

Considerando ahora el tiempo asociado al tiempo de proceso ($\overline{R}_p$), podemos partir de la siguiente expresión:

$$\overline{R}_p = \frac{1}{M}\sum_{i=1}^{M}\left\{ \sum_{j=1}^{P}\left(\frac{p_i^j}{\mu_p^j - \sum_{l=1}^{M}\lambda_l^j} \right) + \frac{\left(1-\sum_{j=1}^{P} p_i^j\right)}{\mu_s - \sum_{l=1}^{M}(1-p_l)\lambda_l} \right\} \qquad \text{(E 3.32)}$$

Dado que se ha fijado el mismo valor de factor de capacidad k para todos los clientes, la expresión previa puede simplificarse de la siguiente forma:

$$\overline{R}_p = \frac{P+1}{(k-1)\cdot\lambda\cdot M} \qquad , ya\ que \sum_{i=1}^{M}\sum_{j=1}^{P}\frac{p_w^{i,j}}{\sum_{l=1}^{M} p_w^{l,j}} = P \qquad \text{(E 3.33)}$$

La primera propiedad de $\overline{R}_p$ que llama la atención es el hecho de que es independiente de la probabilidad de acierto, es decir, permanece constante con esta. Esto puede apreciarse en la Figura 59 y en la Figura 60.

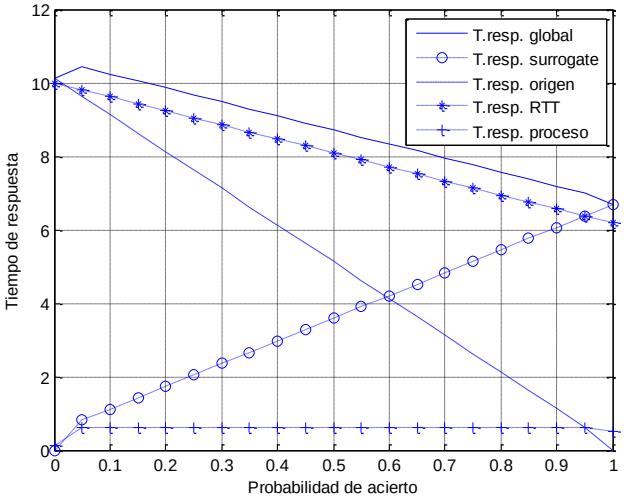


Figura 60. Tiempos de respuesta (M =8, P =4, N=5, α=1.01, k=1.2, hit_ratio=0..1, τ0min =τ0max = 2 seg, τdmin =τdmax =1 seg, λmin=λmax = 100

La expresión anterior es válida para valores de probabilidad de acierto en el intervalo $]0..1[$. Si *p=0*, entonces el tiempo de respuesta asociado al procesado es el requerido por el servidor origen:

$$\overline{R}_p = \frac{1}{M}\sum_{i=1}^{M}\left\{\frac{1}{(k-1)\cdot\sum_{l=1}^{M}\lambda_l}\right\} = \frac{1}{(k-1)\cdot M\cdot\lambda} \qquad \text{(E 3.34)}$$

Dado que $\overline{R}_p(p=0) = \frac{1}{P+1}\overline{R}_p(p\neq 0)$, se justifica el salto o discontinuidad notoria aparecida en la gráfica del tiempo de respuesta asociada al procesado en la Figura 60. Un comportamiento similar tiene también lugar en el escenario de la Figura 59, aunque en este caso el efecto es imperceptible debido al valor reducido del tiempo de proceso.

Por otro lado, si *p=1*, el tiempo de proceso es el requerido por los *surrogates*:

$$\overline{R}_p = \frac{1}{M}\sum_{i=1}^{M}\left\{\sum_{j=1}^{P}\left(\frac{p_i^j}{\mu_p^j - \sum_{l=1}^{M}\lambda_l^j}\right)\right\} = \frac{P}{(k-1)\cdot M\cdot\lambda} \qquad \text{(E 3.35)}$$

El efecto de discontinuidad es menos perceptible en la gráfica, dado que la reducción relativa es menor en el caso de *p=1* que cuando *p=0*. La Figura 61 representa el mismo escenario de una CDN pero con un valor más reducido el parámetro k. Este valor supone el trabajar en condiciones de carga extremas en los servidores, que no es un caso real. Sin embargo, representa claramente las discontinuidades antes indicadas para los valores *p=0* y *p=1*. Gráficas similares podrían obtenerse para otros valores de *M*, *P* y *λ*.

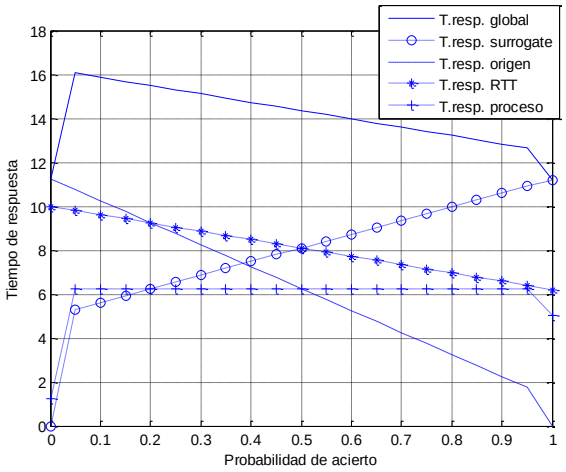


Figura 61. Tiempos de respuesta (M =8, P =4, N=5, α=1.01, k=1.001, hit_ratio=0..1, τ0min =τ0max = 2 seg, τdmin =τdmax =1 seg, λmin=λmax = 100

De las expresiones anteriores resulta sencillo obtener los tiempos de respuesta asociados al contactar sólo con los *surrogates* (y ser servido por estos) y al contactar sólo con el servidor origen. De esta forma, se puede observar en ambos casos la dependencia lineal entre tiempo de respuesta y probabilidad de acierto.

$$\overline{R}_{srrgt} = p \cdot N \cdot \tau_p^* + \frac{P}{(k-1) \cdot M \cdot \lambda} \qquad \text{(E 3.36)}$$

$$\overline{R}_{origin} = (1-p) \cdot N \cdot \tau_0 + \frac{1}{(k-1) \cdot M \cdot \lambda} \qquad \text{(E 3.37)}$$

Un valor fijo del factor de capacidad k no representa de manera realista el estado de carga del servidor, ya que, de no modificarlo, la capacidad siempre se incrementaría con el mismo factor independientemente de la tasa de tráfico que esté soportando, lo que no supone un caso real. Pese a que esta característica nos ha ayudado a simplificar la expresión global en conclusiones parciales, ha conducido a escenarios de trabajo donde el tiempo de respuesta asociado al proceso permanece prácticamente constante, independientemente de que otros parámetros se hayan tomado fielmente a un entorno real de una CDN.

Los servidores disponen de una capacidad limitada que es complicado incrementar de forma dinámica (exceptuando los sistemas elásticos que no se consideran en este modelo analítico). Conforme aumenta el número de peticiones que el servidor debe soportar también lo hará el tiempo de proceso requerido, hasta que se alcance un límite de sobrecarga donde los paquetes puedan ser simplemente descartados (o no se acepte la petición de conexión). Por el contrario, un valor fijo del parámetro k en nuestro modelo supone que una tasa de tráfico de llegada al servidor inmensa resultaría en un tiempo de respuesta reducido, algo poco realista.

Con el propósito de corregir este inconveniente, se ha de fijar un valor fijo de la capacidad de los servidores involucrados en la CDN. Un dimensionamiento en el caso peor será el utilizado en el modelo. Para el caso del servidor origen, el peor caso corresponde a la situación en la que debe soportar todo el tráfico, es decir, cuando *p=0*. La tasa de servicio (μ_s) será:

$$\mu_s = k \cdot \sum_{i=1}^{M} \lambda_i \qquad \text{(E 3.38)}$$

Para el caso de los *surrogates*, la situación más desfavorable se alcanza cuando son estos los que absorben todo el tráfico, esto es, cuando *p=1*. Dado que se asignan diferentes pesos en la comunicación, así como distintas tasas de generación de tráfico a cada uno de los clientes, cada *surrogate* percibirá una tasa de tráfico de llegada diferente, lo que supondrá un valor de capacidad distinto para cada uno de ellos. De esta forma, sería posible construir un vector de capacidades para los *surrogates* de la siguiente forma:

$$\vec{\mu}_p = k \cdot (P_w^T \cdot \vec{\lambda}) \qquad \text{(E 3.39)}$$

Donde $P_w{}^T$ es la matriz de pesos transpuesta y $\vec{\lambda}$ es un vector que contiene la tasa de tráfico de cada uno de los clientes (se trata de un valor aleatorio entre los valores límite λ_{min} y λ_{max}. Nótese que las dimensiones de las matrices (PxM y Mx1) permiten el uso de la multiplicación de matrices, resultando en un vector de dimensión Px1.

La Figura 62 muestra un nuevo escenario de una posible CDN donde la capacidad de los servidores se ha fijado en base a las consideraciones previamente citadas.

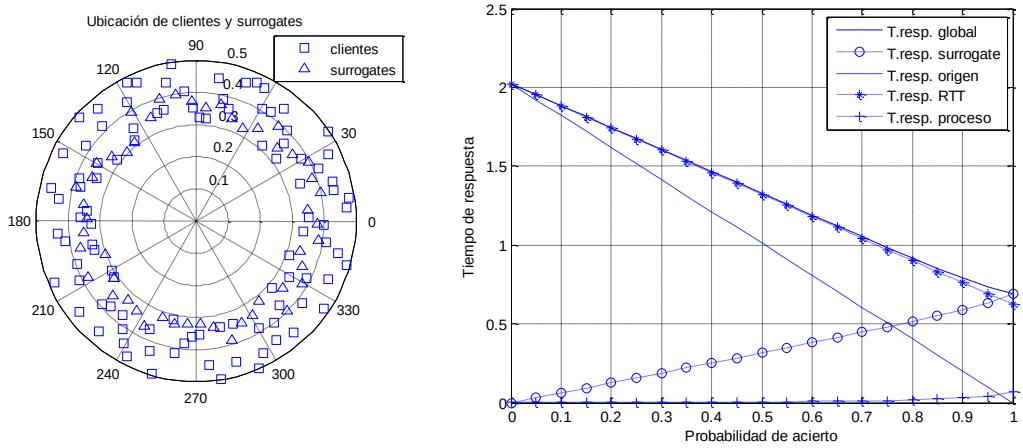


Figura 62. Tiempos de respuesta (M =100, P =50, N=5, α=0.7, k=1.1, hit_ratio=0..1, τ0min =0.3 seg, τ0max = 0.5 seg, τdmin =0.3 seg τdmax =0.4 seg, λmin=50, λmax = 100)

Puede observarse cómo el tiempo de proceso es bastante reducido, por lo que el tiempo de respuesta global está principalmente afectado por el tiempo de ida y vuelta (RTT).

Esto es debido a que los servidores se encuentran sobredimensionados como consecuencia de un diseño en el caso peor. De hecho, esta situación es realista si las tasas de tráfico de los clientes (o las tasas de llegada percibidas en los *surrogates*) pueden ser estimadas. En el caso de una CDN que usa Internet como infraestructura de comunicación, la cantidad de solicitudes de clientes es impredecible. Esto resulta notorio cuando aparece el denominado efecto *flash-crowd*, que podría ser simulado de una forma simplificada mediante un valor reducido del factor de capacidad *k*. Además, la variable independiente para la simulación de un efecto *flash-crowd* debe ser el tiempo, y no la probabilidad de acierto. La Figura 63 muestra una posible simulación donde el factor de capacidad efectivo *k* es una variable aleatoria, aunque siempre superior a la tasa de tráfico de llegada para garantizar el cumplimiento del factor de estabilidad en un sistema M/M/1. Sus márgenes de variación se encuentran entre un valor bastante reducido ($k_{min}=1.001$) y un valor máximo dado (k_{max}). Además, el efecto *flash-crowd* afecta también a la velocidad efectiva de transferencia en la red, ya que aumenta la congestión puntualmente. Esto supone normalmente que las distancias (en términos de latencia) entre los clientes y los *surrogates* aumentan, y probablemente algunos paquetes puedan ser descartados en algún nodo intermedio. Por simplicidad, este efecto se simulará mediante una variación del parámetro *N* (entre un valor dado y el doble de éste) que representaba el número necesario de retransmisiones para satisfacer una petición de un cliente. La Figura 63 muestra un posible efecto *flash-crowd* a las 17:00 h aproximadamente. El valor absoluto del tiempo de respuesta no es importante en este caso, sino el incremento relativo con el tiempo medio de respuesta en todo el intervalo (todo el día). Para este escenario, la sobrecarga en los servidores ha determinado principalmente el tiempo medio global mientras que el ancho de banda en los enlaces apenas ha variado en la simulación. Sin embargo, podría haber sucedido el caso contrario.

Cabe plantearse otras consideraciones acerca de la Figura 63. Un efecto *flash-crowd* puede ser local a una cierta zona, o puede ser global a toda la red de distribución. El primero de los casos puede ser tratado con éxito, al menos parcialmente, por el proveedor de la CDN sin más que redirigir clientes a otros *surrogates* más alejados pero que proporcionan un tiempo de respuesta menor al evitar rutas congestionadas o sobrecarga en los servidores. Un efecto *flash-crowd* global afecta a todo el sistema y resulta difícil reaccionar, ya que la congestión y la sobrecarga se encuentran distribuidos por todo el sistema y los mecanismos de balanceo son poco exitosos.

Mientras que la Figura 63 muestra los tiempos de respuesta obtenidos para múltiples *surrogates*, la Figura 64 muestra un escenario de una posible CDN donde el tiempo de respuesta está representado para cada *surrogate* y servidor origen, así como el tiempo medio de respuesta experimentado por un cliente. Nótese que, dado que cada cliente contacta con varios *surrogates* y con el servidor origen, no resulta trivial obtener la gráfica del tiempo medio de respuesta a partir de las otras funciones, ya que los pesos de probabilidad de contacto también deben ser considerados.

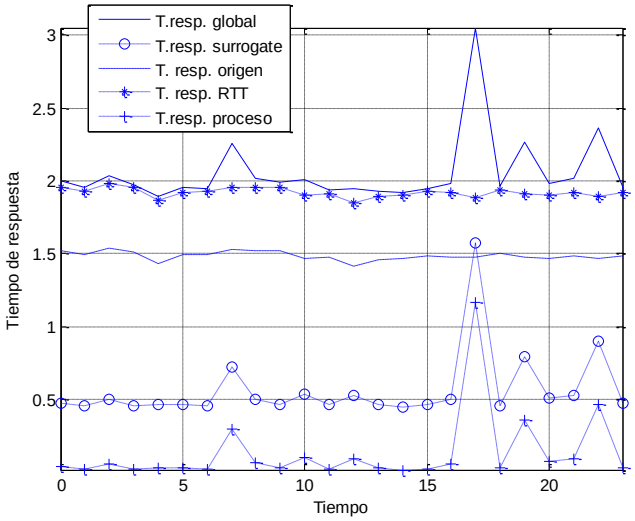


Figura 63. Tiempos de respuesta (M =100, P =50, N=5, α=0.7, kmax=1.4, hit_ratio=0.5, τ0min =0.3 seg, τ0max = 0.5 seg, τdmin =0.3 seg τdmax =0.4 seg, λmin=50, λmax = 100)

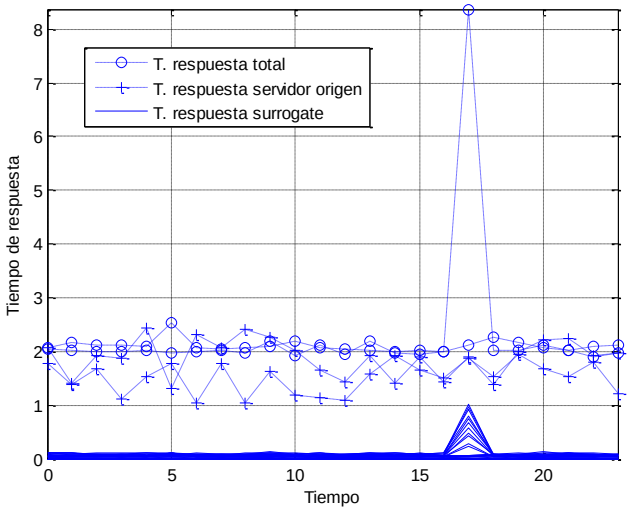


Figura 64. Tiempos de respuesta (M =100, P =50, N=5, α=0.7, kmax=1.3, hit_ratio=0.5, τ0min =0.3 seg, τ0max = 0.5 seg, τdmin =0.3 seg τdmax =0.4 seg, λmin=50, λmax = 100)

3.2.5. Conclusiones parciales del modelo analítico

En esta sección se ha presentado un modelo analítico de una CDN con ciertas simplificaciones donde se ha pretendido caracterizar el funcionamiento básico de este esquema de distribución, marcando una especial atención sobre el retardo medio experimentado por los clientes. Se han realizado múltiples simulaciones para varios valores de M, P, N, etc. para verificar la coherencia de los resultados, si bien en esta tesis se presentan sólo algunos sencillos ejemplos con la finalidad de profundizar en las expresiones matemáticas que dan fundamente teórico a los resultados obtenidos.

En virtud de los resultados (analíticos) y gráficas mostradas se puede concluir que, desde el punto de vista del parámetro RTT, es deseable descargar todo el contenido a los *surrogates*. El parámetro $\overline{R}_t$ analizado en el modelo correspondía a una gráfica decreciente conforme aumentaba la probabilidad de acierto *p* o, en otros términos, el *hit ratio*. Esto encaja perfectamente con un modelo real, ya que el número de enlaces entre un cliente y un *surrogate* siempre es menor que entre el cliente y el servidor origen, por lo que el efecto de la variabilidad del retardo disminuye.

En cuanto al tiempo de proceso (parámetro $\overline{R}_p$), las simplificaciones iniciales del modelo analítico introducían una independencia con la probabilidad de acierto que sólo resultaría realista en condiciones de baja carga. Para poder simular situaciones de alta carga en los servidores ha sido necesario introducir en el modelo un nuevo parámetro variable (*k*) capaz de reproducir efectos del tipo *flash-crowds*.

Otros parámetros podrían resultar también de interés, como la carga del sistema, la ubicación de réplicas en la red, los mecanismos de redirección, etc. En realidad, la carga del sistema se puede obtener fácilmente a partir del tiempo de respuesta del servidor origen y de los *surrogates*. Nótese que la carga media del sistema siempre será la misma en la simulación, y lo que interesa es la variabilidad de las cargas entre *surrogates* y también respecto al servidor origen para identificar si se está realizando adecuadamente un balanceo de carga. Un correcto balanceo de carga está vinculado a un correcto mecanismo de redirección. El algoritmo de redirección no se ha introducido en este modelo analítico con la finalidad de obtener un modelo sencillo y simplificado para estudiar y evaluar conceptos de funcionamiento simples. La complejidad de la redirección posiblemente requeriría introducir otro modelo analítico adicional; de lo contrario, se perdería la sencillez del modelo analítico presentado si se introdujeran mecanismos de redirección, tanto locales como globales (véase capítulo 2).

Sin embargo, el mecanismo de redirección representa un tema de estudio fundamental en esta tesis y sí se ha introducido en el modelo de simulación y en la implementación de la CDN, que se describirán en los subcapítulos 3.3 y 3.4 respectivamente. La complejidad de los modelos de CDN presentados en esta tesis aumenta desde el modelo analítico, el modelo de simulación y el modelo real implementado. El modelo analítico sirve para definir un modelo sencillo basado en los componentes más básicos de la arquitectura general de una CDN (véase capítulo 2) y simplificando otros. A medida que se avanza con el modelo de simulación y el modelo real, los componentes de la arquitectura cobran mayor entidad y las simplificaciones van desapareciendo. Este desarrollo gradual permite contrastar y verificar los resultados entre modelos para posteriormente evolucionar, de tal forma que la implementación real sea completamente operativa (al menos, desde el punto de vista del análisis).

Los principales resultados de este capítulo se han reflejado en dos publicaciones propias [Mol_04b] [Mol_12].

3.3. Modelo de simulación

3.3.1. Introducción

El objetivo principal de esta sección consiste en crear un modelo más avanzado y realista que el analítico, con el cual poder describir características más complejas de una CDN, además de validar, por otro lado, las ventajas de una CDN que ya se describían en el apartado anterior (capítulo 3.2). Adicionalmente, ampliar un modelo de simulación suele ser más sencillo que ampliar un modelo analítico, ya que para la simulación se dispone de herramientas o entornos base que ya emulan el funcionamiento básico de una red de comunicaciones, incluso con los principales protocoles de red, transporte y aplicación. Esto va a permitir que en la presente tesis se desarrollen dos modelos de simulación de una CDN: (i) uno para contenido web que se describe en esta sección; y (ii) otro para contenido streaming que se describirá en el apartado 4.2.

Los principales parámetros de análisis en este modelo de simulación serán:

- La latencia media percibida por los clientes de la CDN.
- La carga media en los servidores o *surrogates*.

Para ambos parámetros, se variará el número de clientes, el número de servidores, la distribución de acceso, el patrón de tráfico, el porcentaje de replicación, etc. y se obtendrán los resultados de la simulación para poder analizarlos y/o compararlos.

Adicionalmente, y con el fin de validar también los resultados proporcionados por el modelo analítico, se realizarán varias simulaciones donde se podrá evaluar tanto el tiempo medio de respuesta como la carga de los servidores en escenarios con y sin CDN, de tal forma que quede reflejada la utilidad de dicha CDN.

En la actualidad apenas existen modelos de CDNs en los entornos de simulación debido, principalmente, a la diversidad de configuraciones que una CDN puede adoptar, como se describió en el capítulo 2.5.4 referente a la taxonomía. Esto obliga al investigador a crear su propio modelo de simulación centrándose en algunos parámetros de funcionamiento y asumiendo o simplificando el ajuste de otros. Entre los modelos de simulación, de ámbito reducido, se puede mencionar CDNsim [WWW_CDN2], basado en OMNetT++ [WWW_OMN], y CDN Simulator [WWW_CDN1], basado en NS-2 [WWW_NS2]. En la medida de lo posible, tras presentar los resultados del modelo de simulación desarrollado, se comparará con estos otros modelos; téngase en cuenta que, al tratarse de modelos distintos, hay que prestar especial precaución en las condiciones bajo las cuales los resultados obtenidos en cada simulación pueden ser comparables.

3.3.2. Herramientas de simulación. NS-2

Las redes y sistemas de telecomunicaciones han alcanzado un grado de sofisticación y complejidad que requieren desde hace años herramientas que permitan simular y, en algunos casos, predecir su comportamiento. Este hecho resulta común en otros entornos de la ingeniería y en arquitectura. Los altos costes de los grandes proyectos hacen necesario un correcto estudio dotado en no pocas ocasiones de modelos de simulación adaptados.

Desde otro punto de vista, las herramientas de simulación también permiten un enfoque didáctico, académico y de investigación. Efectivamente, las simulaciones permiten ilustrar fácilmente, sin necesidad del despliegue de una infraestructura, el comportamiento de grandes sistemas, facilitando la labor pedagógica. La facilidad para cambiar variables y parámetros de diseño y observar el resultado no suponen coste alguno (excepto el tiempo de simulación). Por otro lado, es posible incorporar nuevas variables en el sistema que aún no han sido implementadas físicamente, con la finalidad de estudiar su utilidad y viabilidad antes de proceder a la implementación.

En el ámbito telemático, muchas veces lo que se estudia y evalúa es el correcto funcionamiento de un determinado algoritmo y/o protocolo, o el de varios de ellos que ofrecen conjuntamente un servicio concreto. En esta tesis se estudia el funcionamiento de una CDN, concretamente su efectividad respecto a su modelo de redirección de usuarios, y cómo repercute en el tiempo de respuesta percibido por los usuarios. Téngase en cuenta que una CDN puede constar de un elevado número de nodos (*surrogates*) y la introducción de una herramienta de simulación es especialmente útil.

Existen actualmente una gran variedad de entornos de simulación [WWW_AND], que se pueden clasificar de muy diversas formas. En el ámbito de las redes, algunos de ellos son de propósito general, como NS-2 [WWW_NS2], OPNET [WWW_OPN] y OMNetT++ [WWW_OMN], que permiten modelar un gran número de redes y topologías, o incluso añadir nuevos modelos. Otros entornos son más específicos y se centran en aspectos o sistemas concretos, como pueden ser las redes de sensores o los sistemas móviles, como GlomoSim [WWW_GLO]. Actualmente la tendencia es que estos últimos se integran como módulos adicionales a los simuladores de ámbito general. Por ejemplo, Castalia [WWW_CAST] es un simulador de redes de sensores basado en OMNetT++

Prácticamente los tres simuladores de propósito general mencionados anteriormente son válidos para proceder a la simulación de una red de distribución de contenido. Cuando comenzó esta tesis (comienzos de 2005) no existía en la comunidad científica ninguna herramienta ni módulo dentro de estos simuladores para trabajar con redes de distribución de contenido, por lo que se elaboró uno propio en NS-2. La motivación de este hecho era doble. Por un lado, el desarrollo completo de un modelo de simulación para CDNs permitía dotarle de un enfoque pedagógico para su futuro en prácticas de laboratorio, de tal forma que se dispondría de un control y conocimiento total de todas

(o casi todas) las variables involucradas en el diseño de una CDN (número de clientes, número de *surrogates*, distribución de tráfico, etc.). Por otro lado, NS-2 es una herramienta de libre distribución (*open source*) ampliamente utilizada en el mundo científico y, en especial, en el grupo de investigación SATRD [Mar_04], que funciona en múltiples plataformas (Unix, Linux, Windows).

En los últimos años se han producido ciertos avances en el ámbito de la simulación en general y en el de la simulación de CDNs en particular. En el primer caso, ha aparecido el NS-3 [WWW_NS3] como futura versión de NS-2. Se trata de un cambio sustancial en la forma de simular (por ejemplo, el lenguaje de script Otcl desaparece), aunque la base de programación en C++ se conserva. Sin embargo, no todos los modelos de simulación de NS-2 están portados a NS-3. Cabe indicar que se han realizado numerosos esfuerzos por parte del grupo desarrollador para agilizar o acelerar la transición desde NS-2 a NS-3. La última versión estable de NS-2 (v2.35) no ha variado desde 2011, mientras que la última versión estable de NS-3 (v3.15) data de Agosto de 2012.

En segundo lugar, centrándonos en las redes de distribución de contenido, se han desarrollado ciertos estudios o módulos en cada uno de los simuladores. Si bien OPNET no cuenta con un módulo concreto de CDN disponible, si se han producido ciertos estudios (simulaciones) en este ámbito como es la redirección de usuarios [Pres_08]. NS-2 cuenta desde 2008 con un modelo simplificado de CDN denominado CDN Simulator [WWW_CDN1], disponible en *SourceForge*, aunque su uso no parece muy extendido y no forma parte de la propia distribución de NS-2 (de hecho el módulo sólo es aplicable a una versión concreta de NS-2, la versión 2.33). Finalmente, basado en OMNetT++, existe desde 2009 una herramienta independiente denominada CDNsim [WWW_CDN2], donde la interfaz es aparentemente sencilla pero requiere la introducción de datos a partir de ficheros externos (configuración compleja).

Con la finalidad de centrar la tesis en la producción científica, se aborda directamente el diseño y jerarquía de clases en la siguiente sección. No obstante, una breve descripción de NS-2 y su funcionamiento puede encontrarse en los anexos.

3.3.3. Diseño y jerarquía de clases

Para la implementación de la CDN en NS-2 se ha partido del modelo general descrito en la sección 2.5.2.1, donde se identifican los diferentes bloques de una CDN y se estudia su portabilidad a un entorno de simulación como es NS-2. En este caso, el sistema de tarificación no se tendrá en cuenta ya que no tiene efectos sobre las prestaciones. Por otro lado, el sistema de contabilidad (*logs*) se puede implementar directamente mediante herramientas internas de NS-2 que permiten realizar trazas de eventos. En relación al resto de bloques, es necesario implementar tres componentes básicos en la simulación: los clientes, los *surrogates* y el servidor origen. Desde el punto de vista de la simulación

en NS-2, los tres componentes se pueden considerar como nodos en una red de comunicaciones, y dicha comunicación queda establecida a nivel de aplicación a través de los denominados *agentes*. Sin embargo, es necesario describir el intercambio de información entre nodos (patrón de tráfico generado) así como el proceso de servicio en los servidores (verificación de caché y porcentaje de acierto).

Los bloques restantes según la arquitectura de la CDN general son el bloque redirector y el bloque de distribución. En relación a la redirección, se han creado nodos adicionales en la red de simulación que implementan (mediante el correspondiente *agente* de NS-2) la función de redirección (nodo redirector). Por otro lado, el mecanismo de distribución se ha simplificado en un primer momento, ya que puede resultar bastante complejo si se consideran todas las posibilidades descritas en la sección 2.5.4. En este modelo de simulación, se va a considerar un esquema de *pull no cooperativo* (véase apartado 2.5.4.2) donde todos los *surrogates* se comportan como *proxy caches* con capacidad de replicación parcial y total con respecto al servidor origen.

La implementación en NS-2 de la CDN ha sido desarrollada en base a las siguientes premisas:

- Un cliente accede a recursos web ubicados en *surrogates* de acuerdo a un *determinado patrón de tráfico* configurable, que se puede particularizar para cada cliente. Los parámetros que lo definen son: instante de inicio de sesión, número de páginas que solicita el cliente, distribución que define el intervalo entre peticiones de páginas y la distribución que define el número de objetos por página.
- Cuando un cliente desea obtener un recurso web, el primer paso consiste en *acceder a su Redirector* asociado. El Redirector es un elemento asociado a un cliente o a un conjunto de ellos y es el que se encarga de llevar a cabo el proceso de redirección del cliente. Para ello, cuando el Redirector recibe la petición de un cliente, analiza su identidad, y le responde con la dirección del servidor web (*surrogate o réplica*) con el que debe contactar para obtener el recurso. Esta decisión se realiza, en nuestro modelo, en base a los criterios de proximidad topológica y de balanceo de carga. Este mecanismo de redirección ha pretendido simular el esquema de redirección basado en DNS en el que el Redirector desempeñaría la función de servidor de nombres autoritativo. De hecho, el protocolo de comunicación entre cliente y Redirector se ha implementado sobre UDP y, del mismo modo, se ha tomado un tamaño de mensaje intercambiado equivalente (en media) con los del servicio DNS.
- El cliente *accede al surrogate* indicado para obtener el recurso. Una vez el cliente recibe del Redirector la dirección del servidor (*surrogate*) con el que debe contactar, establece con éste una conexión TCP y envía el mensaje de petición de página web (recurso).

- El servidor *verifica si el recurso solicitado está disponible* en su caché. Cuando el servidor (*surrogate*) recibe la solicitud de una página web, se verifica la disponibilidad de ésta localmente. En caso afirmativo, el servidor proporciona al cliente todos los objetos que componen la página web y cierra la conexión una vez finalizada la descarga. En caso contrario, el servidor establece, a su vez, una nueva conexión TCP con el servidor origen del cual obtiene la página en cuestión. Una vez dispone de dicho recurso en su totalidad, se lo proporciona al cliente. La probabilidad de que el *surrogate* almacene o no el recurso de forma local depende del porcentaje de replicación que, como veremos, constituye uno de los muchos valores parametrizables en este modelo de CDN implementado. Más concretamente, este porcentaje de replicación establece el porcentaje de objetos residentes en el servidor origen que son replicados en el resto de *surrogates* que conforman la CDN. Así, por ejemplo, si este valor es del 100% la información almacenada en el servidor origen y en los *surrogates* es la misma.

Para explicar cómo se ha creado el modelo de simulación en C++, resulta básico crear un árbol de clases en el que queden reflejadas de forma visual las relaciones entre las diferentes clases. También es importante conocer las funciones y parámetros de los ficheros generados, así como las características que éstos implementan.

El modelo de simulación de la CDN está formado por un total de 7 clases nuevas implementadas, relacionadas entre sí dentro de la estructura interna de NS-2 según se muestra en la Figura 65.

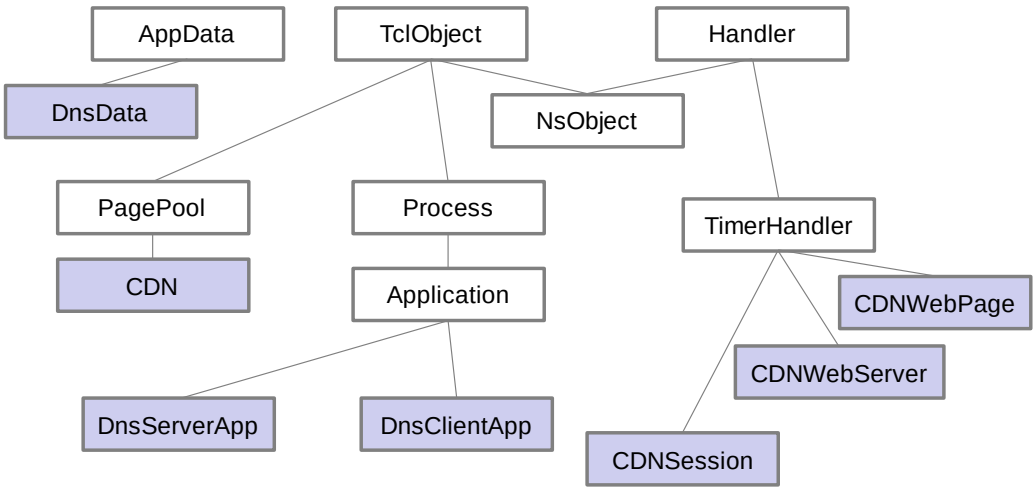


Figura 65. Diagrama de clases del modelo de simulación CDN.

A continuación se describirá brevemente la función de cada una de estas clases y su relación con la clase de NS-2 a la que extienden, de forma que permita entender la motivación de este diseño. No se pretende proporcionar una descripción exhaustiva –

simplemente bastaría acudir al código fuente -, pero sí suficiente. Tampoco se ha considerado necesaria la formalización de la arquitectura de clases de la Figura 65 en un diagrama UML, por dos motivos:

- UML describe de una manera formal arquitecturas y relaciones entre componentes o clases, dando una visión global o particular de la funcionalidad del sistema. En nuestro caso, al trabajar con NS-2 ya se parte de la arquitectura de desarrollo proporcionada por éste simulador, donde simplemente se añaden las clases correspondientes en la jerarquía adecuada. Por lo tanto, un diagrama UML global describiría más la arquitectura de NS-2 que la del modelo de CDN, sin aportar un valor añadido significativo.
- Si bien el modelo de simulación puede resultar complicado durante la implementación en NS-2, no es objeto en la redacción de esta tesis describirlos detalladamente, sino centrarnos en los objetivos principales, que son la obtención de resultados y su posterior análisis. De esta forma, se agiliza la lectura de la misma destacando los resultados y su interpretación.

La clase *DnsData* define una estructura de mensaje con diversos campos para la comunicación entre el cliente y el Redirector que facilitan el proceso de redirección del cliente. Al estar empleando como premisa inicial un esquema de redirección basado en DNS, se ha escogido el nombre *DnsData*. Como se puede apreciar, la nueva clase *DnsData* implementada hereda de la clase NS-2 *AppData*. La clase *AppData* constituye una interfaz para la definición de unidades de datos de nivel de aplicación genéricas. Así, para la transmisión de datos en NS-2, lo más habitual es crear una unidad de datos de nivel de aplicación (ADU) propia que herede de la clase base *AppDat*a. Este hecho existe actualmente en NS-2 para otro tipo de paquetes de nivel de aplicación (HTTP fundamentalmente), como se muestra en la Figura 66.

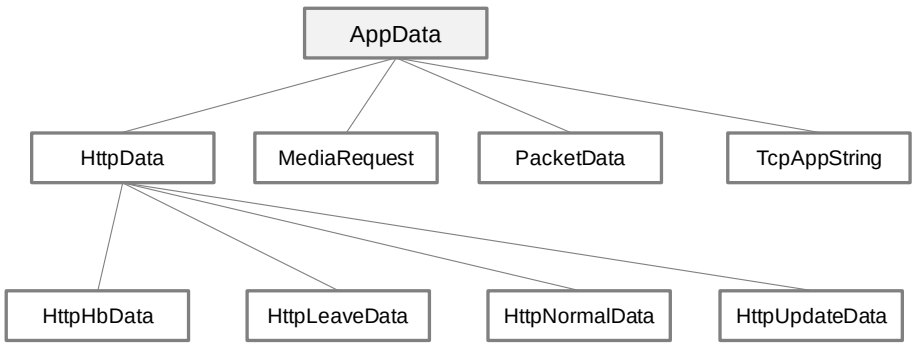


Figura 66. Referencia de clases en NS-2 del componente AppData.

El intercambio de información de mensajes *DnsData* se produce entre una entidad cliente y otra servidora (véase Figura 67). La parte cliente viene representada por la clase *DnsClientApp*, responsable de contactar con el Redirector para obtener la dirección del *surrogate* del cual obtener el recurso web. Por otro lado, la clase

DnsServerApp representa la capa de aplicación del Redirector, implementando los mecanismos de redirección del cliente. El algoritmo redirector sigue un esquema adaptativo, es decir, se adapta a las condiciones de carga de los servidores y la red (véase sección 2.5.4.3) y toma como parámetros de entrada (*inputs*) dos variables. Por un lado, se tiene en cuenta el tiempo de ida y vuelta (RTT) entre cada pareja cliente-*surrogate*. De esta forma se pueden conocer aquellos servidores que están más cercanos en términos de retardo. Por otro lado, se tiene en cuenta la carga de cada *surrogate* para realizar un correcto balanceo entre estos.

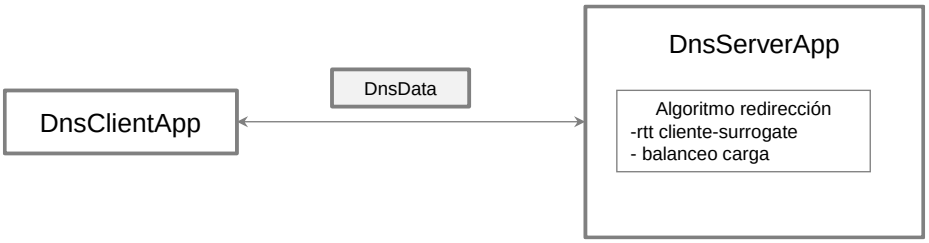


Figura 67. Proceso de redirección de un cliente en la CDN de la simulación.

Hay que tener en cuenta las implicaciones que representa considerar una medida como el RTT. En una topología de red fija, con retardos fijos entre enlaces sin cortes, sería posible hacer una estimación precisa de los *surrogates* más cercanos a cada cliente a priori, de tal forma que con un cálculo inicial sería suficiente. Este no es el caso de Internet, donde la topología es dinámica y el retardo altamente variable. Esto implica calcular el estado de los retardos ante cada petición de un cliente, de forma que es posible asegurar que, en ese momento, el *surrogate* más cercano o adecuado es aquel con un RTT menor.

Obviamente, realizar este cálculo puede implicar un tiempo significativo y una carga excesiva del sistema, sobre todo si se aborda el problema de la escalabilidad con un número elevado de clientes realizando continuas peticiones. Sin embargo, esto no constituye un serio problema en un entorno de simulación por eventos discretos como NS-2, ya que el tiempo de simulación y el tiempo de computación son independientes, y por ello el impacto es mínimo o nulo, dependiendo de los detalles de implementación. En nuestro caso, el impacto es mínimo ya que para calcular el RTT entre cada par se envía un ping empleando la propia simulación (tiempo de simulación). El efecto de la carga del sistema durante la simulación simplemente hace que ésta última tarde más en calcularse (mayor tiempo de computación).

Este reducido impacto sobre el sistema al calcular el RTT supone una ventaja durante la simulación, pero habrá que considerarlo en un modelo real, de tal forma que las medidas sobre los retardos se obtendrán en intervalos de tiempo a través de un sistema de monitorización. Esto se abordará en futuras secciones (3.4 y 4.3) orientados al modelo real. En cualquier caso, esta ventaja de la simulación permite mostrar resultados didácticos muy claros, como se apreciará en el apartado de resultados y análisis de la

simulación. Desde otro punto de vista, se trata de un modelo que dispone de conocimiento global en el mecanismo de redirección (GSLB), como se describió en el apartado 2.5.4.3.

Retornando a la jerarquía de clases y su distribución en la estructura interna de NS-2, las clases *DnsServerApp* y *DnsClientApp* derivan ambas de la clase NS-2 *Application*, que es la clase genérica utilizada para la definición de nuevas aplicaciones, como se observa en la Figura 68.

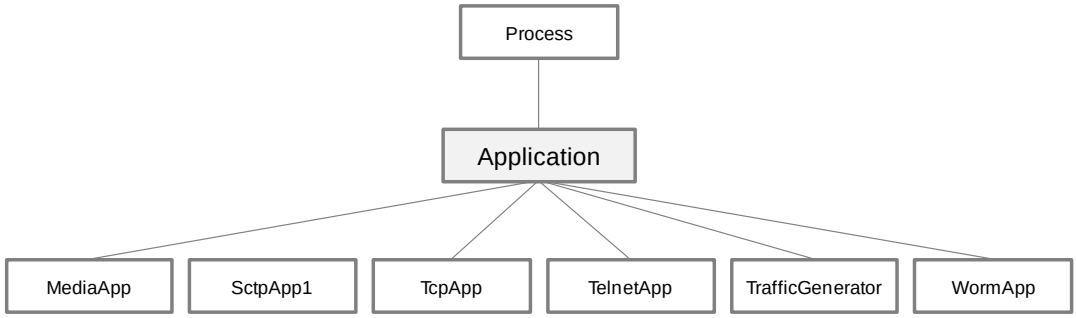


Figura 68. Referencia de clases en NS-2 del componente Application.

Las clases *CDNWebPage*, *CDNSession* y *CDNWebServer* heredan todas ellas de *TimerHandler*. La clase *TimerHandler* constituye la clase base en la definición de cualquier planificador de eventos, y dispone de un elevadísimo número de subclases, algunas de ellas representadas en la Figura 69.

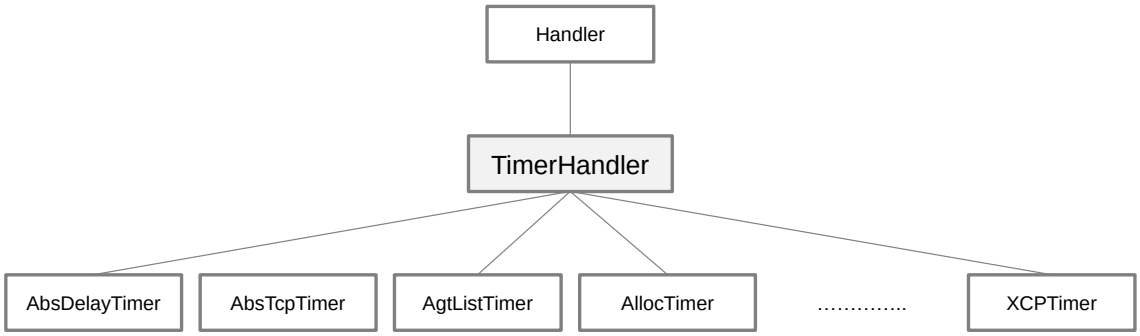


Figura 69. Referencia de clases en NS-2 del componente TimerHandler.

La clase *CDNWebPage* permite crear instancias que representan a páginas web. Una página web, en este contexto, está compuesta por un conjunto determinado de objetos cuyos tamaños siguen una determinada distribución. La clase *CDNWebPage* dispone de métodos que permiten, entre otros, enviar de forma consecutiva mensajes de petición para cada uno de los objetos que componen la página.

La clase *CDNSession* representa una sesión en la CDN. Una sesión se define a partir de su instante de inicio, del cliente que realiza las peticiones y de su patrón de tráfico (número de páginas solicitadas e intervalo entre peticiones de páginas). La clase *CDNSession* se encarga de invocar los distintos métodos del cliente para que éste lleve a cabo las acciones pertinentes. Así, cada vez que vence el temporizador y, por tanto, el cliente requiere

solicitar una nueva página, la clase *CDNSession* ejecuta las funciones oportunas para que éste contacte con el Redirector (sin *caching* en el cliente) y, una vez recibida la respuesta, lo haga con el *surrogate* indicado. Por otro lado, también se encarga de obtener una instancia de la clase *CDNWebPage* que representa la página solicitada por el cliente.

Por su parte, la clase *CDNWebServer* es la que representa a un servidor web (en este caso englobando tanto a los *surrogates* como al servidor origen) de la CDN. La clase en cuestión (si se trata de un *surrogate*) recibe las peticiones de los clientes y se encarga de atenderlas si dispone del recurso solicitado. En caso contrario, contacta con el servidor origen (*pull no cooperativo*) mediante una nueva conexión TCP a través de la cual obtiene el recurso que posteriormente proporciona al cliente. La clase *CDNWebServer* permite modificar un amplio número de parámetros relacionados con el servidor, como la tasa de procesamiento, el modo de funcionamiento, la política de gestión de colas, el tamaño de dichas colas, etc.

Por último, se dispone de la clase *CDN* que deriva de *PagePool*. La clase *PagePool* (véase Figura 70) en NS-2 puede ser utilizada por los servidores para generar información de páginas (nombre, tamaño, instante de modificación, tiempo de vida, etc.), por las *caches* para describir qué páginas se almacenan, y por los clientes para generar flujos de peticiones.

En nuestro caso, la clase *CDN* representa, como su propio nombre indica, a la CDN en su totalidad. De esta forma, incluye parámetros tales como el número de clientes, número de *surrogates*, número de redirectores, número de objetos, distribuciones que definen el número de objetos por página y sus tamaños, porcentaje de replicación de objetos, etc. La clase *CDN* incorpora, además, los métodos necesarios para la creación y destrucción de las sesiones mencionadas anteriormente.

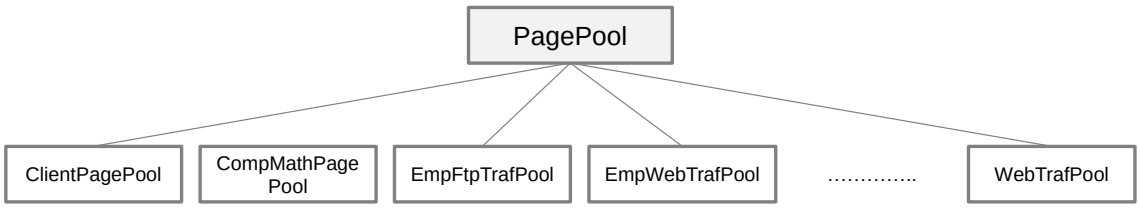


Figura 70. Referencia de clases en NS-2 del componente PagePool.

Todas estas clases han sido implementadas en NS-2 en un total de 11 ficheros, como se observa en la Figura 71. Existen ocho clases en C++ asociadas a las siete clases descritas con anterioridad, así como tres ficheros Otcl que permiten y facilitan la posterior simulación de la CDN. Con la finalidad de proporcionar una visión de alto nivel del desarrollo para centrarse en los resultados.

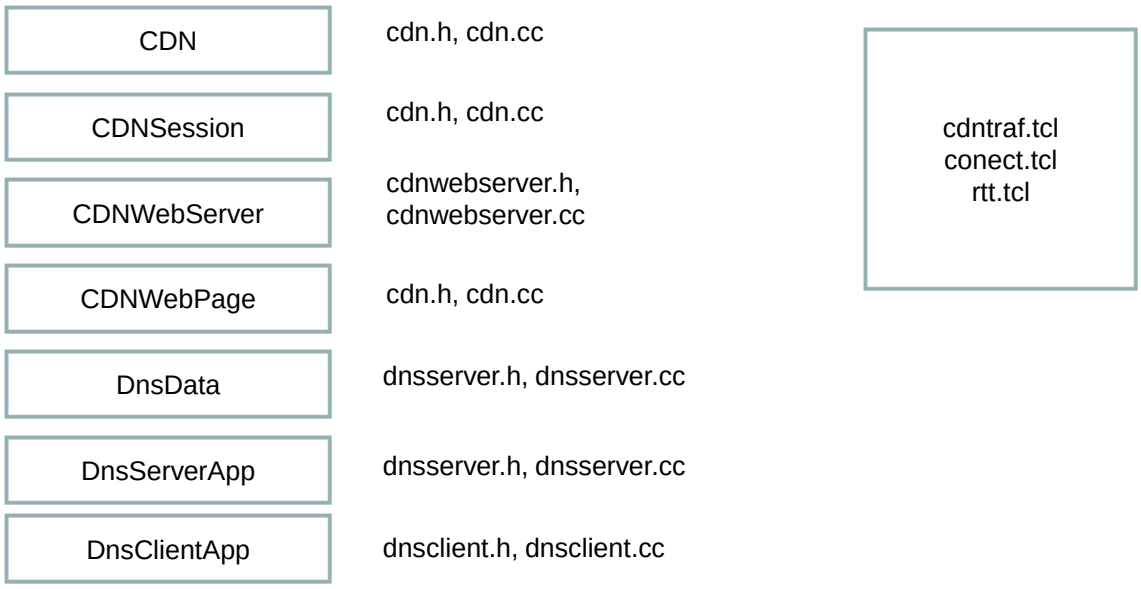


Figura 71. Ficheros implementados y su relación con las clases del simulador CDN.

Respecto a los ficheros TCL, el fichero ***cdntraf.tcl*** implementa los métodos que permiten realizar las peticiones de objetos así como sus respuestas asociadas. Más concretamente, se asocian los agentes TCP (origen y destino) a los nodos respectivos, se conectan entre sí, se envían los paquetes y, finalmente, se reciclan dichos agentes. También se ubican aquí las llamadas al método correspondiente (*job_arrival)* del servidor para informarle de la recepción de una solicitud, así como al método oportuno (*doneObj)* del cliente para advertirle de la recepción del objeto solicitado.

El fichero ***connect.tcl*** dispone de un procedimiento que permite crear dos agentes UDP, los cuales se asocian respectivamente a las aplicaciones DNS cliente y servidora que se conectan entre sí. Esto se emplea durante el mecanismo de redirección.

Finalmente el fichero ***rtt.tcl*** implementa un método que permite estimar el RTT entre dos nodos de una topología de red concreta. Esto permite determinar el retardo entre un cliente y los *surrogates* y, de esta forma, disponer de una lista ordenada de *surrogates* (de menor a mayor RTT) para cada cliente. Esta lista ordenada se puede calcular inicialmente para toda la simulación o bien de forma dinámica. En el primer caso se está haciendo una ordenación y posterior asignación de *surrogates* a clientes a priori, de forma muy parecida a cómo se actuó en el modelo de simulación matemático de la sección 3.2. En el segundo caso, se estaría procediendo a un entorno más realista donde se tiene en cuenta posibles caídas de enlaces y/o *surrogates*; éste es un modo de actuar más parecido al modelo real, que se presentará en el apartado 3.4. La forma de obtener el RTT puede realizarse o bien mediante la simulación de un ping a través de la propia herramienta, o bien de forma matemática a partir del ancho de banda del canal entre cliente y *surrogate* en un momento determinado. Si bien la primera opción es más realista, la segunda opción es más rápida y permite agilizar la simulación, especialmente cuando el número de nodos es elevado como es el caso de una CDN.

3.3.4. Escenarios de simulación. Topologías de red y parámetros.

En este apartado se describen los distintos entornos de red empleados como escenarios de simulación para realizar las distintas simulaciones. Estos entornos, que constituyen el punto de partida para las simulaciones, se han realizado con los denominados generadores de topologías. Es importante en este apartado establecer una breve relación entre los protocolos y los diferentes tipos de generadores de topologías históricamente propuestos.

Los protocolos de red son (o al menos deberían ser) diseñados para ser independientes de la topología de red subyacente. Sin embargo, mientras que la topología no debería tener ningún efecto sobre el nivel de corrección de los protocolos de red, sí tiene, en ocasiones, un gran impacto en el rendimiento de dichos protocolos de red. Por esta razón, a menudo se utilizan generadores de topología de red para crear topologías realistas para las simulaciones. Estos generadores de topologías no aspiran a producir réplicas exactas de la Internet actual; en cambio, simplemente intentan crear topologías de red que representen características fundamentales de las redes reales.

El primer generador de topologías de red que fue ampliamente utilizado en simulaciones de protocolos fue desarrollado por Waxman [Wax_88]. Este generador es una variante del clásico grafo aleatorio de Erdos-Rényi [Bol_85], donde las probabilidades de creación de enlaces están basadas en la distancia euclídea entre los extremos del enlace. Una línea posterior de investigación, percatándose del hecho que las topologías reales de red no tienen una estructura aleatoria, hizo especial hincapié en la generación jerárquica de topologías. Este nuevo modelo fue ampliamente aceptado y durante varios años, los modelos de generadores de red resultantes de este tipo, como Transit-Stub [Cal_97] y Tiers [Doa_96], se consideraron de última generación. A estos generadores se les denominó *estructurales* porque estaban muy enfocados en la estructura jerárquica de las redes, y fueron ampliamente utilizados hasta 1999, cuando un estudio inter-AS e intra-AS ponía de manifiesto que la distribución de grado de los grafos seguía un tipo especial de relación matemática denominada ley de potencias [Fal_99]. Los generadores estructurales no producían distribuciones de grado en base a una ley de potencias, por lo que algunos investigadores opinaban que no eran adecuados para modelar Internet y propusieron otro tipo de generadores de topologías orientados a cumplir el requisito de distribución de grado, sin tratar de modelar la estructura jerárquica de Internet [Jun_00] [Med_01] Pal_00] [Bu_02]. Así pues, se pueden distinguir dos tipos de generadores de topologías:

- ***Estructurales***: generan grafos jerárquicos orientados a modelar la estructura a gran escala de Internet. Un ejemplo de este tipo de generadores es GT-ITM [WWW_Gtitm].
- ***Con distribución de grado en base a una ley de potencias***: genera grafos de red orientados a modelar propiedades locales de redes. Ejemplos de este tipo de generadores son BRITE [WWW_Brite] e INET [WWW_Inet].

Si bien es posible generar topologías de red desde dos puntos de vista, en el caso de las CDNs resulta más importante modelar la estructura a gran escala de Internet (estructura jerárquica) ya que se trata de grandes redes, y el rendimiento de los protocolos se ve más afectado por estructuras de gran escala que por propiedades locales [Tan_02]. No obstante, existe una amplia bibliografía en relación a la generación de topologías de red capaces de simular el funcionamiento de Internet, sin que exista un consenso generalizado en la comunidad científica, ya que existen diferentes métricas (espectro, distribución del espacio, distribución de grado, clustering, etc.) que condicionan los resultados. En esta tesis se trabajará fundamentalmente con GT-ITM como generador de topologías, por tres motivos:

- Permite generar estructuras jerárquicas de una forma sencilla (N-level, transit-stub), aunque también permite generar grafos planos aleatorios (Waxman 1, Waxman 2, Pure random, Doar-Leslie, Exponential, Locality).
- Viene integrado con NS-2, de forma que es posible generar topologías (formato SGB) e importarlas a NS-2 (formato TCL).
- Permite posteriormente una mejor comparación del modelo de simulación de CDN propuesto con CDNsim, que también emplea GT-ITM. El otro simulador, CDN Simulator, al estar implementado en NS-2, también emplea GT-ITM.

En la Tabla 13 se muestran los principales parámetros que definen la topología de red en general, a partir de la cual se insertarán y configurarán los clientes y los servidores para formar una CDN. Cabe destacar que el ancho de banda troncal se ha segmentado en varias categorías según el nivel en la jerarquía de la red, abarcando lo que serían redes troncales y de distribución. En cuanto al retardo de los enlaces, se ha optado por una distribución exponencial, que describe mejor el retardo que una distribución normal o normal truncada [Suk_09].

Nodos	2050
Enlaces	2083
Nodos hoja (leaf nodes)	1028
Protocolo de transporte	TCP (Tahoe)
Algoritmo de gestión de colas	DropTail
Ancho de banda (troncal)	10 Gbps, 2.5 Gbps, 622 Mbps, 155 Mbps, 100 Mbps
Ancho de banda (clientes)	E1 (2 Mpbs)
Ancho de banda (servidores)	E2 (34 Mbps)
Retardo (enlaces troncales)	Distribución exponencial con media = 1 ms
Retardo (enlaces de acceso)	Distribución exponencial con media = 0.5 ms

Tabla 13. Ejemplo de Topología de red en NS-2.

Los parámetros que caracterizan a la CDN de simulación son diversos y variados y se listan en la Tabla 14.

Servidores	Número de *surrogate*s en la CDN
Clientes	Número de clientes en la CDN
Redirectores	Número de Redirectores (o Gestores de Contenido)
Sesiones	Número de sesiones en la CDN
Objetos web	- Número de objetos web - Tamaño de objetos web (distribución en bytes)
Replicación	Porcentaje de replicación de objetos web
Surrogates	- Número de *surrogates* a considerar por cliente - Distribución de acceso a los *surrogates*
Modo	Modo de funcionamiento de los *surrogate*s y servidor origen: - Sin retardo de procesamiento (NO_DELAY) - Con retardo de procesamiento y política FCFS (FCFS_DELAY) - Con retardo de procesamiento y política STF (STF_DELAY)
Tasa	Tasa de procesamiento de los *surrogates* y servidor origen (KB/s)
Buffer	Tamaño máximo de cola de los *surrogates* y servidor origen
Tráfico	Patrón de tráfico de los clientes: - Instante de inicio de sesión - Número de páginas por sesión - Número de objetos por página (función de distribución) - Intervalo entre peticiones de páginas (función de distribución)

Tabla 14. Parámetros de simulación generales en NS-2.

Cada vez que se define un nuevo cliente en la CDN, se crea un nodo nuevo en el grafo original generado que se asocia, mediante el correspondiente enlace *full-duplex*, a un nodo hoja (*leaf node*) elegido de forma aleatoria de entre los que dispone la topología. Por ello, en sentido estricto, el número de nodos en cada simulación se corresponde con el número de nodos de la topología original más el número de clientes definidos, como se observa en la Figura 72. Por otro lado, este nodo hoja al cual se conecta el cliente se ha tomado como Redirector (servidor DNS local) asociado a dicho cliente. Si asumimos que cada nodo cliente se conecta con nodos hoja diferentes (cada nodo hoja representa una subred distinta), entonces el número de Redirectores será coincidente con el número de clientes. En este caso, se ha simplificado la comunicación entre el cliente y el Redirector, asumiendo que el algoritmo de redirección se ejecuta en el servidor DNS local. Si bien esto no es realista, la única diferencia en términos de simulación es la ventaja en el ahorro de mensajes entre nodos, de forma que la resolución DNS (con algoritmo de redirección incluido) sólo atraviesa un enlace en lugar de varios.

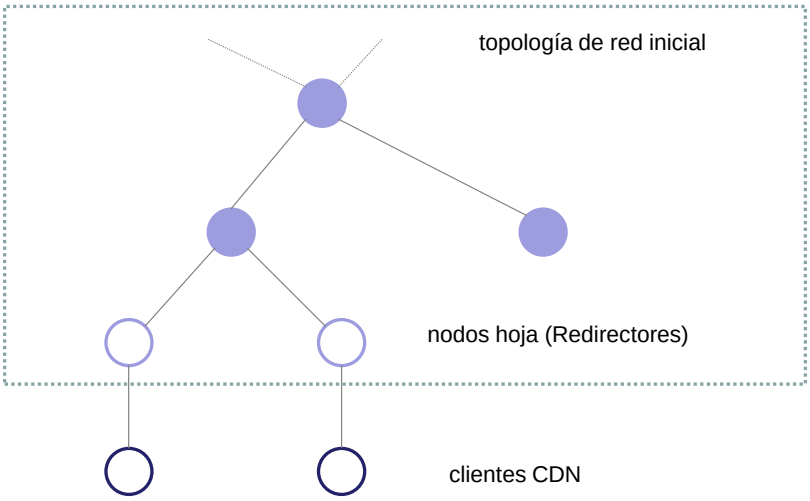


Figura 72. Inserción de nodos clientes en la topología inicial de red.

La simplificación anterior no es la única que se ha considerado en este modelo, ya que, debido al gran número de variables implicadas en el modelo (véase Tabla 14), es preciso fijar ciertos valores para presentar los resultados. Estos valores son:

- En primer lugar, en todas las simulaciones se establecerán 50 sesiones por cada cliente que se defina en la CDN. Aunque es posible introducir más sesiones por cliente, si el número de clientes es elevado los valores medios de retardo no varían significativamente al tener el mismo patrón de tráfico.

- En segundo lugar, el número de objetos web con los que contará la CDN será de 2000. Evidentemente, una CDN real dispone de muchos más objetos web, aunque lo más importante es la proporción entre los objetos solicitados por los clientes con respecto a la totalidad de objetos en términos de replicación de objetos en los *surrogates*. El número de objetos no afectará significativamente en los resultados ya que se empleará una distribución de tipo Zipf para modelar el acceso web [Ser_00]. En nuestro modelo esto queda simplificado mediante un porcentaje de replicación en los *surrogates*. Por otro lado, cabe señalar que una página web no es más que un contenedor de objetos que integra tanto texto como imágenes. Así, si las imágenes son de gran tamaño, el texto puede incluso ser despreciado. Algunas páginas web populares, según se ha observado, disponen de imágenes y gráficos desde 20 KB a 200KB. También resulta habitual encontrar en las páginas componentes de Macromedia Flash (en lugar de imágenes) incorporando contenido multimedia (audio y vídeo). Estos objetos serán omitidos en esta parte de la tesis y se estudiarán en el capítulo 4 como objetos multimedia. De igual forma, muchos de los scripts incluidos en numerosas páginas, si son de cierta complejidad, disponen de un tamaño comparable al de una imagen.

- En tercer lugar, conviene precisar qué se entiende por (a) número de *surrogates* a considerar por cliente y por (b) distribución de acceso a dichos *surrogates*. En

la implementación de la CDN, se ha pretendido que la elección del *surrogate* óptimo se realice en base a los criterios de proximidad topológica y de balanceo de carga. El primero de ellos se ha conseguido mediante una ordenación, por cada cliente, de las réplicas de menor a mayor RTT. Para simular el balanceo de carga, se ha hecho que el *surrogate* seleccionado no sea siempre el más próximo al cliente, sino que el acceso se realice de acuerdo a la distribución de probabilidades (b), que dará un mayor peso a las réplicas con menor RTT.

3.3.5. Resultados y análisis de la simulación

A continuación se describirán las diferentes simulaciones que se han llevado a cabo. Para ello, se han ido modificando uno o varios parámetros de los definidos con anterioridad, manteniendo fijo el resto. Las variables que se han medido han sido principalmente dos: la carga media de los servidores, y el tiempo medio de respuesta experimentado por los clientes. La primera variable se ha calculado en términos del número medio de conexiones activas en los servidores a lo largo de la simulación. El segundo parámetro, por su parte, representa el retardo medio que experimentan los clientes desde que solicitan una página hasta que la descargan en su totalidad.

3.3.5.1. Carga media en los servidores vs. Número de clientes

La Figura 73 muestra la evolución de la carga experimentada por los servidores frente al incremento del número de clientes. Para este último parámetro, se ha utilizado un rango de 0 a 2000 usuarios, y la representación se ha obtenido, a su vez, para distintos valores en el número de *surrogates* que conforman la CDN (5, 20 y 40 respectivamente). Cabe señalar que, para evaluar la carga experimentada por los servidores, se ha considerado el número medio de páginas web distribuidas por cada uno de éstos.

Los parámetros particulares para esta simulación se especifican en la Tabla 15. Cabe destacar que el tamaño de los objetos sigue una distribución de tipo Pareto [Mar_03] y el contenido se encuentra replicado totalmente (posteriormente se estudiará el caso en que varía la replicación). En cuanto al tráfico generado por los clientes, estos solicitan 50 páginas por sesión, los objetos por página y el intervalo entre páginas sigue una distribución de tipo Pareto [Mar_03].

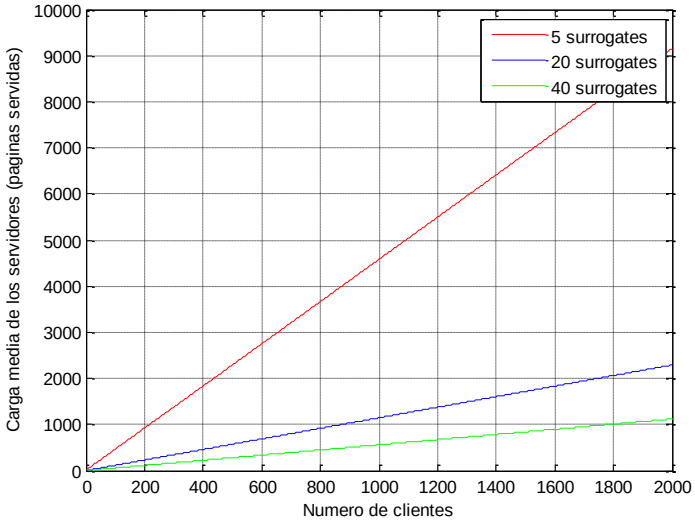


Figura 73. Carga media vs número de clientes.

Número de objetos web	2000
Tamaño de los objetos web	Pareto (avg =12KB, shape=1.2)
Porcentaje de replicación de objetos	100 %
Número de réplicas a considerar por cliente	4
Distribución de acceso a los surrogates	55% 25% 15% 5%
Modo de funcionamiento de los servidores	NO_DELAY
Patrón de tráfico de los clientes	- Nº de páginas por sesión: 50 - Nº objetos/página: Pareto (avg=4, shape=1.2) - Intervalo entre páginas: Pareto (avg=50 s, shape=2)

Tabla 15. Parámetros NS-2. Carga media vs número de clientes.

De los resultados obtenidos se pueden derivar diversas conclusiones:

- Por una parte, existe un incremento proporcional de la carga con el número de clientes. Conforme aumenta el número de clientes, los servidores reciben más peticiones que tienen que servir. Si bien esta relación es lineal, existe un umbral que no es deseable sobrepasar para no cargar un servidor. Este hecho no se ha considerado en la simulación para centrarse en la evolución de la carga, sin establecer dicho umbral de carga, que dependería de la capacidad de cada *surrogate*.

- Por otro lado, lo más relevante es la disminución que experimenta esta carga mediante un incremento en el número de *surrogates*, lo cual refleja una de las ventajas más destacadas de las CDNs. Si establecemos en la Figura 73 un número de 1000 clientes, entonces la CDN con 5 *surrogates* debe absorber un máximo de 4557 páginas web por servidor, mientras que las CDNs con 20 y 40 *surrogates* simplemente absorberán 1145 y 560 páginas respectivamente. Dado que el conjunto de páginas web en la simulación se 'reparte' entre todos los

servidores, es posible establecer una relación de proporcionalidad entre el número de *surrogates* y la carga media (medida en términos de páginas web servidas). De esta forma, la proporción en el aumento del número de *surrogates* equivale a la disminución proporcional en el número de páginas servidas.

Evidentemente, el número medio de páginas web que un servidor soporta durante una simulación es un indicador vinculado a la carga de los servidores, pero también puede medirse si se está produciendo algún tipo de congestión en la red, ya sea en la parte de los servidores o de los clientes. La Tabla 16 ilustra la velocidad máxima en términos de páginas web por segundo que un *surrogate* y un cliente pueden soportar para los anchos de banda fijados en la simulación (34 Mbps y 2 Mbps respectivamente). Como tamaño medio de página, se toma la media fijada en la simulación (12 KB) y se le añade adicionalmente un 5% en términos de sobrecarga de protocolos (*protocol overhead*). En el caso de los clientes, la velocidad máxima (pag/seg.) no representa un problema ya que su patrón de generación de tráfico web no es tan exigente.

Ancho de banda del enlace	Tamaño medio de las páginas	Velocidad máxima
34 Mbps	12 KB (+5%)	337 pag/seg.
2 Mbps	12 KB (+5%)	20 pag/seg.

Tabla 16. Máxima velocidad (pag/seg) en enlaces.

Para el caso de los *surrogates*, si se supera esta tasa de servicio, entonces se produce congestión en la red. Normalmente esto no es un caso real, ya que típicamente el ancho de banda de los *surrogates* es más que suficiente para su capacidad, de tal forma que se alcanzaría antes la sobrecarga en el servidor que la congestión en la red debido a su conexión directa a la red (sin embargo, sí es posible que haya congestión en otros tramos de la red). En las simulaciones realizadas no se ha superado en ningún caso la velocidad máxima calculada en la Tabla 16, sino que los valores medios de velocidad obtenidos eran muy inferiores (alrededor de un 70% inferior en el peor de los casos, es decir, cuando el número de clientes es de 2000).

Otro aspecto a considerar es la evolución de la carga del sistema en un escenario con y sin CDN. En un escenario sin CDN, todas las peticiones son encaminadas al servidor origen, y el valor de carga es constante: se requiere pues que la capacidad máxima de dicho servidor origen sea superior a este valor constante. Por el contrario, en un escenario con CDN, a medida que se dispone de un mayor número de *surrogates* la carga media de los servidores disminuye (tendencia asintótica a 0). Este efecto se puede apreciar en la Figura 74.

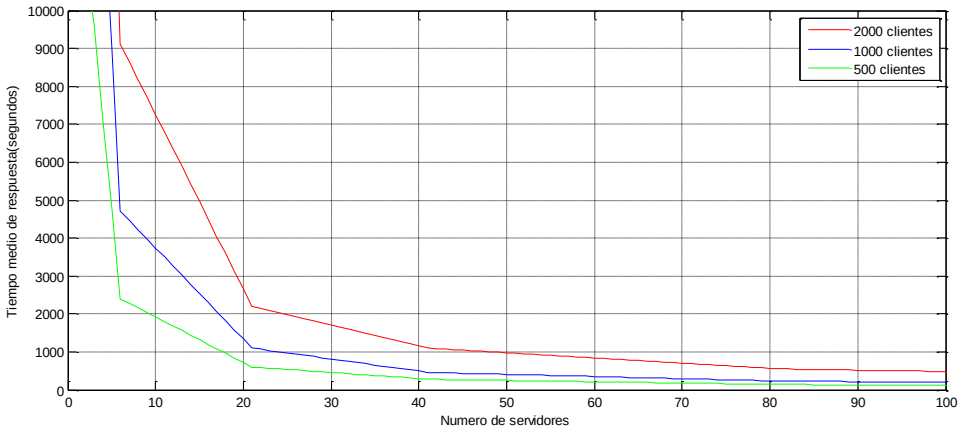


Figura 74. Carga media vs número de surrogates

3.3.5.2. Tiempo medio de respuesta vs. Número de servidores y clientes

La Figura 75 relaciona el tiempo medio de respuesta experimentado por los clientes frente al incremento en el número de *surrogates* que conforman la CDN. Como rango para este último parámetro se ha utilizado el intervalo de 0 a 40 servidores, similar a la simulación anterior, ya que 40 es un número significativamente elevado como para poder apreciar la evolución del tiempo medio de respuesta (como se observa en la Figura 75). A su vez, las representaciones se han obtenido para diferentes valores en el número de clientes, 30, 200 y 400 respectivamente. Estos valores también permiten observar su efecto en el tiempo de respuesta. El criterio seguido es tomar un valor máximo de clientes (400) 10 veces mayor que el valor máximo de *surrogates* (40).

En cuanto al tiempo de respuesta, éste se ha definido como el tiempo medio de descarga de página y representa, por tanto, el tiempo comprendido entre que el cliente contacta con el Redirector (servidor DNS local) hasta que recibe la página en su totalidad. En relación a los parámetros particulares para esta simulación, son los mismos que en la simulación anterior (véase sección 3.3.5.1) que quedaban reflejados en la Tabla 15.

Los resultados obtenidos en este caso ponen de manifiesto las ventajas de las redes de distribución de contenido en términos de disminución en el tiempo medio de respuesta:

- En primer lugar, se observa que a media que el número de réplicas es mayor, se produce una disminución en la latencia percibida por los usuarios. Esta reducción es más pronunciada en el intervalo de 0 a 6 servidores, y más suave en el resto. Esto es así debido a que, cuando existen pocos servidores, un pequeño incremento en su número provoca que los clientes puedan acceder a nuevas réplicas potencialmente próximas. No obstante, cuando éstas alcanzan un número razonable (15), los servidores que se añaden pueden no encontrarse más cercanos respecto a cualquier cliente que alguno de los ya existentes o, si lo

hacen, con unas diferencias mínimas, por lo que el tiempo medio permanecerá prácticamente inalterado. Evidentemente, la carga media disminuirá, aunque este aspecto no se analiza en esta sección.

- En segundo lugar, en relación con el aumento del número de clientes, observamos un comportamiento coherente si se tiene en cuenta que, conforme éste es mayor, el tráfico en la red y la carga de los servidores son superiores, lo que repercute directamente en un mayor tiempo de respuesta. Desde un punto de vista del diseño y dimensionamiento de la CDN, si establecemos como umbral máximo de espera un tiempo de 1 segundo, dicha CDN requeriría un mínimo de 5 *surrogates* para poder soportar hasta un máximo de 400 clientes.

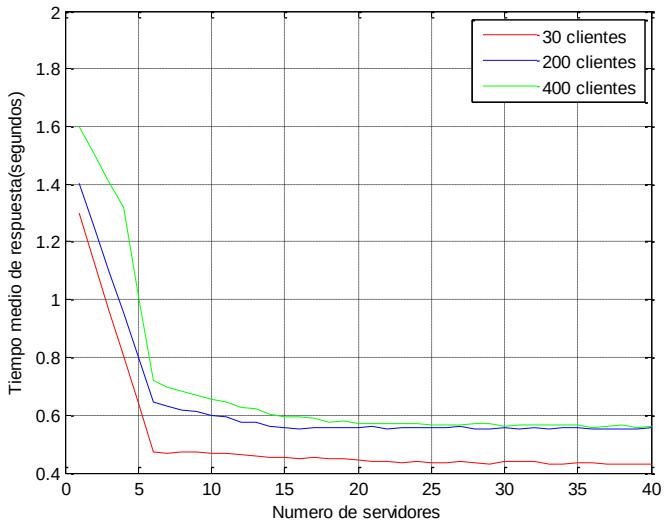


Figura 75. Tiempo medio de respuesta vs número de servidores y clientes.

3.3.5.3. Tiempo medio de respuesta vs. Número de servidores y Distribución de acceso a surrogates

Otro de los aspectos interesantes a analizar en la simulación es cómo afecta el patrón de acceso de los clientes a los *surrogates*, es decir, evaluar en qué medida un cliente debería ser redirigido siempre a su *surrogate* más cercano o, por el contrario, debería contactar también con otros *surrogates* próximos. Para ello, se ha establecido un total de 4 servidores por cliente, ordenados de menor a mayor RTT (véase Figura 76), de tal forma que dicho cliente contacta con los *surrogates* según tres tipos de patrones de acceso, como se observa en la Tabla 17:

- En el patrón tipo A, la mayor parte de las peticiones (90%) son encaminadas al *surrogate* S_1, mientras que un pequeño grupo son atendidas por los *surrogates* S_2, S_3 y S_4. Este patrón representa el extremo de encaminar las peticiones al servidor más próximo.

- En el patrón tipo C, todas las peticiones se encaminan con idénticos pesos a los *surrogates* S_1, S_2, S_3 y S_4. Este patrón representa el extremo del balanceo equilibrado entre *surrogates*.
- El patrón tipo B representa una situación intermedia donde el *surrogate* más cercano absorbe aproximadamente la mitad de las peticiones mientras que la otra mitad es absorbida por los otros servidores.

El tiempo de respuesta se ha definido (nuevamente) como el tiempo medio de descarga de página, y los parámetros particulares para esta simulación se reflejan en la Tabla 18.

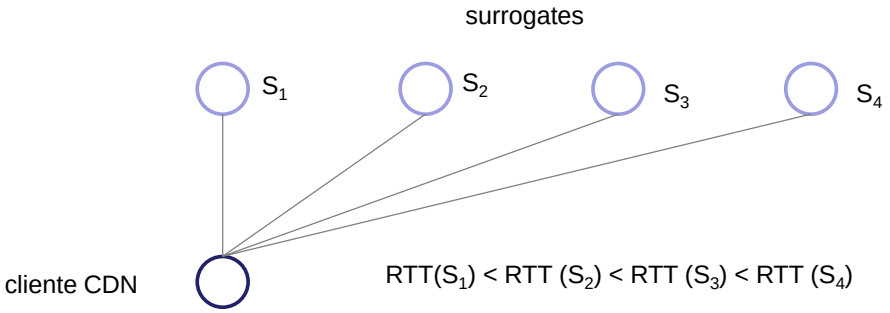


Figura 76. Esquema de acceso a 4 surrogates.

Tipo	S_1	S_2	S_3	S_4
A	90	5	3	2
B	55	30	10	5
C	25	25	25	25

Tabla 17. Patrón de acceso a los surrogates.

Número de clientes	200
Número de Redirectores	200
Número de sesiones	50
Número de objetos web	2000
Porcentaje de replicación de objetos	100 %
Tamaño de los objetos web	Pareto (avg =12KB, shape=1.2)
Modo de funcionamiento de los servidores	NO_DELAY
Patrón de tráfico de los clientes	- Nº de páginas por sesión: 50 - Nº objetos/página: Pareto (avg=4, shape=1.2) - Intervalo entre páginas: Pareto (avg=50 s, shape=2)

Tabla 18. Parámetros de simulación para distribución de acceso a réplicas.

La Figura 77 ilustra los resultados obtenidos en la simulación. Las principales conclusiones son las siguientes:

- Para pocos servidores, la distribución que ofrece unos pesos más homogéneos (tipo C) para todas los *surrogates* ofrece mejores resultados (menor RTT) que aquella que prioriza el acceso al *surrogate* más próximo al cliente (tipo A).

Dicho efecto se comprende mejor si tenemos en cuenta el relativamente elevado número de clientes (200) utilizado en la simulación. En concreto, si se emplean distribuciones destinadas a seleccionar un *surrogate* muy por encima del resto, resulta que toda la carga de tráfico es gestionada por unos pocos *surrogates* con lo que la latencia percibida por los clientes sufre una notable penalización. Sin embargo, con distribuciones más equitativas, se lleva a cabo un balanceo de carga que compensa el hecho anterior con tiempos de respuesta inferiores.

- Para muchos servidores, la distribución tipo A parece la óptima ya que ofrece tiempos medios inferiores. Esto es debido a que se dispone de *surrogates* distintos para cada clúster de clientes, de forma que ningún *surrogate* llega a saturarse. Este hecho será usado en la implementación del algoritmo de redirección, particularmente en el caso de contenido streaming, y se abordará en futuras secciones.
- La distribución tipo B es una solución intermedia entre ambas, que ofrece mejores resultados en una franja determinada (4-7 servidores). Todo esto conduce a pensar en la idoneidad de un algoritmo de redirección adaptativo que permite variar la asignación de pesos a los *surrogates* dependiendo del número de clientes y del total de *surrogates* para servirlos. Este aspecto se tendrá en cuenta cuando se describa el algoritmo de redirección en las secciones 3.4 y 4.3.

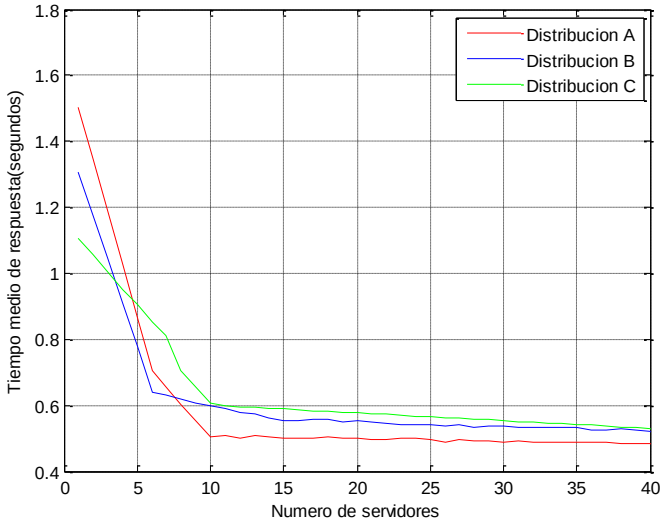


Figura 77. Tiempo medio de respuesta vs número de servidores y distribución de acceso a surrogates.

3.3.5.4. Tiempo medio de respuesta vs. Porcentaje de replicación y retardo de procesamiento

En las simulaciones previas se estaba asumiendo un porcentaje de replicación de objetos del 100%, lo que equivale a que todo el contenido del servidor origen se encuentra replicado en todos los *surrogates*. Esto no es un escenario real práctico ya que la capacidad de los *surrogates* es limitada, el contenido web es creciente con el tiempo y,

además, una CDN ofrece soporte a varios servidores origen, por lo que queda evidente que los recursos de almacenamiento en los *surrogates* están limitados. Este hecho ya se mencionó en la sección del estado del arte de las CDNs (sección 2.5) donde se describía la evolución de las CDNs desde una mera replicación hasta un esquema híbrido entre replicación y caching.

Esta sección se centra en analizar el efecto de la replicación en el tiempo medio de respuesta. En las simulaciones previas, todo el contenido estaba replicado en los *surrogates*, con lo cual no era necesario contactar con el servidor origen ya que el contenido estaba disponible (*cache hit*). Esto representaba el mejor caso posible, con lo cual los resultados obtenidos representaban el caso más favorable, al no ser necesaria la comunicación entre *surrogate* y servidor origen. Adicionalmente, los resultados de las simulaciones previas se han obtenidos para un retardo de procesamiento en los servidores mínimo (NO_DELAY), como se podía apreciar en la Tabla 15 y Tabla 18.

En este caso, se va a considerar también el efecto de la carga de los *surrogates*, introduciendo un retardo de procesamiento con una política de tipo FCFS (*First Come, First Served*), como se describía en la Tabla 14.

Número de clientes	200
Número de Redirectores	200
Número de sesiones	50
Número de objetos web	2000
Número de surrogates	15
Tamaño de los objetos web	Pareto (avg =12KB, shape=1.2)
Número de réplicas a considerar por cliente	4
Distribución de acceso a las réplicas	90% 5% 3% 2%
Patrón de tráfico de los clientes	- Nº de páginas por sesión: 50 - Nº objetos/página: Pareto (avg=4, shape=1.2) - Intervalo entre páginas: Pareto (avg=50 s, shape=2)

Tabla 19. Parámetros de simulación para porcentaje de replicación.

Los parámetros de simulación en este caso se recogen en la Tabla 19. Dado que tomamos 15 *surrogates* en la simulación, se ha escogido la distribución tipo A que obtenía mejores resultados (véase sección 3.3.5.3).

La Figura 78 muestra el comportamiento de la latencia percibida por los clientes respecto al incremento del porcentaje de replicación de objetos. Dicho porcentaje se ha incrementado del 0 al 100% con saltos del 10%. En cuanto a las distintas representaciones, éstas se han obtenido para diferentes configuraciones de los *surrogates*. En concreto los modos utilizados han sido:

- sin retardo de procesamiento (NO_DELAY)
- con retardo de procesamiento (FCFS, tasa de proceso de 200 kB/s)
- con retardo de procesamiento (FCFS, tasa de proceso de 100 kB/s)

Las principales conclusiones que se desprenden de la Figura 78 son las siguientes:

- El tiempo medio de respuesta se ve reducido ante un incremento en el porcentaje de replicación de objetos. Esto es debido a que, con un mayor porcentaje, el *hit rate* es superior, con lo que la probabilidad de que el *surrogate* deba descargar el recurso del servidor origen es menor y, por tanto, el retardo que experimentan los clientes se reduce de igual forma.

- En lo que respecta a la configuración de los *surrogates*, observamos cómo su tasa de procesamiento tiene una influencia directa sobre el retardo experimentado por los clientes, de forma que cuanto menor es dicha tasa, mayor es el tiempo que se invierte en la obtención del recurso. El caso más favorable corresponde con un retardo de procesamiento mínimo (NO_DELAY), en cualquier otro caso, el retardo medio se ve incrementado. Si, por ejemplo, se desea garantizar un tiempo medio de respuesta máximo de 1 segundo, entonces:

    - Se requiere un porcentaje de replicación del 60% para unos *surrogates* con política FCFS de 100 KB/s.

    - Se requiere un porcentaje de replicación del 40% para unos *surrogates* con política FCFS de 200 KB/s.

- Si comparamos estos resultados con los obtenidos en la Figura 75, se puede observar que, en este caso, los valores del tiempo medio de respuesta obtenidos para un porcentaje de replicación nulo (valores entre 1.8 y 2.1 segundos) son superiores a los obtenidos en la Figura 75 para el caso en que no hay *surrogates*. Esto se debe a que, en el escenario de la Figura 78, el cliente accede en primer lugar al *surrogate* más próximo, el cual, a su vez, solicita el recurso al servidor origen; en el escenario de la Figura 75, por el contrario, el cliente accede directamente al servidor origen sin contactar previamente con ningún *surrogate*.

Aunque no aparece reflejado en la Figura 78, es importante destacar que la carga de los *surrogates* también se ve influenciada por el porcentaje de replicación, ya que si el porcentaje de replicación es reducido, una parte significativa de la carga se destina a contactar con el servidor origen para obtener el recurso. Por el contrario, si el porcentaje de replicación es elevado, el *hit rate* es también elevado, y la mayor parte de la carga de los *surrogates* está destinada a gestionar los objetos sin necesidad de contactar con el servidor origen.

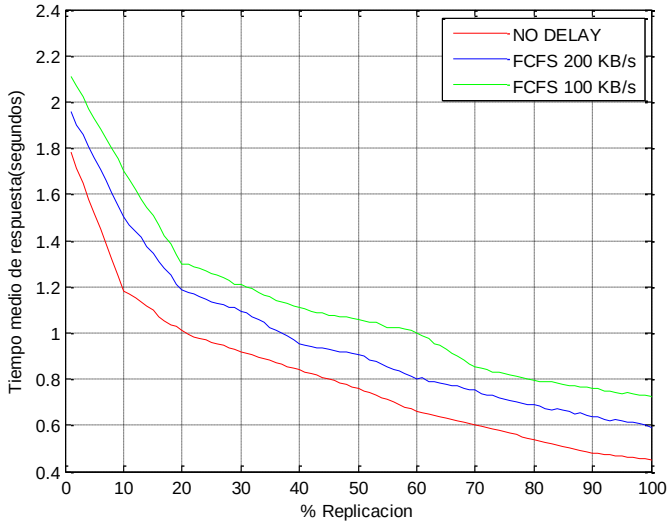


Figura 78. Tiempo medio de respuesta vs porcentaje de replicación y retardo de procesamiento.

3.3.5.5. Tiempo medio de respuesta vs. Número de servidores y porcentaje de replicación

En secciones anteriores se ha estudiado el tiempo medio de respuesta variando el número de *surrogates*, el número de clientes, la distribución de acceso y el retardo de procesamiento. En esta sección, se compara el efecto combinado del número de *surrogates* y porcentaje de replicación sobre el tiempo medio de respuesta. Los parámetros específicos de esta simulación se recogen en la Tabla 20.

Número de clientes	200
Número de Redirectores	200
Número de sesiones	50
Número de objetos web	2000
Tamaño de los objetos web	Pareto (avg =12KB, shape=1.2)
Número de réplicas a considerar por cliente	4
Distribución de acceso a las réplicas	90% 5% 3% 2%
Modo de funcionamiento de los servidores	NO_DELAY
Patrón de tráfico de los clientes	- Nº de páginas por sesión: 50 - Nº objetos/página: Pareto (avg=4, shape=1.2) - Intervalo entre páginas: Pareto (avg=50 s, shape=2)

Tabla 20. Parámetros de simulación para tiempo de respuesta (con y sin CDN).

La Figura 78 ilustra los resultados de la simulación, donde se han aplicado diferentes valores de porcentaje de replicación (100%, 70% y 50% respectivamente). Adicionalmente se ha introducido el tiempo de respuesta obtenido al contactar únicamente con el servidor origen, de forma que se representa un escenario con CDN y

otro sin CDN, que permite una mejor comparación. Para la obtención de estas representaciones, por cada nodo cliente se ha generado un segundo nodo cliente adicional con las mismas características y conectado al mismo punto de la red. De esta forma, durante la simulación, ambos clientes solicitan el mismo conjunto de páginas pero, en el primer cliente hace uso de la CDN (escenario con CDN) y el segundo cliente accede directamente al servidor origen (escenario sin CDN). Las principales conclusiones son las siguientes:

- Conforme aumenta el número de servidores, el tiempo de respuesta disminuye, con un mayor efecto perceptible cuando el número de *surrogates* es pequeño. Este efecto ya se analizaba en la sección 3.3.5.2, si bien ahora conviene incluir el efecto de la replicación. En este caso, la replicación total (100%) permite obtener una cota mínima de retardo correspondiente al caso más favorable, mientras que si el porcentaje de replicación disminuye el tiempo de respuesta aumenta, ya que son necesarias más comunicaciones entre los *surrogates* y el servidor origen.

- Si comparamos los escenarios con y sin CDN tomando el caso más favorable (100% de replicación), entonces se aprecia una disminución significativa (hasta un 63% menor) del tiempo medio al emplear una CDN, sobre todo conforme aumenta el número de *surrogates*. Téngase en cuenta que en el escenario sin CDN los clientes contactan directamente con el servidor origen, por ello su retardo medio permanece inalterado en la Figura 79, ya que el número de *surrogates* no afecta.

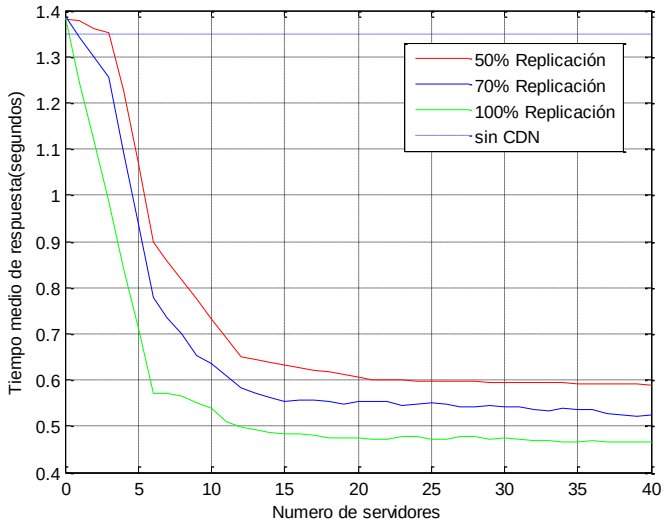


Figura 79. Tiempo medio de respuesta vs número de servidores y porcentaje de replicación

3.3.5.6. Tiempo medio de respuesta vs. Caching y replicación

En las simulaciones anteriores se asumía que los *surrogates* eran capaces de almacenar todo el contenido disponible del servidor origen por motivos de simplicidad (en el modelo y en la simulación). Sin embargo, esto no es una situación real ya que el tamaño de los discos en los *surrogates* es limitado y, en no pocas ocasiones, se tiene que compartir con varios servidores origen que corresponden a empresas (clientes) distintas. Es por ello por lo que se tendrá en cuenta en este momento un tamaño de disco limitado en los *surrogates*, inferior al volumen de objetos disponible en el servidor origen. Por otro lado, este espacio de disco en los *surrogates* se va a dividir en dos partes, siguiendo el modelo propuesto en [Sta_06]:

- ***Contenido estático (replicación)***: en esta parte se replica contenido estático del servido origen.
- ***Contenido dinámico (caching)***: esta parte actúa como un *proxy cache* replicando contenido dinámicamente.

Cuando se recibe una petición de un cliente se examina en primer lugar la caché estática. Si se produce un acierto se devuelve el objeto solicitado al cliente; en caso contrario se busca en la caché dinámica. Si el contenido se encuentra disponible en esta última, se sirve al cliente y se actualiza la caché. En caso contrario, se obtiene del servidor origen. El objetivo de disponer de una parte (cache) estática es para garantizar una cierta disponibilidad de contenido sin necesidad de contactar con el servidor origen, lo que mejorará los tiempos medios de respuesta como se mostrará a continuación. Para caracterizar adecuadamente esta nueva situación es necesario ampliar el modelo de simulación, teniendo en cuenta los parámetros presentes en [Sta_06], que quedan reflejados en la Tabla 21.

Parámetro	Descripción
W	Sitio web
N	Número de objetos del sitio web W
W(s)	Tamaño del sitio web W
U_k(s)	Tamaño del k-ésimo objeto
M	Número de surrogates
M_i(s)	Capacidad de almacenamiento del i-ésimo surrogate ($M_i(s) = p*W$)
f_{ik}	Función que indica si el k-ésimo objeto está ubicado en el i-ésimo surrogate
r	Porcentaje de replicación (r + c =1)
c	Porcentaje de caching (r + c =1)

Tabla 21. Variables de caracterización de caching y replicación.

Como algoritmo de replicación se ha tomado el algoritmo il2p [Pal_06b] que tiene en cuenta la carga de los servidores. Más concretamente, il2p emplea dos fases o iteraciones. En la primera iteración, se selecciona el *surrogate* apropiado para cada

objeto minimizando la latencia de red. En la segunda fase, dado un par (objeto, *surrogate*) se selecciona aquel que minimiza una función de utilidad dependiente de la carga del servidor. Como políticas de caching se han empleado LRU, LFU y SIZE.

La Figura 80 muestra los resultados de la simulación para varios valores de replicación y caching. Los límites del rendimiento quedan determinados por:

- ***Replicación total (full mirroring):*** corresponde al caso en que todo el contenido del sitio web está replicado en todos los *surrogates*. Como se ha mencionado anteriormente esto no es un caso real, pero representa el valor más pequeño que se obtendría para el tiempo medio de respuesta percibido por el cliente.

- ***Discos vacíos:*** corresponde a la situación en que los discos de los *surrogates* están completamente vacíos, tanto su parte para replicación como para caching. Este sería el caso más desfavorable y representa el valor más elevado para el tiempo medio de respuesta, ya que se debe contactar primero con el servidor origen y es necesario generar más tráfico.

Como se puede observar en la Figura 80, la replicación total no parece ofrecer el mejor rendimiento en términos de tiempo medio de respuesta. Esto es debido a que si W(s) es mucho mayor que $M_i(s)$, entonces un número relativamente pequeño de *surrogates* (20 en esta simulación) no pueden producir suficientes aciertos y es necesario contactar con el servidor origen. Sin embargo, para un 80% de replicación y un 20% de caching, se obtiene entre un 60% y un 70% de reducción en el tiempo de respuesta, y supone un pico para las tres políticas de caching consideradas. La opción de caching total (sin replicación) también parece una buena opción ya que el tamaño medio de los objetos es bastante pequeño (12 KB) y es posible almacenar bastantes objetos en caché. Sin embargo en la práctica se prefiere disponer de una cierta disponibilidad de contenido.

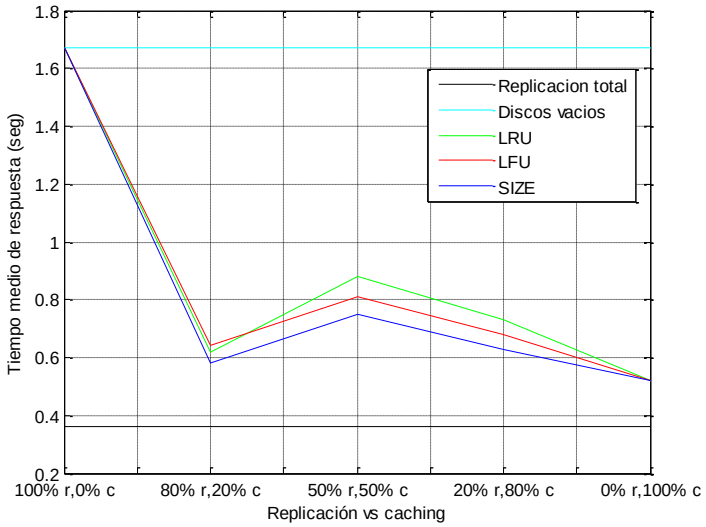


Figura 80. Tiempo medio de respuesta con replicación y caching.

La función de distribución DCF (*Distributed Cumulative Function*), que indica la probabilidad de disponer de tiempos de respuesta menores (o iguales) a un cierto valor. Téngase en cuenta que, en una CDN, uno de los objetivos fundamentales es aumentar la probabilidad de tener tiempos de respuesta cerca del límite teórico inferior.

La Figura 81 muestra la función DCF para un 80% de replicación y un 20% de caching. Puede apreciarse como, para las tres políticas de caching, esta configuración se acerca al límite ideal (replicación total), si bien la política SIZE es ligeramente mejor. En estas condiciones (con cualquiera de las tres políticas de caching), la probabilidad de que un cliente cualquiera obtenga una respuesta inferior a un segundo es mayor del 99%, si bien en el caso de discos vacíos solamente un 40% de clientes obtendrían una respuesta por debajo de un segundo.

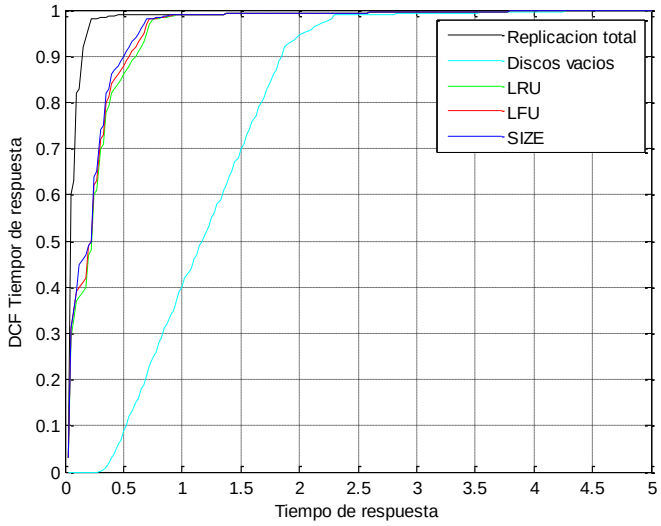


Figura 81. DCF Tiempo de respuesta.

3.3.5.7. Hit ratio y byte hit ratio

Otro de los parámetros que se puede estudiar en el análisis del rendimiento del modelo es el *hit ratio* y el *byte hit ratio*. Como se describió en la sección 2.3.1, el *hit ratio* se define como la fracción de aciertos en relación con el número total de peticiones. Si se mide en unidades de bytes, el concepto recibe el nombre de *byte hit ratio*. Estos dos conceptos tienen sentido cuando se emplean diferentes políticas de caching para poder comparar mejor su efectividad, por eso se seguirá con la misma simulación que en la sección anterior (3.3.5.6), donde se empleaban políticas LRU, LFU y SIZE.

Tanto en la Figura 82 como en la Figura 83 se puede observar la presencia de un pico para un 80% de replicación y un 20% de caching, donde la política SIZE ofrece mejores resultados en hit ratio (debido al pequeño tamaño de los objetos) aunque peores en byte hit ratio. Para el caso de una replicación total, los valores de hit ratio y byte hit ratio son prácticamente nulos ya que la capacidad de almacenamiento (parámetro p=0.015) resulta bastante limitante y, en estas circunstancias, el algoritmo de replicación no presenta un buen funcionamiento. La motivación de este valor tan reducido del parámetro p es doble. Por un lado (aspecto práctico de simulación), el algoritmo il2p requiere un mayor tiempo de proceso que aumenta con la capacidad de almacenamiento. Por otro lado (aspecto real), en la práctica se prefiere valores pequeños de la memoria caché porque, de esta forma, se agilizan las búsquedas y las actualizaciones.

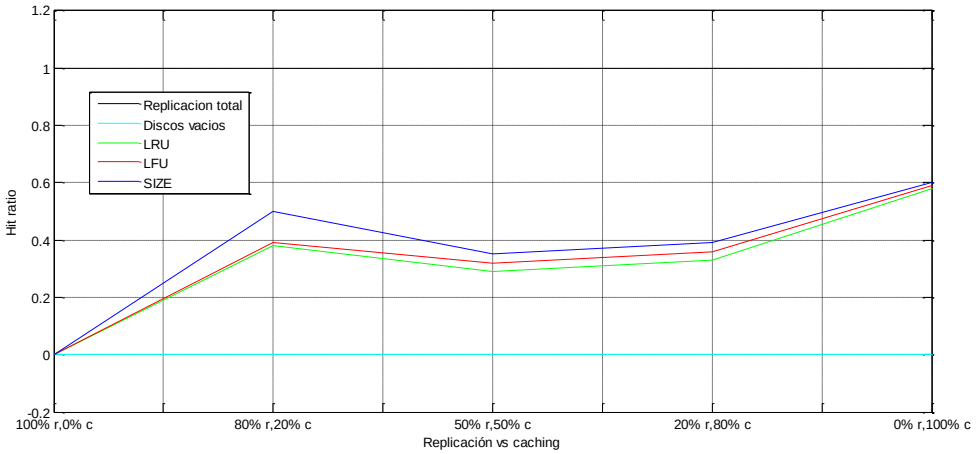


Figura 82. Hit ratio con replicación y caching.

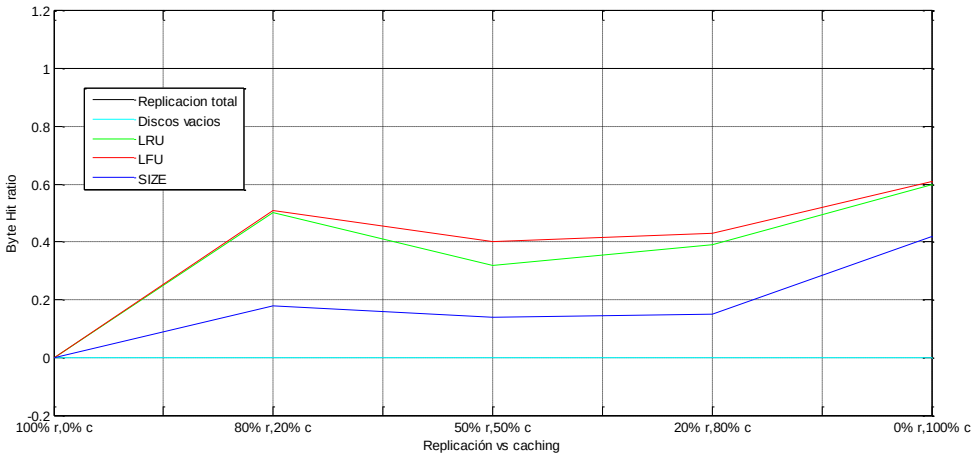


Figura 83. Byte hit ratio con replicación y caching

3.3.5.8. Efecto *flash-crowd*

Es importante analizar el comportamiento del modelo de simulación de CDN cuando se produce un *flash-crowd*, de igual forma que se hizo en el caso del modelo analítico. En este caso, es necesario generar una gran cantidad de peticiones con la finalidad de observar cómo la CDN es capaz de 'absorber' este impacto.

La Figura 84 muestra los resultados para el tiempo medio de respuesta. La situación de discos vacíos conduce a un estado inestable donde los tiempos de respuesta resultan desproporcionados. Por el contrario, el caso de replicación total ofrece un rendimiento similar a cuando no se produce un *flash-crowd* ya que se evita la generación de tráfico en la red troncal (y los *surrogates* no llegan a saturarse). En esta situación, se puede observar como una CDN adecuadamente configurada es capaz de proporcionar unos tiempos de respuesta muy reducidos, ya sea mediante replicación total, caching o una combinación de ambas.

Comparando los resultados de este experimento con los de la Figura 82, se puede observar que, en este caso, se obtiene un mejor rendimiento en términos de tiempo de respuesta para un porcentaje de replicación del 80% y un porcentaje de caching del 20%, en lugar de un caching al 100%, más acentuado en el caso de una política de caching tipo SIZE. Esto es debido a que, cuando se produce un efecto *flash-crowd*, es muy importante disponer del contenido (replicación) sin necesidad de generar tráfico adicional. Un pequeño porcentaje de caching (20%) permite adaptarse a las variaciones en la popularidad de los objetos y, de esta forma, obtener mejores tiempos de respuesta que con un disco replicado y sin caching.

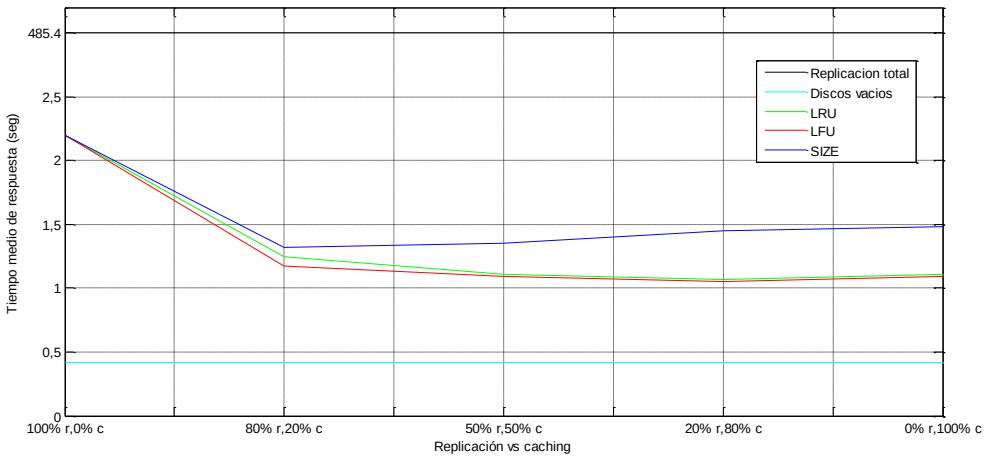


Figura 84. Tiempo de respuesta (*flash-crowd*).

3.3.6. Comparación con otros modelos de simulación

Una vez realizado el análisis del modelo de simulación, es necesario comparar los resultados obtenidos con el modelo analítico, así como con otros simuladores (CDNsim y CDN Simulator), disponibles y descritos en la sección 3.3.2. La finalidad de este estudio comparativo es doble:

- Por un lado, permite validar el primer modelo analítico propuesto (sección 3.2) y establecer en qué medida las hipótesis asumidas en dicho modelo eran válidas o razonables.

- Por otro lado, la comparación con otros modelos propuestos por la comunidad científica permite validar el propio modelo de simulación, detectar las propiedades de unos y otros y establecer en qué medida se complementan. Tampoco tendría demasiado sentido crear dos simuladores idénticos.

Es importante destacar que se debe ser muy cuidadoso a la hora de comparar y evaluar resultados entre dos sistemas diferentes (en nuestro caso dos modelos distintos), ya que muchas veces las hipótesis iniciales son diferentes y los resultados solamente son comparables en un determinado rango de valores. De hecho, las variantes en los resultados muchas veces permiten detectar las diferencias en el diseño y en la implementación de ambos modelos, sin que necesariamente ninguno de los dos sea erróneo.

3.3.6.1. Comparación con el modelo analítico

En esta sección se realizará una comparación del modelo de simulación desarrollado en NS-2 con el modelo analítico analizado en la sección 3.2. En dicho modelo analítico se proponía un modelo de CDN y se establecía una ecuación que permitía obtener el tiempo de respuesta, que es el parámetro fundamental en el análisis del rendimiento de una CDN y, por ello, se presta una especial atención es esta tesis. El modelo analítico permitía dividir el tiempo de respuesta global desde dos puntos de vista:

- Como el tiempo de respuesta de los *surrogates* y el servidor origen.

- Como el tiempo de respuesta asociado al tiempo de ida y vuelta (RTT) y tiempo de proceso en los servidores (tanto *surrogates* como servidor origen).

A partir de esta visión del tiempo de respuesta, se realizaban diversas simulaciones donde la variable independiente era la probabilidad de acierto *p*, que indicaba en qué medida un cliente (clúster de clientes) contactaba bien con un *surrogate* cercano o bien con el servidor origen. Si *p=0*, el cliente contactaba con el servidor origen, mientras que si *p=1* el cliente contactaba con un *surrogate* cercano. Desde el punto de vista de la

arquitectura de funcionamiento de una CDN, se está asumiendo un algoritmo de encaminamiento inmediato (tiempo de respuesta nulo) capaz de indicar a un cliente en qué medida debe contactar con un *surrogate* o contactar directamente con el servidor origen. En otros términos, si p=0.1 para un cliente C quiere indicar, de una manera muy simplificada, que la efectividad del *caching* de los *surrogates* asignados al cliente C es del 10%.

En el modelo de simulación propuesto, por el contrario, se introduce una entidad adicional a clientes, *surrogates* y servidor origen, denominado redirector y capaz de ejecutar un algoritmo de encaminamiento. En este modelo de simulación también se ha simplificado la carga de procesamiento de dicho algoritmo, que se ha supuesto nula, aunque sí se ha considerado el tiempo en el envío de mensajes desde el cliente al redirector. Considerar la carga de procesamiento nula no representa una limitación sustancial si se desea centrar el análisis en los tiempos de respuesta debidos a la congestión de red y de los *surrogates*. Dicha carga de proceso podría considerarse un valor constante e igual al tiempo necesario en obtener la información sobre la disponibilidad de servidores y realizar las correspondientes asignaciones por cliente (o clúster de clientes). Si consideramos que este cálculo puede realizarse como parte de un servicio, tarea programada (Windows) o demonio (Unix), este valor puede ser descartado inicialmente en el cálculo del tiempo de respuesta.

Por otro lado, el modelo de comunicación en el modelo de simulación es más complejo, ya que se establecen más canales de comunicación que en el modelo analítico:

- El cliente contacta con el redirector y con el *surrogate* correspondiente.
- El cliente no contacta con el servidor origen, sino con el *surrogate*. Es el *surrogate* quien, realizando funciones de servidor *proxy cache*, contacta con el servidor origen, obtiene el contenido solicitado y se lo proporciona al cliente.

Sin embargo, pese a las grandes diferencias entre modelos, sí es posible realizar una comparación a modo cualitativo entre ambos modelos (analítico y de simulación). Para ello, se puede ilustrar si el tiempo medio de respuesta se ve reducido conforme aumenta el número de *surrogates*. La Figura 85 muestra esta comparación para un número de 400 clientes. Los resultados del modelo de simulación (NS-2) se han obtenido directamente de la Figura 75. En el caso del modelo analítico, se han realizado varias adaptaciones para poder proceder a la comparación:

- En primer lugar, se asume que la replicación es total, por lo que el *hit_ratio* es igual a la unidad. Nótese, por un lado, que en el modelo analítico la probabilidad de acierto era una variable en el análisis, mientras que en este caso permanece fija. Por otro lado, este hecho (replicación total) permite que ambos escenarios sean más homogéneos, ya que, de esta forma, los clientes sólo contactan con los *surrogates* en los dos modelos.

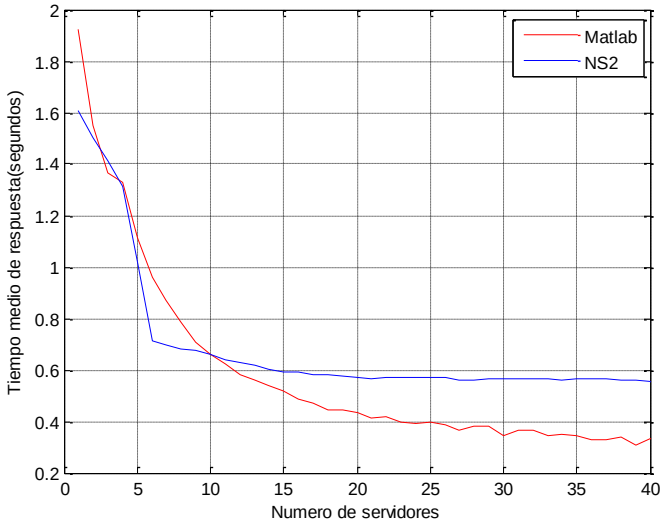


Figura 85. Comparación modelo analítico y modelo de simulación.

- En segundo lugar, el parámetro α (α<1) debe modificarse con la finalidad de establecer las mismas condiciones de acceso desde un cliente a sus respectivos *surrogates*, así como el porcentaje de acceso a estos. De esta forma, y para el escenario de comparación, en el modelo analítico cada cliente contactará siempre con los cuatro *surrogates* más cercanos, con una probabilidad de acceso, según cercanía, de 55%, 25%, 15% y 5% respectivamente. Esto permite realizar una comparación entre modelos más adecuada.

En la Figura 85 se observa cómo el retardo medio experimentado por los clientes disminuye en ambos modelos a medida que el número de *surrogates* se ve incrementado. La reducción sigue siendo más pronunciada para un número de *surrogates* reducido, como ya se comentaba en la sección 3.3.5.2. Esta reducción es aún más acusada en el escenario de comparación, ya que el porcentaje de replicación es del 100%, y no se requiere contactar con el servidor origen en ninguno de los casos. Conforme aumenta el número de *surrogates*, se puede apreciar cómo en el modelo de simulación el retardo medio se mantiene más o menos constante, mientras que en el modelo analítico, aunque ligeramente, aun se produce una reducción del retardo. La explicación es la siguiente: en el modelo de simulación, a partir de un cierto número de *surrogates*, al añadir nuevos servidores en la topología de red afecta muy poco en el retardo, ya que dichos *surrogates* pueden no estar necesariamente más cerca que los ya existentes, como se comentaba en la sección 3.3.5.2. En el modelo analítico, por otro lado, la generación de la topología de red (véase sección 3.2.2) implica necesariamente que, conforme aumenta el número de *surrogates*, la distancia entre clientes y *surrogates* disminuye, y por ello su retardo también. Evidentemente, la generación de la topología de red en el modelo de simulación es mucho más realista que en el modelo analítico.

3.3.6.2. Comparación con CDNsim

CDNsim es un sistema de simulación de CDNs basado en el entorno de simulación OMNetT++ (Objective Modular Network Test-bed in C++) [WWW_OMN] y el marco INET [WWW_Inet]. INET es una extensión de OMNeT++ para proporcionar protocolos de red como TCP/IP. CDNsim emplea OMNeT++ únicamente para las operaciones básicas de red como las transmisiones TCP/IP, así como para la planificación de eventos discretos. El encaminamiento, la distribución de contenido y la gestión están simulados directamente por CDNsim cuyas características principales, asimilables a las descritas en la sección 2.5.4, se presentan de forma resumida en la Tabla 22.

Configuración	Organización	Surrogates y elementos de red
	Servidores	Servidor origen y surrogates
	Relaciones	Cliente → surrogate → origen
	Interacción	Red, intercaché
	Tipo de contenido	Contenido estático, streaming, servicios
Distribución de contenido y gestión	Ubicación de surrogates	Cualquier política o configuración
	Selección de contenido y distribución	Cualquier política o configuración
	Externalización de contenido	Push cooperativo; push no cooperativo, pull cooperativo, pull no cooperativo
	Organización de cache	Bajo demanda, actualizaciones automáticas
Encaminamiento	Mecanismo de encaminamiento	Basado en DNS

Tabla 22. Principales características de CDNsim.

Si bien la configuración en la Tabla 22 ya se describió en la sección 2.5.4, cabe comentar los aspectos de distribución de contenido y gestión, así como del encaminamiento. En el ámbito de la configuración, cabe destacar que tanto nuestro modelo de simulación como CDNsim son muy parecidos, y cada uno de los diferentes tipos de nodo especificados (clientes, *surrogates*, servidores origen, routers y redirectores DNS) realizan idénticas funcionalidades.

En relación a la ubicación de *surrogates*, que es un problema ya conocido y afecta al rendimiento (véase sección 2.5.4.4) se debe tratar de encontrar aquella topología que minimice la latencia percibida por el usuario así como el coste de infraestructura. CDNsim no implementa ninguna política en este sentido de manera nativa, sino que ofrece una forma genérica de introducirle mediante un fichero de texto (NED, *Network Description*) una topología de red donde se haya aplicado algún tipo de algoritmo de ubicación de *surrogates*, como:

- *Greedy*, que ubica servidores de manera incremental [Kri_00b].

- *Hot Spot*, que ubica servidores junto a los clientes que generan mayor tráfico [Qui_01].

- *Árbol*, que asume que la topología subyacente son árboles, y modela la estrategia de ubicación como un problema de programación dinámico [Li_98].

En relación a la selección de contenido y distribución, que también es un problema conocido y afecta al rendimiento, CDNsim tampoco implementa de manera nativa ningún mecanismo, y es el usuario quien debe configurar esta política de gestión, pudiéndose implementar mecanismos como comunidades de páginas web [Kat_08] [Sid_08].

CDNsim soporta cuatro tipos de externalización de contenido:

- ***Entorno cooperativo (surrogate más cercano)***: en este caso, el contenido se distribuye de manera proactiva desde el servidor origen hacia los *surrogates*. Inicialmente, el contenido se almacena en caché antes de ser accedido (*prefetching*) y, posteriormente, los *surrogates* cooperan con la finalidad de reducir los costes de replicación y actualización. En este esquema, CDNsim mantiene un mapeo entre el contenido y los *surrogates*, y cada petición se encamina al *surrogate* más cercano que dispone de dicho contenido; en caso contrario se encamina al servidor origen.

- ***Entorno no cooperativo (origen más cercano)***: en este esquema, el contenido se envía de forma proactiva desde el servidor origen a los *surrogates*. Las peticiones serán satisfechas por un *surrogate* local cercano. Si dicho *surrogate* no dispusiera del contenido solicitado, lo obtiene del servidor origen más cercano, pero no coopera con otros *surrogates* cercanos. Nótese que CDNsim soporta la posibilidad de disponer de varios servidores origen.

- ***Entorno cooperativo (surrogate aleatorio)***: en esta situación, el cliente es redirigido a un *surrogate* aleatorio. Si el contenido no está en caché, se obtiene de forma cooperativa de un *surrogate* aleatorio que disponga de dicho contenido. Si el contenido no ha sido externalizado, entonces se obtiene el contenido del servidor origen más cercano antes de servirlo al cliente.

- ***Entorno cooperativo (balanceo entre surrogates)***: en este escenario, el cliente es redirigido al *surrogate* más cercano en términos de saltos de red. Si el *surrogate* presenta una carga superior al 95% el cliente es redirigido al *surrogate* menos cargado. Si el contenido no se encuentra en caché, se obtiene de forma cooperativa del *surrogate* más cercano que contenga dicho contenido. Nuevamente, si este *surrogate* contactado presenta una carga superior al 95% se contacta con el *surrogate* menos cargado que disponga del contenido solicitado. Si el contenido no ha sido externalizado, se contacta con el servidor origen más

cercano, pero si su carga es superior al 95% se contacta con el servidor origen menos cargado.

En relación a la organización de la caché, CDNsim es capaz de soportar actualizaciones tanto bajo demanda como periódicas, aunque no las implementa de forma nativa. También es posible desarrollar otras políticas o emplear heurísticas para determinar la organización de la caché [Lao_05] [Sta_06].

Finalmente, relativo al encaminamiento descrito en la Tabla 22, cabe mencionar que CDNsim emplea un esquema basado en DNS. Cuando se genera una petición, el servidor DNS interno en CDNsim devuelve una IP con la dirección de los *surrogates* que disponen de dicho contenido. El cliente DNS (*resolver*) envía una prueba a todos los *surrogates* y contacta con aquel que contesta en un tiempo menor.

El rendimiento de CDNsim ha sido analizado en la literatura científica [Sta_10] en base a varios parámetros. A continuación se describen dichos parámetros y se procede a su comparación con nuestro modelo de simulación:

- ***Tamaño de red***: el experimento consiste en crear topologías para un número de nodos elevado (hasta 3037) y observar la memoria consumida en la simulación. Se puede observar un crecimiento lineal de la memoria requerida conforme aumenta el número de nodos (véase Figura 86), y OMNetT++ resulta más eficiente en este sentido que NS-2. Generando la topología mediante GT-ITM en tres modos distintos (aleatorio, AS, transit-stub), se comprueba que el factor más destacado en el consumo de memoria es el número de enlaces. El modo aleatorio es el que presenta un mayor consumo de memoria, pues genera más enlaces, mientras que el modo AS presenta un menor consumo de memoria, ya que, al crearse una estructura jerárquica, el número medio de saltos entre nodos no aumenta significativamente conforme aumenta el tamaño de la topología. En términos relativos, el modo AS consume casi un 50% menos de memoria que el modo aleatorio.

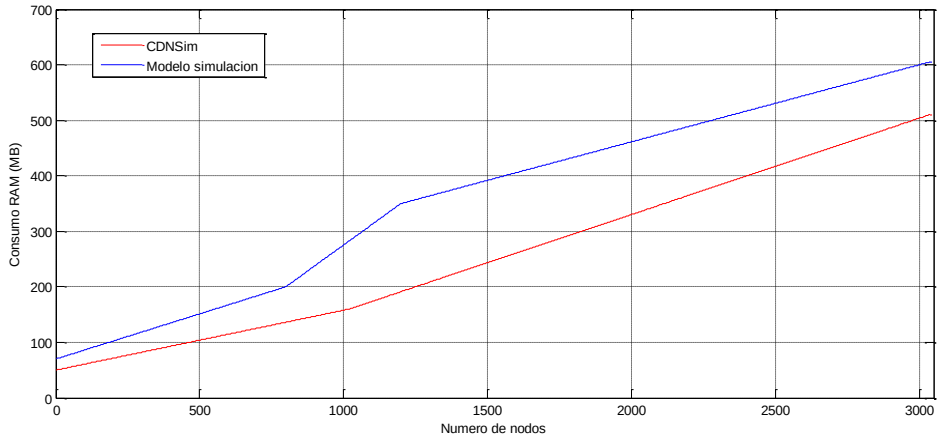


Figura 86. Uso de memoria vs. Topología de red.

- ***Número de peticiones***: en este caso, se fija una estructura de red (1000 nodos) y se calcula el tiempo requerido por la CPU para ejecutar una simulación conforme aumenta el número de peticiones (hasta 1 millón), sobre un mismo equipo (Pentium Core 2 Duo). Se puede observar (véase Figura 87) como el comportamiento sigue un esquema lineal. Tanto en CDNsim como en nuestro modelo de simulación los tiempos requeridos por la CPU son similares, sin grandes diferencias.

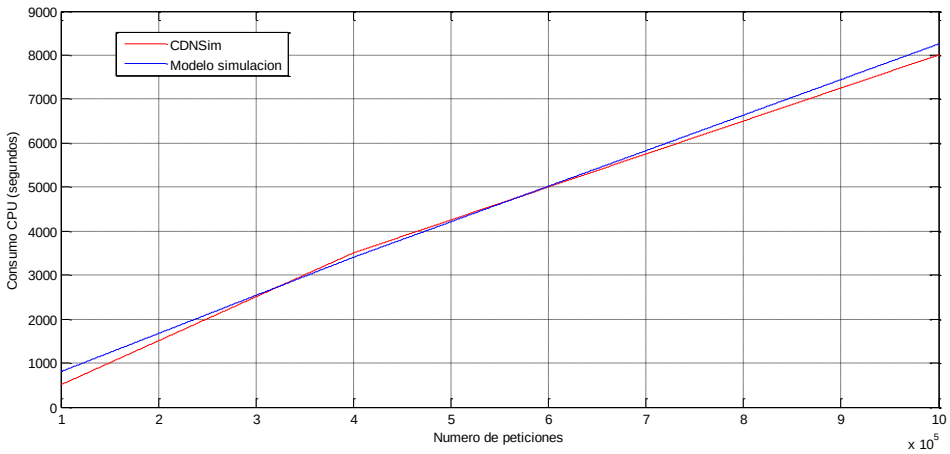


Figura 87. Tiempo de CPU vs. Número de peticiones.

- ***Modelo de peticiones***: en este caso, para producir diferentes modelos de demanda se ha empleado un generador de peticiones que captura las características más particulares de la Web y se encuentra descrito en [Kat_08]. Este generador crea una carga de peticiones según unos modelos matemáticos y procesos estocásticos. Para el experimento, se han generado varios modelos de demanda variando los siguientes parámetros (que son los que mayor impacto tienen en la carga):

    - La pendiente Zipf de la distribución de popularidad.
    - El índice de cola pesada (*heavy tailed*) Pareto de la distribución de tamaños.
    - El porcentaje de objetos dentro de cada petición.

En dicho experimento se ha tomado una topología de 1008 *routers*, 100 *surrogates* y una política LRU (para el caso de CDNsim). Los valores obtenidos se reflejan en la Tabla 23 de manera comparativa entre CDNsim y nuestro modelo de simulación. Los parámetros de simulación en el caso de CDNsim se han tomado directamente de [Kat_08], y se han replicado las condiciones para nuestro modelo de simulación, incluyendo el caso de la política de caching (LRU).

					CDNsim		Modelo simulación	
Nº peticiones	**objetos**	**Zipf**	**Pareto**	**% objetos distintos**	**Memoria (MB)**	**CPU (seg)**	**Memoria (MB)**	**CPU (seg)**
985810	100000	0.75	1.2	10	1274.95	4881.80	1428.54	4979.78
990792	300000	0.75	1.2	30	2927.99	5903.35	3259.38	6048.34
920694	800000	0.75	1.2	80	7098.92	6774.57	7893.64	6867.23
970446	300000	0.01	1.2	30	2928.84	6690.19	3256.34	6679.32
965839	300000	0.25	1.2	30	2927.65	5896.69	3255.94	5912.87
954703	300000	0.5	1.2	30	2922.51	5561.09	3244.23	5602.29
990792	300000	0.75	1.2	30	2927.99	5903.35	3253.18	6035.34
950211	300000	1	1.2	30	2929.66	5090.92	3259.24	5187.46
990792	300000	0.75	1	30	2910.62	5744.53	3239.71	2919.56
990792	300000	0.75	1.2	30	2927.99	5903,35	3245.56	2994.88
990792	300000	0.75	1	30	2930,45	5358,11	3257.01	2978.23

Tabla 23. Comparación CDNsim y modelo de simulación (consumo vs. número de peticiones)

Para el porcentaje de objetos distintos, se han tomado como valores 10%, 30% y 80%. A medida que dicho porcentaje aumenta, se puede apreciar un incremento en el tiempo de CPU. Este dato es en cierta medida coherente ya que la política LRU tiene una tasa de fallo (*miss ratio*) elevada. Estos fallos conducen a la generación de tráfico entre *surrogates*, lo que aumenta el consumo de CPU. Por otro lado, el consumo de memoria aumenta conforme aumenta el número de objetos. Desde el punto de vista de la pendiente de popularidad Zipf, donde se han empleado diversos valores desde 0.0.1 hasta 1, se puede observar un menor consumo de CPU conforme dicha pendiente aumenta. También puede notarse como la distribución Pareto no parece afectar al consumo de memoria ni de CPU. Como conclusión, el factor dominante que afecta (aumenta) el consumo de memoria es el número de objetos, mientras que el uso de CPU se ve afectado por las políticas de reemplazo de caché.

Desde el punto de vista de la comparación entre modelos, puede apreciarse una cierta eficiencia en memoria de CDNsim con respecto a nuestro modelo de simulación, en torno al 10- 15%. En relación al consumo de CPU, los resultados son muy similares, si bien en CDNsim es ligeramente inferior.

3.3.6.3. Comparación con CDN Simulator

El simulador CDN Simulator tiene un menor uso y documentación disponible que CDNsim, y su única referencia bibliográfica es relativamente reciente [Cec_10]. CDN Simulator está implementado como una extensión de NS-2, más concretamente sobre la versión 2.33 (la última versión disponible es la 2.35). La motivación para el desarrollo de este simulador está originada en el estudio de los mecanismos de balanceo de carga que tienen lugar en el proceso de selección de *surrogates* ante una petición por parte de un cliente dentro de una CDN. Como se comentó en la sección del estado del arte (sección 2.5.4.3), los mecanismos de encaminamiento y redirección se pueden

clasificar en tres grandes bloques: (i) basados en DNS, (ii) basados en el nivel de transporte (conmutador de nivel 4) y (iii) basados en redirecciones a nivel de aplicación (URL *rewriting* y redirección HTTP). Desde este punto de vista, CDN Simulator se basa en la redirección de nivel de aplicación (redirección HTTP), mientras que el modelo de simulación propio está basado en la redirección DNS. Esto tiene una cierta implicación en la estructura topológica de la CDN. CDN Simulator requiere que cada *surrogate* disponga de una lista de todos los demás *surrogates* que conforman la CDN, por lo que se establece una estructura plana peer-to-peer entre servidores. Nuestro modelo de simulación, al estar basado en la redirección DNS, permitiría una estructura jerárquica por el propio funcionamiento del servicio DNS. Es más, sería posible incorporar la redirección HTTP pero solamente con una lista reducida de *surrogates*, de forma que el sistema resultara escalable.

Si bien CDNsim (ver sección anterior) permite en teoría la incorporación de cualquier política de balanceo de carga, actualmente sólo tiene implementados esquemas muy sencillos de los mecanismos "servidor aleatorio" y "servidor menos cargado". CDN Simulator, por el contrario, dado que está orientado a este aspecto, incorpora un mayor número de mecanismos de balanceo de carga, como son, entre otros:

- ***LL (Least Loaded Server):*** se encamina la petición al *surrogate* menos cargado.
- ***RR (Round Robin):*** se encamina la petición a un *surrogate* de forma encadenada de entre una lista de servidores.
- ***RAND (Random):*** la petición es encaminada a un *surrogate* aleatorio.
- ***R2C (Two Random choices)***: se toman dos *surrogates*, y la petición se encamina a aquel con menor carga [Mit_01].

Desde el punto de vista de la arquitectura, CDN Simulator introduce dos novedades principales como extensión de NS-2:

- Un nuevo cliente y servidor HTTP que actúan como nodos de la CDN. Si bien el intercambio de información se realiza sobre mensajes HTTP GET y REDIRECT, el nuevo cliente es capaz de tratar estos últimos para contactar con otro servidor. Por otro lado, los servidores intercambian información del estado de carga de cada uno de ellos. La consistencia de los datos (actualizaciones de contenido) no está contemplada en CDN Simulator, y se supone que cada *surrogate* dispone de una copia coherente (actualizada) de los datos.
- Un nuevo mecanismo para introducir mecanismos de balanceo de carga en los servidores. La implementación de CDN Simulator dispone de tres métodos que separan las acciones para otorgarle una implementación modular y extensible:

- *Send_cdn_data():* extrae información de estado y la envía al resto de *surrogates*.
- *Process_cdn_data():* obtiene los datos de estado de los *surrogates* y realiza un procesado para hacer una estimación de la carga.
- *Choose_server():* selecciona el *surrogate* más adecuado para una petición.

Desde el punto de vista de la implementación, se puede comparar CDN Simulator con nuestro modelo de simulación, ya que ambos están basados en NS-2. Sin embargo, es interesante mencionar las métricas empleadas en la evaluación del rendimiento en CDN Simulator [Cec_10], que se recogen en la Tabla 24.

Métrica	Descripción
Tiempo de ida y vuelta o RTT	Incluye el tiempo de propagación por los enlaces, el tiempo de espera en los routers intermedios, el tiempo de espera en los *routers* y el tiempo de servicio.
Desviación típica del tiempo de respuesta	Proporciona información acerca de las situaciones con mejor y peor tiempo de respuesta.
Índice no balanceado (unbalanced index)	Permite evaluar la capacidad de balanceo del algoritmo en cuestión, que se obtiene, para cada tiempo muestreado, tomando la varianza de las longitudes de las colas en los surrogates y después calculando la media con el número de muestras.
Sobrecarga por redirección múltiple	Consiste en el ratio entre el número total de mensajes GET redirigidos con respecto al número total de mensajes GET generados.

Tabla 24. Métricas en CDN Simulator.

Resulta interesante el hecho de que en CDN Simulator [Cec_10] se ha modelado cada servidor como un sistema M/M/1 con tasa de servicio μ y tasa de llegada λ (proceso de Poisson), lo que enlaza con el modelo analítico presentado en la sección 3.2 de la tesis.

Existen ciertos aspectos adicionales que diferencian CDN Simulator de nuestro modelo de simulación, que se indican a continuación:

- ***Carga de los servidores***: la carga media de los servidores se evalúa en [Cec_10] como la longitud de paquetes en las colas de los servidores. En nuestro modelo de simulación, por el contrario (véase sección 3.3.5.1), se tenía en cuenta el número de páginas servidas. En nuestro modelo de simulación las páginas solicitadas por los clientes son variables en base a una determinada distribución indicada, mientras que en [Cec_10] se emplea el mismo tipo de petición a todos los *surrogates*.
- ***Algoritmo de selección***: el algoritmo de selección de *surrogate* está completamente enfocado al estudio del balanceo de carga en CDN Simulator. Si

bien se ofrecen resultados para los mecanismos de balanceo LL, R2C y RAND antes comentados, ninguno de ellos se emplea en nuestro modelo de simulación, donde se tiene en cuenta la proximidad entre *surrogates* y clientes y, una vez seleccionados los *surrogates* más próximos, se establece una asignación de pesos para garantizar un cierto balanceo de carga (véase sección 3.3.5.3).

- ***Escalabilidad***: los resultados simulados en [Cec_10] se han obtenido para un número muy reducido de clientes (10) y *surrogates* (10), por lo que resulta cuestionable su validez para una CDN real si no se realizan más simulaciones con un mayor número de clientes y servidores.

Una vez establecidas las diferencias entre ambos modelos de simulación, es importante establecer un escenario de aproximación entre ambos para poderlos comparar en la medida de lo posible. Las comparaciones que se han hecho anteriormente con CDNsim no tienen demasiado sentido establecerlas en este caso, ya que ambos modelos están implementados sobre NS-2. Es por ello por lo que tanto el uso de memoria (en comparación con los resultados de la Figura 86) como de CPU (en comparación con los resultados de la Figura 87) resultan muy similares ya que, fundamentalmente, está condicionado por el propio NS-2 y sus requerimiento de memoria y de CPU.

La comparación entre ambos modelos se va a llevar a cabo en términos de carga y tiempo de respuesta. La Figura 88 muestra la desviación típica de la carga en comparación con el número de *surrogates*. Dado que se están midiendo valores medios, los resultados entre ambos modelos podrían resultar muy similares, y es conveniente apreciar los desajustes entre *surrogates* a través de la desviación típica de la carga, medida en este caso como número de peticiones servidas (y no a través de la longitud de la cola en los servidores). Para simplificar la comparación, se va a tomar el mismo tipo de petición para todos los clientes, como se realizaba en [Cec_10]. En la Figura 88 puede observarse un escenario con cuatro simulaciones, dependiendo del algoritmo de carga empleado: (i) dos de ellas correspondientes a CDN Simulator (LL, R2C) y (ii) las otras dos correspondientes a nuestro modelo de simulación, con criterio de cercanía de *surrogate* pero con diferentes pesos (casos A y C de la Tabla 17). Las principales conclusiones son las siguientes:

- Cuando el número de servidores es reducido, el algoritmo LL reparte mejor la carga entre servidores que cualquier otro escenario. Sin embargo, esto puede conducir, en comparación, a unos tiempos de respuesta poco óptimos, como se verá en la Figura 89. El algoritmo LL no redirige necesariamente a un cliente a un *surrogate* cercano, aunque éste último esté poco cargado.

- Conforme aumenta el número de *surrogates*, la carga global del sistema se reparte mejor entre los *surrogates* en todos los casos, por lo que las diferencias entre simulaciones no son tan distantes.

- El algoritmo R2C balancea mejor la carga que nuestro modelo solo en el caso de pocos servidores y con unos pesos muy desiguales (caso A). Para pesos más equitativos (caso C) nuestro modelo tiene un comportamiento ligeramente superior. Esto es debido a que se reparte mejor la carga entre todos los servidores de forma determinista que de manera aleatoria. Conforme aumenta el número de *surrogates*, el algoritmo R2C parece el menos óptimo, aunque es el más sencillo de implementar.

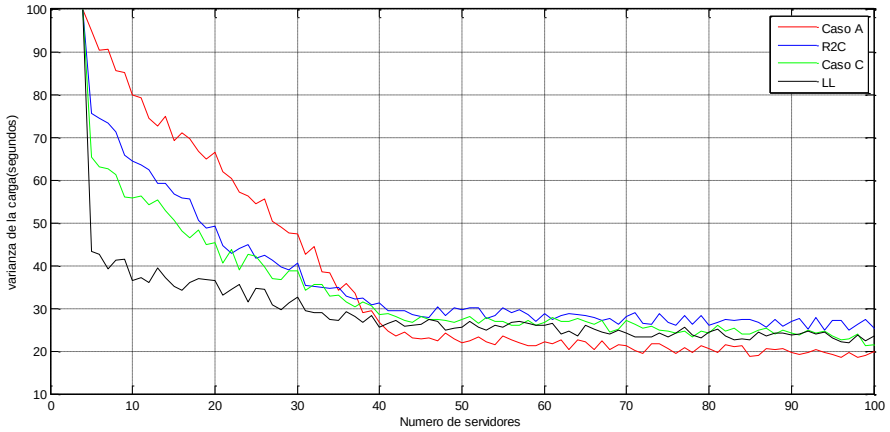


Figura 88. Varianza de la carga (500 clientes).

Una vez comparados los modelos en términos de carga, se procede a evaluarlos en términos del tiempo media de respuesta percibido por los clientes. En la Figura 89 se pueden apreciar estas cuatro simulaciones nuevamente. Se puede concluir lo siguiente:

- El tiempo medio de respuesta en nuestro modelo es claramente mejor en comparación con CDN Simulator. Esto es debido a que las peticiones se encaminan en nuestro modelo a un *surrogate* cercano, mientras que en CDN Simulator, con los algoritmos LL y R2C no se asegura este hecho, y una petición de un cliente puede ser encaminada a un *surrogate* alejado en términos de red. La diferencia en el retardo puede no parecer significativa, pero eso es debido a que los enlaces no están saturados y el contenido solicitado es web, de poco tamaño. En el caso de contenido de vídeo, con volúmenes de información muy elevados, esta diferencia de retardo es crucial, como se estudiará en la sección 4.

- El tiempo de respuesta es ligeramente mejor con el algoritmo R2C que LL. Esto debe ser debido a que, de manera aleatoria, se asigna un *surrogate* poco cargado más cercano que con el algoritmo LL, que siempre busca el servidor menos cargado. Dado que en ningún caso los servidores experimentan un *flash-crowd*, el parámetro determinante en la obtención del tiempo de respuesta es el retardo de la red. Obviamente, en una situación donde se produjera un *flash-crowd* generalizado, el algoritmo LL sería siempre capaz de encontrar un *surrogate* poco cargado (a no ser que todos estuvieran congestionados).

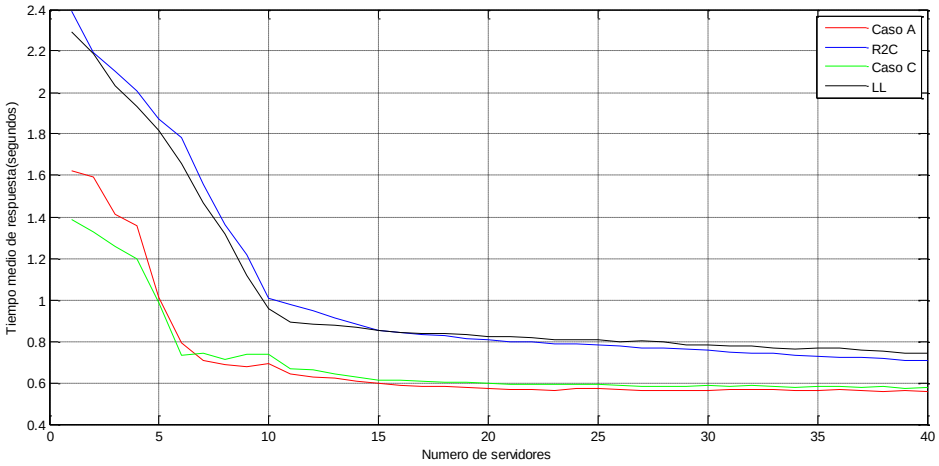


Figura 89. Tiempo medio de respuesta (comparación).

3.3.7. Conclusiones parciales del modelo de simulación

El modelo de simulación permite ampliar significativamente el modelo analítico introduciendo muchos más factores en la caracterización de una CDN, pudiendo analizar mejor su comportamiento desde diversos puntos de vista, aunque fundamentalmente los parámetros más destacados son la carga media en los servidores (en el análisis de los *surrogates*) y el tiempo medio de respuesta (en el análisis de los clientes). Adicionalmente, el modelo de simulación también permite una comparación con otros modelos de simulación existentes, aunque de forma limitada, como CDNsim y CDN Simulator.

En nuestro modelo de simulación se ha estudiado la carga media de los *surrogates*, el tiempo medio percibido por los clientes, la distribución de acceso, el porcentaje de replicación, el retardo de procesamiento, el hit ratio, el byte hit ratio y el efecto *flash-crowd*. Uno de los aspectos más destacados es la integración de replicación y caching en los *surrogates*, lo que conduce a un rendimiento mejorado en términos de tiempo de respuesta percibido, hit ratio y byte hit ratio. De una forma más concreta, se pueden derivar las siguientes conclusiones en este aspecto:

- El modelo integrado (replicación y caching) es capaz de ofrecer unos tiempos de respuesta hasta un 70% mejores que en el caso de únicamente replicación.
- Los tiempos de respuesta en el modelo integrado ofrece tiempos de respuesta similares al caso de solamente caching en condiciones normales; sin embargo, en situaciones de *flash-crowd*, los tiempos de respuesta son hasta un 15% mejores (80% replicación, 20% caching) que únicamente con caching.

- En términos de hit ratio y byte hit ratio, la mera replicación conduce a un rendimiento deficiente, aunque dependiente del tamaño del disco. El empleo de caching mejora sustancialmente el hit ratio. Con un reducido porcentaje de caching (20%) ya se obtienen valores similares de hit ratio y byte hit ratio que con porcentaje de caching del 100%.

- Los picos obtenidos en las gráficas aparecen con independencia de la política de caching empleada. Por otro lado, si bien la relación 80-20 en la replicación y caching ofrece unos valores muy convenientes en el estudio del rendimiento, debido a las múltiples configuraciones no se puede establecer un porcentaje de ambos valores que proporcione un máximo en el rendimiento en todos los escenarios.

Pese a que los resultados obtenidos en esta sección son bastante clarificadores en cuanto al funcionamiento de una CDN, es importante centrarse en esta sección en las limitaciones del modelo de simulación propuesto. En nuestro modelo de simulación el tiempo de respuesta del Redirector es inmediato, y se debería modelar mejor, o introducir un valor fijo en base a CDNs comerciales (por ejemplo entre 500 ms en la redirección DNS).

Otro aspecto susceptible de interés es que, en los resultados de nuestro modelo de simulación, a partir de 15 *surrogates* los resultados apenas varían, y esto no parece extrapolable a una CDN real, pues depende de la topología de red. Posiblemente, con un mayor número de nodos hubieran sido necesarios más *surrogates*. En esta tesis se han tomado estos valores en los nodos de la topología porque son los que típicamente se emplean en la literatura científica, y permiten una mejor comparación a posteriori.

En esta sección se ha estudiado y evaluado el modelo de simulación de CDN, donde se han introducido nuevos parámetros que permiten caracterizar mejor el comportamiento de una CDN. Si bien algunos aspectos se han simplificado (tiempo de respuesta del Redirector), otros se han implementado de manera bastante realista (políticas de caching) de tal forma que la aproximación obtenida representa un punto de partida bastante avanzado para proceder a la implementación real de una CDN. Este es el objetivo de la siguiente sección, donde se describirá el diseño de una CDN real y se estudiará su comportamiento de manera análoga a estas secciones previas. Esto permitirá validar definitivamente cada uno de los dos modelos anteriores (modelo analítico y modelo de simulación). Adicionalmente, se comparará la CDN implementada con otras implementaciones disponibles (académicas), en la medida en que se pueden comparar.

3.4. Implementación de una CDN

3.4.1. Introducción

Si bien en la sección 3.2 se describía un modelo analítico que permitía un estudio simplificado de las CDNs, en la sección 3.3 se ha mejorado el modelo a través de la simulación, que permite introducir nuevas variables que representan un funcionamiento más realista. La introducción de todas estas variables en la formulación matemática de la sección 3.2 conduciría a un escenario difícilmente analizable, y por ello se ha recurrido a la simulación mediante la herramienta NS-2. Los resultados obtenidos permiten conocer mejor el funcionamiento y las características de las CDNs, y han sido comparados con los principales simuladores disponibles (CDNsim y CDN Simulator). Una vez concluidas las fases de estudio analítico y simulación, se ha realizado el estudio en un modelo de implementación real, que se analizará en esta sección. En primer lugar, se describirá la arquitectura de implementación propuesta. Dicha arquitectura tiene un enfoque general y resulta fácilmente asimilable o comparable al modelo de arquitectura de una CDN descrito en la sección 2.5.2.1. Posteriormente, se evaluará el rendimiento de dicha CDN para entornos o aplicaciones web, analizando especialmente el comportamiento del algoritmo de redirección implementado. Este algoritmo implica disponer de un conocimiento actualizado y permanente de toda la red y aplicar un criterio de decisión u otro dependiendo de esta información continua y variable. En última instancia, se comparará este modelo con otras implementaciones reales, fundamentalmente con Globule [WWW_Glob], si bien cabe mencionar que este tipo de comparaciones siempre son limitadas porque hay diferencias significativas entre ambas implementaciones (recuérdense las comparaciones y las limitaciones en los modelos de simulación de la sección anterior).

De una manera concreta y resumida, los objetivos de esta sección son los siguientes:

- Especificar y describir una arquitectura de CDN.
- Realizar una implementación de las funcionalidades sobre dicha arquitectura.
- Diseñar un banco de pruebas que permita analizar el rendimiento de la CDN y sirva de base para ampliaciones (servicio streaming) y mejoras.
- Estudio del rendimiento (algoritmo de redirección) y comparación con otras CDNs.

Los principales resultados de este capítulo se han descrito en dos publicaciones propias [Mol_04] [Mol_06], que también aborda el servicio de streaming.

3.4.2. Descripción de la arquitectura

Siguiendo el modelo general de arquitectura propuesto en la sección 2.5.2.1, la arquitectura de alto nivel de nuestro sistema está formada fundamentalmente por cinco componentes funcionales, como se observa en la Figura 90. Estos son: *Cliente, Surrogate, Redirector, CDN Manager y Servidor Origen*. La elaboración de esta arquitectura se ha basado en [Pen_03] y [Pie_01] para servicios web, aunque posteriormente también se abordará un escenario de servicios de streaming por lo que se ha tomado adicionalmente como referencia la arquitectura propuesta en [Cra_00]. El modelo de arquitectura propuesto en esta tesis ha sido publicado en varios artículos propios [Mol_04] [Mol_06].

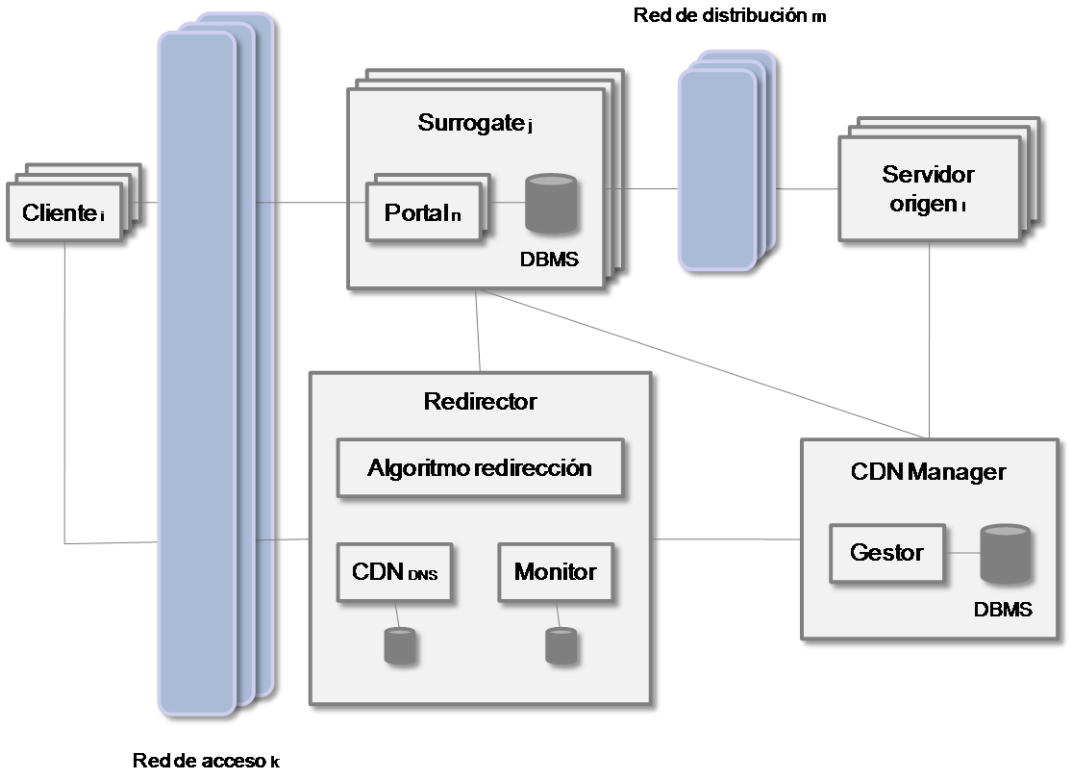


Figura 90. Arquitectura funcional de la CDN real implementada.

En los sucesivos subapartados de describirán cada uno de estos componentes, los módulos de los que constan y la forma en que interactúan unos con otros.

3.4.2.1. Cliente

El *Cliente* consiste básicamente en un navegador web, que se conecta a alguno de los portales de la CDN (ubicados en un *surrogate*) para solicitar un servicio, aplicación o

contenido. Un portal no es más que el punto de entrada a una serie de servicios ofrecidos por un proveedor de contenido a través de una CDN. La primera vez que el cliente accede a la CDN, contactará (mediante un mecanismo DNS) con el *Redirector*, responsable de redirigir al cliente al *surrogate* más adecuado. En condiciones normales, el cliente es redirigido al portal local, esto es, el más próximo a éste o, en su defecto, al que tenga una mejor conectividad y pueda proporcionarle un mejor servicio.

Una vez el cliente contacta con el portal correspondiente, el primer paso consiste en la autenticación. Se ha optado por una autenticación web básica, ampliamente soportada por la mayoría de navegadores y servidores web, dado que no es un objetivo de la presente tesis abordar este aspecto en profundidad, si bien en la sección 5.2.3 dedicada a líneas futuras se describirá el esquema SAML (*Security Assertion Markup Language*) para la autenticación CDNs federadas. Tras la autenticación, el usuario puede solicitar cualquier contenido ofrecido, típicamente recursos web. Nótese por otro lado que, desde un punto de vista general dependiendo de la configuración de la CDN, es posible que la autenticación se realice con posterioridad al acceder a ciertos contenidos restringidos, si hay contenidos de libre acceso. Esto también podría ser complementario con la existencia de perfiles de usuario con diferentes configuraciones relativas a la autorización. En cualquier caso, como se ha descrito anteriormente, el aspecto de la seguridad no se aborda en profundidad, sino que el modelo se limita a emplear técnicas de uso común ya disponibles en la mayoría de navegadores y servidores web (autenticación web y uso de HTTPS si fuera preciso).

3.4.2.2. Surrogate

La función principal de un *surrogate* es servir de punto de acceso a la CDN para los clientes, y suministrar los contenidos que éstos soliciten. Estos contenidos serán típicamente recursos web estáticos. De manera más directa, un cliente se conecta a uno o más portales web e interactúa con ellos. Ello quiere decir que un *surrogate* puede albergar uno o más portales (véase Figura 90), que representarían el contenido servido por cada uno de los diferentes proveedores de contenido que ofrecen sus servicios a través de una CDN.

Posteriormente (véase sección 4) se abordará la introducción de contenido enriquecido (multimedia). Sin embargo, cabe hacer en este punto una aclaración respecto a ambos tipos de contenido. Los contenidos estáticos (recursos web) ocupan poco espacio en el servidor, por lo que se puede asumir (por simplicidad) una replicación inicial en todos los portales, de tal forma que los usuarios acceden a éstos desde su portal local (aquel más cercano). Por el contrario, los contenidos streaming necesitan más espacio en disco y consumen más recursos, por lo que no es aconsejable (ni muchas veces posible) disponer de copias de todos los contenidos en todos los portales. Cada portal sólo dispondrá de un número limitado de videos, que almacenarán en su *caché*. Esta doble

aproximación permite hacer un análisis sencillo inicial de la CDN sin abordar en profundidad las políticas de distribución y *caching* para, posteriormente, estudiar el comportamiento de una CDN bajo el impacto de un servicio multimedia.

De una manera similar a la redirección global y local que se explicó en la sección 2.5.2.2 (sistema de encaminamiento), hay que tener en cuenta que existen dos niveles de redirección en la CDN desarrollada:

- Por un lado, la redirección DNS cuando el cliente contacta con el Redirector, en la que se centra esta sección.
- Por otro lado, la redirección por parte del *surrogate* si éste se encuentra muy cargado o no dispusiera del contenido solicitado. La última premisa (carecer del contenido solicitado) se considerará sólo en la sección 4 al introducir el servicio de streaming; en caso de recursos web se supondrá, por simplicidad, replicación total inicialmente. Así pues, el *surrogate* acepta una petición si es capaz de ofrecer un buen servicio; en caso contrario, redirige al cliente a otra ubicación (otro *surrogate*). En el caso del protocolo HTTP, este mecanismo se puede realizar a través de los códigos 30x.

La clave de este segundo nivel de redirección consiste en que cada *surrogate* debe tener conocimiento acerca del estado de los demás *surrogates* en la CDN, lo cual requiere un mecanismo de intercambio de información entre ellos. Para abordar esta situación, se ha tomado un nodo centralizado, el Redirector, quien tiene conocimiento del estado de los servidores y es el encargado de tomar las decisiones. De esta forma, si un *surrogate* se encuentra excesivamente cargado y estima que no puede ofrecer un servicio adecuado, lanza una nueva consulta al Redirector para obtener un nuevo *surrogate* al que redirigir al cliente. Todo esto se puede apreciar visualmente en la Figura 91. En ella, el *surrogate* K tiene establecidas N+1 sesiones web, donde cada cliente le va solicitando contenido de manera continuada. En un momento determinado, la carga en el *surrogate* K supera un umbral establecido, lo cual desencadena que ante una nueva petición (en la figura representado por el cliente N+1) se redirija la consulta a otro *surrogate* P, previa consulta al Redirector. Tras esta redirección, el cliente N+1 contactará con el *surrogate* P en sucesivas peticiones de su sesión web. El resto de N clientes seguirán contactando con el *surrogate* K. Téngase en cuenta que, por simplicidad, se asume que la redirección del cliente N+1 permite reducir la carga del *surrogate* K por debajo de su umbral, permaneciendo inalterables el resto de N sesiones web. Esto, en la práctica, no es una situación habitual, sino que típicamente el *surrogate* K, en condiciones de excesiva carga, redirigirá a varios clientes al *surrogate* K. También hay que considerar que, en condiciones de elevada carga, el *surrogate* K redirigirá al primer cliente que contacte con él; en la Figura 91 es el cliente N+1, pero podría haber sido perfectamente cualquier otro cliente de los N anteriores, ya que cada sesión tiene intercambios de información independientes en el tiempo.

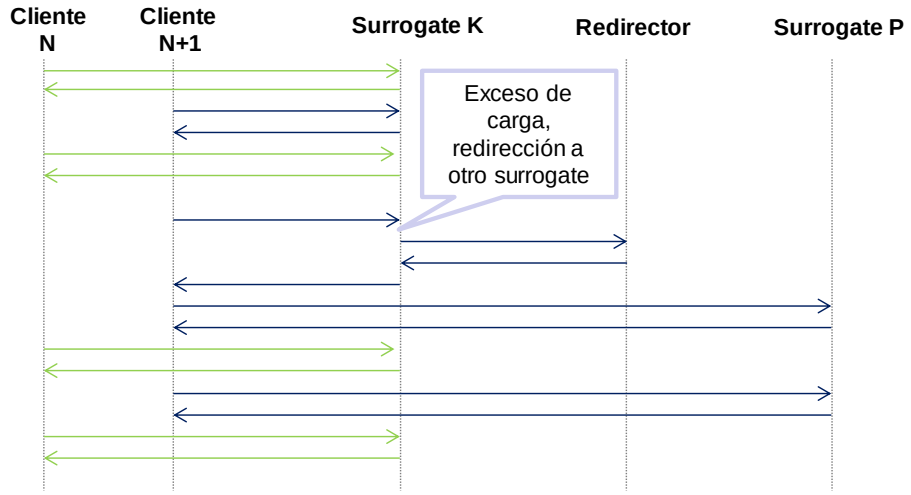


Figura 91. Redirección a nivel de surrogate ante carga extrema.

3.4.2.3. Redirector

El Redirector representa la inteligencia principal de toda la CDN propuesta, y es el encargado de determinar el *surrogate* más adecuado para un cliente en un momento determinado, proceso que se realiza a través del algoritmo de redirección, quien requiere información actualizada del estado de cada *surrogate* y de los enlaces de red. Esto se consigue a través de un proceso continuo de monitorización que obtiene y registra información de estado de servidores y de la red.

En general, en la mayoría de los casos se considerará que el *surrogate* óptimo será aquel más próximo al cliente, aunque también hay que tener en cuenta otros factores además de la distancia: los recursos disponibles en el servidor, la congestión de la red entre cliente y *surrogate*, etc. Todos estos factores se evalúan mediante un algoritmo, a partir de la información recogida por el Monitor. Si el surrogate propuesto inicialmente se encuentra muy cargado, se consulta el estado del resto de servidores (información que se obtiene periódicamente), y se toma una decisión balanceando la carga entre todos los *surrogates*, pero considerando también el factor distancia.

El Redirector recibirá peticiones del servidor DNS y de los portales. Cuando un usuario se conecta a la CDN, el servidor DNS de la CDN recibirá una petición del navegador del cliente. Entonces, el servidor DNS mandará una petición al Redirector, solicitando que se devuelva un registro correspondiente a un *surrogate* (o una lista de *surrogates*). Una vez el usuario accede al portal (*surrogate*) y desea visualizar alguno de los contenidos ofrecidos, el portal envía un mensaje al Redirector, en el que se indica el identificador de contenido solicitado y la dirección IP del cliente. Si el portal dispone del contenido solicitado y puede ofrecer un servicio con una calidad aceptable,

proporcionará el contenido al usuario. En caso contrario, bien porque el portal no disponga del contenido, bien porque esté muy cargado, el Redirector establecerá una lista de los *surrogates* candidatos, que son aquellos que disponen de una copia del contenido solicitado. A continuación, se envía un mensaje al Monitor para que recoja la información necesaria de todos estos *surrogates* y así determinar cuál de ellos es el servidor óptimo para el cliente.

El Redirector es el módulo más relevante desde el punto de vista de la arquitectura (y de la carga de desarrollo), por lo que se describirá con mayor detalle cada uno de sus componentes. Se tratará una especial atención al algoritmo de redirección, ya que es determinante en la posterior evaluación del rendimiento de la CDN.

3.4.2.3.1. Servidor DNS (CDN_{DNS})

La redirección DNS es una redirección global donde se decide el *surrogate* óptimo a contactar por un cliente en un momento determinado. Típicamente un proveedor de servicios de CDN dispone de varios nombres de dominio, que los utiliza para identificar los objetos de un proveedor de contenido que un cliente puede demandar, y aplicar el algoritmo de redirección durante el proceso de resolución. Desde este punto de vista, la resolución DNS se realiza en el "último tramo", es decir, el cliente contacta en primer lugar con su servidor DNS local, quien contacta con un servidor raíz y, en última instancia, con el servidor DNS del proveedor de servicios de CDN, que invoca el proceso de resolución. Este esquema se ilustra en la Figura 92.

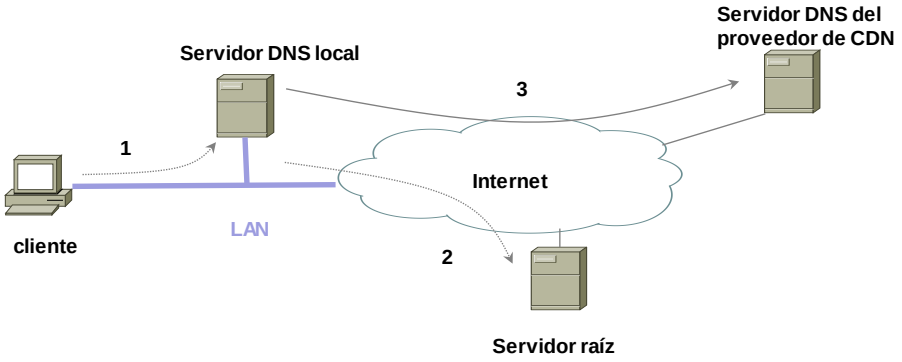


Figura 92. Resolución DNS en el último tramo.

Es posible, por otro lado, una solución de "primer tramo", donde se realiza el proceso de redirección en la red local (LAN), evitando de esta forma dar de alta un dominio nuevo en Internet y registrarlo en el servidor raíz. Esto agiliza la operación y evita costes, y será la aproximación empleada en nuestro esquema de pruebas. A continuación se describe con mayor detalle esta configuración, partiendo de la arquitectura de referencia (Figura 90).

El servidor DNS incluido en el módulo Redirector atiende las peticiones DNS dirigidas a los dominios gestionados por nuestra CDN; en caso contrario, actúa como un simple servidor DNS caché retransmitiendo la petición al servidor que se encuentra por encima de él en la jerarquía de servidores DNS (servidor DNS local). El mecanismo transparente de redirección basado en el propio DNS se puede observar en la Figura 93. En este caso, si un cliente envía una petición DNS sobre un dominio gestionado por la CDN (por ej. www.cdnupv.upv.es), el servidor CDN_{DNS} atenderá directamente la petición consultando con el módulo Redirector (activando su algoritmo de redirección). Por el contrario, si un cliente envía una petición sobre un dominio no gestionado por la CDN (por ej. www.elmundo.es) la petición será reenviada al servidor DNS local, comenzando la típica resolución iterativa del proceso DNS. Puede apreciarse que esta configuración es completamente transparente a la red y el único cambio que requiere es cambiar en el cliente la dirección IP del servidor DNS local por la del servidor CDN_{DNS}. Podría entenderse este último servidor CDN_{DNS} como un servidor DNS de "primer tramo", sin modificar en absoluto el proceso de resolución DNS.

Se podría haber modificado el Servidor DNS local, pero muchas veces esto no es posible (o deseable) a no ser que se tenga control completo del sistema. En nuestro caso, se han realizado pruebas experimentales sobre la propia red de la UPV, donde el servidor DNS local de la UPV no es manipulable. Por otro lado, la Figura 93 separa de manera lógica el servidor CDN_{DNS} y el Redirector, aunque físicamente podrían compartir LAN o incluso equipo físico. Básicamente, depende del número de subredes que conformen la CDN: típicamente habrá un servidor CDN_{DNS} por subred, mientras que sólo habrá un Redirector encargado de ejecutar el algoritmo de redirección.

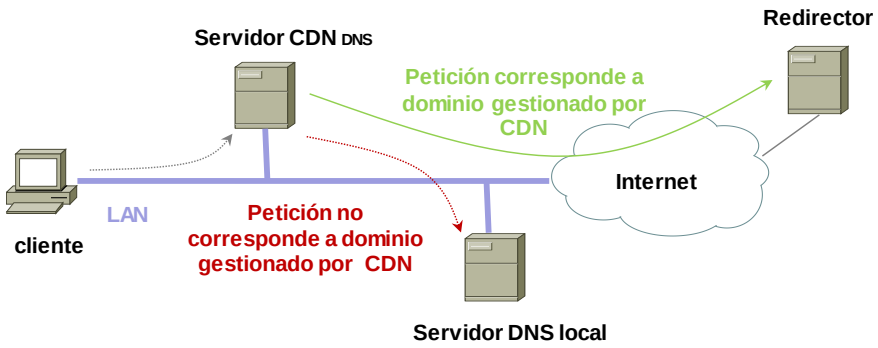


Figura 93. Redirección transparente mediante DNS.

Cuando un cliente acceda a la CDN, el servidor CDN_{DNS} debe proporcionarle la dirección IP de un portal en el que iniciar la sesión web. Para seleccionar este portal, se pueden seguir varios criterios:

- Devolver siempre la misma dirección, es decir, elegir el portal que se encuentre en la misma subred (si lo hubiera) que el servidor CDN_{DNS}.

- Devolver una dirección siguiendo una estrategia *round robin*: cada vez se retorna la dirección IP correspondiente a un *surrogate* de la CDN de forma cíclica.

- Devolver la dirección IP del *surrogate* menos cargado (criterio de carga mínima).

- Devolver la dirección IP del *surrogate* más cercano en función de la distancia cliente-servidor (criterio de distancia mínima, similar a la primera).

Las dos primeras soluciones pueden ser implementadas directamente en el propio servidor CDN_{DNS}, ya que no es necesaria ninguna información adicional sobre la red. Por el contrario, las dos últimas opciones requieren un conocimiento del estado de la red y /o los *surrogates*. En esta tesis se ha desarrollado una combinación de las dos últimas soluciones debido a su mayor flexibilidad: nótese que las dos primeras opciones no son operativas si alguno de los *surrogates* falla o no está en buenas condiciones para dar servicio. El empleo de las dos últimas opciones, por el contrario, permite realizar un balanceo de carga de forma dinámica, en función del estado de los servidores y de la red. Esta información es recogida por el módulo Monitor.

3.4.2.3.2. Monitor

La misión de este módulo es la obtención de información sobre el estado de los *surrogates* (portales) y de la red. Para ello, hace uso del protocolo SNMP (*Simple Network Management Protocol)* para interrogar a los agentes instalados en los portales y solicitar dos tipos de datos:

- ***Recursos disponibles en el surrogate***: carga de CPU, memoria RAM disponible, número de conexiones establecidas (sesiones iniciadas), espacio de almacenamiento disponible en disco.
- ***Estado de la red entre los clientes y los portales***: retardo temporal, distancia topológica (número de saltos).

Toda esta información se almacena en una base de datos, que es procesada por el Redirector para determinar, en cada momento, el *surrogate* más adecuado para cada cliente. El uso de SNMP es debido a su sencillez y uso extendido.

La información sobre los recursos de los portales se obtiene periódicamente. Cada cierto intervalo de tiempo se interroga al agente SNMP situado en los *surrogates* de la CDN para obtener los datos. En la base de datos se guarda, principalmente, los valores recogidos durante el último intervalo de medición, si bien se hace un pequeño procesado por ventanas temporales configurables (horas, días) para almacenar la evolución en el tiempo. Actualmente esto se emplea con una finalidad de seguimiento (véase el módulo

de contabilidad de la arquitectura general en la sección 2.5.2.1), aunque podría considerarse también como un una variable más (*input*) en el algoritmo de redirección.

La información sobre el estado de la red se obtiene tanto de forma periódica como aperiódica. Periódicamente el Monitor realiza mediciones del retardo entre los *surrogates*, para evaluar el estado de los enlaces entre las distintas subredes en las que se ubican los portales. En este caso se guardarán los valores de la última medición, y también se guarda un histórico para uso contable, como en el caso anterior. Además de estas mediciones periódicas, si el Redirector necesita determinar cuál es el *surrogate* adecuado para ofrecer un contenido a un cliente, se realiza una medición puntual entre dicho cliente y los servidores de la CDN, para evaluar aquel que dispone de mejor conectividad con el cliente. En la base de datos también se guarda esta información, que se puede utilizar con otros clientes que estén en la misma subred que el primero.

Es posible medir la distancia de varias formas entre clientes y *surrogates*. En esta tesis se ha empleado la distancia en saltos y la distancia temporal. La primera métrica suele ser constante en el tiempo, ya que suele haber pocos cambios en la topología de la red. La segunda métrica varía en función de la carga de la red (y del servidor) en cada instante, aunque en algunos entornos (LAN) puede resultar prácticamente constante.

El método más sencillo para obtener distancias de red es el empleo del protocolo ICMP. Mediante la utilidad *ping*, a partir del paquete ICMP de respuesta se obtiene el tiempo en recibir la respuesta, así como la distancia en saltos (campo TTL). Nótese que los *routers* de diferentes fabricantes y versiones (Cisco, HP, IBM, etc.) emplean diferentes valores máximos de TTL (63, 127, 255, etc.) en el protocolo ICMP. Cuando un *router* recibe un paquete ICMP con un TTL mayor que su valor máximo, transforma el valor, y lo devuelve en base a su valor máximo. Esto puede conducir a cierta confusión si no se emplean otros mecanismos (*traceroute*) para estimar el número de saltos.

3.4.2.3.3. Algoritmo de redirección

El Redirector emplea varios mecanismos para determinar el servidor óptimo a partir de los datos recogidos por el Monitor. Estos algoritmos emplean ciertos valores o parámetros que ofrecen una medida del estado de los servidores y de la red. A partir de estos valores se calculan una serie de coeficientes que indican la calidad del servicio que pueden ofrecer estos servidores a un cliente. Se han definido dos tipos de coeficientes: el primer tipo (coeficientes *f*) ofrece información sobre el estado de los servidores, mientras que el segundo tipo (coeficientes g) ofrece datos del estado de la red. El *surrogate* óptimo se tomará en función de los valores de estos coeficientes. A continuación se describe cómo se calculan estos coeficientes.

3.4.2.3.3.1. Coeficientes del estado de los servidores (f)

Para el cálculo de estos coeficientes se ha partido del algoritmo descrito en [Cas_99]. Este algoritmo fue diseñado para el balanceo de carga entre servidores web distribuidos; en esta tesis se han introducido ciertas modificaciones con la finalidad de hacerlo más general y por ello válido para nuestro caso de estudio (CDN). La principal diferencia estriba en la utilización de una función *x(i,j)*, que representa mediante una combinación lineal la información sobre los recursos disponibles en un servidor (carga de CPU, memoria RAM disponible, número de conexiones establecidas), en lugar de emplear únicamente la carga de CPU o la longitud de la cola de conexiones, como se usaba en [Cas_99].

Para la descripción del algoritmo que estima el estado de los servidores se va a emplear la nomenclatura descrita en la Tabla 25 :

Variable	Significado
T	intervalo entre mediciones
N_s	número de servidores de la CDN
l(i,j)	carga de CPU del servidor i-ésimo en el intervalo de medida j-ésimo
m(i,j)	porcentaje de memoria RAM ocupada en el servidor i-ésimo en el intervalo de medida j-ésimo
c(i,j)	número de conexiones TCP que se han establecido con el servidor i-ésimo durante el intervalo j-ésimo
L(i)	carga máxima deseable en el servidor i-ésimo
C(i)	número máximo de conexiones deseable en el servidor i-ésimo
x(i,j)	Combinación lineal de m(i,j), l(i,j) y c(i,j), utilizado como parámetro de entrada en nuestro algoritmo. El valor de esta función es inversamente proporcional a los recursos disponibles en el servidor. Un valor de x(i,j) = 1 indica que todos los recursos del servidor están siendo utilizados.
f(i,j)	Valor de salida del algoritmo, proporcional al número de peticiones de nuevas sesiones que se envían al servidor i-ésimo durante el intervalo de medida j-ésimo. Indica la probabilidad con la que una petición será enviada al servidor, y se cumple que $\sum_i f(i, j+1) = 1$
f_{min}	valor mínimo de salida para cualquier f(i,j)
f_{max}	valor máximo de salida para cualquier f(i,j)

Tabla 25. Nomenclatura para el cálculo de los coeficientes f.

El algoritmo se aplica en ventanas temporales de duración T segundos. En el intervalo de medida j-ésimo, se recoge la información sobre la carga de CPU del i-ésimo *surrogate* (*l(i,j)*), su memoria usada (*m(i,j)*) y el número de conexiones establecidas (*c(i,j)*), de manera normalizada. A partir de estos datos se calcula la función *x(i,j)* como:

$$x(i, j) = \alpha_1 \cdot m(i, j) + \alpha_2 \cdot \frac{l(i, j)}{L(i)} + \alpha_3 \cdot \frac{c(i, j)}{C(i)} \qquad \text{(E 3.40)}$$

donde $0 < \alpha_1 , \alpha_2, \alpha_3 < 1$, $\alpha_1 + \alpha_2 + \alpha_3 = 1$, $0 < x(i,j) < 1$

La función *x(i,j)* indica un nivel compuesto de carga que sirve para determinar si en el siguiente intervalo debe (puede) aceptar peticiones de clientes. Es decir, si está muy cargado, parece lógico que no acepte peticiones (o muy pocas) de clientes, y estas últimas sean encaminadas a otros *surrogates*. Por otro lado, si el valor de *x(i,j)* es reducido (poca carga) para el *surrogate* i-ésimo, parece lógico que en el siguiente intervalo *(j+1)* acepte más peticiones que otros *surrogates* que están poco cargados. De esta forma, la fracción de nuevas sesiones que se encaminarán al servidor i-ésimo en el intervalo *(j + 1)* se establece mediante la siguiente función:

$$f(i, j+1) = k[x(i, j)] \cdot f(i, j) \qquad 0 < i < N_s \qquad \text{(E 3.41)}$$

siendo **k**[] una función que depende de *x(i,j)*, y se escoge de tal forma que su valor es mayor que 1 para los servidores poco cargados (se enviará más carga a estos servidores), y menor que 1 para los servidores cuya carga supere un determinado umbral (la carga disminuirá en estos servidores). En la Figura 94 se muestra una de las posibles funciones que se pueden utilizar y que se han escogido debido a su simplicidad.

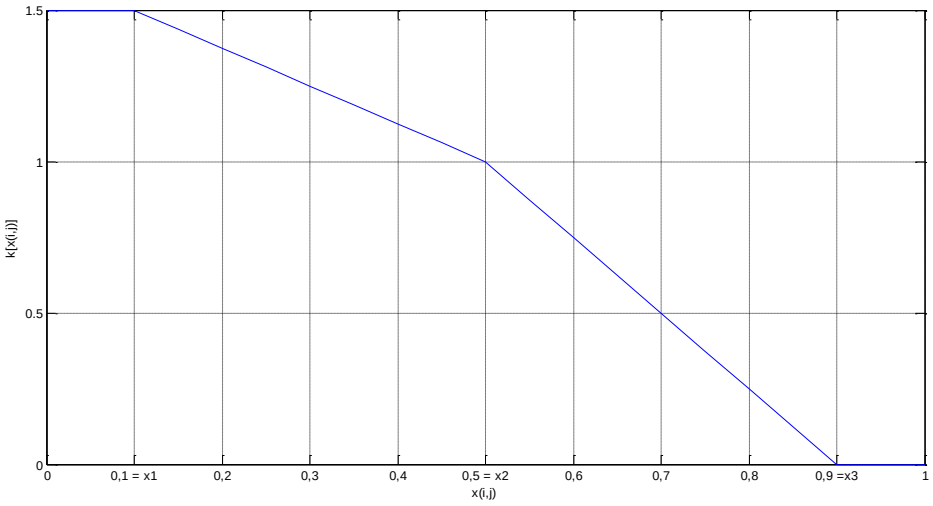


Figura 94. Función k[x(i,j)].

A partir de la función *k[x(i,j)]*, se establecen varios niveles de funcionamiento:

- ***x(i,j) = 0*** . En este caso no se ha recibido información sobre el servidor debido a algún fallo en las comunicaciones o en el propio servidor, por lo que no se puede asegurar un buen servicio a los usuarios. Por fiabilidad, se toma *f(i,j+1) =0* hasta que se vuelva a recibir información del servidor y, por tanto, no se considerará por el Redirector a la hora de elegir un *surrogate*. Una vez el servidor se recupere, se dará a *f(i,j)* un valor inicial f_{min}.

- ***0 < x(i,j) < x1***. En este caso el servidor i-ésimo dispone de un índice carga combinado reducido (inferior al 10%), por lo que es susceptible de recibir un mayor número de peticiones en el siguiente intervalo. En este caso, se ha fijado un valor de k=1.5, lo que equivale a un incremento del 50% de peticiones entrantes con respecto a su estado anterior.

- ***x1< x(i,j) <x2.*** En este caso el servidor i-ésimo también dispone de un índice de carga reducido, por lo que aceptará más peticiones entrantes en el siguiente intervalo. Sin embargo, su carga no es tan reducida como en el primer caso, por lo que su incremento de peticiones es dependiente de su índice de carga, de tal forma que, en el límite (*x(i,j) = x2*) el *surrogate* i-ésimo aceptará el mismo número de peticiones entrantes que tenía establecido. Expresado en otros términos, si el índice de carga combinado es del 50%, el servidor siempre recibirá el mismo número de peticiones en cada intervalo de tiempo. Esto se trata de una situación ideal de diseño donde se fija un estado estable y permanente de los *surrogates* con un índice de carga del 50%. Nótese que este valor de x2 es un parámetro de diseño que permite garantizar un buen servicio. En esta tesis se ha tomado x2=0.5.

- ***x2< x(i,j) <x3.*** En este caso el servidor i-ésimo dispone de un índice de carga combinado considerable, por lo que aceptará menos peticiones entrantes en el siguiente intervalo. Conforme aumenta dicho índice de carga combinado, el número de peticiones entrantes en el siguiente intervalo disminuye, de tal forma que, en el límite (*x(i,j) = x3*) el *surrogate* i-ésimo no acepta nuevas peticiones.

- ***x(i,j) >x3.***En este caso el servidor i-ésimo dispone de un índice de carga considerable, y no aceptará nuevas peticiones en el siguiente intervalo de tiempo. Esto le permitirá dar servicio a las sesiones pendientes e ir reduciendo poco a poco su carga, hasta que su índice de carga combinado sea inferior a x3 y pueda volver a recibir peticiones entrantes.

La función *k[x(i,j)]* se aplica para cada uno de los *surrogates*, por lo que se trata de una estimación tentativa por parte del Redirector del porcentaje de nuevas peticiones que aceptará cada uno de los *surrogates*. Sin embargo, esta decisión no se puede tomar de forma independiente, sino que hay que considerar al resto de servidores de forma que se establezca la condición de normalización:

$$\sum_i f(i, j+1) = 1 \qquad \text{(E 3.42)}$$

Además de la condición de normalización, también hay que garantizar en las consideraciones globales del algoritmo (es decir, teniendo en cuenta todos los *surrogates*) otros aspectos importantes, como son:

- $f_{min} < f(i,j+1) < f_{max}$. Es importante comprobar que ningún valor de *f(i,j+1)* sobrepasa los límites máximos y mínimos establecidos. En caso contrario, se podría llegar a una situación donde el algoritmo diera cada vez más peso a los servidores con mayor *f(i,j)* y menos a los que tienen menor *f(i,j)*, lo que conduciría precisamente al efecto contrario al perseguido, que no es otro que un correcto balanceo de carga.

- ***f(i,j+1) ≈ f(l,j+1) si f(i,j) ≈ f(l,j).*** Cuando dos servidores presenten índices de carga combinada similares, los valores de *f(i,j+1)* deben ser similares, por los que las nuevas conexiones serán dirigidas con similar probabilidad a ambos servidores.

3.4.2.3.3.2. Coeficientes del estado de la red (g)

Para el cálculo de los coeficientes del estado de la red, a diferencia de los coeficientes *f*, no se usa un algoritmo con memoria, es decir, no se tienen en cuenta los valores calculados anteriormente. Esto es debido a que los parámetros de red son variables con el tiempo, sobre todo en Internet, y el estado de un enlace en un momento determinado no condiciona su estado en un momento posterior. El cálculo se hace a partir de la información sobre el estado de la red en un instante concreto, que el Monitor recoge y almacena en su base de datos. Esta información puede medirse respecto a un cliente en concreto (en el caso de que conozcamos su dirección IP), o puede medirse también respecto al servidor DNS local de dicho cliente. En este último caso, se supone que la distancia del servidor DNS al cliente es pequeña, ya que en otro caso la información no sería fiable. Como se analizará posteriormente en la sección 3.4.4, en el entorno de pruebas de esta tesis se asume que esta distancia es mínima, ya que en todas las subredes a considerar se dispone de un servidor DNS.

Para la descripción del algoritmo que estima el estado de los servidores se emplea la nomenclatura descrita en la Tabla 26.

Variable	Significado
T	intervalo entre mediciones
N_s	número de servidores de la CDN
d(i,j)	distancia en saltos entre el servidor i-ésimo, y el cliente j-ésimo
p(i,j)	distancia temporal (ping) entre el servidor i-ésimo, y el cliente j-ésimo
D	Máxima distancia en salto deseable entre un servidor y cualquier cliente. El no sobrepasar esta distancia garantizará un funcionamiento óptimo de la CDN
P	Máxima distancia temporal deseable entre cualquier servidor y cualquier cliente para garantizar que se da un buen servicio.

y(i,j)	Combinación lineal de d(i,j) y p(i,j), que se utiliza como parámetro de entrada en nuestro algoritmo. El valor de esta función será directamente proporcional a la distancia entre el servidor i-ésimo y el cliente (o servidor DNS) j-ésimo. Por tanto, los mejores servidores para un cliente serán aquellos para los que el valor de la función y(i,j) sea mínimo.
g(i,j)	Valor de salida del algoritmo, proporcional al número de peticiones de nuevas sesiones que se envían al servidor i-ésimo durante el intervalo de medida j-ésimo. Indica la probabilidad con la que una petición será enviada al servidor, y se cumple que $\sum_i f(i, j+1) = 1$
g_{min}	valor mínimo de salida para cualquier g(i,j)
g_{max}	valor máximo de salida para cualquier g(i,j)

Tabla 26. Nomenclatura para el cálculo de los coeficientes g.

El algoritmo se aplica en ventanas temporales de duración T segundos. En el intervalo de medida j-ésimo, se recoge la información sobre la distancia en saltos entre clientes y *surrogates* (*d(i,j)*), así como su distancia temporal (*p(i,j)*). A partir de estos valores se calcula la función *y(i,j)* como:

$$y(i, j) = \gamma_1 \cdot \frac{d(i, j)}{D} + \gamma_2 \cdot \frac{p(i, j)}{P} \qquad \text{(E 3.43)}$$

donde $0 < \gamma_1, \gamma_2 < 1$, $\gamma_1 + \gamma_2 = 1$, $0 < y(i,j) < 1$

A partir de *y(i,j)* se calcula *g(i,j)* para asignar adecuadamente los pesos:

$$g(i, j+1) = a(i, j) \cdot \frac{\sum_i y(i, j)}{y(i, j)} \qquad \text{(E 3.44)}$$

donde $0 < a(i,j) < 1$

Sobre esta última expresión cabe mencionar ciertos aspectos significativos:

- ***g(i,j+1) = 0 para y(i,j) =0***. Si *y(i,j)* es cero se produce una indeterminación en la expresión (E 3.44) que hay que resolver. Una distancia *y(i,j)* nula no tiene sentido, ya que indicaría que el servidor está en el propio cliente, así que sólo puede producirse como consecuencia de algún tipo de error en la medición. En esta situación, se toma *g(i,j+1)* =0 para evitar que se elija este servidor en este intervalo de tiempo.

- ***g_{min}< g(i,j+1) <g_{max}*** . Es importante comprobar que ningún valor de g*(i,j+1)* sobrepasa los límites máximos y mínimos establecidos, similar al caso de los coeficientes f. Para ello, los valores que excedan (por arriba o por debajo) se truncan a g_{min} y g_{max}. De esta forma, el *surrogate* más cercano toma un valor de g_{max} (si hay truncamiento) o cercano, mientras que el resto de *surrogates* tendrán un valor inferior.

- ***Criterio de surrogates***. La variable a(i,j) varía en función del servidor y el cliente, de forma que se otorga más peso a ciertos servidores siguiendo algún criterio concreto (conocimiento de servidores cercanos, conocimiento de servidores que disponen del contenido, etc.).

Nótese que en la expresión (E.43) se medían distancias entre clientes y *surrogates*. Este modo de funcionamiento no es viable porque el número de clientes puede ser muy elevado (Internet), con lo que tendríamos un sistema poco escalable. Adicionalmente, los clientes pueden no estar disponibles (conectados) de forma permanente, por lo que no sería posible obtener datos de distancia de forma periódica. Es por ello por lo que se debe recurrir a agrupar clientes en los denominados clústeres de clientes, y tratar de obtener una medida entre este clúster de clientes y los *surrogates*. Esta es precisamente la estrategia abordada, de tal forma que los clústeres de clientes quedan asociados a su servidor DNS local. Nuestra CDN propuesta sólo deberá conocer las distancias entre los servidores DNS locales y los *surrogates*, con lo que se reduce el almacenamiento en memoria requerido por el Redirector. Así pues, la expresión (E.43) es de aplicación en dos tipos de escenarios:

- Para calcular las distancias entre servidores DNS locales y nuestros *surrogates*. Este cálculo se puede hacer de manera periódica, de tal forma que cuando se ejecuta el algoritmo de redirección ya se dispone de esta información y no hay retardo adicional.

- Para calcular las distancias entre un cliente en concreto y los *surrogates*. Este cálculo se realiza de forma asíncrona, y en este caso el algoritmo de redirección debe esperar a que esta información esté disponible (se envía un mensaje al módulo Monitor para que realice las medidas), produciéndose un retardo adicional.

Como se mostrará en la sección 3.4.4, en la CDN implementada se dispone de un servidor DNS local por subred, por lo que lo clientes de cada subred quedan parcialmente determinados por su correspondiente servidor DNS local. No obstante, si el módulo Redirector lo requiere, puede invocar al módulo Monitor para que obtenga la distancia entre un cliente concreto y un *surrogate*. Esta situación podría pasar cuando el cliente en cuestión no está ubicado en una subred asociada a un servidor DNS que se esté monitorizando, o bien cuando el supuesto *surrogate* más cercano al cliente no está funcionando adecuadamente y es necesario redirigir al cliente a otro *surrogate*. También es posible emplear esta medida explícita como mecanismo de fiabilidad (a posteriori) para verificar que el *surrogate* seleccionado es una buena opción. Esto sería una medida de rendimiento interna que se describió en la sección 2.5.4.4.

3.4.2.3.3.3. Coeficientes conjuntos (h)

En las secciones previas se han descrito dos tipos de mecanismos de selección de *surrogates* o portales, bien teniendo en cuenta la carga de los servidores (coeficientes *f*), o bien considerando la distancia entre el servidor y el cliente (coeficientes *g*). Se ha estudiado una tercera alternativa, que es una combinación lineal ambas. Definimos los coeficientes *h* mediante la siguiente expresión:

$$h(i,j) = \beta_1 \cdot f(i,j) + \beta_2 \cdot g(i,j) \qquad \text{(E 3.45)}$$

donde $0 \leq \beta_1, \beta_2 \leq 1$, $\beta_1 + \beta_2 = 1$, $0 < h(i,j) < 1$

La ecuación (E.45) representa la expresión más general del algoritmo de redirección, y permite aplicar de manera independiente y aislada el criterio de servidor menos cargado ($\beta_1 = 1$, $\beta_2 = 0$) o el criterio de servidor más cercano ($\beta_1 = 0$, $\beta_2 = 1$). Cualesquiera otros valores de β_1 y β_2, simplemente permitirán ponderar un aspecto sobre el otro.

3.4.2.4. CDN Manager

Si bien desde el punto de vista de la investigación académica el módulo Redirector (con el algoritmo de redirección) es el aspecto más importante, desde el punto de vista de la operación el módulo CDN Manager es el más importante, ya que se encarga de administrar los servidores origen (si es posible) y *surrogates*, los contenidos y los usuarios de la CDN. Sin ánimo de entrar en demasiado detalle en este aspecto, las principales funcionalidades se recogen en la Tabla 27, Tabla 28, Tabla 29 y Tabla 30.

Operación de CDNs
Creación, registro y eliminación de CDNs
Parametrización de la CDN. Configuración de variables de funcionamiento.
Monitorización de la CDN y evolución del rendimiento

Tabla 27. Funcionalidades en la gestión de la CDN.

Operación de Servidores
Creación, registro y eliminación de servidores origen y surrogates
Parametrización del servidor origen y surrogate. Configuración de variables de funcionamiento.
Monitorización del surrogate y evolución del rendimiento

Tabla 28. Funcionalidades en la gestión de servidor origen y surrogate.

Operación de Contenidos
Creación, registro y eliminación de contenidos
Adaptación del formato de los contenidos. Esto puede ser debido a varios motivos: (i) adaptación de ancho de banda del usuario, (ii) adaptación del tipo de terminal y (iii) adaptación de streaming. El tercer caso no se considerará en este capítulo, ya que se analizará en la sección 4.3, pero es importante destacar que algunos servidores de streaming requieren marcas temporales en los contenidos, que habría que generar si no las incorporan de origen).
Políticas de gestión y distribución. Típicamente los contenidos se clasifican en varios grupos, el más común es en base a la demanda, y para cada uno de estos grupos se define una política de gestión configurable por varios parámetros (tiempo máximo de permanencia en cache, número máximo y mínimo de copias distribuidas, etc.).
Gestión de caché. Se controla el tiempo que permanecen en caché los contenidos dependiendo de las políticas de gestión.
Transferencias entre portales. Cuando se requiere transferir una nueva copia de un contenido a alguno de los portales, el CDN Manager decide el servidor (servidor origen o surrogate) que debe transferir el contenido. Este aspecto será especialmente relevante en la CDN funcionando como distribuidor de contenido multimedia; con contenido web y replicación no resulta relevante.

Tabla 29. Funcionalidades en la gestión de contenidos.

Operación de Usuarios
Creación, registro y eliminación de usuarios
Autenticación. Para acceder a los contenidos de la CDN, el usuario debe estar registrado en la base de datos de usuarios. La autenticación sólo se exige al iniciar sesión.
Autorización. Se comprueba si el usuario puede acceder o no a un contenido en particular. Esto permite generar perfiles de usuarios y perfiles contenidos, y establecer correspondencias entre ambas mediante mecanismos de autorización.
Facturación: Contabiliza el acceso a cada contenido por parte de un usuario. Esto permite generar estadísticas e información de realimentación para la gestión de contenidos, identificando los contenidos más demandados como entrada (input) para las políticas de gestión y distribución.

Tabla 30. Funcionalidades en la gestión de usuarios.

3.4.2.5. Servidor origen

El servidor origen representa el almacenamiento original de todos los contenidos ofrecidos por la CDN. Si se desea introducir un nuevo contenido en la CDN, se ubica siempre en este servidor. Las políticas de distribución y gestión serán las encargadas de realizar sucesivas copias de dicho contenido entre todos los *surrogates* de la CDN, dependiendo fundamentalmente de la demanda esperada y generada. El servidor origen sólo ofrecerá servicio directo a un *surrogate* si ninguno de los otros *surrogates* puede ofrecer el contenido al cliente.

3.4.3. Tecnologías de implementación utilizadas

La implementación del sistema propuesto se ha llevado a cabo en el lenguaje de programación Java por ser un lenguaje de programación sencillo, potente y reutilizable, con gran utilización en entornos web (servidores de aplicaciones). Los módulos correspondientes a la arquitectura descrita anteriormente (sección 3.4.2) han sido desarrollados como servicios web (esquema SOA) y como aplicaciones independientes. Es por ello por lo que no se describirá el código de cada una de las clases desarrolladas (mayor de 50) para centrarnos en la producción científica (a diferencia del modelo de simulación). Como en toda aplicación web, se puede distinguir la capa de presentación orientada a la interactividad con el usuario (portal), y la lógica de negocio que alberga los servicios web orientados al proceso y la gestión (Redirector, Monitor).

Las aplicaciones web en Java desarrolladas han sido desplegadas en un servidor de aplicaciones, como Tomcat [WWW_Tom], Glassfish [WWW_Glas], Jboss [WWW_Jbos] o Jonas [WWW_Jonas]. Por simplicidad y sencillez, se empleó Tomcat (v7.x) aunque, propiamente dicho, es un contenedor de servlets (soporte para servlets y JSPs), sin soportar completamente la especificación J2EE (*Java 2 Enterprise Edition*). Esto lo hace más sencillo y ligero desde el punto de vista de carga de los *surrogates*. Normalmente, en los contenedores de servlets como Tomcat se despliegan aplicaciones de tipo WAR (*Web Archive*), mientras que en los servidores de aplicaciones soportan aplicaciones EAR (*Enterprise Archive*).

Pese a que Tomcat no soporta toda la especificación J2EE, sí soporta un gran número de librerías Java de forma nativa, así como otras que se pueden agregar. Por ejemplo, los módulos de Redirector (servidor DNS y Monitor) y CDN Manager requieren una interfaz de comunicación que les permita intercambiar mensajes. Esta interfaz se ha desarrollado mediante el API SAAJ (*SOAP with Attachments API for Java*) que aporta un conjunto de funciones que permiten intercambiar mensajes SOAP entre aplicaciones Java.

Otra librería empleada, sobre todo para la parte de monitorización, es JDMK (*Java Dynamic Management Kit*) basado en JMX (*Java Management Extensions*). En realidad es un API (*Application Programming Interface* que proporciona un conjunto de herramientas de desarrollo para diseñar e implementar aplicaciones de gestión, con soporte para el protocolo SNMP (*Simple Network Management Protocol* que es el estándar más utilizado para la gestión de redes y dispositivos.

Para el desarrollo de las bases de datos de la CDN se ha empleado MySQL (v5.x) [WWW_MyS] como sistema gestor de bases de datos, al tener una arquitectura compacta y estable. Para acceder a la base de datos desde aplicaciones Java se requiere un conector JDBC (*Java DataBase Connectivity*). MySQL está disponible para sistemas Linux y Windows, por lo que, junto con Java, se conserva la característica multiplataforma del desarrollo.

3.4.4. Escenarios de evaluación. Topologías de red y parámetros

El sistema que se ha desarrollado en esta tesis se ha desplegado en la red corporativa de la UPV, cuya estructura simplificada se representa en la Figura 95. En realidad existen más puntos de interconexión menores que se puede consultar en [WWW_UPV], pero para las pruebas realizadas en esta tesis son suficientes tres sedes. Como se puede observar en la Figura 95, esta red se compone de tres grandes subredes. La red del campus de Valencia es la más grande y la que tiene un mayor número de usuarios, mientras que las otras dos (Alcoy y Gandía) disponen de un menor número de usuarios. En esta tesis se propone desplegar la CDN implementada con un *surrogate* en cada una de las tres sedes. Dado que la sede de Valencia cuenta con un mayor número de usuarios, es conveniente desplegar también ahí el servidor origen, ya que se reducirá la latencia para los clientes en la sede de Valencia si tienen que contactar con el servidor origen, en proporción con los clientes de Gandía y Alcoy. Razonando de manera análoga, los módulos CDN Manager y Redirector también se han desplegado en la sede de Valencia, ya que se reduce el retardo general si la sede de Valencia es la que genera más tráfico en proporción con la sede Gandía y Alcoy. Por otro lado, en cada una de las sedes se encuentra un cliente que, en realidad, emula el comportamiento de un clúster de clientes, por lo que es capaz de generar diferentes patrones de tráfico para la evaluación de la CDN.

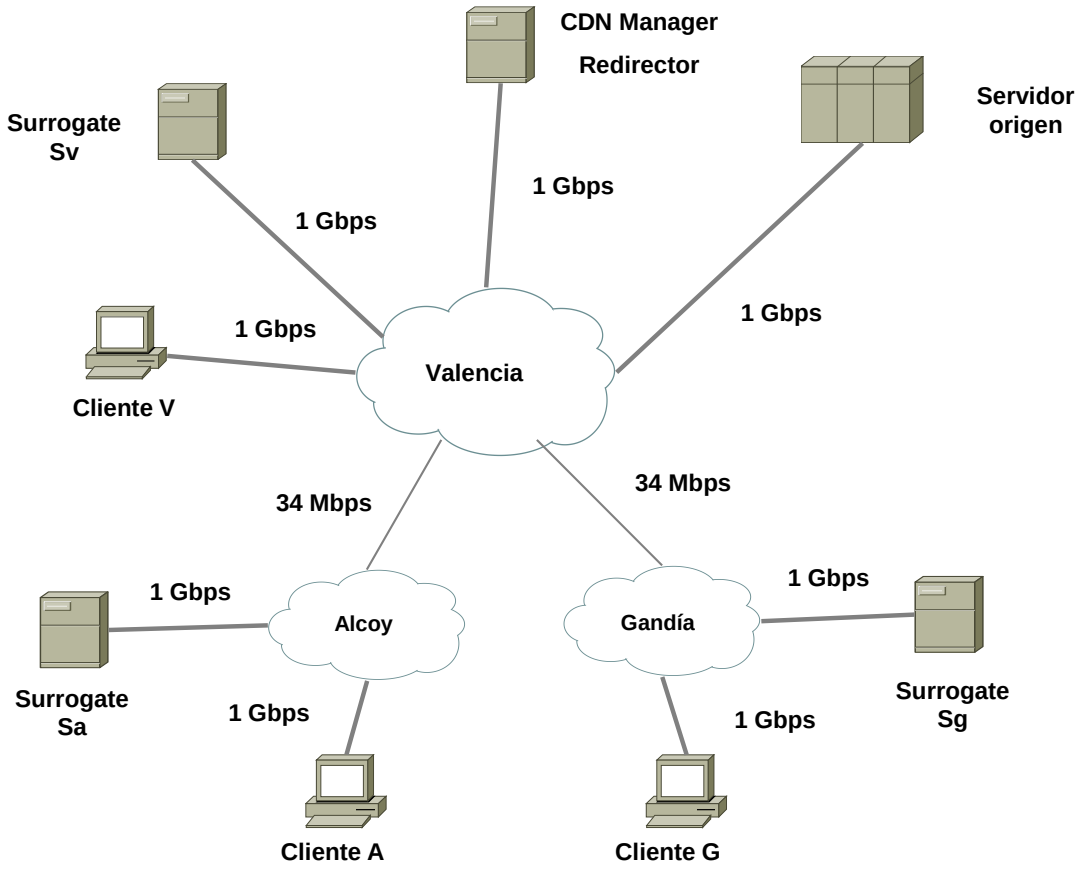


Figura 95. Despliegue de CDN en la UPV.

Antes del despliegue real, y en una primera fase inicial de pruebas, se pretende probar que las funciones básicas del sistema han sido desarrolladas correctamente y se cumple con el objetivo de funcionamiento deseado o, al menos, esperado. Por esta razón, se ha creado en una maqueta que permita testear la CDN, para portarla posteriormente a la red corporativa de la UPV.

La maqueta simplificada es funcionalmente equivalente, y consta de tres redes, separadas entre sí mediante dos *routers*. En los puertos de los *routers* se conecta un conmutador (*switch*) donde se conectan los equipos que pertenecen a cada una de las subredes. El esquema se muestra en la Figura 96. El servidor origen y el módulo CDN Manager se han incluido dentro de un mismo equipo físico por simplicidad. Esto no afectará en el rendimiento del sistema ya que el servidor origen apenas servirá contenido en las pruebas que se realizarán, y no sobrecargará el proceso del módulo CDN Manager y Redirector.

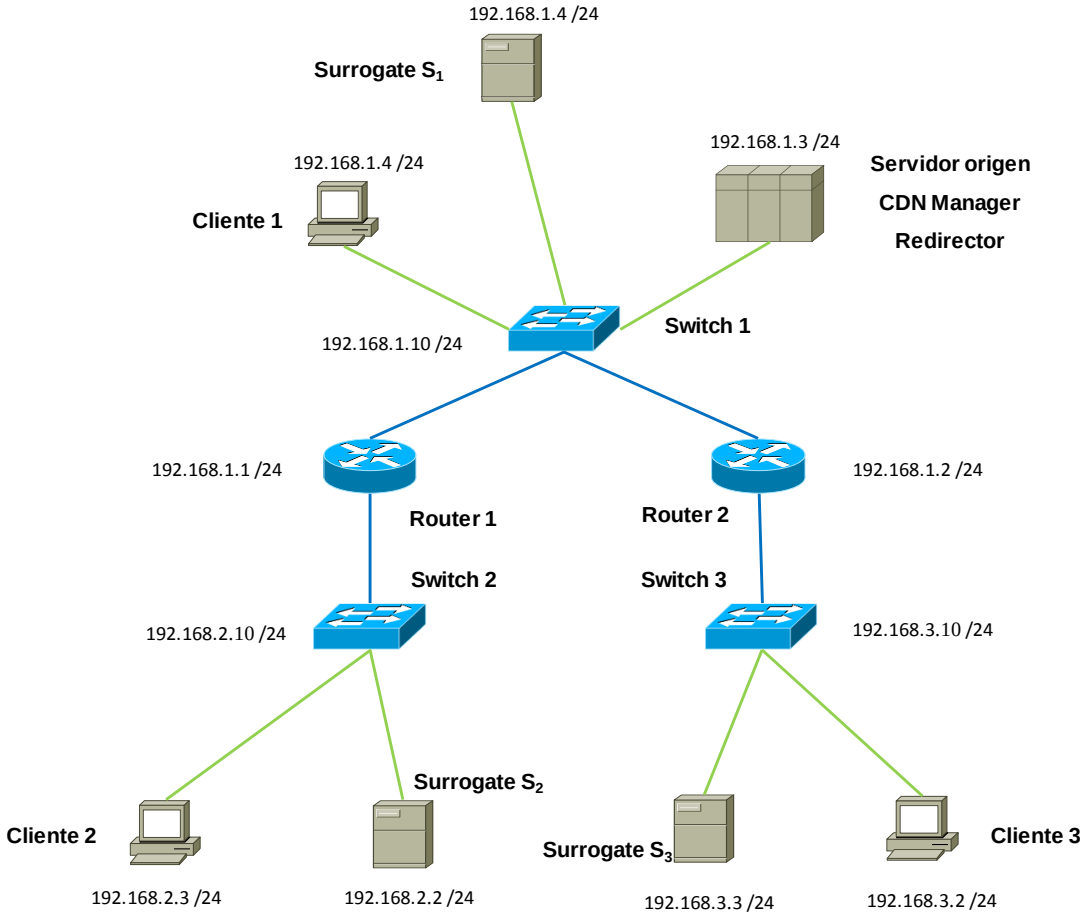


Figura 96. Maqueta de pruebas simplificada.

Las características de los equipos y dispositivos que conforman la arquitectura de la CDN se recogen en la Tabla 31. Cabe hacer una serie de puntualizaciones:

- Los routers son de gama baja (Ethernet a 10Mbps) para reducir al máximo la velocidad y emular en la medida de lo posible el tráfico WAN.

- Los *swicthes* son de gama media-baja pero permiten una elevada capacidad de configuración. Por ejemplo, es posible monitorizar el tráfico en cada uno de los puertos. Incluso es posible redirigir el tráfico entrante/saliente en uno de los puertos y enviarlo por otro.

Dispositivo	Descripción		
CDN Manager	**CPU**: P-IV 1.8 GHz	**RAM**: 4 GB	**HD**: 500 GB
	S. O.: W2000 Server SP4	**IP**: 192.168.1.3 /24	
	Programas instalados: MySQL 5.X; JDK 1.7.X, Servidor DNS, Redirector, MonitorSNMP, CDNControl, CDNManager		
Surrogate 1	**CPU**: P-IV 1.5 GHz	**RAM**: 4 GB	**HD**: 500 GB
	S. O.: Win XP SP3	**IP**: 192.168.1.4 /24	
	Programas instalados: MySQL 5.X; JDK 1.7.X, Tomcat 6, IIS, Agente SNMP java, Darwin, Aplicación Portal		
Surrogate 2	**CPU**: P-IV 1.8 GHz	**RAM**: 4 GB	**HD**: 500 GB
	S. O.: Win XP SP3	**IP**: 192.168.2.2 /24	
	Programas instalados: MySQL 5.X; JDK 1.7.X, Tomcat 6, IIS, Agente SNMP java, Darwin, Aplicación Portal		
Surrogate 3	**CPU**: P-IV 1.8 GHz	**RAM**: 4 GB	**HD**: 500 GB
	S. O.: Win XP SP3	**IP**: 192.168.3.2 /24	
	Programas instalados: MySQL 5.X; JDK 1.7.X, Tomcat 6, IIS, Agente SNMP java, Darwin, Aplicación Portal		
Cliente 1	**CPU**: P-III 500 MHz	**RAM**: 2 GB	**HD**: 500 GB
	S. O.: Win XP SP3	**IP**: 192.168.1.5 /24	
	Programas instalados: JDK 1.4.1_01; QuickTime 6 (QT for Java); TCPView, Jmeter,		
Cliente 2	**CPU**: P-III 500 MHz	**RAM**: 2 GB	**HD**: 500 GB
	S. O.: Win XP SP3	**IP**: 192.168.2.3 /24	
	Programas instalados: JDK 1.7.X; QuickTime 6 (QT for Java); TCPView, Jmeter		
Cliente 3	**CPU**: P-III 500 MHz	**RAM**: 2 GB	**HD**: 500 GB
	S. O.: Win XP SP3	**IP**: 192.168.3.3 /24	

	Programas instalados: JDK 1.7.X, QuickTime 6 (QT for Java); TCPView, Jmeter
Router 1	Modelo: 1605-R (Serie 1600), 2 Ethernet 10 Mbps, 1 conexión serie IPs (subredes): 192.168.1.1 /24 ; 192.168.2.1 /24
Router 2	Modelo: 1605-R (Serie 1600), 2 Ethernet 10 Mbps, 1 conexión serie IPs (subredes): 192.168.1.2 /24 ; 192.168.3.1 /24
Switch 1	Cisco Catalyst 2950 24 puertos, IP = 192.168.1.10
Switch 2	Cisco Catalyst 2950 24 puertos; IP = 192.168.2.10
Switch 3	Cisco Catalyst 2950 24 puertos; IP = 192.168.3.10

Tabla 31. Equipamiento de la maqueta de pruebas simplificada.

3.4.5. Resultados y análisis de la evaluación

A continuación se describirán las diferentes pruebas que se han llevado a cabo. Para ello, se han ido modificando uno o varios parámetros en la configuración y se ha evaluado el rendimiento. Nótese, por un lado, que en este caso el número de *surrogates* está acotado, y no podemos introducirlos tan fácilmente como en los modelos analíticos y de simulación. El número de clientes, por otro lado, sí que puede ser simulado con programas capaces de generar el tráfico equivalente. En la maqueta descrita en la Figura 96 los tres clientes incluyen el programa JMeter [WWW_JMe] capaces de generar varios tipos de carga de tráfico de forma multihilo emulando un número configurable de clientes.

En este entorno de pruebas el mayor interés radica en el estudio y evaluación del algoritmo de redirección. Es importante comprobar que el algoritmo funciona correctamente ante varias configuraciones del sistema (número de clientes, patrón de acceso, indisponibilidad de un servidor, etc.). Dado que en el algoritmo se encuentran implícitos los valores de carga en los *surrogates* e incluso distancias de red (véase sección 3.4.2.3.3), es relativamente simple obtener o deducir resultados análogos que en los modelos analíticos y de simulación. En este sentido se ha sometido a nuestro sistema CDN a tres tipos de pruebas:

- ***Estudio de la redirección mediante DNS.*** Esta prueba estriba en evaluar si las peticiones DNS generadas por los clientes son procesadas correctamente por el módulo Redirector, redirigiendo a los clientes a su servidor local correspondiente, siempre que éste no tenga una carga excesiva y pueda ofrecer un buen servicio. La prueba consiste en generar desde todos los clientes una carga constante de peticiones DNS (1 petición por segundo) y, al mismo tiempo, generar una carga variable sobre todos los servidores. De esta forma,

se puede comprobar cómo evoluciona el sistema y el algoritmo de redirección ante los distintos casos de funcionamiento que se pueden presentar.

- ***Estudio de la redirección basada en contenidos.*** En esta prueba se verifica que el módulo Redirector funciona correctamente ante las peticiones de contenidos de los clientes. De forma análoga al caso anterior, se genera una carga variable sobre los servidores y, al mismo tiempo, se generan peticiones para descargar alguno de los contenidos de la CDN, para comprobar que los clientes son dirigidos al servidor con el que tienen mejor conectividad.

- ***Estudio del tiempo de respuesta.*** Todo el proceso realizado por nuestro sistema, iniciado cuando se recibe una petición de un cliente, y que finaliza cuando el cliente recibe la respuesta a su petición, añade cierta latencia, que puede ser determinante. Por este motivo se ha realizado un estudio del incremento en los tiempos de respuesta introducido para comprobar si es realmente importante. Para este análisis se ha empleado el programa WAPT [WWW_Wapt].

3.4.5.1. Redirección DNS

En este estudio se analiza el funcionamiento del algoritmo de redirección para las peticiones DNS, y se muestran los resultados mediante una serie de gráficas. Concretamente, se puede apreciar cómo el algoritmo se adapta dinámicamente a las condiciones de carga de los *surrogates* en cada momento, de forma que la probabilidad de enviar una petición a uno de los servidores se basa en las condiciones de carga de cada uno de ellos con respecto a todos los demás y no únicamente en las de cada servidor separado. El análisis se divide en grupos de gráficas en las que se observan las distintas situaciones que se pueden presentar durante el funcionamiento habitual de la CDN.

En primer lugar, se muestra un proceso de inicialización de la CDN y, posteriormente, cómo evoluciona el algoritmo conforme aumenta y disminuye la carga de los *surrogates*.

En el siguiente escenario describe la situación en que un *surrogate* deja de responder a las peticiones del Monitor. Esta situación puede deberse a diferentes causas, tales como que el servidor haya dejado de funcionar (por un fallo de software o un fallo eléctrico) o que hayan sido los propios enlaces de comunicaciones los que dejen de responder, impidiendo el acceso de los clientes a los *surrogates*.

Posteriormente, se estudia la situación inversa al caso anterior, es decir, se recupera el contacto con un *surrogate* que había dejado de responder, y se observa cómo el módulo Redirector es consciente de esta recuperación y encamina peticiones a este servidor nuevamente disponible.

Finalmente, se muestran las variaciones que experimenta el algoritmo cuando se somete a los *surrogates* de la CDN a cambios muy bruscos y rápidos en sus condiciones de carga. Esto permite apreciar la capacidad de adaptación del algoritmo de redirección, aunque también es un parámetro de la frecuencia de recogida de datos de monitorización en relación a la frecuencia con la cual varían los *surrogates*.

Cada uno de estos casos se divide a su vez en dos partes. En la primera parte se explican los eventos que se producen y cómo evoluciona el algoritmo de redirección frente a estos eventos, presentando las gráficas de los parámetros de entrada y salida. En la segunda parte se presentan las gráficas en las que se muestra cómo, efectivamente, las peticiones son redirigidas hacia el *surrogate* correcto.

3.4.5.1.1. Funcionamiento normal

En esta sección se presentan una serie de gráficas correspondientes a un modo de funcionamiento habitual de la CDN. En esta situación, las condiciones de carga de los servidores varían de forma suave y la probabilidad de encaminamiento a cada *surrogate* se adapta a estos cambios.

Las sucesivas gráficas se representan con tres colores distintos. Cada color se corresponde con uno de los servidores de nuestra maqueta de pruebas. En concreto, el color azul se corresponde con el *surrogate* 1, el rojo con el *surrogate* 2 y el verde con el *surrogate* 3. Este código de colores se mantendrá en todas las gráficas de las siguientes secciones. Por otro lado, en las gráficas se han superpuesto una serie de líneas verticales de color azul que indican cuando se produce un evento significativo en el funcionamiento de la CDN. Estas líneas se utilizan para facilitar la identificación de un mismo evento en las diferentes gráficas vinculadas a diferentes parámetros.

La Figura 97 muestra una serie de gráficas del funcionamiento de la CDN, relativas a los parámetros que rigen la ejecución del algoritmo de redirección. El punto de partida son los datos brutos de monitorización, esto es, el porcentaje de CPU utilizado y las conexiones TCP pasivas (en adelante conexiones). A partir de estos valores de entrada, el Redirector calcula los valores de x y, a partir de estos, se calculan los valores de k. Finalmente, se obtiene el valor del coeficiente f de cada *surrogate* a partir del valor de k y del valor anterior de f.

Inicialmente, en un tiempo inferior a t=120 s, todos los *surrogates* presentan una carga muy baja de CPU (inferior al 10%) y apenas reciben conexiones. Esto implica un valor reducido del parámetro x (cercano a 0.1) y un valor del parámetro k cercano a 1.5 (véase Figura 94). En estas condiciones de similaridad en la carga, el valor del parámetro f es prácticamente el mismo para todos los servidores y adopta un valor cercano a 0.33. Esto quiere decir que las nuevas peticiones durante este período de monitorización se reparten de manera equitativa entre los tres *surrogates*. Esta situación se mantiene hasta el primer evento significativo (t=120 s).

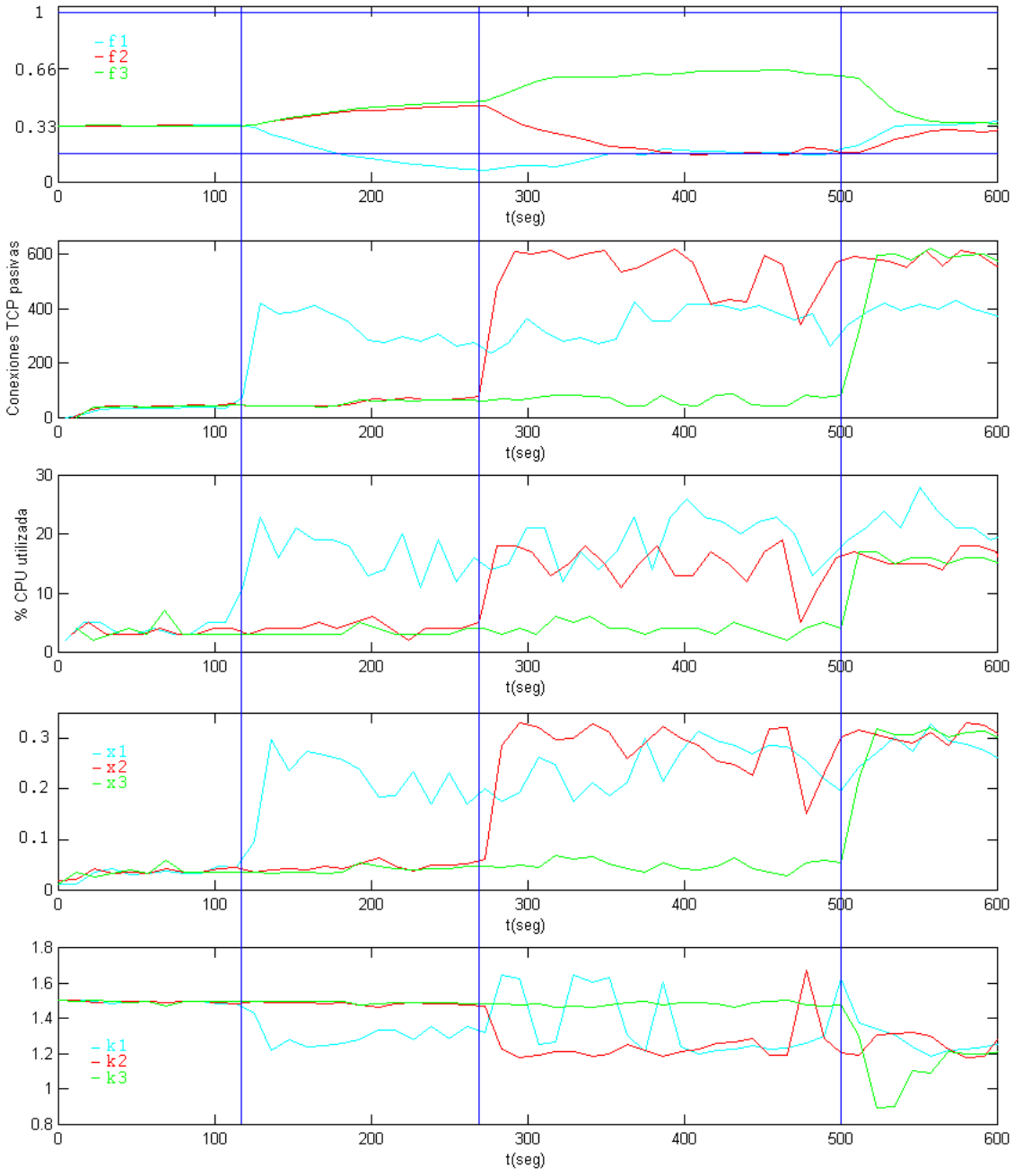


Figura 97. Funcionamiento normal.

Hito 1 (120 segundos)

El primer evento (t=120 s) consiste en una variación de las condiciones de carga del *surrogate* 1. Como se puede ver en la gráfica, el número de conexiones que el *surrogate* 1 comienza a recibir se incrementa hasta un valor de 400 durante un periodo de muestreo. En las pruebas realizadas, se ha tomado un periodo de muestreo de 10 segundos, por lo que el *surrogate* 1 está recibiendo 40 conexiones por segundo de media. Este aumento de las conexiones implica un aumento del porcentaje de CPU utilizado (se ha de atender a más clientes), como se puede apreciar en la gráfica. Al producirse este aumento en los parámetros de entrada, el valor de *x* asociado aumenta

también y se sitúa en valores cercanos a 0.2-0.3, lo que se traduce en una disminución en el valor de *k*, y por tanto del valor de *f*.

Dada la interdependencia entre servidores, el efecto producido en el *surrogate* 1 tiene un efecto en los coeficientes *f* de los otros dos servidores, tal y como se puede observar en la Figura 97. Ya que los *surrogates* 2 y 3 permanecen descargados mientras que el *surrogate* 1 soporta un nivel de carga alto y continuado, los coeficientes *f* de estos dos (*surrogates* 2 y 3) aumentan de forma similar, mientras que el del *surrogate* 1 disminuye. Esto indica que el algoritmo se ajusta a estas condiciones de carga, es decir, dado que el *surrogate* 1 está más cargado, el porcentaje de nuevas conexiones en los siguientes instantes será menor (coeficiente *f* menor), mientras que los *surrogates* 2 y 3 atenderán más peticiones(coeficientes *f* mayores)

Hito 2 (270 segundos)

El segundo evento (t=270 s) lo constituye el aumento de la carga del *surrogate* 2, ya que empieza a recibir unas 60 conexiones por segundo. A partir de este momento, se puede observar cómo el valor de *f* correspondiente al *surrogate* 2 comienza a disminuir, mientras que el coeficiente *f* para los *surrogates* 1 y 3 aumenta. Otro efecto interesante producido por el algoritmo de redirección en este segundo hito es la "igualación" de niveles cuando se supera un determinado umbral. En este caso, en la gráfica correspondiente a los valores *f* se ha dibujado un umbral inferior de valor 0.15. Esto quiere decir que, en condiciones de funcionamiento, incluso con alta carga, cualquier *surrogate* de la CDN es capaz de absorber un 15% del total de conexiones (peticiones) previstas en un intervalo. Evidentemente, se trata de un parámetro de diseño que se fija inicialmente en base a la capacidad de los *surrogates* y la estimación de peticiones esperada. En la gráfica de los coeficientes f se observa cómo el *surrogate* 1 ha descendido (t=180 s) por debajo de este umbral; sin embargo, en este segundo hito se aprecia cómo el algoritmo se ajusta de tal forma (t= 400 s) que los *surrogates* 1 y 2 igualan su coeficiente f sobre este valor umbral. Este ajuste tiene lugar, principalmente, a través del parámetro *k* que, tras una serie de oscilaciones, su valor se iguala para el *surrogate* 1 y 2 (niveles de carga similares). La velocidad de adaptación en referencia a este umbral (0.15) no es importante para los niveles de carga de los servidores: en este caso, no se requiere una adaptación inmediata, ya que es más importante la relación de los coeficientes f de los *surrogates* para cada instante de muestreo.

Hito 3 (500 segundos)

En el tercer evento (t= 500 s), se iguala la carga de los tres *surrogates*, aumentando la carga del servidor 3, que también recibe aproximadamente 60 conexiones por segundo. El efecto producido es el esperado: el valor de *f* de este servidor disminuye para igualarse a la de los otros dos alrededor del valor 0.33. Se trata, en la situación final, de

un caso muy parecido al punto de partida, con algunas muy ligeras variaciones en los valores de *f*. Por otro lado, como sucedía análogamente en el hito anterior, el valor de *k* disminuye rápidamente cuando se incrementa la carga del servidor 3 (capacidad e autoajuste del algoritmo), pero va aumentando gradualmente hasta que se alcanza a la situación de equilibrio, momento en el que el valor asignado a todos los *surrogates* es similar.

Desde un punto de vista general, se puede apreciar que el valor de *f* tiene un valor de 0.33 tanto al principio (t=0 s) como al final (t=600 s). Sin embargo, las condiciones de carga son distintas en ambos instantes. Al principio no hay carga en ninguno de los *surrogates*, mientras que al final todos ellos se encuentran cargados. Se puede ver que el algoritmo implementado en el módulo Redirector asigna el valor de *f* atendiendo a la diferencia de carga entre los *surrogates* y no a las condiciones individuales de carga.

Una vez estudiado el comportamiento general básico del algoritmo de redirección, es importante comprobar que, efectivamente, las peticiones se redirigen correctamente a los surrogates correspondientes dependiendo de cada situación de carga de los servidores. Esto se puede observar en las gráficas representadas en la Figura 98. En este caso, los eventos significativos son los cruces por el umbral de *f* de cada *surrogate*. Es decir, el umbral inferior fijado en 0.15 es sobrepasado en varias ocasiones por los *surrogates* 1 y 2, lo que se traduce en una redirección de usuarios, como se describirá.

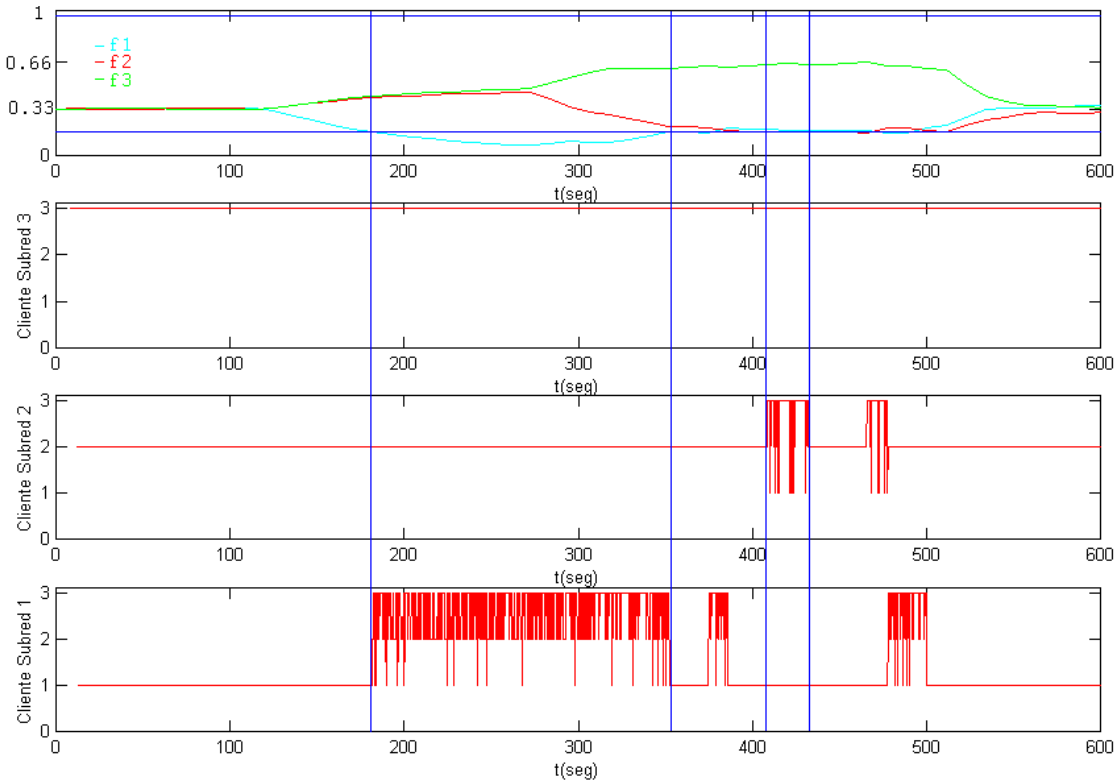


Figura 98. Redirección en funcionamiento normal.

En la Figura 98 se representan los coeficientes *f* y los *surrogates* a los que se redirigen las peticiones para tres clientes distintos situados en cada una de las tres subredes de la maqueta. Cada uno de estos clientes tiene asociado un *surrogate* local que es al que se redirigen las peticiones en condiciones normales.

En la gráfica correspondiente al cliente situado en la subred 3 (cliente 3), se observa que todas sus peticiones siempre se redirigen hacia el *surrogate* 3, su servidor local correspondiente, ya que el valor de *f* para el *surrogate* 3 siempre se encuentra por encima del umbral inferior (0.15).

La gráfica asociada al cliente situado en la subred 1 (cliente 1) es la que presenta un comportamiento más variable. Mientras el valor del coeficiente *f* del *surrogate* 1 está por encima del umbral, las peticiones se redirigen hacia él. En el momento en que se produce el cruce por el umbral inferior (t= 180 s), la mayor parte de las peticiones son redirigidas hacia los *surrogates* 2 y 3, y sólo un pequeño porcentaje son encaminadas al servidor local (*surrogate* 3). Esto es debido al componente aleatorio que se añade en la elección del *surrogate* adecuado cuando el local se encuentra por debajo del umbral. Puede observarse que el número de peticiones que recibe es pequeño en proporción con el que reciben los *surrogates* 2 y 3, y siempre será dependiente del valor de *f*. Esta situación permanece así hasta que se produce un nuevo cruce por el umbral (t= 350 s), pero de modo ascendente esta vez, por lo que gran parte de las peticiones vuelven a dirigirse al *surrogate* local. Dado que este cruce es suave (baja pendiente), existen unos pequeños periodos donde el valor de *f* puede, ocasionalmente, volver a disminuir por debajo del umbral. Esto se traduce en unas redirecciones ocasionales a los *surrogates* 2 y 3 (t =370 s, t= 40 s), si bien el algoritmo de redirección se estabiliza una vez se ha cruzado de manera clara dicho umbral inferior y, al final, todas las peticiones generadas en la subred 1 por el cliente 1 son encaminadas al *surrogate* 1.

Si se observa ahora la gráfica correspondiente al cliente 2, entre los instantes t=410s y t= 430 s, se puede ver otra situación característica. En este caso, el valor de *f* del *surrogate* 2 está por debajo del umbral, mientras que los valores de *f* del *surrogate* 1 y 3 se encuentran por encima. Sin embargo, el *surrogate* 1 está muy próximo al umbral, mientras que el servidor 3 está bastante alejado de este valor umbral. Esto se corresponde con una situación en la que los *surrogates* 1 y 2 están cargados y el servidor 3 descargado. En este caso, no sería correcto redirigir las peticiones desde el *surrogate* 2 de forma equitativa hacia los servidores 1 y 3, ya que las condiciones de carga del *surrogate* 1 son claramente más desfavorables que las del *surrogate* 3. Este balanceo equitativo puede apreciarse en la gráfica correspondiente, ya que existen más redirecciones al *surrogate* 3 (zonas sin variación) que al *surrogate* 1. Como el número de peticiones enviadas a cada *surrogate* depende del valor de f, el *surrogate* 3 recibe más peticiones procedentes del cliente 2, que es el comportamiento deseable en esta situación. En t=465 s se produce una pequeña oscilación debido al cruce por el umbral, pero el algoritmo se estabiliza en la situación final (análogo al cliente 1).

3.4.5.1.2. Caída de un *surrogate*

En el funcionamiento normal de un servidor se pueden presentar diferentes situaciones por las cuales la conectividad con éste se ve interrumpida. Esto puede estar motivado por varias razones: cortes de electricidad, fallos de software o hardware, ataques de denegación de servicio, virus que bloquean el sistema o, simplemente, cortes en las comunicaciones que impiden acceder al servidor.

Ante este problema, es necesario que el algoritmo de redirección detecte este tipo de eventualidades y se adapte a esta situación para no redirigir ninguna petición hacia un servidor no accesible. En esta sección se muestra cómo reacciona el algoritmo ante estos eventos y cómo modifica las probabilidades de reenvío de cada *surrogate* para adaptarse a la nueva situación.

La Figura 99 se representa esta situación. En ésta, se pueden observar dos eventos significativos. El primero de ellos corresponde con la caída del *surrogate* 3, mientras el segundo de ellos muestra la caída del *surrogate* 2.

Analizando las gráficas desde el principio (t= 3200 s, ya que se sigue con la simulación anterior) se observa que todos los *surrogates* presentan una carga similar, por lo que sus valores de *f* son similares y cercanos a 0.33. En la gráfica de conexiones se puede ver que, en un momento determinado (t= 3300 s), se dejan de recibir conexiones en el *surrogate* 1. Esto se traduce en una reducción de la carga (*x1*), por lo que comienza a aumentar el valor de *f* en este servidor (f1) mientras disminuye en los otros dos (*f2, f3*). Se trata de una situación similar a la descrita en la sección anterior, por lo que no se entrará en más detalles.

En el instante t=3815 s, se puede observar que la el valor de *f* del *surrogate* 3 cae bruscamente a cero, mientras que las de los otros dos *surrogates* aumentan. Es en este momento cuando se ha producido la pérdida de conectividad con el servidor 3, por lo que desde este momento no se redirige ninguna petición hacia este servidor.

Se puede apreciar que, tras la caída del *surrogate* 3, aparece un aumento de las conexiones recibidas por el *surrogate* 2. Esto no se debe solamente a la caída del *surrogate* 3, sino que también se ha producido un aumento de las conexiones del *surrogate* 2. Una vez finaliza este pico de conexiones, la carga que están soportando los *surrogates* 1 y 2 es similar, por lo que el valor de *f* de ambos tiende a igualarse alrededor del valor 0.5 (ya que ambos servidores deben soportar todas las peticiones a la CDN), mientras que la del s*urrogate* 3 permanece en el valor cero.

Una vez se han igualado los valores de *f* de los *surrogates* 1 y 2, se provoca la caída del *surrogate* 2 (t =4260 s), lo que implica que el valor de *f* de este *surrogate* (*f2*) pasa a valer cero y el valor de *f* del *surrogate* 1 es 1. A partir de este momento, todas las peticiones que se reciban en la CDN serán redirigidas al *surrogate* 1 que es el único disponible.

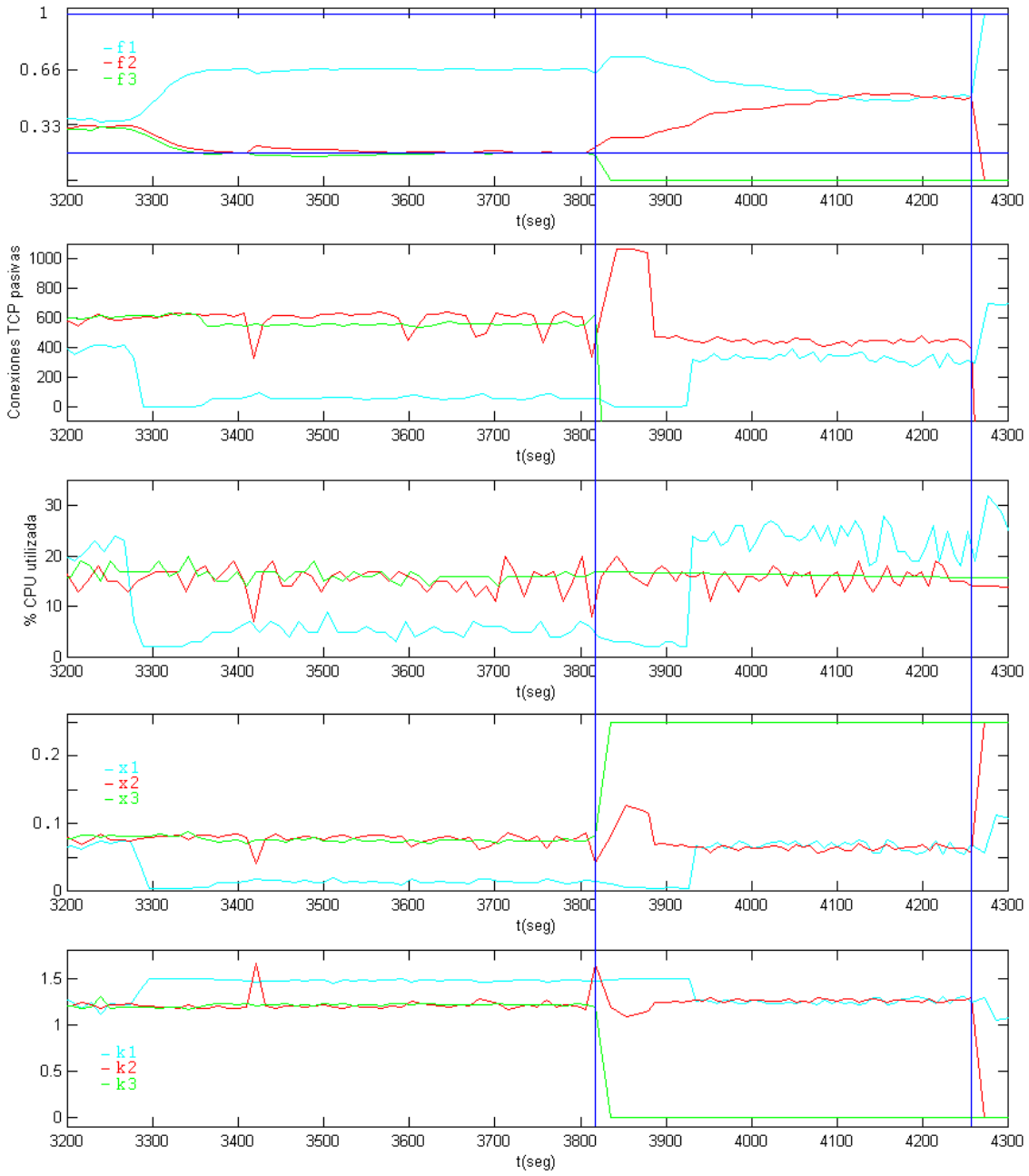


Figura 99. Caída de un surrogate.

En la Figura 100 se puede comprobar que la redirección de usuarios se produce en base al valor del coeficiente *f* (*f1*, *f2* y *f3*). Cuando se produce la caída del *surrogate* 3 (t=3815 s), todas las peticiones que realiza el cliente 3 son redirigidas hacia los *surrogates* 1 y 2, mientras que las de los clientes 1 y 2 son redirigidas hacia sus respectivos *surrogates* locales. Cuando se produce la caída del *surrogate* 2 (t =4260 s), todas las peticiones de todos los clientes son redirigidas hacia el *surrogate* 1 que es el único disponible.

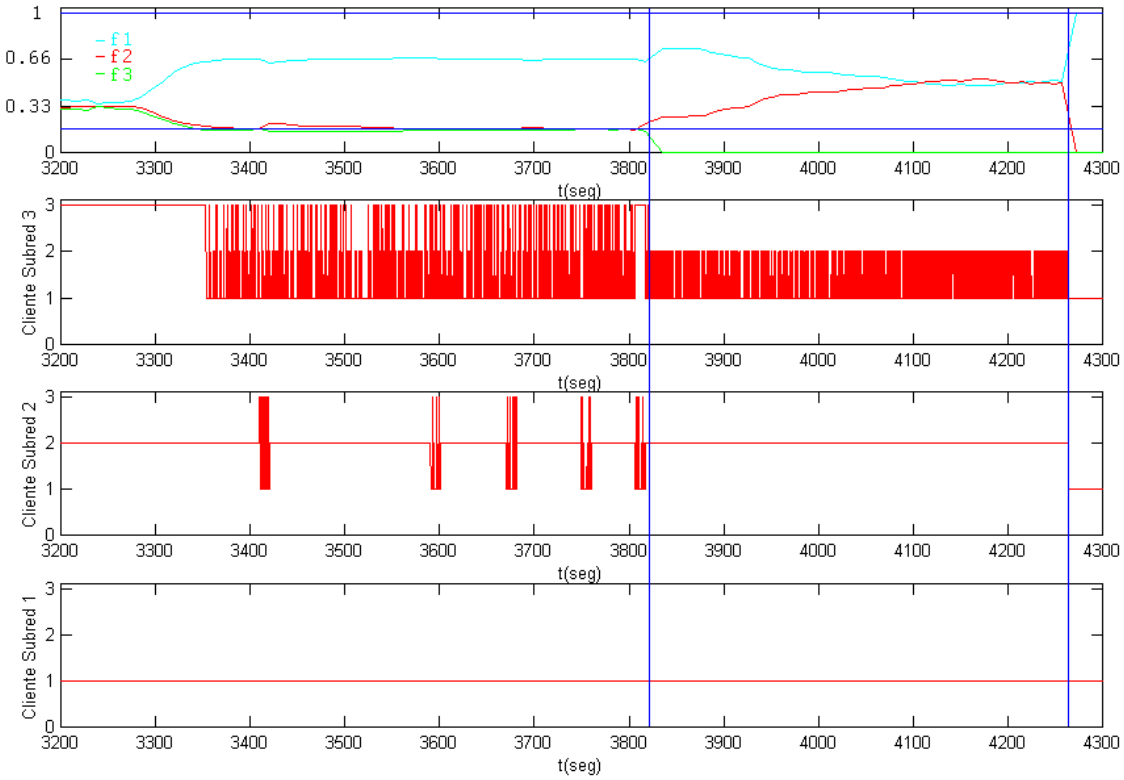


Figura 100. Redirección en caída de un surrogate.

3.4.5.1.3. Recuperación de *surrogates* caídos y variaciones abruptas de la carga

Las gráficas de las secciones anteriores presentaban un comportamiento de los *surrogates* bajo condiciones de carga controlada. Este control se ha establecido mediante un generador de peticiones HTTP (JMeter), por lo que es posible obtener un número de peticiones más o menos constante, lo que se refleja en una carga de CPU con un comportamiento estable.

En una situación real, por el contrario, las peticiones se reciben de una forma aleatoria, con picos de tráfico no predecibles a priori. Adicionalmente, los *surrogates* pueden presentar aumentos puntuales del porcentaje de CPU utilizado por otros procesos ejecutándose diferentes al propio procesado de las peticiones. A continuación se muestra este tipo de situaciones y cómo el algoritmo de redirección se adapta a los cambios que se producen.

Las gráficas representadas en la Figura 101 continúan con la situación descrita en la sección anterior. El único *surrogate* accesible es el 1 ya que se había perdido la conectividad con los otros dos servidores. En el instante t= 7200 s, se recupera la conectividad con el *surrogate* 2. El valor de *f2* aumenta rápidamente hasta situarse por encima de *f1*, ya que se encuentra menos cargado.

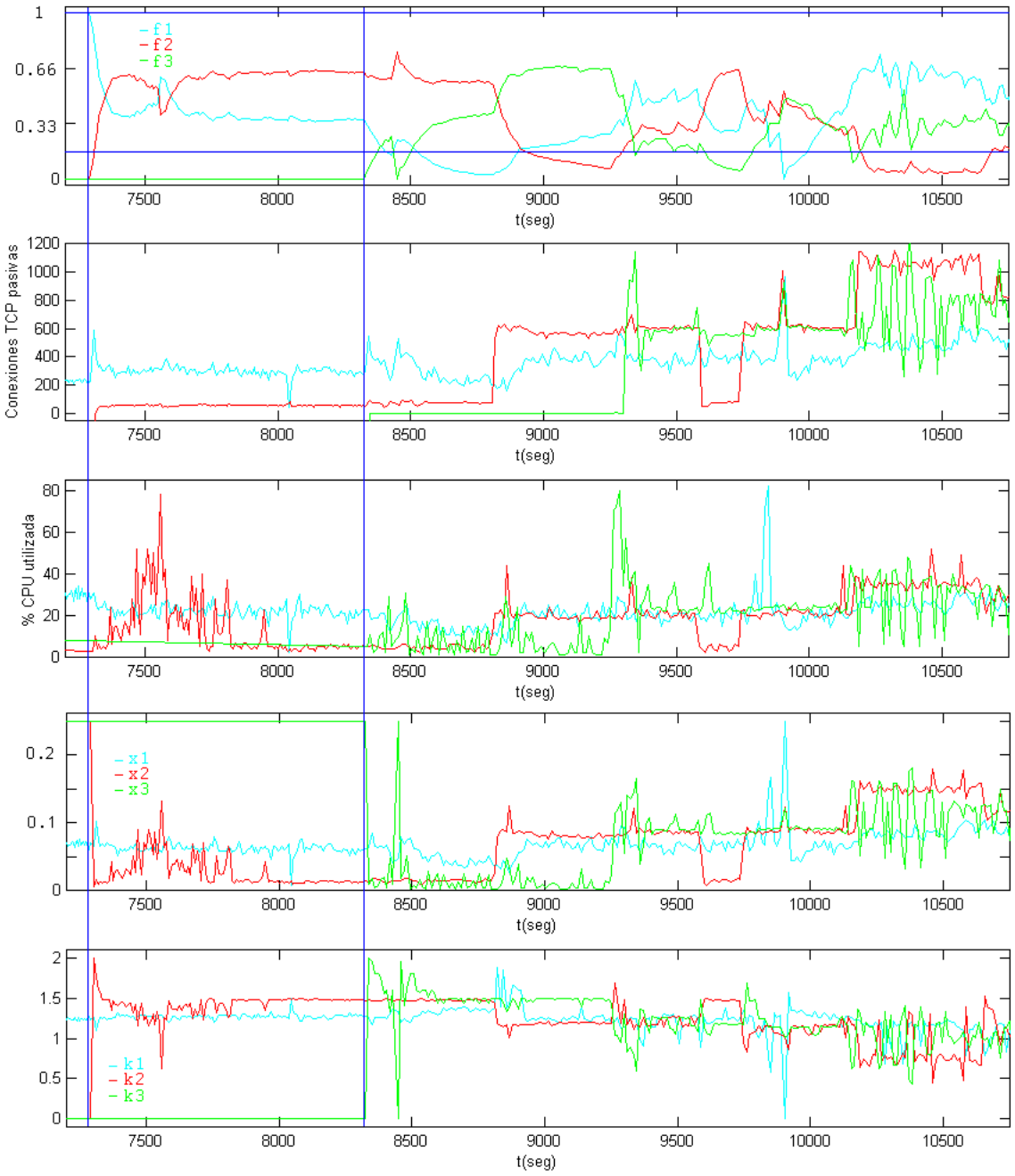


Figura 101. Recuperación de surrogate y variaciones de carga abruptas.

Durante este tiempo el *surrogate* 2 experimenta ciertos picos en la utilización de CPU, lo que se refleja en disminuciones de *f2*. En el instante t= 8350 s se recupera la conectividad con el *surrogate* 3 y *f3* comienza a aumentar. A partir de este momento, se somete a los *surrogates* a cambios rápidos y aleatorios en sus condiciones de carga. Se puede observar que, cuando un *surrogate* tiene picos de CPU, su valor de *f* disminuye y viceversa. Lo mismo sucede con las conexiones que recibe cada servidor. Es importante destacar que se comienza con los servidores sometidos a una carga moderada y progresivamente se va aumentando hasta que al final es bastante elevada y con variaciones rápidas. El algoritmo de redirección se adapta rápidamente a estos cambios de carga de los *surrogates* aumentando o disminuyendo el valor de los coeficientes *f*.

3.4.5.2. Redirección basada en contenidos

En el apartado anterior se ha analizado el funcionamiento de la CDN (su algoritmo de redirección) cuando el servidor DNS recibe peticiones de los clientes. En este apartado el objetivo es analizar el comportamiento de la CDN cuando un cliente solicita un contenido web almacenado en la CDN. Por simplicidad en el estudio, se han tenido en cuenta las siguientes consideraciones al hacer las pruebas:

- Todos los *surrogates* poseen el contenido solicitado (replicación al 100%). En caso contrario, sería necesario estimar si es mejor redirigir al cliente a un *surrogate* que posea el contenido o mandar el contenido al *surrogate* que se encuentra en mejores condiciones para servirlo al cliente, pero que no posee una copia y debe conseguirla. Estas consideraciones se dejan para siguientes capítulos donde se analiza el contenido de streaming (véase sección 4.3).

- Los parámetros calculados para seleccionar el *surrogate* óptimo que servirá el contenido web dependen, en este caso, no sólo del estado de los servidores, sino también del cliente que lo ha solicitado (su ubicación de red en referencia a los *surrogates*). De este modo, por cada cliente tendremos una gráfica de los coeficientes *y*, *g* y *h* (véase sección 3.4.2.3) de cada *surrogate*, tres en este caso.

Es importante destacar que los valores de los coeficientes *y*, *g* y *h* únicamente se calculan cuando algún *surrogate* está por debajo del umbral inferior de *f* (en nuestro caso establecido en 0.15, aunque parametrizable) y un cliente local le solicita un contenido web. Cuando el *surrogate* se encuentra por encima de dicho umbral no se realizan estos cálculos, por sencillez, eficiencia y escalabilidad. Es por ello por lo que, en las gráficas que se mostrarán, existen intervalos de tiempo donde estos coeficientes (*y*, *g* y *h*) no se muestran. Téngase en cuenta que por encima de este umbral las peticiones se redirigen al *surrogate* local, que será el que mejor conectividad en términos de red ofrezca. Cuando se está por debajo del umbral, hay que estimar aquella subred no local que ofrece unas mejores garantías de servicio.

En la Figura 102 se representan las gráficas obtenidas al realizar las peticiones desde un cliente ubicado en la subred 3. Un primer aspecto a destacar es, sin duda, la fluctuación que presentan los parámetros *g* de cada *surrogate*. Esto es debido al comportamiento del algoritmo descrito en la sección 3.4.2.3.3.2. Este algoritmo asigna al servidor con mejor conectividad un valor de 1, y al resto le asignará valores inferiores, proporcionales al valor de *y*. Dado que inicialmente (hasta t=1000 s) todos los *surrogates* presentan valores de *y* muy similares, el valor de *g* oscila mucho. En el último tramo (entre 1015 s y 1100 s), los valores de *y* se distancian más entre sí, lo que se traduce en que las fluctuaciones de los coeficientes *g* son menores. Cuando los valores de *y* son lo suficientemente distintos, las variaciones de *g* son pequeñas.

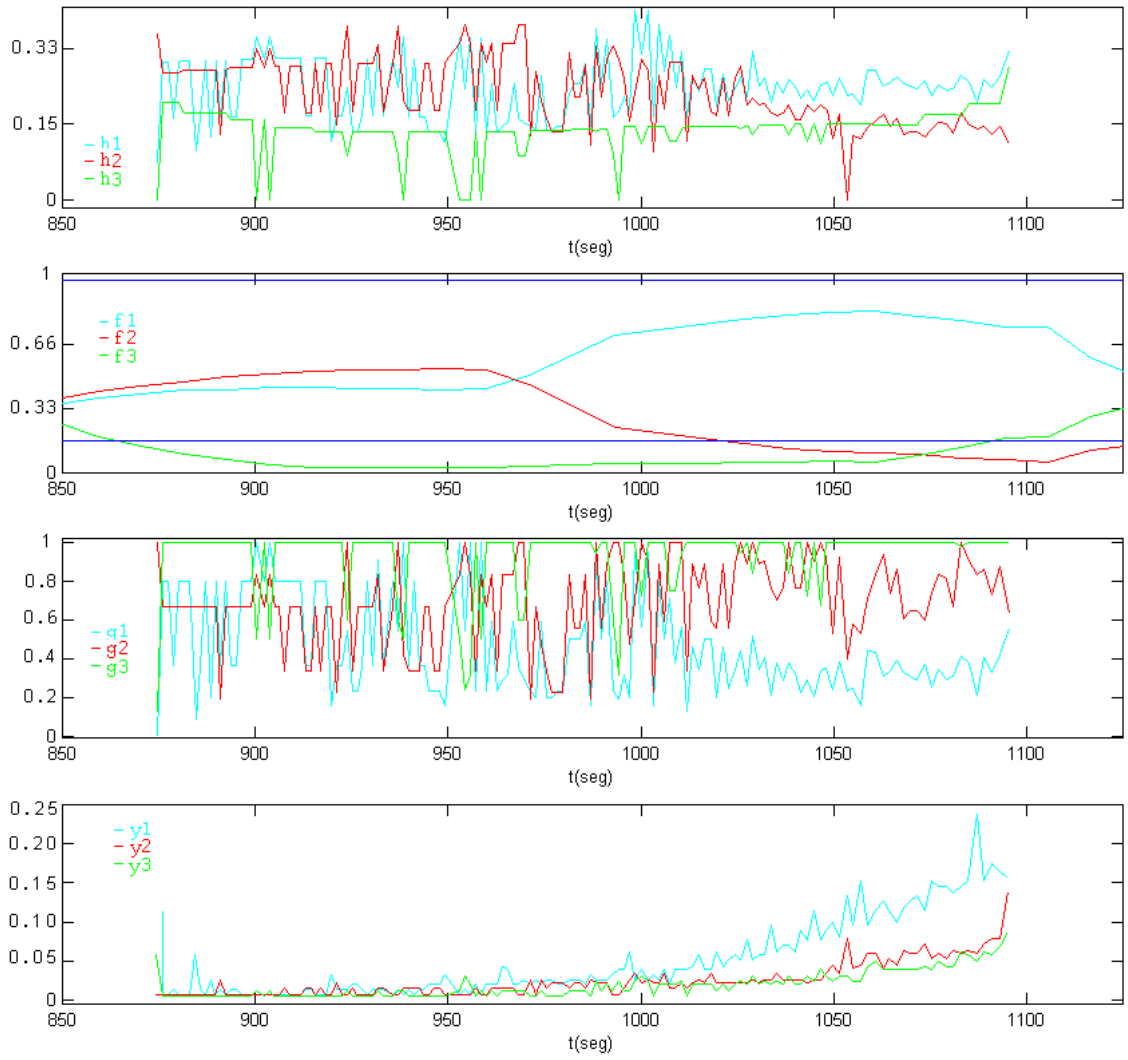


Figura 102. Redirección basada en contenidos. Coeficientes *g* y *h* (cliente 3).

Otro aspecto que se puede observar es que, dado un cliente, siempre hay uno de los *surrogates* para el cual predomina el valor de 1 para *g*. En el caso de la Figura 102 es el *surrogate* 3 el que presenta mejores condiciones de red con el cliente. Estos resultados son, evidentemente, los que cabría esperar ya que se encuentran en la misma subred. Por otro lado, en las gráficas se puede apreciar que, si bien el valor de *g3* es elevado para un cliente en la subred 3, el valor de *f3* es muy bajo (claramente por debajo del umbral). Esto conduce a una situación final de coeficientes *h3* bajos, ya que los pesos asignados a *f* y *g* en la expresión de *h* son similares (véase sección 3.4.2.3.3.3).

En Figura 103 se realiza otro experimento donde las peticiones parten de un cliente situado en la subred 2. En este caso, el *surrogate* 2 es claramente el que presenta unas condiciones de red mejores para el cliente en la subred 2, con respecto a los demás *surrogates* (*g2* siempre es mayor que *g1* y *g3*, ya que *y2* siempre es menor que *y1* e *y3*). A diferencia de la Figura 102, en la Figura 103 los valores de *f* están más próximos, lo que se refleja en el hecho de que *h2* mantiene un valor elevado durante mayor tiempo y, por tanto, será el elegido para transmitir el contenido solicitado.

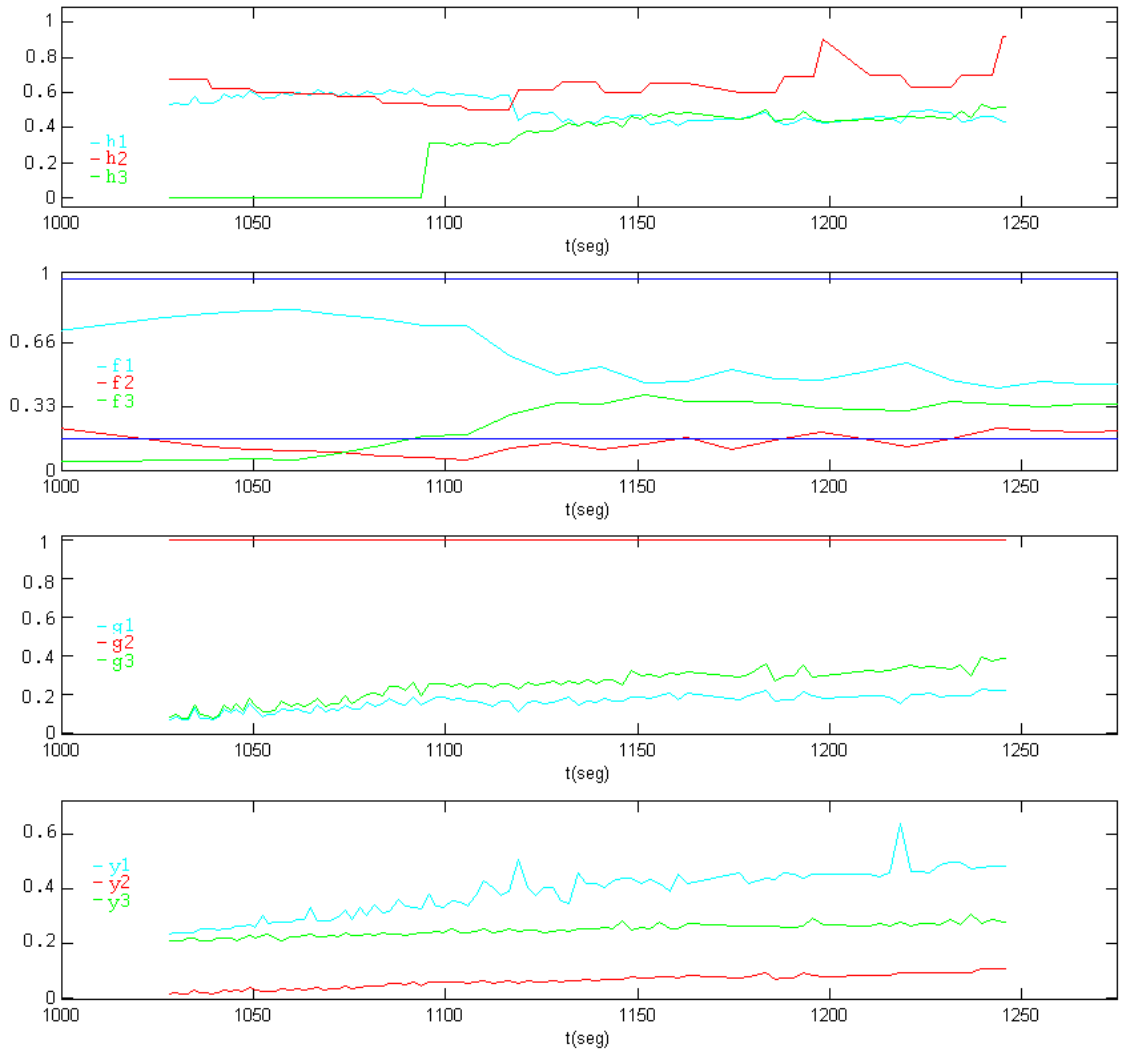


Figura 103. Redirección basada en contenidos. Coeficientes *g* y *h* (cliente 2).

Continuando con estos dos casos descritos, se va a mostrar a continuación cómo se han redirigido las peticiones de contenidos de los dos clientes.

En primer lugar, en la Figura 104 se representan las gráficas correspondientes al cliente situado en la subred 3. Se observa que las peticiones sólo se encaminan a un *surrogate* distinto al local cuando el valor de *f* del servidor local es inferior al umbral, y sólo entonces se calcula el valor de *g* y *h*. Durante este periodo, las peticiones son redirigidas hacia el *surrogate* que tenga un mayor valor de *h*. La mayoría de peticiones se dirigen a los *surrogates* 1 y 2, pero dos de las peticiones son redirigidas al *surrogate* 3, porque como se puede ver, el valor de *h3* es mayor en ciertos instantes (t=890 s, t= 980 s).

En la Figura 105 se puede ver que el cliente ubicado en la subred 2 siempre es redirigido al *surrogate* local salvo en un pequeño periodo de tiempo durante el cual el valor de *h* del *surrogate* 1 es algo mayor. En este caso, el *surrogate* 2 presenta una carga elevada como se refleja en el valor de *f2*, pero también es el que presenta menor retardo con el cliente 2, con lo que el valor de *g2*, y por tanto el de *h2*, sea más alto que el de los otros dos *surrogates*.

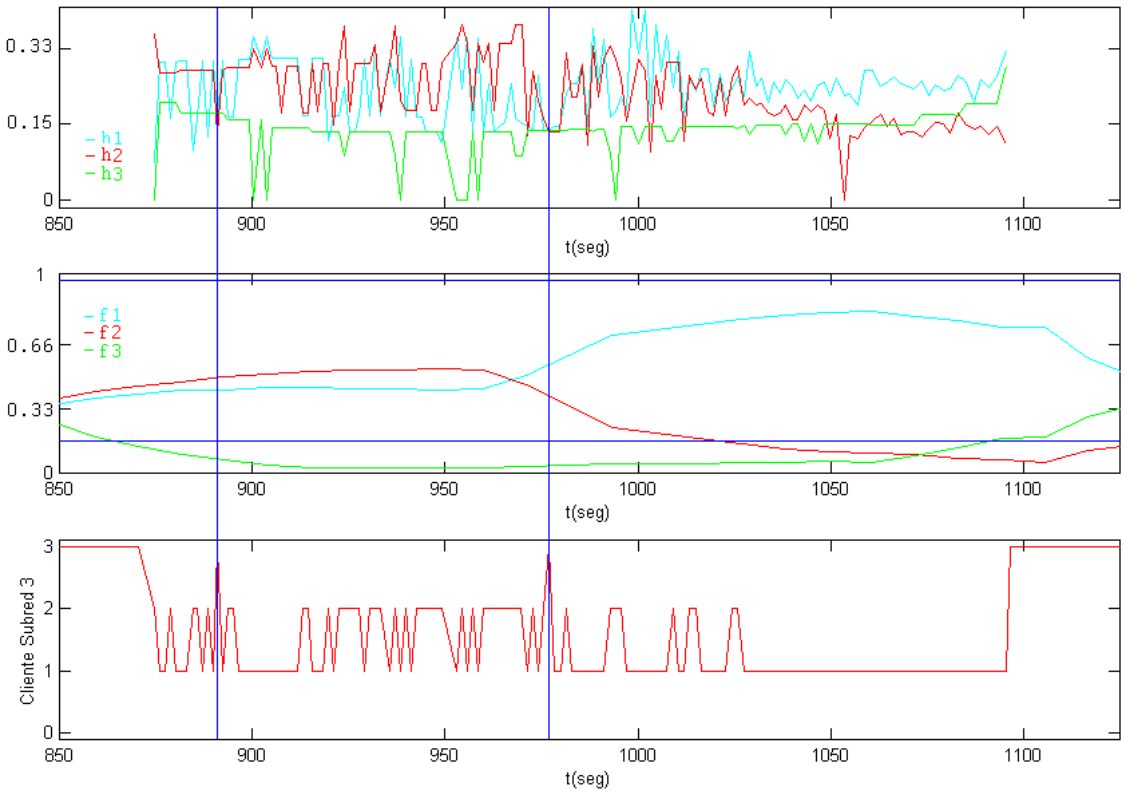


Figura 104. Redirección basada en contenidos (redirección cliente 3).

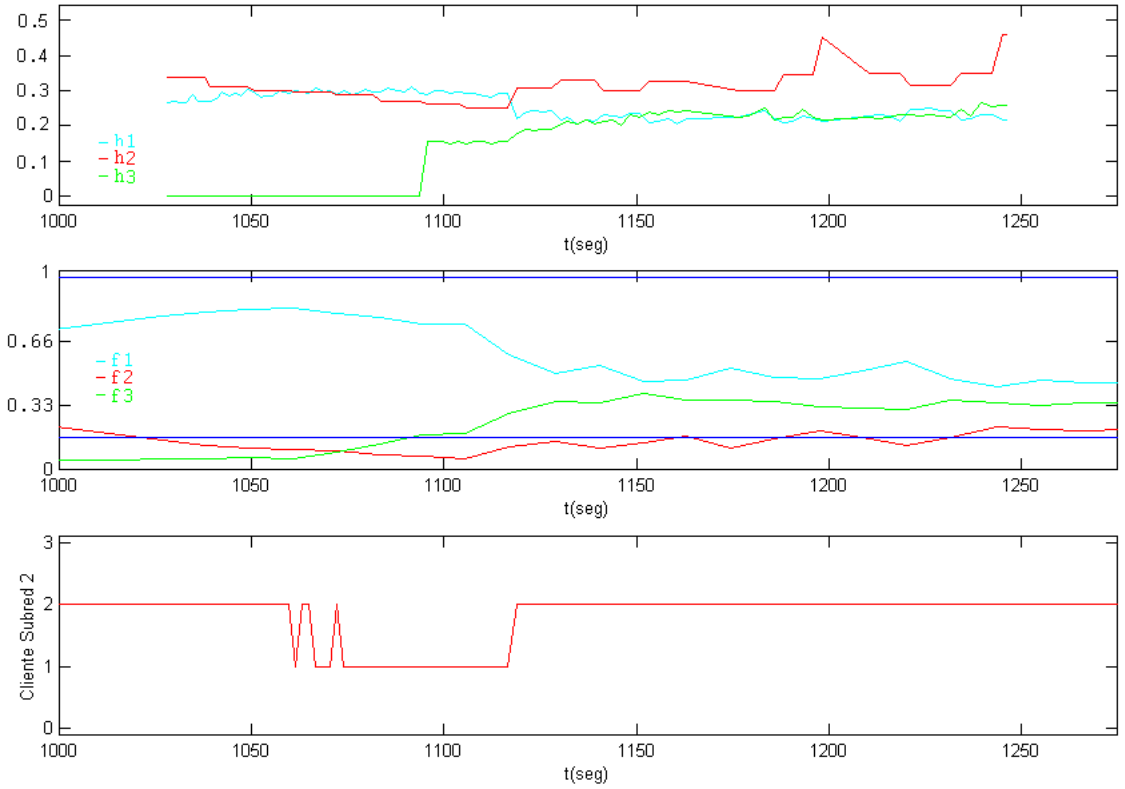


Figura 105. Redirección basada en contenidos (redirección cliente 2).

3.4.5.3. Estudio de los tiempos de respuesta

Uno de los factores críticos en el rendimiento de una aplicación web es el tiempo que debe esperar el cliente hasta que ese contenido se muestra en la pantalla de su ordenador. Esto se conoce como latencia percibida. Si este tiempo de espera se alarga excesivamente, el cliente seguramente cancelará su petición.

El proceso de selección del *surrogate* introduce un retardo adicional en el proceso de petición. Si bien esto es contrario a lo que en un principio se pretende, que es la minimización del tiempo de espera, al seleccionar el *surrogate* más adecuado, nos aseguramos de que el cliente correspondiente va a estar conectado al servidor que menores tiempos de respuesta le proporcionará en las siguientes peticiones.

En esta sección se mostrarán los incrementos en los tiempos al introducir los mecanismos de selección de *surrogate*. Como se podrá comprobar, el aumento del tiempo de espera es relativamente pequeño si se considera las mejoras que se introducen en las sucesivas peticiones.

En primer lugar, se estudiará el retardo introducido por la resolución DNS; en segundo lugar, se analizará el tiempo requerido en recibirse una página web cuando se solicita un contenido en concreto (consulta a la base de datos).

3.4.5.3.1. Tiempo medio de la consulta DNS

Cada vez que un cliente se conecta por primera vez a la CDN lo hace a través de su navegador introduciendo una URL, por ejemplo http://cdn.upv.es. En este proceso, el primer paso por parte del navegador es una petición al servidor DNS de su red solicitando la resolución IP de la máquina que está registrada con el nombre 'cdn.upv.es'. En esta consulta DNS se introduce el primer mecanismo de selección de *surrogate*. El nombre 'cdn.upv.es' está asociado en el servidor DNS con múltiples direcciones IP, correspondientes a los *surrogates* que forman parte de la CDN. En el proceso de resolución de direcciones, se realizan los cálculos necesarios y se elige la dirección IP del *surrogate* con mejores condiciones de carga.

Estos cálculos previos implican un aumento del tiempo de espera del cliente. A continuación se presentan los tiempos medios de espera de un cliente para acceder a esta primera página web de la CDN, que es la que implica una resolución DNS. Téngase en cuenta que, las restantes peticiones por parte del navegador, usarán la propia caché interna del navegador, al menos durante toda la sesión.

	Secuencial 1x100	Concurrente 1x100c	Implosión 100x1	Secuencial 1x200	Concurrente 1x200c	Implosión 200x1
Surrogate 1	4.9326	7.7684	8.4357	5.0717	8.4521	8.5158
Surrogate 2	4.4802	4.4199	4.8338	3.6674	4.915	5.1228
Surrogate 3	3.5851	4.1097	4.4831	3.62405	4.4673	4.5502

Tabla 32. Tiempos medios de respuesta web (en ms).

En la Tabla 32 se muestran varios tiempos de respuesta (medidos en ms) para diferentes pruebas que se describen a continuación:

- Se han realizado los mismos experimentos para todos los *surrogates* que constituyen el banco de pruebas, a fin de comprobar las diferencias entre ellos. En la primera columna aparece un identificador de cada *surrogate*.
- Los tiempos medios que se muestran en la Tabla 32 representan peticiones directas a los *surrogates*, es decir, que no se realiza una resolución DNS. Representan únicamente los tiempos medios de solicitud y transferencia del contenido web.
- En la primera fila aparece un identificador del tipo de prueba realizada:
    - *Acceso secuencial*: Se trata de un mismo cliente que ha solicitado una misma página web 100 (1x100) y 200 (1x200) veces respectivamente. El cliente solicita una página, espera a recibir completamente la respuesta y cuando ha finalizado pasa a solicitar la siguiente página. Se ha empleado un cliente web sin caché interna para que no almacene la página web.
    - *Acceso concurrente (1x100c y 1x200c)*: Es un caso similar al anterior pero con la diferencia de que ahora hay varios clientes que están solicitando esta misma página. Se trata de una medida en condiciones de carga alta.
    - *Efecto implosión (100x1 o 200x1)*: Se intenta medir en este caso la capacidad del *surrogate* para soportar una avalancha de peticiones y cómo afecta al tiempo de respuesta. En este caso se emulan 100 o 200 usuarios solicitando la misma página web simultáneamente una única vez.

Como se puede apreciar en la Tabla 32, el tiempo medio que se debe esperar para recibir la página web es muy pequeño, en torno a los 4 o 5 milisegundos. Hay diferencias entre los tiempos medios de cada *surrogate* debidas al tipo de hardware de cada servidor, pero incluso en el peor de los casos (*surrogate* 1) el incremento del tiempo de espera es imperceptible por el cliente.

Cuando se han realizado las pruebas de carga (acceso concurrente), los resultados obtenidos han sido muy similares a las pruebas sin carga. Incluso en las peticiones simultáneas (100x1 y 200x1) los resultados son relativamente bajos.

	1x100c	1x200c
DNS	68.455	70.01

Tabla 33. Tiempo medio de respuesta del servidor DNS (en ms).

Una vez descritos los resultados de la Tabla 32, se evalúa el retardo introducido en la consulta DNS. En este caso se ha medido el tiempo de respuesta de nuestro servidor DNS cuando se ha de seleccionar el *surrogate* adecuado. Los resultados (medidos en ms) se muestran en la Tabla 33, donde únicamente se han realizado dos medidas, solicitando 100 y 200 resoluciones DNS respectivamente. En este caso, se ha optado por la opción concurrente ya que es el caso más realista para una CDN.

Se puede apreciar que el tiempo medio adicional introducido para seleccionar el *surrogate* está alrededor de los 70 ms, lo cual constituye un retardo significativamente superior a la propia petición web. Sin embargo, este retardo sigue siendo asumible para el cliente, ya que es imperceptible para él y puede mejorar mucho el tiempo de respuesta en las siguientes peticiones.

3.4.5.3.2. Tiempo medio de un recurso web con consulta a base de datos

La motivación de esta medida está originada en una posterior comparación con la siguiente sección. Se trata de una petición que implica una consulta a la base de datos. Se pretende disponer de una referencia del tiempo necesario por parte de un *surrogate* en responder a una petición de una página web donde no realiza ningún tipo de selección de *surrogate* y, posteriormente, comparar los resultados con los obtenidos cuando sí que se realizan estos cálculos. De esta forma, se puede conocer el incremento de tiempos en todo el proceso de una sesión web.

La Tabla 34 muestra los valores de los tiempos de respuesta (medidos en ms). En esta prueba, se han tomado los dos casos mejor (secuencial) y peor (implosión). Se puede apreciar como los retardos son mayores que los reflejados en la Tabla 32, aunque aún bastante menores que la resolución DNS.

	Secuencial 1x100	Implosión 100x1	Secuencial 1x200	Implosión 200x1
Surrogate 1	13.0195	21.4079	12.97205	22.8182
Surrogate 2	9.7356	10.5569	9.56405	10.736
Surrogate 3	9.2362	10.5169	8.88535	10.24875

Tabla 34. Tiempo medio de respuesta con consulta a base de datos (en ms).

3.4.5.3.3. Tiempo medio en la redirección de contenido

En esta sección se va a determinar el tiempo medio que se introduce cuando un cliente solicita un contenido a un *surrogate* concreto (ya se ha producido la redirección por DNS) y se debe calcular el *surrogate* más adecuado que atenderá la petición (segundo nivel de redirección). En este caso, cabe diferenciar entre dos situaciones diferentes:

- En el primer caso, el *surrogate* inicial contactado se encuentra en buenas condiciones de carga, por lo que se le indica al cliente que se conecte al mismo *surrogate* para acceder al contenido solicitado. Esta situación se corresponde con las pruebas 1x100 y 1x200 reflejadas en la Tabla 35 .
- En el segundo caso, el *surrogate* se encuentra cargado y la petición debe ser atendida por otro *surrogate*. En esta situación, el módulo Redirector solicita a los *surrogates* que realicen un *ping* al cliente para obtener medidas de tiempo de respuesta y distancia en saltos. Esto, lógicamente, supone un incremento del tiempo de respuesta. En este caso, los tiempos medios obtenidos se reflejan en la Tabla 35 para 100 peticiones concurrentes.

En el primero de los casos se puede apreciar que los tiempos medios de espera dependen del *surrogate* en cuestión, pero que en el peor de los casos (*surrogate* 1), están alrededor de los 100 ms. Considerando los tiempos medios de descarga de una página simple obtenidos en la sección anterior (alrededor de 10 ms), se puede concluir que el retardo introducido está entre 60 y 80 ms. De forma similar a la resolución DNS, se trata un retardo casi imperceptible para el cliente.

Los resultados cambian en el segundo caso. El tiempo de respuesta aumenta hasta los 500 milisegundos aproximadamente. Esto es lógico si se piensa en que se realizan varios intercambios de información entre el Redirector, los *surrogates* y el cliente. La mayor parte de este tiempo es debida a la realización de *pings* desde los *surrogates* hacia el cliente.

La cuestión que cabe plantearse es si este retardo, que empieza a ser considerable, es asumible por el cliente. Posiblemente para ofrecer servicio web únicamente no parece muy adecuado. Sin embargo, para un servicio de streaming de video, si parece adecuado. Esto requiere un ancho de banda elevado y preservar unas comunicaciones entre cliente y *surrogate* adecuadas, para que la calidad de la reproducción sea baja y con cortes. Esto se describirá en la sección 4.3 dedicado al servicio de streaming.

	Caso 1 **1x100**	**Caso 2** **1x100c**	**Caso 1** **1x200**
Surrogate 1	101.2671	520.2214	96.9026
Surrogate 2	68.3909	500.988	67.4194
Surrogate 3	81.9514	378.5787	74.67285

Tabla 35. Tiempo medio en la redirección de contenido (en ms).

3.4.6. Comparación con otros modelos de simulación

Una vez realizado el análisis de la implementación, es necesario comparar los resultados obtenidos con los modelos analítico y de simulación, disponibles y descritos en las secciones 3.2 y 3.3. Asimismo, se realizará una comparación de la implementación desarrollada con otras implementaciones disponibles. La finalidad es doble:

- Por un lado, permite validar el modelo analítico propuesto y el modelo de simulación, y establecer en qué medida las hipótesis asumidas en dichos modelos eran válidas o razonables.
- Por otro lado, la comparación con otras implementaciones propuestas permite validar la propia implementación, detectar las propiedades de unos y otros y establecer en qué medida se complementan.

Es importante destacar que se debe ser muy cuidadoso a la hora de comparar y evaluar resultados entre dos sistemas diferentes (en nuestro caso modelos e implementaciones distintos), ya que muchas veces las hipótesis iniciales son diferentes y los resultados solamente son comparables en un determinado rango de valores o situaciones. De hecho, las variantes en los resultados muchas veces permiten detectar las diferencias en el diseño y en la implementación, sin que necesariamente ninguno de los dos sea erróneo.

3.4.6.1. Comparación con el modelo analítico

En esta sección se realizará una comparación de la implementación de la CDN desarrollada con el modelo analítico propuesto y analizado en la sección 3.2. En dicho modelo analítico se proponía un modelo de CDN y se establecía una ecuación que permitía obtener el tiempo de respuesta, que es el parámetro fundamental en el análisis del rendimiento de una CDN y, por ello, se le presta una especial atención es esta tesis. El modelo analítico permitía dividir el tiempo de respuesta global de dos formas diferentes desde dos puntos de vista:

- Como el tiempo de respuesta de los *surrogates* y el servidor origen.
- Como el tiempo de respuesta asociado al tiempo de ida y vuelta (RTT) y tiempo de proceso en los servidores (tanto *surrogates* como servidor origen).

A partir de esta descripción del tiempo de respuesta, se realizaban diversas simulaciones donde la variable independiente era la probabilidad de acierto *p*, que indicaba en qué medida un cliente (clúster de clientes) contactaba bien con un *surrogate* cercano o bien con el servidor origen. Si *p=0*, el cliente contactaba con el servidor origen, mientras que si *p=1* el cliente contactaba con un *surrogate* cercano. Desde el punto de vista de la arquitectura de funcionamiento de una CDN, se está asumiendo un algoritmo de

redirección inmediato (tiempo de respuesta nulo) capaz de indicar a un cliente en qué medida debe contactar con un *surrogate* o contactar directamente con el servidor origen.

En la implementación desarrollada, por el contrario, no es posible hacer simplificaciones y se introducen entidades adicionales, como es el módulo Redirector (integrado dentro de un servidor DNS), capaz de ejecutar un algoritmo de redirección. En esta implementación existe una carga de procesamiento de dicho algoritmo nada despreciable, como se ha descrito en la sección 3.4.5.3.

Por otro lado, el modelo de comunicación en la implementación es más complejo, ya que se establecen más canales de comunicación que en el modelo analítico:

- El cliente se comunica con el módulo Redirector (a través del servidor DNS) y con el *surrogate* correspondiente.
- El cliente no contacta con el servidor origen, sino con el *surrogate*. Es el *surrogate* quien, realizando funciones de servidor *proxy cache*, contacta con el servidor origen, obtiene el contenido solicitado y se lo proporciona al cliente. En la implementación web se ha asumido una replicación al 100% para simplificar el análisis, pero en la implementación de streaming se considerará una replicación parcial (véase sección 4.3)
- El *surrogate*, si se encuentra excesivamente cargado, es capaz de redirigir al cliente a otro *surrogate*, introduciéndose así un segundo nivel de redirección no contemplado en el modelo analítico, pero si empleado de forma extendida en diferentes modelos reales de CDNs.

Por todo ello, la comparación entre ambos modelos se debe hacer en casos muy simplificados de la implementación, aquellos escenarios de implementación de la CDN propuesta donde: (i) se pueda obviar o no computar la redirección DNS y (ii) no se realice un segundo nivel de redirección en el *surrogate* contactado. En este escenario, se pueden calcular los parámetros característicos del tiempo de respuesta representados en la expresión (E 3.3) descrita en la sección 3.2.2.

$$R = p \cdot \left[N \cdot \tau_p + \frac{1}{\mu_p - \lambda_p} \right] + (1-p) \cdot \left[N \cdot \tau_s + \frac{1}{\mu_s - \lambda_s} \right] \qquad \text{(E 3.3)}$$

Si asumimos una replicación total de contenido en todos los *surrogates* (p=1) el segundo factor de la expresión E3.3 desaparece y podemos centrarnos en calcular los valores de los parámetros τ_p, μ_p y λ_p para cada uno de los *surrogates*. Si bien τ_p se puede calcular relativamente fácil a través de un *ping*, los valores de μ_p y λ_p dependen del número de clientes y su concurrencia, por lo que se ha generado una gráfica de rendimiento para cada uno de los *surrogates* para diferentes peticiones mediante la herramienta *httperf* [WWW_Hperf].

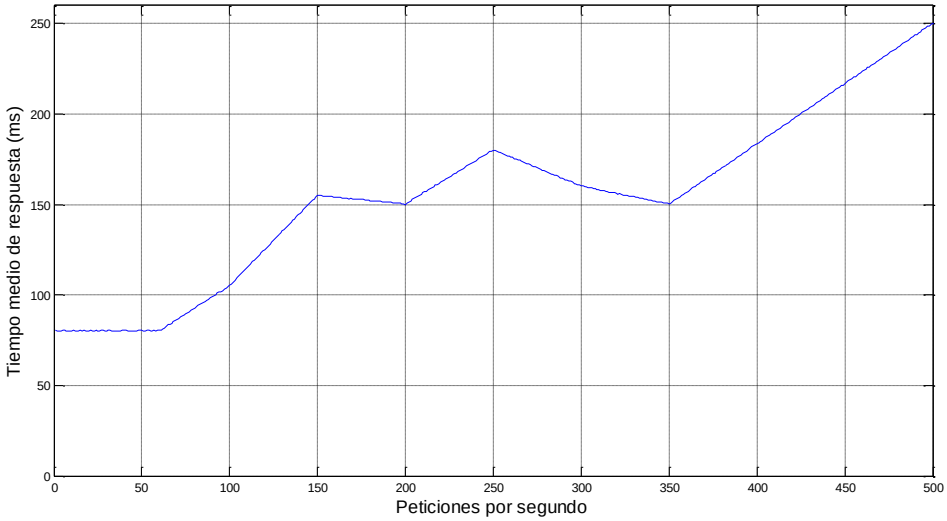


Figura 106. Tiempos de respuesta (surrogate 1).

Los resultados se muestran en la Figura 106 para el *surrogate* 1. Se puede comprobar cómo el tiempo de respuesta es prácticamente constante hasta un cierto número de peticiones concurrentes, donde empieza a crecer de una forma un tanto oscilante. Aumentando el número de peticiones, se observa que llega un momento en que el tiempo de respuesta crece lineal y significativamente. Esto correspondería al tramo donde el *surrogate* se puede considerar cargado.

A partir de estas gráficas se podrían calcular los parámetros τ_p, μ_p y λ_p para cada uno de los *surrogates* de una forma relativamente sencilla. No obstante, cabe mencionar que el modelo M/M/1 no es totalmente aplicable a un servidor real y, en nuestro caso, el proceso de llegada de las peticiones web no sigue una distribución de Poisson. No es objetivo de esta tesis establecer las diferencias entre un servidor real y un proceso M/M/1, sobre todo con las CPUs actuales que soportan varios núcleos internamente, pero a nivel teórico se sigue empleando esta aproximación para hacer una estimación de los retardos. Es decir, se sigue un proceso inverso al descrito en esta sección: los parámetros τ_p, μ_p y λ_p son conocidos a priori y se hace una estimación del retardo medio (en ocasiones omitiendo el parámetro τ_p), para posteriormente comprobar si se cumplen dichas estimaciones con la realidad.

3.4.6.2. Comparación con el modelo de simulación

En esta sección se realiza una comparación de la implementación de la CDN desarrollada con el modelo de simulación analizado en la sección 3.3. En dicho modelo de simulación se ha propuesto un modelo de CDN más complejo que el modelo

analítico, introduciendo ciertos aspectos más realistas como la redirección DNS, los patrones de tráfico parametrizables (típicamente siguiendo una distribución Pareto o Zipf en entornos web), así como escenarios de red variables mediante generadores de topologías. El modelo de simulación ha permitido obtener un gran número de resultados y gráficas para el análisis, como la carga media en los *surrogates*, el tiempo medio de respuesta, la distribución de acceso, el porcentaje de replicación, caching estático y dinámico, hit ratio, byte hit ratio e incluso el efecto *flash-crowd* (véase sección 3.3.5)

En el modelo de simulación propuesto se introducía, a diferencia del modelo analítico, el módulo Redirector, capaz de ejecutar un algoritmo de encaminamiento, de forma similar a la implementación. Sin embargo se asumía en el modelo de simulación, por simplicidad, que la carga de procesamiento de dicho algoritmo era nula, si bien si se consideraba el tiempo en el envío de mensajes desde el cliente al Redirector. En la implementación real se puede apreciar que esta carga de procesamiento sí es significativa (véase sección 3.4.5.3) en el siguiente sentido:

- En el primer nivel de redirección (DNS), los valores de los tiempos de respuesta son moderados. Su impacto en el tiempo de respuesta global no es trascendente. Desde este punto de vista, la asunción para el modelo de simulación es válida.

- En el segundo nivel de redirección (*surrogate*), los valores de los tiempos de respuesta son más elevados, sobre todo si el *surrogate* está cargado. En este caso, su impacto en el tiempo de respuesta global no resulta para nada despreciable. Este segundo nivel de redirección no se ha implementado en el modelo de simulación, pero habría que tenerlo en cuenta en una futura extensión (véase sección 5).

Por otro lado, el modelo de comunicación en el modelo de simulación es más complejo que el modelo analítico, y se establecen dos canales de comunicación:

- El cliente contacta con el Redirector y con el *surrogate* correspondiente.

- El cliente no contacta con el servidor origen (modelo analítico), sino con el *surrogate*. Es el *surrogate* quien, realizando funciones de servidor *proxy cache*, contacta con el servidor origen, obtiene el contenido solicitado y se lo proporciona al cliente.

Este comportamiento es similar en la implementación de la CDN. Si asumimos un escenario con replicación total, podemos comparar ambos modelos. Ya que la topología de red en la implementación no es fácilmente modificable, se ha modificado la topología de red en el modelo de simulación, generando dos topologías equivalentes desde el punto de vista de los clientes, los *surrogates* y los enlaces de red entre ellos. Nótese que, en el modelo de simulación, los coeficientes *f, g* y *h* no están implementados, por lo que hay que establecer unos escenarios en cierto modo similares. Para ello, en el modelo de simulación se tomará una distribución de acceso del tipo [90

5 5] (véase sección 3.3.5.3) de forma que la mayor parte de las peticiones se encaminen al *surrogate* de la misma subred. De igual forma, se tomarán unos patrones de tráfico que no sobrecarguen a los servidores. Dado que el tiempo de carga en el modelo de simulación se ha tomado nulo, se tomarán los valores de la Tabla 33 como media y se introducirá una función de distribución normal. Los resultados de los tiempos medios de respuesta obtenidos se ilustran en la Tabla 36. Los resultados de la implementación se reflejan en la Tabla 32 y en la Tabla 33, mientras que para el modelo de simulación se han realizado los cambios correspondientes y se ha ejecutado una nueva simulación.

Se puede apreciar como los valores son similares, si bien en el modelo de simulación son ligeramente inferiores ya que se asume una serie de simplificaciones dentro del funcionamiento de NS-2. El efecto de la carga sobre los *surrogates* en la implementación es también diferente al modelo de simulación, ya que en este último caso, el uso de CPU y su efecto en el retardo no está simulado. Tampoco está simulado el hardware de *surrogate*, por otro lado, por lo que los valores de simulación son más parecidos entre servidores.

	Implementación 1x100c	**Implementación 1x200c**	**Simulación 1x100c**	**Simulación 1x200c**
Surrogate 1	76.2234	78’4561	56.4521	63.8713
Surrogate 2	72.8749	74’925	58.2110	62.9210
Surrogate 3	72.5647	74’4773	54.9863	60.3411

Tabla 36. Comparación implementación y modelo de simulación (ms).

3.4.6.3. Comparación con otras implementaciones

En esta sección se realizará una comparación de la implementación de la CDN desarrollada con otras implementaciones reales disponibles. Tal y como se mencionaba en la introducción de esta tesis, existen actualmente escasas implementaciones libres y abiertas, como Globule [Pie_01] [Pie_03], CoDeen [Wan_04] u OpenCDN [WWW_Ope]. Desgraciadamente, estos proyectos han sido abandonados por varios motivos, fundamentalmente porque fueron creados desde una base puramente de investigación que posteriormente no tuvo una salida comercial. No obstante algunos proyectos, como Globule, han evolucionado hacia servicios en nube (*cloud computing*), aunque siguen ofreciendo el software de CDN (aunque sin soporte). En cualquier caso, Globule es el proyecto de CDN libre mejor documentado, tanto en su instalación como configuración, y por ello se ha realizado una comparación entre implementaciones.

Globule es una CDN orientada a ofrecer servicios de distribución web mediante la replicación de objetos web entre servidores. Las principales características son las siguientes [WWW_Glob]:

- ***Replicación***: los sitios web (o parte de ellos) pueden ser replicados entre múltiples servidores, aunque estos servidores pertenezcan a entidades (personas, empresas) distintas.
- ***Redirección de usuarios***: los clientes que acceden a un sitio web son redirigidos automáticamente a una de las réplicas. Esto se puede realizar empleando redirección HTTP o DNS. Globule soporta varias políticas de que permiten decidir a qué replica redirigir a un cliente.
- ***Tolerancia a fallos***: cada servidor Globule comprueba periódicamente la disponibilidad de aquellos otros servidores réplica. En caso de que uno de estos servidores esté 'caído' o mal configurado, deja de redirigir a los clientes a dicha réplica hasta que se recupere. Globule también permite la creación de copias de seguridad (*backups*) del servidor origen de tal forma que el sitio web pueda funcionar incluso si el servidor origen está caído.
- ***Monitorización***: Globule ofrece a los administradores la capacidad de monitorizar el comportamiento del sistema en varios aspectos. En primer lugar, los ficheros de *log* de las peticiones efectuadas a las réplicas se envían de vuelta al servidor origen para generar un fichero global de *log*. En segundo lugar, Globule permite añadir una *cookie* con cada documento servido, de forma que contiene información acerca de cómo se atendió la petición. En tercer lugar, es posible obtener estadísticas internas sobre los patrones de uso y procesarlos mediante filtros configurables.
- ***Replicación adaptiva***: existen muchas formas mediante las cuales un documento (objeto web) puede ser replicado, y la actualización de estos objetos debe ser considerada. A diferencia de otros sistemas, Globule no presupone que haya una sola política que sea óptima en todos los casos [Pie_02]. En lugar de ello, soporta múltiples políticas, y periódicamente comprueba para cada documento qué política ofrecerá un mejor rendimiento.
- ***Replicación dinámica de documentos***: Globule no sólo es capaz de replicar documentos, sino también *scripts* que se pueden ejecutar en las réplicas para generar contenido, como los *scripts* PHP. Si estos últimos acceden a una base de datos (MySQL), entonces Globule también es capaz de *cachear* las peticiones a la base de datos para incrementar el rendimiento [Siva_05].
- ***Servidor de configuración***: El módulo GBS (*Globe Broker System*) es un sitio web donde los usuarios de Globule se pueden registrar, anunciarse y decidir si desean replicar el contenido de otro servidor. En este caso, los ficheros de configuración se generan de forma automática, de forma que los usuarios no necesitan leer la mayor parte de la documentación.

Adicionalmente a todas estas características, dentro del proyecto Globule se introdujeron parcialmente (como resultado de investigaciones no finalizadas) los siguientes aspectos:

- ***Estimación de la latencia***: Globule pretendía desarrollar un método ágil para estimar la latencia entre dos nodos cualesquiera en Internet. En lugar de enviar una gran cantidad de mensajes entre ambas máquinas, su método estaba basado en un número muy reducido de medidas [Szy_04]. Dicha investigación culminó en un prototipo que finalmente no terminó de integrarse dentro del sistema Globule.
- ***Ubicación de réplicas***: En base a la estimación de las latencias, Globule es capaz de analizar la ubicación de los clientes, y proporcionar un conjunto de localizaciones donde se pueda ubicar de forma óptima servidores réplica [Szy_05].
- ***Predicción de flash-crowd y gestión proactiva***: la idea básica es desarrollar técnicas que permitan identificar las primeras fases de un efecto *flash-crowd* (tasa de llegada de las peticiones, referencia en un BBS de impacto, etc.), de forma que se replique de manera proactiva un sitio web para satisfacer de manera eficiente futuras peticiones [Bar_05].

Desde el punto de vista de la implementación, Globule funciona únicamente sobre Apache 2.0.x, y no está portada a versiones anteriores (1.3.x) ni posteriores (2.x).

Una vez vistas las principales características de Globule, en la Tabla 37 se comparan éstas con las de nuestra implementación propuesta en esta tesis.

Característica	CDN implementada	Globule
Modelo de redirección	DNS/HTTP ,Basado en agentes	DNS/HTTP , Basado en medidas de red (BGP, configurable)
Caching	Estrategia global	Estrategia por documento [Pie_02]
Monitorización	Logs, Envío SNMP periódicos, Medidas de red	Logs , Cookies
Tolerancia a fallos	Detección de servidores caídos por agentes SNMP periódicos	Detección de servidores caídos a través de Apache
Configuración	Servidor origen incorpora el proceso de configuración	CGB, réplicas deciden unirse a la CDN
Flash-crowd	Predicción mediante agentes (carga)	No disponible
Replicación	Estático (FTP, HTTP)	Estático (HTTP), Dinámico (GlobeDB [Siv_05b])
Ubicación de surrogates	Manual	Manual (Hotzone[Szy_05])

Tabla 37. Comparación Globule y CDN implementada.

3.4.7. Conclusiones parciales de la implementación web

La implementación de CDN descrita y analizada en esta sección permite ampliar significativamente el modelo analítico y el modelo de simulación descritos en las primeras secciones del capítulo 3, introduciendo todos los factores reales (o casi todos) que caracterizan una CDN; de esta forma, se puede analizar mejor su comportamiento desde diversos puntos de vista, aunque fundamentalmente el aspecto que se ha estudiado con mayor profundidad es el correcto funcionamiento del algoritmo de redirección, teniendo en cuenta parámetros como la carga media en los servidores (monitorización de *surrogates*) y el tiempo medio de respuesta (monitorización de clientes). Adicionalmente, la implementación propuesta también se ha comparado con el modelo analítico y el modelo de simulación previamente analizados, así como con otra implementación real (Globule). Los principales resultados de este capítulo se han descrito en dos publicaciones propias [Mol_04] [Mol_06].

En nuestra implementación se ha estudiado la redirección DNS bajo un funcionamiento normal; la caída y recuperación de un *surrogate* y variaciones abruptas en la carga de dichos *surrogates*. Posteriormente se ha analizado la redirección basada en contenidos y, finalmente, el estudio de los tiempos de respuesta en las consultas DNS, consultas a base de datos y redirecciones HTTP. De este trabajo, se pueden derivar las siguientes conclusiones:

- El algoritmo de redirección está basado en la captura inteligente de parámetros de funcionamiento obtenidos por agentes distribuidos en los *surrogates*. Los coeficientes *f* permiten obtener de manera periódica el estado de los servidores para determinar sus condiciones de carga. Esta monitorización periódica permite, adicionalmente, conocer si algún servidor está caído y reaccionar en consecuencia. Los coeficientes *g* permiten obtener, por otro lado, el estado de la red para determinar las rutas menos cargadas. Finalmente, los coeficientes *h* fusionan los coeficientes <*f,g*> asignando mayor peso a uno u a otro dependiendo de la configuración o servicio ofrecido.

- El algoritmo de redirección es capaz de adaptarse de forma dinámica al estado de carga de los *surrogates* en cada instante, y la probabilidad de enviar una petición a un servidor se basa no sólo en sus condiciones de carga de forma independiente, sino también en las condiciones de carga de forma relativa al resto de *surrogates*, con la finalidad de disponer de la decisión óptima a nivel global.

- Existen umbrales superiores e inferiores parametrizables en las condiciones de carga de un *surrogate* (función *k[]*). El umbral superior impide que un servidor tenga excesivas peticiones (evita implosiones), mientras que el umbral inferior permite aumentar rápidamente el número de nuevas conexiones (permite un rápido crecimiento hacia el punto de trabajo óptimo).

- En la redirección basada en contenidos (redirección HTTP) los valores de los coeficientes *y*, *g* y *h* únicamente se calculan cuando algún *surrogate* está por debajo del umbral inferior de *f* (parametrizable) y un cliente local le solicita un contenido web. Por encima de dicho umbral no se realizan estos cálculos, por sencillez, eficiencia y escalabilidad, y las peticiones se redirigen al *surrogate* local, que será el que mejor conectividad en términos de red ofrezca. Por debajo del umbral, es necesario estimar la mejor subred no local.

- Si bien las medidas de red son típicamente oscilantes, en las gráficas obtenidas se aprecia una excesiva oscilación de los coeficientes *g* cuando los valores de los coeficientes *y* son parecidos. Podría resultar útil analizar alguna especie de 'ciclo de histéresis' para evitar tanta fluctuación y variabilidad de la red, lo que permitiría diferenciar mejor el servidor óptimo.

- En cuanto a los tiempos de respuesta obtenidos, se observa que la mayor parte del retardo está motivado por el algoritmo de redirección, cuyo procesado requiere más de 10 veces el tiempo de envío de una página web (para páginas web de pequeño tamaño usadas en esta tesis). No obstante el tiempo global es inferior al medio segundo y es asumible por el cliente.

Pese a que los resultados obtenidos en esta sección son bastante clarificadores en cuanto al funcionamiento de una CDN, es importante centrarse también en las limitaciones o posibles mejores de la implementación propuesta.

En nuestra implementación, para el cálculo de los coeficientes *f*, se emplea un esquema con memoria, pero sólo a partir del intervalo de monitorización anterior. Dado que se encuentra disponible, se puede usar todo (o parte) del histórico de datos de monitorización para alimentar al algoritmo de redirección, o bien para mantenimiento y así detectar de manera proactiva servidores con problemas (por ejemplo, aquellos que caen varias veces de forma anómala).

Por otro lado, para el cálculo de los coeficientes *g* se puede usar información ya disponible de un cliente para asociarlo a clientes de la misma subred (o cercanas). El histórico de datos de red también se puede usar para detectar la evolución de la red; si cayera un enlace también se podría emplear *multihoming* en los *surrogates*.

En este capítulo se ha analizado tanto un modelo analítico como un modelo de simulación y una implementación de CDN para servicios web. Si bien los resultados mostrados en su estudio indican un comportamiento correcto de la CDN, es importante abordar otro tipo de servicios ofrecidos actualmente por la segunda generación de CDNs (véase sección 2.5.5), que es el servicio de streaming, con más requisitos en términos de ancho de banda y jitter que el contenido web. Esto afecta a las políticas de despliegue y configuración de una CDN y, en consecuencia, a su algoritmo de redirección. En el siguiente capítulo se abordará el estudio de una CDN para servicios de streaming, ampliando el modelo de simulación y la implementación.

4. DESCRIPCIÓN Y EVALUACIÓN DEL SERVICIO MULTIMEDIA

4.1. Planteamiento general y estructura

Como ya se ha descrito en capítulos anteriores, el objetivo principal de esta tesis es el estudio y análisis comparativo de los principales parámetros de funcionamiento de una CDN (retardo medio, carga del sistema, mecanismo de redirección, etc.) en entornos web y en servicios de streaming. Este estudio se ha realizado de una forma progresiva, de menor a mayor complejidad. En el capítulo 3 se ha hecho un análisis y estudio de una CDN con tráfico web, mientras que en este capítulo se analiza el tráfico multimedia ya que, por sus requerimientos de tiempo real y características de ancho de banda y *jitter*, no puede tratarse de igual forma que el tráfico web, y la configuración de la CDN y su sistema de redirección se ven afectados.

En este capítulo no existe un modelo analítico para el servicio de streaming análogamente a la sección 3.2, sino que se presentan directamente el modelo de simulación y la implementación. Esto es debido a dos motivos fundamentales. El primer motivo es que el modelo analítico presentado en la sección 3.2 es bastante generalista en el modelo de interacción cliente-servidor, por lo que debería ampliarse en su parametrización de variables para contemplar y distinguir entre servicios web y multimedia. Si bien el modelo analítico se está ampliando actualmente, como se presentará en las líneas futuras de investigación (véase sección 5), no lo ha hecho en esta diferenciación de servicios, sino en la capacidad de proporcionar una arquitectura de despliegue más realista con clústeres de *surrogates* y su capacidad de comunicación mediante mecanismos de caching colaborativos. El segundo motivo es que el modelo de simulación y la implementación son ya de por sí más complejos y realistas que el modelo analítico, por lo que se ha preferido comenzar directamente con estos dos y estudiar las implicaciones y diferencias entre servicios web y streaming en una CDN.

En la segunda parte (sección 4.2) se presenta un modelo de simulación de una CDN que introduce capacidades de streaming, a partir del modelo descrito en la sección 3.3 para servicios web. Por ello, se ha empleado nuevamente el simulador de redes de comunicaciones NS-2, y se ha adaptado y configurado para poder simular sesiones de streaming por parte de los clientes. A partir de este modelo, se han realizado numerosas simulaciones para evaluar parámetros de interés como son la carga media y los tiempos medios de respuestas, dependiendo del número de clientes, número de servidores, modo de funcionamiento, etc., como se describe en la sección 4.2.4.

En la tercera parte (sección 4.3) se presenta un prototipo de implementación de CDN. Nuevamente se ha partido del modelo de implementación web descrito en la sección 3.4, dotándolo de capacidades de streaming. De manera análoga, se ha hecho un especial hincapié en el algoritmo de redirección y los criterios que deben establecerse para garantizar un correcto funcionamiento cuando se desea ofrecer un servicio de streaming.

4.2. Modelo de simulación

4.2.1. Introducción

El objetivo principal de esta sección consiste en crear un modelo avanzado de simulación con el cual poder describir y estudiar servicios de streaming ofrecidos por una CDN, partiendo del modelo de simulación para servicio web descrito en la sección 3.3. Así pues, en la presente tesis se han desarrollado dos modelos de simulación de una CDN: (i) uno para contenido web descrito en la sección 3.3; y (ii) otro para contenido streaming que se describirá en esta sección.

Los principales parámetros de análisis en este modelo de simulación serán:

- La latencia media percibida por los clientes de la CDN.
- La carga media en los servidores o *surrogates*.

Para ambos parámetros, se variará el número de clientes, el número de servidores, el porcentaje de replicación, etc., y se obtendrán los resultados de la simulación para poder analizarlos y/o compararlos. Como se puede apreciar, el estudio es análogo al llevado a cabo para servicios web, lo que permitirá una mejor comparación entre ambos modelos para identificar más fácilmente las semejanzas y las diferencias entre ambos.

Adicionalmente, y con el fin de validar también los resultados proporcionados por el modelo analítico, se han realizado diferentes simulaciones donde se ha evaluado tanto el tiempo medio de respuesta como la carga de los servidores en escenarios con y sin CDN, de tal forma que quede reflejada la utilidad de dicha CDN.

En relación a la comparación con otros simuladores existentes, se comparará nuestro modelo de simulación con CDNsim [WWW_CDN2] y CDN Simulator [WWW_CDN1], de forma similar a la sección 3.4.6.2. No obstante, cabe mencionar que, en este caso, las condiciones de comparación entre ambos modelos son incluso más diferentes que en la sección 3.4.6.2, ya que el servicio de streaming sólo está parcialmente soportado en estos simuladores, y no existe ninguna publicación disponible donde se haya estudiado este tipo de servicio. Por ejemplo, CDN Simulator está diseñado para acceso web, e incluso la redirección se realiza mediante este mecanismo (HTTP). En este sentido, resulta complicado comparar este esquema con nuestro modelo propuesto en esta sección, que emplea protocolos estándares de streaming como son RTP y RTSP. Si bien el streaming mediante HTTP es posible, el funcionamiento es diferente, lo que dificulta la comparación entre modelos. No obstante, esto se describirá con mayor detalle en la sección 4.2.5.

4.2.2. Diseño y jerarquía de clases

Para la simulación de la CDN con capacidad de streaming se ha partido del modelo diseñado en el apartado anterior, al cual se ha añadido la funcionalidad necesaria para la transmisión de tráfico con requerimientos de tiempo real. Entre otros, se ha implementado un módulo en C++ que simule el funcionamiento del protocolo RTSP (*Real Time Streaming Protocol*), que es el protocolo de *streaming* utilizado en el estudio. A nivel de aplicación, y para la transmisión de los flujos multimedia, se ha hecho uso de las implementaciones de los protocolos RTP (*Real Time Transport Protocol)* y RTCP (*Real Time Control Protocol*) ya incluidas en la distribución del NS-2 y a las cuales se les ha introducido ciertas modificaciones (mejoras).

A continuación se describe el funcionamiento básico de la CDN con capacidad de streaming que se ha diseñado con las siguientes características:

- Un cliente accede a recursos *multimedia* ubicados en *surrogates* de acuerdo a un *determinado patrón de tráfico*, que se puede particularizar para cada cliente. Los parámetros que lo definen son: instante de inicio de sesión, número de recursos que solicita el cliente y distribución que define el intervalo entre peticiones de objetos multimedia.
- Cuando un cliente desea obtener un recurso *multimedia*, el primer paso consiste en *acceder a su Redirector* asociado. El Redirector, como en el modelo anterior, es un elemento encargado de llevar a cabo el proceso de redirección del cliente a través del servicio DNS. Esta decisión, en el caso de la CDN de streaming, se ha hecho algo más compleja. Más concretamente, depende del modo de funcionamiento de la CDN. Si el modo es estático (*static*), el Redirector sólo considera aquellos *surrogates* que dispongan del objeto multimedia solicitado por el cliente. Entre estos, descarta aquellos *surrogates* cuyo número de sesiones activas iguale a un umbral máximo y, de los restantes, selecciona el más próximo al cliente en términos de RTT. En el modo dinámico (*dynamic*), el Redirector proporciona directamente al cliente la dirección del *surrogate* más cercano a éste (siempre que no iguale el número máximo de sesiones activas permitidas). Debido a ello, es posible que, en este último modo, el *surrogate* carezca del objeto multimedia solicitado en caché, en cuyo caso lo descarga del servidor origen mediante un esquema que simula el servicio de transferencia de ficheros FTP (*File Transfer Protocol*).
- El cliente *accede al surrogate* indicado para obtener el recurso multimedia. Una vez el cliente recibe del Redirector la dirección del *surrogate* con el que debe contactar, establece con éste una conexión TCP y, por encima de ésta, la conexión de nivel de aplicación RTSP. El cliente, entonces, inicia el envío de los comandos de control (OPTIONS, DESCRIBE, SETUP, PLAY) propios de este

protocolo de streaming, que derivan en el establecimiento de una sesión RTP/RTCP a través de la cual el cliente obtiene el objeto multimedia solicitado.

- El cliente interactúa con el *surrogate* para detener la reproducción o incluso abandonarla. Cuando un cliente accede a un recurso multimedia en la modalidad de *streaming*, es muy probable que éste detenga la reproducción en reiteradas ocasiones o que incluso la termine definitivamente. En el modelo de CDN de streaming que se ha implementado es posible definir el patrón o frecuencia de generación de mensajes de tipo PAUSE por parte del cliente, así como la probabilidad de que el usuario abandone una sesión RTSP en curso mediante el correspondiente mensaje de TEARDOWN.

Para entender cómo se ha creado el modelo de simulación en C++, tal y como ya se hizo en la sección 3.3.3, resulta básico crear un árbol de clases en el que queden reflejadas de forma visual las relaciones entre las diferentes clases. También es importante conocer las funciones y parámetros de los ficheros generados, así como las características que éstos implementan.

El modelo de simulación de la CDN de streaming está formado por un total de 10 clases nuevas implementadas, relacionadas entre sí dentro de la estructura interna de NS-2 según se muestra en la Figura 107.

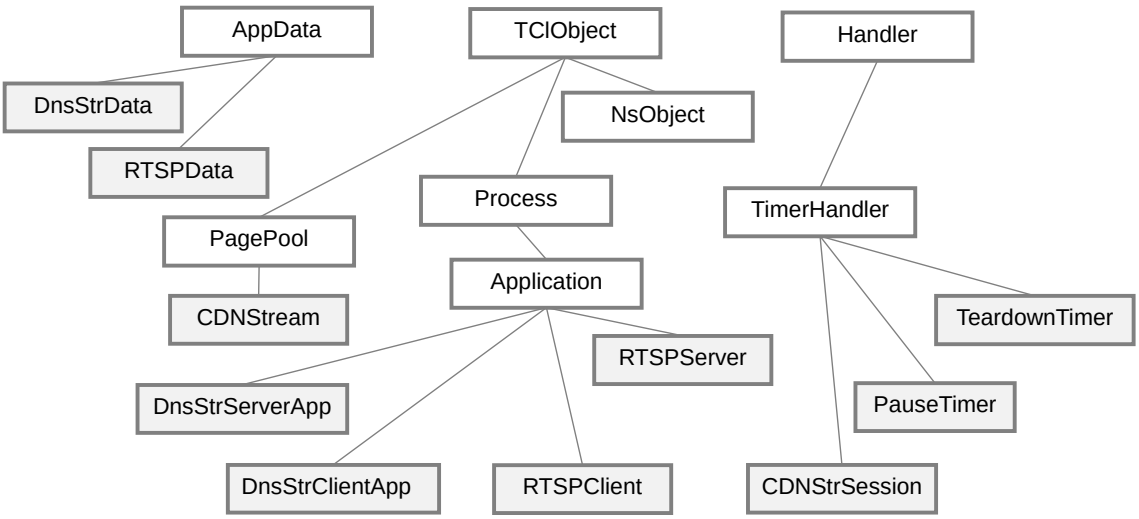


Figura 107. Diagrama de clases del modelo de simulación CDN de streaming.

A continuación se describirá brevemente la función de cada una de estas clases y su relación con la clase de NS-2 a la que extienden, de forma que permita entender la motivación de este diseño.

La clase *DnsStrData* hereda de *AppData* que constituye, como ya se ha descrito anteriormente (sección 3.3.3), una interfaz genérica que se utiliza como base para la definición de nuevas unidades de datos de nivel de aplicación (ADUs). Así, la clase

DnsStrData representa una estructura con diversos campos que facilita la comunicación entre el cliente y el Redirector (véase Figura 109). La parte cliente viene representada por la clase *DnsStrClientApp*, responsable de contactar con el Redirector (redirección DNS) para obtener la dirección del *surrogate* del cual obtener el recurso multimedia. Por otro lado, la clase *DnsStrServerApp* representa la capa de aplicación del Redirector, implementando los mecanismos de redirección del cliente.

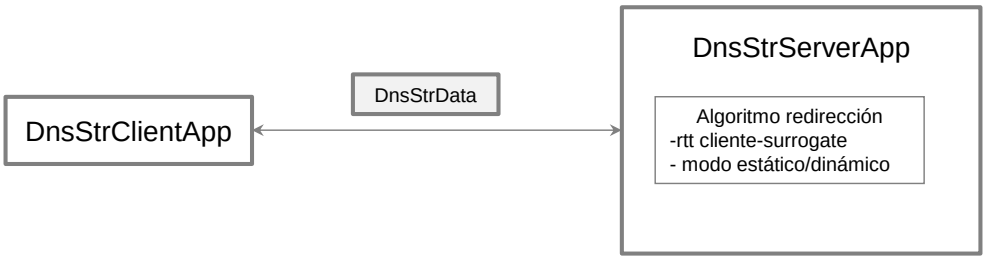


Figura 108. Proceso de redirección de un cliente en la CDN de simulación (modo streaming).

Nótese el paralelismo con las clases desarrolladas en la CDN de la sección 3.3 (modo web). En este caso, el tipo de información intercambiada (*DnsStrData*) es ligeramente diferente en el modo streaming, ya que hay que tener en cuenta el tipo de objeto solicitado. En el modo web se supuso, por simplicidad, una replicación total en los *surrogates*, por lo que el objeto web solicitado no era importante a la hora de tomar la decisión de redirección, y podía ser omitido en la clase *DnsData*.

En este modo de streaming el objeto multimedia, debido a su potencial tamaño (varios órdenes de magnitud superiores a un objeto web), es significativo en la decisión de redirección y debe incluirse como campo del mensaje, tal como muestra la Figura 109.

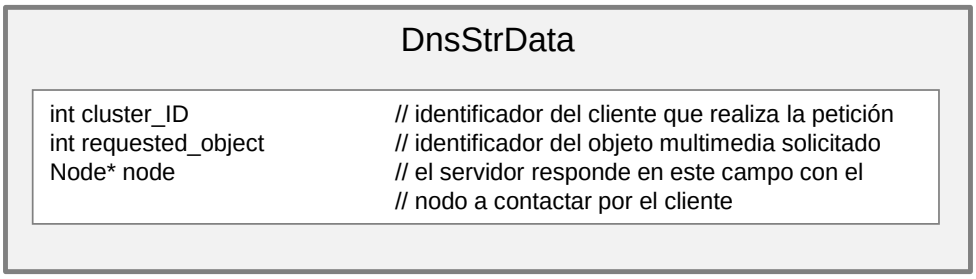


Figura 109. Estructura de datos de la clase DnsStrData.

En este intercambio de mensajes *DnsStrData*, la clase *DnsStrServerApp* es la más importante al implementar el algoritmo de redirección a partir de los dos primeros campos del paquete *DnsStrData* (identidad del cliente y objeto multimedia solicitado). A partir de esta información se calcula el nodo del *surrogate* (o servidor de streaming RTSP) más apropiado. Para ello, por cada cliente, se mantiene una ordenación de las

réplicas de menor a mayor RTT y, por cada *surrogate*, se mantiene un registro de los objetos multimedia que almacena. Dependiendo del modo de funcionamiento de la CDN, el criterio de selección de *surrogate* varía, verificándose siempre que un servidor de streaming RTSP no supera un número máximo de clientes (sesiones) a los que puede dar servicio simultáneamente. Este número máximo es un parámetro configurable en la clase *DnsStrServerApp*.

El algoritmo de redirección dispone de dos modos de funcionamiento, que determina el criterio de selección del *surrogate* cuando se recibe la petición de un cliente. Los modos admitidos son:

- **Modo estático**. El Redirector devuelve, de entre aquellos *surrogates* que disponen del objeto multimedia en caché, aquel más próximo al cliente.
- **Modo dinámico**. El Redirector devuelve la dirección del *surrogate* más cercano al cliente ignorando el objeto multimedia solicitado por éste.

El primer modo de funcionamiento (modo estático) es el esquema más sencillo y presupone que el estado de la red entre cliente y *surrogate* es aceptable (no afectará significativamente al retardo), por lo que el cliente recibe directamente el contenido multimedia sin necesidad de contactar con el servidor origen. Este modo puede resultar válido cuando hay un gran número de servidores distribuidos en la red, de forma que la probabilidad de contactar con un *surrogate* cercano en términos de red (RTT) con dicho objeto multimedia sea relativamente elevada. Cuando el número de servidores es escaso o el número de objetos multimedia es elevado, es preferible redirigir a un cliente a un *surrogate* cercano, aunque dicho objeto multimedia no esté disponible. El primer cliente tendrá un retardo elevado, pero los siguientes clientes ubicados en el mismo clúster verán reducido el tiempo de respuesta, con lo que el retardo medio se verá minimizado.

El punto anterior se clarifica mediante el ejemplo representado en la Figura 110 y en la Figura 111. En la primera, el cliente C_1 solicita el objeto multimedia *M*. El Redirector, en el modo estático, consciente que dicho objeto no está disponible en un *surrogate* cercano (S_B), redirige al cliente hacia S_D, que es el *surrogate* más cercano que dispone del objeto multimedia solicitado. Supongamos que el servidor central está topológicamente lejano, por lo que el valor de RTT_3 es mucho mayor que el de RTT_1, RTT_2 y RTT_4. Sucesivas peticiones de otros clientes ubicados en el mismo clúster (C_2) también atravesarán la ruta P_1 con un *RTT* global medio expresado por:

$$RTT^{M}_{estatico} = RTT^{M}_{1} + RTT^{M}_{2} + RTT^{M}_{4} \qquad \text{(E 4.1)}$$

Nótese que RTT_1, RTT_2 y RTT_4 son variables con el tiempo, por lo que hay que tomar su valor medio.

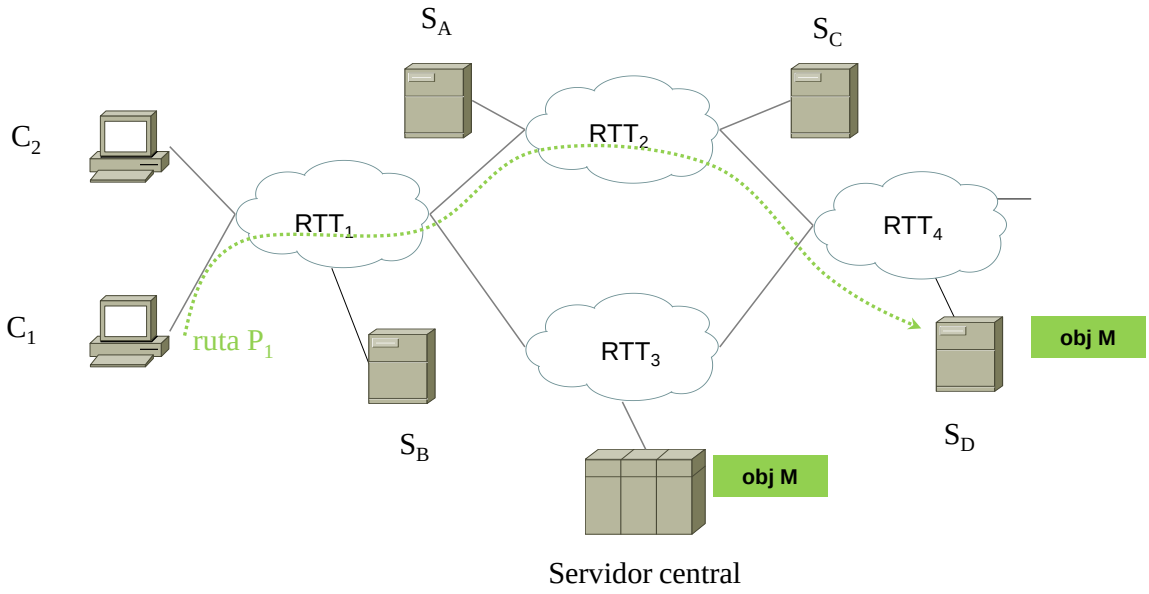


Figura 110. Modo de funcionamiento estático en el algoritmo de redirección.

En el modo dinámico, representado en la Figura 111, el cliente C_1 es redirigido al *surrogate* S_B, y obtiene el objeto multimedia de éste. Dado que no está almacenado en caché, el *surrogate* S_B debe solicitar previamente el objeto multimedia M del servidor origen y posteriormente servirlo al cliente. Sucesivas peticiones de otros clientes accederían directamente al *surrogate* S_B sin necesidad de volver a contactar con el servidor origen. En este caso, el *RTT* global medio sería:

$$RTT^{M}_{dinámico} = \frac{1}{N}\left(RTT^{M}_{1} + RTT^{M}_{3}\right) + \frac{N-1}{N} RTT^{M}_{1} \qquad \text{(E 4.2)}$$

Siendo N el número de peticiones en el intervalo de tiempo de observación.

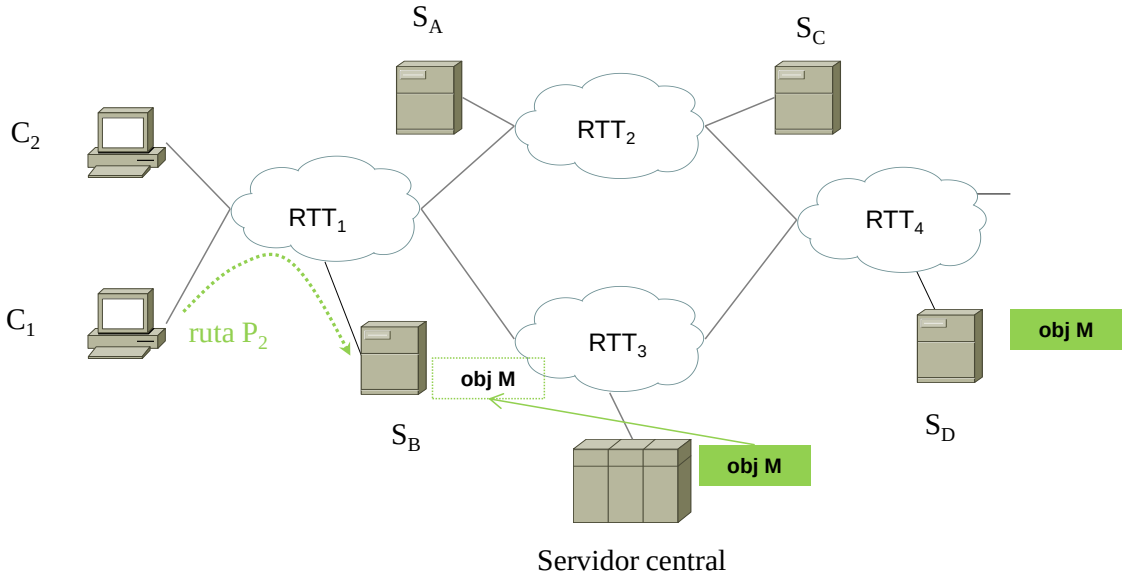


Figura 111. Modo de funcionamiento dinámico en el algoritmo de redirección.

Si el número de peticiones es elevado, como en el caso de una CDN, el valor de RTT_3 se puede despreciar de tal forma que quedaría:

$$RTT^{M}_{dinámico} \cong RTT^{M}_{1} \quad \text{(E 4.3)}$$

Esto supone un menor retardo que en el caso estático. Evidentemente, la percepción del retardo del primer cliente que accede en el modo dinámico puede ser considerable. Para minimizar esta situación, se suelen crear enlaces entre *surrogates* (*cache meshes*) de forma que el *surrogate* contacta con otro *surrogate* más cercano que el servidor origen para obtener el objeto multimedia solicitado. En este modelo de simulación, por simplicidad, no se crearán grupos de cache distribuidos.

Volviendo a nuestro diagrama de clases, la sesión de streaming tiene lugar entre dos entidades de nivel de aplicación, *RTSPClient* y *RTSPServer*, que intercambian información representada por la clase *RTSPData*, como se muestra en la Figura 112.

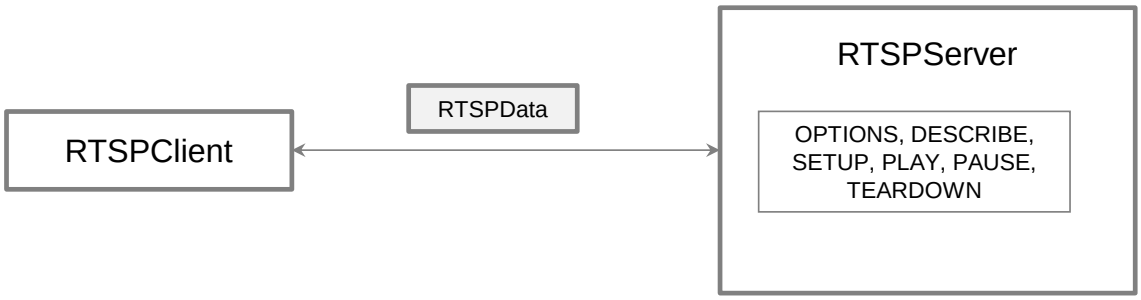


Figura 112. Sesión RTSP en el modelo de simulación.

RTSPClient implementa la aplicación que ejecuta el cliente para comunicarse con el servidor de streaming haciendo uso de los comandos que define el protocolo RTSP. Típicamente, el cliente envía un mensaje OPTIONS al servidor e inicia la transmisión de los comandos que definen el funcionamiento del protocolo: OPTIONS, DESCRIBE, SETUP, PLAY, PAUSE y TEARDOWN. Durante las sesiones RTSP es posible que el cliente envíe mensajes de PAUSE que detengan temporalmente la visualización del flujo, o mensajes de TEARDOWN que terminen con la sesión. La clase *RTSPClient* dispone de métodos que permiten regular estos mensajes de PAUSE y TEARDOWN

En realidad el cliente RTSP está implementado como una máquina de estados para actuar en consecuencia dependiendo del tipo de mensaje RTSP intercambiado con el servidor:

- Si se trata de la respuesta a un mensaje OPTIONS, el cliente construye un mensaje de tipo DESCRIBE en el que indica el identificador del objeto multimedia del que se desea obtener su descripción, y lo envía al servidor RTSP. Tras esto, el cliente cambia de estado.

- Si la respuesta es a un mensaje DESCRIBE, el cliente obtiene de ésta el número de flujos de que consta el objeto multimedia y, a continuación, envía un mensaje de SETUP por cada uno de ellos. Tras esto, el cliente cambia de estado nuevamente.
- Si se trata de la respuesta a un mensaje SETUP, quiere decir que la sesión RTSP para dicho flujo se ha establecido correctamente y, por tanto, el cliente puede enviar un mensaje de PLAY.
- Si la respuesta es a un mensaje PLAY, el cliente asume que ya está recibiendo el flujo multimedia y, entonces, programa el planificador de eventos para enviar un siguiente mensaje de PAUSE de acuerdo con el patrón indicado como parámetro.
- Si se trata de la respuesta a un mensaje PAUSE, el cliente asume que la sesión se ha detenido y, entonces, programa el planificador de eventos para reanudarla (envío de un mensaje PLAY) transcurrido un intervalo de tiempo determinado.
- Finalmente, si se trata de la respuesta a un mensaje TEARDOWN, el cliente asume que dicha sesión RTSP ha concluido y la elimina.

La clase *RTSPServer*, por su parte, implementa la aplicación RTSP servidora que se ejecuta en los *surrogates*. Los comandos RTSP que reconoce son OPTIONS, DESCRIBE, SETUP, PLAY, PAUSE y TEARDOWN, y los estados del servidor durante una sesión son INIT, READY, PLAYING y PAUSED. Cuando llega un nuevo mensaje, la actuación del servidor es la siguiente:

- Si se trata de un mensaje OPTIONS, el servidor contesta con un mensaje en el que se simula el envío de los comandos RTSP soportados por éste.
- Si recibe un mensaje del tipo DESCRIBE, el servidor obtiene de éste el identificador del objeto multimedia del cual se desea obtener su descripción. Con dicho valor, busca la instancia que representa a este objeto y obtiene el número de flujos que lo componen y, con esta información, construye el mensaje de respuesta que envía al cliente.
- Cuando el mensaje recibido es del tipo SETUP, el servidor crea una nueva sesión RTSP que, tras inicializar, inserta en su lista de sesiones activas. En este punto, se obtienen las instancias de los agentes RTP y RTCP que se encargan de la transmisión del flujo multimedia para el cual se ha establecido la sesión. Tras esto, el servidor envía una respuesta de confirmación.
- Si el mensaje es del tipo PLAY, el servidor obtiene de éste el identificador de sesión y comprueba que ésta se encuentra en un estado preparado (READY) o detenido (PAUSED). Tras esta verificación, el servidor RTSP inicia la transmisión del flujo invocando el método correspondiente del agente RTP y cambia el estado de la sesión (estado PLAYING).

- Si se trata de un mensaje PAUSE, el servidor obtiene de éste el identificador de sesión y comprueba que ésta se encuentra en un estado de reproducción (PLAYING). En caso afirmativo, el servidor RTSP detiene el streaming de dicho flujo invocando el método correspondiente del agente RTP y, a continuación, cambia el estado de la sesión (estado PAUSED).
- Si el mensaje es del tipo TEARDOWN, el servidor obtiene de éste el identificador de sesión y concluye la ejecución de la misma invocando el método *stop* del agente RTP. tras esto, cambia el estado de la sesión (estado INIT). En los tres últimos casos, también se envía el correspondiente mensaje de confirmación al cliente por parte del servidor.

La clase *RTSPServer* dispone de métodos que permiten insertar objetos multimedia en el servidor RTSP, indicando también el número de flujos que lo componen, así como su popularidad. A efectos de la simulación cada flujo puede ser un clip de un determinado tamaño que se transmite a tasa constante, o puede venir definido por una traza que determina el instante de envío de cada paquete y el tamaño de estos. Los campos que lo forman son:

```
int flow_ID;          // es el identificador del flujo
int bandwidth;        // representa el ancho de banda de transmisión del flujo
int size;             // tamaño en bytes del flujo multimedia
char *vname;          // nombre del fichero de traza que define el  flujo
```

Estos flujos multimedia son el componente principal de las sesiones RTSP. Las diferentes sesiones RTSP activas en el servidor se mantienen en una lista para facilitar la eliminación e inserción de nuevas sesiones. Los campos de que consta esta estructura son:

```
int sesión_ID;          // identificador de la sesión
int client_ID;          // identificador del cliente
sState status;          // estado en el que se encuentra la sesión
RTPSession* agentRTP;   // agente RTP utilizado para la transmisión del
                        // flujo a nivel del servidor
Flow* flow;             // flujo multimedia asociado a la sesión
Session *next;          // campo que apunta a la siguiente sesión de la lista
```

El servidor dispone de un número máximo de objetos multimedia que puede almacenar, indicado por un parámetro configurable. Este parámetro representa el tamaño del vector donde el servidor almacena todos los objetos multimedia. Estos objetos quedan representados en una estructura de datos donde básicamente se indica el número de flujos multimedia, así como su popularidad.

Por otro lado, nótese que el servidor RTSP puede, potencialmente, recibir la solicitud de un objeto multimedia del que no dispone en caché. En esta situación, el *surrogate* RTSP solicita dicho objeto al servidor origen de la CDN. La transferencia se realiza mediante un esquema que emula el funcionamiento del servicio FTP. Una vez el servidor RTSP dispone del objeto requerido, lo reemplaza por uno de los que almacena en memoria de acuerdo a una política establecida. Cuando el servidor RTSP dispone del objeto multimedia descargado del servidor origen, puede proporcionar la descripción del mismo al cliente y continuar así con la secuencia de comandos propia del protocolo RTSP.

Las clases *PauseTimer* y *TeardownTimer* heredan de *TimerHandler*, que constituye la clase base para la definición de cualquier planificador de eventos. *PauseTimer* y *TeardownTimer* implementan los temporizadores que determinan el patrón de generación de mensajes de PAUSE y TEARDOWN, respectivamente, por parte de los clientes. De esta forma, se consigue que los clientes interaccionen con el servidor de streaming mediante el envío de estos mensajes de parada o abandono de la sesión, cada vez que vencen los temporizadores respectivos.

La clase *CDNStrSession* también hereda de *TimerHandler* y representa una sesión en la CDN. Una sesión se define empleando los siguientes parámetros: instante de inicio, cliente que realiza las peticiones y su patrón de tráfico (número de objetos solicitados e intervalo entre peticiones consecutivas). La clase *CDNStrSession* se encarga de invocar los distintos métodos del cliente para que éste lleve a cabo las acciones pertinentes. Así, cada vez que vence el temporizador y, por tanto, el cliente requiere solicitar un nuevo recurso multimedia, la clase *CDNStrSession* ejecuta las funciones oportunas para que éste contacte con el Redirector y posteriormente con el *surrogate* indicado.

Por último se dispone de la clase *CDNStream* que deriva de *PagePool*. La clase *CDNStream* representa, como su nombre indica, a la Red de Distribución de Contenido en su totalidad. De esta forma, incluye parámetros tales como el número de clientes, número de *surrogates*, número de redirectores, número de objetos, porcentaje de replicación de objetos, estrategia de replicación de objetos, modo de funcionamiento de la CDN, etc. La clase *CDNStream* incorpora, además, los métodos necesarios para la creación y destrucción de las sesiones que intervienen en la misma

En la clase *CDNStream* se indica el modo de funcionamiento de la CDN, estático o dinámico, como se ha descrito al principio de este subcapítulo. Otro aspecto relevante en la configuración de esta clase es la estrategia de replicación de objetos. En nuestro modelo de simulación se han implementado tres políticas:

- **mostPopular**: en este caso todos los *surrogates* almacenan los mismos objetos multimedia y éstos son los M objetos más populares del servidor origen (donde M depende del porcentaje de replicación).

- **random**: en este caso los objetos del servidor origen se replican de forma aleatoria con la salvedad de que un mismo servidor no almacene dos objetos iguales.
- **all**: este caso es idéntico al anterior salvo que, en este caso, cualquier objeto del servidor origen debe replicarse, al menos, en un *surrogate*.

Todas estas clases han sido implementadas en NS-2 en un total de 15 ficheros, como se observa en la Figura 113. Existen doce clases en C++ asociadas a las diez clases descritas con anterioridad, así como tres ficheros Otcl que permiten y facilitan la posterior simulación de la CDN de streaming. Con la finalidad de proporcionar una visión de alto nivel del desarrollo para centrarse en los resultados, la descripción detallada de estos ficheros puede consultarse directamente en el código fuente.

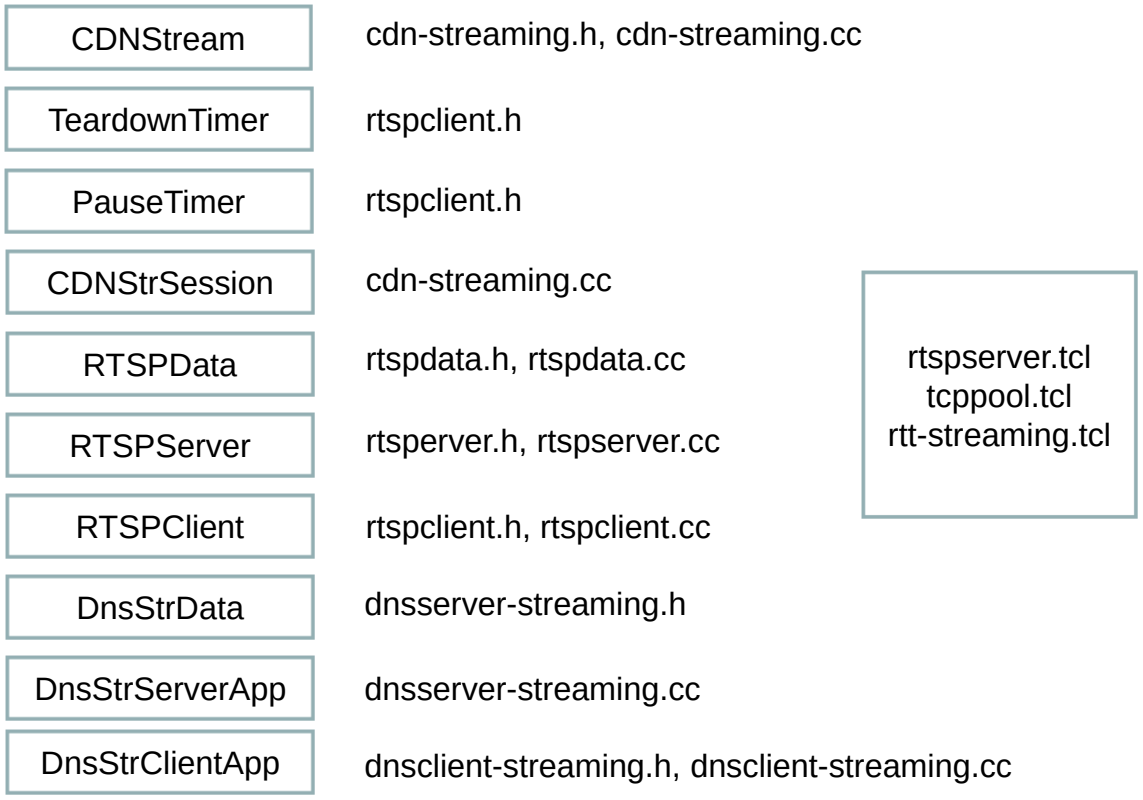


Figura 113. Ficheros implementados y su relación con las clases del simulador CDN de streaming.

Respecto a los ficheros TCL, el fichero **rtspserver.tcl** dispone de tres procedimientos. El primero de ellos permite crear sesiones RTP/RTCP y obtener instancias a los agentes asociados. El método, además, asocia dichas sesiones a los nodos cliente y servidor que recibe como parámetros. Los otros dos procedimientos incluidos son necesarios cuando la CDN trabaja en modo dinámico. Así, estos métodos son los encargados de instanciar sendos agentes TCP, asociarlos al *surrogate* y al servidor origen, conectarlos entre sí y simular el envío, sobre dicha conexión de transporte, de la solicitud de un objeto

multimedia por parte del *surrogate* y de la correspondiente respuesta por parte del servidor origen. Este esquema pretende emular un servicio de transferencia de ficheros del tipo FTP.

El fichero **rtt-streaming.tcl** dispone de un procedimiento que, dados dos nodos de la topología, devuelve como resultado el RTT entre ellos. Este procedimiento resulta muy útil para determinar el retardo entre cada pareja cliente-*surrogate* y, de esta forma, obtener, para cada cliente, una ordenación de los *surrogates* de menor a mayor RTT. Este fichero es equivalente al que se desarrolló en la sección 3.3 de la CDN para entornos web.

El fichero **tcppool.tcl** incluye dos procedimientos de apoyo a distintas clases. El primero de ellos permite obtener una instancia de un agente *FullTcp*, lo cual resulta necesario para la comunicación entre el cliente y el servidor de streaming RTSP. El segundo se utiliza para crear dos agentes UDP que se asocian respectivamente a las aplicaciones DNS cliente y servidora y se conectan entre sí.

Como ya hemos indicado con anterioridad, para la simulación de la transmisión de datos en tiempo real, se ha hecho uso de las implementaciones del protocolo RTP/RTCP incluidas en la distribución de NS-2. No obstante, se han introducido diversas modificaciones en las mismas para mejorar su funcionamiento y adaptarlas a los requerimientos y objetivos de nuestras simulaciones. Estas mejoras se han realizado, concretamente, en los ficheros *rtp.h*, *rtp.cc*, *rtcp.cc*, *session-rtp.cc* y *session-rtp.tcl*, y son las siguientes:

- Posibilidad de establecer sesiones *unicast*.
- Los agentes RTP pueden reducir o incrementar su tasa de emisión en función de la tasa de pérdidas que experimentan sus receptores y que comunican a éstos a través de los paquetes de realimentación RTCP.
- Posibilidad de que los agentes RTP transmitan archivos de un determinado tamaño y a una tasa especificada.
- Posibilidad de que los agentes RTP generen paquetes de acuerdo a lo indicado en un fichero de traza que reciben como parámetro.
- Capacidad de reanudar sesiones a partir del punto en que éstas fueron detenidas (como consecuencia de un mensaje de PAUSE).
- Capacidad para interactuar con los servidores y clientes RTSP.

Cabe mencionar que la implementación estándar del protocolo RTP/RTCP en NS-2 ha sufrido varias modificaciones y adaptaciones para cumplir algún requisito en concreto, dependiendo de la funcionalidad deseada. Algunos ejemplos son [Mon_10] y [WWW_NS2b].

4.2.3. Escenarios de simulación

En este apartado se describen los distintos entornos de red empleados como escenarios para realizar las distintas simulaciones. Estos entornos, de forma análoga a la sección 3.3.4, se han obtenido mediante el generador de topologías GT-ITM.

Los principales parámetros que definen la topología de red en general, a partir de la cual se insertarán y configurarán los clientes y los servidores para formar una CDN, se muestran en la Tabla 13 (sección 3.3.4). Por otro lado, los parámetros que caracterizan a la CDN implementada son diversos y se listan en la Tabla 38.

Servidores	Número de *surrogates* en la CDN
Clientes	Número de clientes en la CDN
Redirectores	Número de Redirectores (o Gestores de Contenido)
Sesiones	Número de sesiones en la CDN
Objetos multimedia	- Número de objetos multimedia - Tamaño de objetos multimedia
Replicación	- Porcentaje de replicación de objetos web - Estrategia de replicación (*mostPopular, random, all*)
Surrogates	- Número de surrogates a considerar por cliente - Número máximo de sesiones simultáneas
Modo	Modo de funcionamiento de la CDN: - Estático: el *surrogate* contactado dispone en memoria del objeto multimedia solicitado - Dinámico: el *surrogate* contactado puede no disponer del objeto multimedia solicitado, por lo que deberá contactar con el servidor origen.
Sesión RTSP	- Probabilidad de abandono de la sesión RTSP mediante mensaje TEARDOWN - Patrón de generación de mensajes PAUSE por el cliente
Tráfico	Patrón de tráfico de los clientes: - Instante de inicio de sesión - Número de peticiones por sesión - Intervalo entre peticiones de páginas (función de distribución)

Tabla 38. Parámetros de simulación generales (streaming)

De manera similar al modelo de simulación web (sección 3.3), cada cliente que se crea en la CDN se añade como un nuevo nodo hijo a un nodo hoja de la topología de red creada.

Para todas las simulaciones, el número de objetos multimedia con los que contará la CDN será también el mismo e igual a 500. Cada uno de estos objetos constará de 2 flujos (audio y video) y dispondrá, igualmente, de un índice de popularidad que determinará la frecuencia en el acceso a cada uno de ellos. Para los flujos de video se leerán ficheros de trazas con secuencias de películas reales codificadas en MPEG4, 10 minutos de duración, resolución 640*480 y con diferentes niveles de calidad (tasa de codificación y cuadros por segundo). Para los flujos de audio, por su parte, se tomará un ancho de banda con tasa constante (128 Kbps), menor que el video. Estos detalles, para cada objeto multimedia, quedan recogidos en la Tabla 39 para los diez primeros objetos.

Objeto Multimedia	Número de Flujos	Popularidad	Flujo		
1	2	1	1	**Audio:** *aladdin.mp3*	
			2	**Video:** *aladdin.mp4*	770*Kbps*, 60*Mb*
2	2	3	1	**Audio:** *beauty.mp3*	
			2	**Video:** *beauty.mp4*	660*Kbps*, 52*Mb*
3	2	4	1	**Audio:** *cars.mp3*	
			2	**Video:** *cars.mp4*	550*Kbps*, 44*Mb*
4	2	2	1	**Audio:** *snowwhite.mp3*	
			2	**Video:** *snowwhite.mp4*	*440Kbps*, 34*Mb*
5	2	5	1	**Audio:** *bambi.mp3*	
			2	**Video:** *bambi.mp4*	*330Kbps*, 23*Mb*
6	2	4	1	**Audio:** *dumbo.mp3*	
			2	**Video:** *dumbo.mp4*	*220Kbps*, 11*Mb*
7	2	5	1	**Audio:** *peterpan.mp3*	
			2	**Video:** *peterpan.mp4*	*110Kbps*, 4*Mb*
8	2	3	1	**Audio:** *pocahontas.mp3*	
			2	**Video:** *pocahontas.mp4*	*700Kbps*, 58*Mb*
9	2	2	1	**Audio:** *robinhood.mp3*	
			2	**Video:** *robinhood.mp4*	600*Kbps*, *49Mb*
10	2	1	1	**Audio:** *lionking.mp3*	
			2	**Video:** *lionking.mp4*	500*Kbps*, *38Mb*

Tabla 39. Objetos multimedia empleados en la simulación de streaming.

El patrón de generación de mensajes de PAUSE por parte del cliente se especifica a través de una variable aleatoria que sigue una determinada distribución y que, para cada sesión, determina el número de mensajes de este tipo que enviará el cliente a su servidor de streaming por cada 60 minutos (1hora) de la misma.

La variable *P2Pause_* representa el intervalo de tiempo entre generación de mensajes PAUSE (véase Figura 114). El tiempo hasta el envío del correspondiente mensaje de PLAY que reanude la sesión debe ser menor que el tiempo entre mensajes PAUSE, y se ha tomado de forma arbitraria en el valor 1/6**P2Pause_*, ya que no existen referencias bibliográficas que caractericen este tipo de distribución.

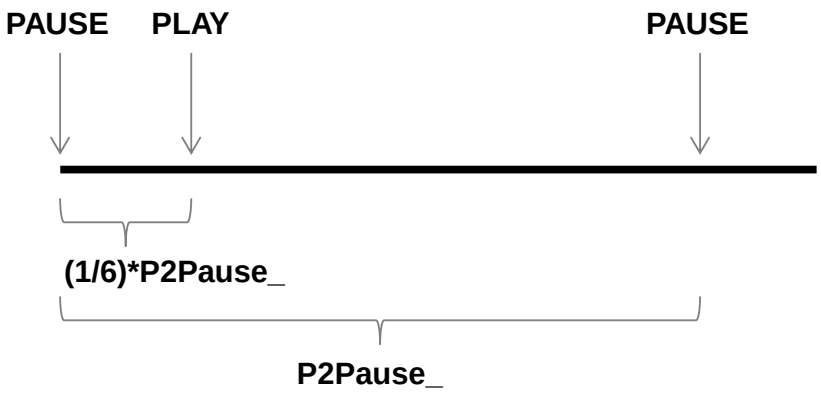


Figura 114. Patrón de generación de mensajes PAUSE mediante la variable P2Pause_.

Tanto el parámetro anterior (*P2Pause_*) como el que determina la probabilidad de abandono de la sesión RTSP mediante el mensaje TEARDOWN se han definido del mismo modo en todas las simulaciones realizadas dada su escasa repercusión en los resultados finales (como se verá). Así, los valores escogidos han sido:

- Número de mensajes PAUSE/hora de sesión: *Normal (avg_= 3, std_=1)*
- Probabilidad de abandono (mensajes TEARDOWN): *20%*

Por último, como ya se describió anteriormente, una de las mejoras introducidas en la implementación del protocolo RTP/RTCP existente en NS-2, ha sido el control de QoS en base a la medida de la tasa de pérdidas. De este modo la fuente, por cada paquete RTCP RR que recibe, calcula la tasa de pérdidas (λ) utilizando un estimador predictivo para suavizar las medidas y evitar oscilaciones de QoS:

$$\lambda = (1-\alpha)\cdot\lambda + \alpha\cdot b \qquad 0 \le \alpha \le 1 \qquad \text{(E 4.4)}$$

donde b es la tasa de pérdidas medida por el receptor y presente en el paquete RTCP

RR. Con esta tasa de pérdidas, la fuente realiza una estimación del estado de la red en base a los umbrales λ_u y λ_c de la forma:

- Si $\lambda \ge \lambda_c \rightarrow$ red congestionada
- Si $\lambda_c > \lambda \ge \lambda_u \rightarrow$ red cargada
- Si $\lambda < \lambda_u \rightarrow$ red descargada

Finalmente, dependiendo del estado de la red, se pueden dar tres situaciones:

- Se disminuye el ancho de banda mediante un factor multiplicativo μ
- Se incrementa el ancho de banda mediante un factor aditivo ν
- El ancho de banda se mantiene igual.

Para las simulaciones, los valores escogidos para estos parámetros han sido:

$$\alpha = 0.3;\ \lambda_c = 10;\ \lambda_u = 3;\ \mu = 0.2;\ \nu = 0.1$$

4.2.4. Resultados y análisis de la simulación

A continuación se describirán las diferentes simulaciones que se han llevado a cabo. Para ello, se ha ido modificando uno o varios parámetros de los definidos con anterioridad, manteniendo fijo el resto. La evaluación se ha centrado en dos variables: la carga media de los servidores/*surrogates*, y el tiempo medio de respuesta experimentado por los clientes. El primero de ellos se ha expresado en términos del número medio de sesiones activas en los servidores a lo largo de la simulación. El segundo parámetro se ha evaluado según tres criterios: (i) retardo inicial (establecimiento de sesión RTSP), (ii) retardo medio durante la sesión RTSP y (iii) *jitter* o variabilidad del retardo.

4.2.4.1. Carga media de los *surrogates* vs. Número de clientes

La Figura 115 muestra la evolución de la carga experimentada por los servidores frente al incremento del número de clientes. Para este último parámetro, se ha utilizado un rango de 1 a 2000 usuarios, y la representación se ha obtenido, a su vez, para distintos valores en el número de *surrogates* que conforman la CDN (5, 20 y 40 respectivamente). Cabe señalar que, para evaluar la carga experimentada por los servidores, se ha considerado el número medio de conexiones RTSP establecidas por el servidor a lo largo de la simulación.

Los parámetros particulares para este escenario de simulación se especifican en la Tabla 40. Cabe destacar que no existen referencias bibliográficas para caracterizar la generación de mensajes PAUSE, por lo que se ha tomado una distribución normal; por otro lado, el contenido se encuentra replicado totalmente (posteriormente se estudiará el caso en que varía la replicación). En cuanto al tráfico generado por los clientes, estos solicitan 10 flujos multimedia por sesión, y el intervalo entre sesiones sigue una distribución de tipo Pareto [Mar_03], suponiendo un patrón de acceso similar al propuesto para el servicio web tradicional (sección 3.3).

Número de objetos multimedia	200
Número máximo de sesiones simultáneas en los servidores de streaming	500
Porcentaje de replicación de objetos	100 %
Estrategia de replicación de objetos	Random
Patrón de generación de mensajes de tipo PAUSE	Normal (avg_=10, std_=1)
Probabilidad de abandono de la sesión RTSP (TEARDOWN)	20%
Modo de funcionamiento de los servidores	static
Patrón de tráfico de los clientes	- Inicio de sesión: 0.1 s - Peticiones por sesión: 25 - Intervalo entre peticiones: Pareto (avg=50 s, shape=2)

Tabla 40. Parámetros NS-2 (streaming).

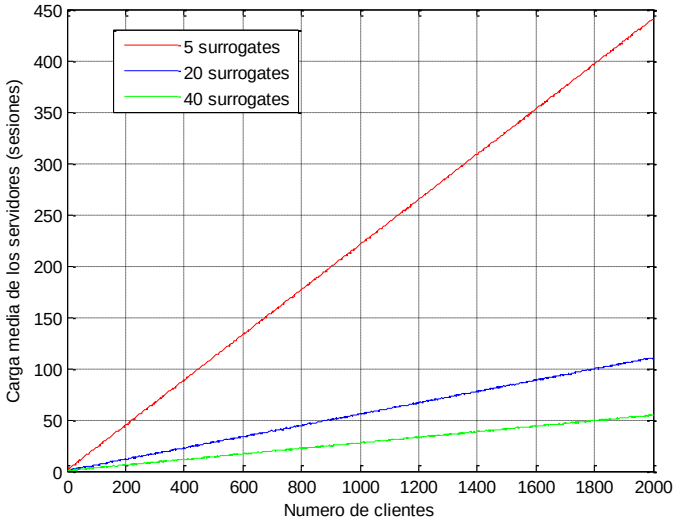


Figura 115. Carga media vs número de clientes (streaming).

De los resultados obtenidos se pueden derivar las siguientes conclusiones:

- Existe un incremento proporcional de la carga con el número de clientes. Conforme aumenta este número de clientes, los servidores reciben más peticiones que tienen que servir. Esta relación es lineal porque depende del número de sesiones RTSP, pero cabe mencionar que la carga en los servidores en términos de utilización de CPU no es lineal conforme aumentan sus sesiones de streaming e incluso las propias sesiones web (como se presenta en la sección 3.4.5 de la implementación real), y conviene fijar un umbral de carga, que dependería de la capacidad de cada *surrogate*. En la simulación se ha tomado un umbral equivalente de 500 sesiones simultáneas que, como se puede comprobar, no se alcanza para 2000 clientes.

- Es relevante la disminución que experimenta esta carga con un incremento en el número de *surrogates*, lo cual refleja una de las ventajas más destacadas de las CDNs. Si establecemos en la Figura 115 un número de 1000 clientes, entonces la CDN con 5 *surrogates* debe absorber por término medio un máximo de 220 sesiones RTSP por servidor, mientras que las CDNs con 20 y 40 *surrogates* simplemente absorberán 58 y 32 sesiones respectivamente. Dado que el conjunto de sesiones RTSP en la simulación se 'reparte' entre todos los servidores, es posible establecer una relación de proporcionalidad entre el número de *surrogates* y la carga media (medida en términos de sesiones RTSP servidas). De esta forma, la proporción en el aumento del número de *surrogates* equivale a la disminución proporcional en el número de sesiones multimedia servidas.

Evidentemente, el número medio de sesiones RTSP que un servidor soporta durante una simulación es un indicador vinculado a la carga de los servidores, pero también puede

medirse si se está produciendo algún tipo de congestión en la red, ya sea en el extremo de los servidores o de los clientes.

La Tabla 41 ilustra las tasas máximas por sesión que son capaces de soportar los enlaces de la topología de red en el peor caso descrito en la Figura 115, correspondiente a 2000 clientes, ya que cuenta con mayor número de sesiones RTSP. La segunda columna (34 Mbps) corresponde a los servidores y el ancho de banda máximo capaz de soportar por sesión: para 5 *surrogates* y 2000 clientes, suponiendo una asignación homogénea entre servidores, cada servidor atiende a 400 clientes, lo que se corresponde con una tasa máxima por sesión de 85 Kbps. Por otro lado, la tercera columna (2Mbps) corresponde al ancho de banda de los clientes. Aunque en principio son capaces de usar completamente (saturar) su ancho de banda, como valor máximo emplearán 898 Kbps (770+128, véase Tabla 39) que corresponde a la sesión con mayores requisitos de ancho de banda. Se puede comprobar fácilmente como se producirá congestión en la red, lo cual implicará retardos excesivos y *jitter*. Evidentemente, el dimensionamiento de los enlaces de red realizado para servicios web (2 Mbps y 34 Mbps) y para clientes y *surrogates* no se puede trasladar a los servicios multimedia, a no ser que el número de sesiones RTSP disminuya considerablemente. Ni siquiera con 40 servidores en la CDN es posible satisfacer los requisitos de ancho de banda de red demandados. Si se ha fijado el umbral de carga de los servidores a 500 sesiones (que no se ha alcanzado en la simulación), se observa que se producirá antes congestión en la red que congestión en los servidores. Hay que tener en cuenta que los valores en la Tabla 41 corresponden al caso peor en que todos los clientes soliciten simultáneamente un flujo de 898 Kbps. Si se toma el valor medio del tamaño de las sesiones de la Tabla 39, se corresponde con 616 Kbps, lo cual es un valor soportable para una CDN con 40 *surrogates*. Dado que los contenidos de la CDN son demandados con la misma probabilidad, en principio el ancho de banda de la red puede parecer suficiente en términos medios. No obstante, puede haber momentos puntuales durante las sesiones donde se aprecie congestión si el ancho de banda puntual demandado excede al disponible.

Número de surrogates	34 Mbps	2 Mbps
5	85 Kbps	898 Kbps
20	340 Kbps	898 Kbps
40	680 Kbps	898 Kbps

Tabla 41. Tasas máximas por sesión RTSP.

Uno de los aspectos que se ha introducido en el simulador de NS-2 es la capacidad de reacción por parte del protocolo RTP ante situaciones de congestión en la red, como se describió en la sección 4.2.3. Si el emisor disminuye su ancho de banda en la emisión de paquetes, se pueden establecer dos situaciones:

- El flujo transmitido se corresponde con los mismos ficheros originales (audio y video), cada vez habrá un menor número de paquetes que llegarán al *buffer* del cliente, con lo que se producirán paradas en la reproducción de video (*buffer*

underrun). El cliente tendrá que esperar hasta que dicho *buffer* vuelva a llenarse para reproducir el vídeo. Si la situación de congestión en la red perdura, el cliente no reproducirá el vídeo satisfactoriamente (de manera fluida), y posiblemente abandonará la sesión. Esta situación es útil desde el punto de vista de la simulación para estimar los retardos y *jitter* en toda la sesión, de tal forma que se puede comprobar que, a medida que se aumenta el número de *surrogates*, estos valores disminuyen. Desde el punto de vista del usuario, lo que éste percibe es un menor número de paradas (o ninguna).

- El flujo de emisión no corresponde con el fichero original de video, sino que se emplean diferentes tasas de codificación para paliar el efecto de las paradas. De esta forma, hay una variación de calidad (número de cuadros/s, resolución, codificación, etc.) pero no afecta (apenas) al retardo ni al *jitter* y el usuario puede seguir reproduciendo un vídeo fluido. Esta situación es útil desde el punto de vista de la capacidad de un único servidor de streaming para adaptarse a las condiciones cambiantes de la red, pero en esta tesis se estudian los beneficios de la CDN en su conjunto (meta-aplicación colaborativa de todos los servidores).

4.2.4.2. Tiempo medio de respuesta vs. Número de servidores

La Figura 116 relaciona el tiempo medio de respuesta experimentado por los clientes frente al incremento del número de *surrogates* que conforman la CDN. Como rango para este último parámetro se ha utilizado el intervalo de 0 a 40 servidores, análogo a la simulación anterior, ya que 40 es un número significativamente elevado como para poder apreciar la evolución del tiempo medio de respuesta (como se observa en la Figura 116). A su vez, las representaciones se han obtenido para diferentes valores en el número de clientes, 30, 200 y 400 respectivamente. Estos valores también permiten observar su efecto en el tiempo de respuesta. El criterio seguido es tomar un valor máximo de clientes (400) 10 veces mayor que el valor máximo de *surrogates* (40, con el propósito de modelar tendencias.

Para la evaluación del tiempo medio de respuesta se ha considerado el intervalo de tiempo transcurrido desde que el cliente contacta con el Redirector hasta que obtiene la correspondiente respuesta al mensaje DESCRIBE. Se ha optado por este mensaje ya que el servidor de streaming sólo responde al mismo una vez dispone del recurso multimedia en caché. En este escenario (replicación total) este aspecto no es significativo, pero sí lo será en otros casos de estudio donde se evalúa el tiempo medio de respuesta con el porcentaje de replicación, y el servidor de streaming deberá obtener una copia del objeto multimedia antes de contestar al cliente.

En relación a los parámetros particulares para esta simulación, son los mismos que en la simulación anterior (véase sección 4.2.4.1) que quedaban reflejados en la Tabla 40.

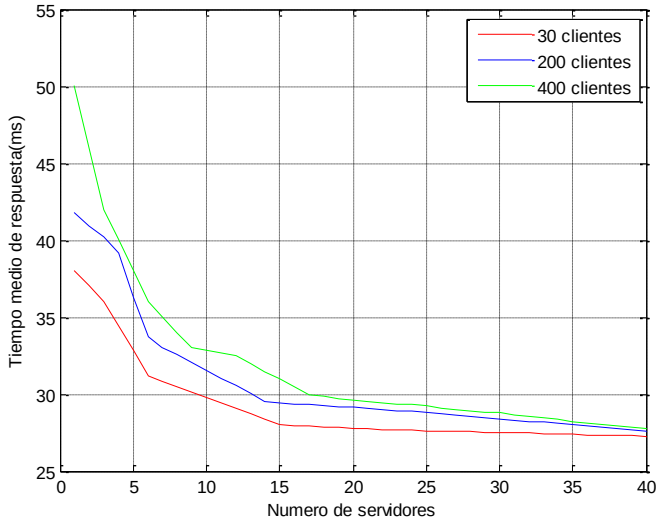


Figura 116. Tiempo medio de respuesta vs número de servidores (streaming).

Los resultados obtenidos en este caso ponen de manifiesto las ventajas de las CDN en términos de disminución en el tiempo medio de respuesta:

- Se observa que a medida que el número de servidores es mayor, se produce una disminución en la latencia percibida por los usuarios. Esta reducción es más pronunciada en el intervalo de 0 a 6 servidores, y más suave en el resto. Esto es así debido a que, cuando existen pocos servidores, un pequeño incremento en su número provoca que los clientes puedan acceder a nuevas réplicas potencialmente próximas. No obstante, cuando éstas alcanzan un número considerable (15), las diferencias no son tan apreciables. Se trata de un comportamiento similar al obtenido en el caso de servicio web, aunque en este caso los retardos son menores, ya que en el intercambio inicial el mensaje RTSP DESCRIBE de petición y respuesta tiene un tamaño muy pequeño.

- En relación con el aumento del número de clientes, observamos un comportamiento coherente si se tiene en cuenta que, conforme éste es mayor, el tráfico en la red y la carga de los servidores son superiores, lo que repercute directamente en un mayor tiempo de respuesta.

Si bien es importante estimar el retardo inicial del cliente (en este caso hay replicación total y no hay que realizar transferencia de vídeo entre servidores), no se trata del parámetro más relevante en una sesión de streaming. En un servicio web, se devuelve el contenido desde el servidor y se consume directamente en el cliente en un solo paso; en este caso, el retardo inicial corresponde al retardo total. Por el contrario, en una sesión de streaming el contenido se consume de forma continua en múltiples pasos, y el parámetro de interés es cada uno de los retardos individuales en cada salto y cómo afecta esto a la reproducción del flujo multimedia. Si el retardo es elevado en alguno de estos pasos, se producirá una parada en la reproducción del vídeo. El método práctico

para tratar de solventar este problema, tal como se explicó en la sección 2.3.4.2, es el empleo de un *buffer* que minimiza el efecto de la variabilidad de la red, a costa de incrementar el retardo inicial. En esta situación, se introduce un nuevo parámetro de interés, el *jitter*, que corresponde con la variabilidad del retardo. Si el *jitter* es elevado, indica que el retardo es muy fluctuante, lo que conducirá a paradas en la reproducción del vídeo (*buffer underrun*).

La Tabla 42 muestra los valores de retardo medio y *jitter* máximo medidos en la simulación de la CDN con soporte de streaming para diferentes valores de servidores RTSP y clientes. El valor del retardo se ha obtenido en cada uno de los clientes cuando se recibe el primer paquete del flujo de vídeo (más limitante que el flujo de audio por su mayor consumo en términos de ancho de banda). Se puede apreciar cómo tanto el retardo medio como el *jitter* aumentan con el número de clientes, mientras que se reducen con el número de *surrogates*. En este sentido, la CDN es efectiva al reducir los retardos. Por otro lado, se puede observar en la Tabla 42 como para 200 clientes la red se encuentra descargada y los clientes podrán reproducir sus flujos de vídeo sin ninguna parada. Sin embargo, para el caso de 2000 clientes la red está cargada y esto se refleja en los valores de retardo medio y *jitter*:

- El retardo medio se incrementa por dos motivos. En primer lugar porque se produce un mayor retardo en los nodos intermedios (*routers*). En segundo lugar, porque los agentes RTP detectan la congestión en la red y reducen su tasa de emisión, con lo que el paquete tarda más tiempo en llegar a su destino.

- El *jitter* aumenta por los motivos anteriormente citados para el caso del retardo, ya que se trata de parámetros de red íntimamente relacionados. Cabe señalar que en la Tabla 42 se muestra el valor máximo de *jitter* medido en la simulación, sin indicar cuantas veces se ha producido. El análisis detallado de este dato requiere el estudio de una sesión de usuario de forma individual, como se presenta a continuación.

El caso más desfavorable en la Tabla 42 es el de 2000 clientes y 5 servidores. Si asumimos un retardo máximo (RTT) de 200 ms, que es un valor típicamente usado en streaming por Internet [Sax_08], se observa que este valor es superado, lo que es un indicador claro de congestión en la red que se traducirá en una o varias paradas en la reproducción del flujo en el cliente.

	200 clientes		400 clientes		2000 clientes	
Servidores	**Retardo medio**	**Jitter máximo**	**Retardo medio**	**Jitter máximo**	**Retardo medio**	**Jitter máximo**
5	**88**	**14**	**128**	**29**	**192**	**43**
20	**84**	**12**	**113**	**21**	**156**	**39**
40	**78**	**10**	**91**	**12**	**134**	**27**

Tabla 42. Valores de retardo y jitter (en ms) para la CDN de simulación.

En la Figura 117 se representan los valores del retardo durante el primer minuto de una sesión de usuario para una simulación de 2000 clientes. Puede apreciarse como, para el caso de 5 servidores, se produce un retardo excesivo (superior a 200 ms en numerosas ocasiones) que conduce a una parada en la reproducción. Por otro lado, incluso con 40 servidores, hay momentos puntuales de congestión, aunque el número es mucho menor. Este último dato no se podía apreciar en la Tabla 42, al emplear valores medios. Los momentos en que el retardo es nulo corresponden a las pausas realizadas por el cliente, cuyo patrón de generación se describía en la Tabla 40.

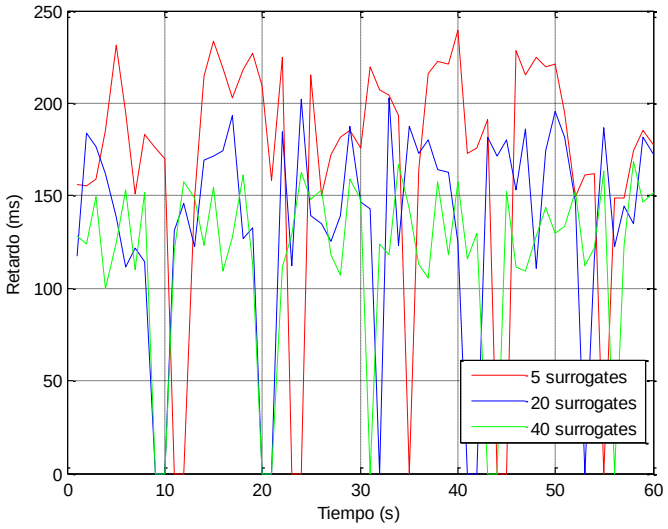


Figura 117. Retardo en tres sesiones multimedia.

Otro aspecto fundamental en la reproducción de un flujo de vídeo no es únicamente evitar las paradas o tiempos de espera, sino también evitar las pérdidas o cortes en la reproducción. Esto está originado por dos motivos:

- El primer caso corresponde a un retardo excesivo en los paquetes enviados. En ocasiones, las aplicaciones de reproducción de los clientes asumen que si un paquete no llega antes de un determinado tiempo se ha perdido, y aunque llegue posteriormente es descartado.

- El segundo caso corresponde con la posibilidad de pérdida de paquetes en la red de comunicaciones, de tal forma que ciertos paquetes o bien no llegan al receptor, o llegan erróneos. En este caso, se introduce una cierta tasa de error tanto en los enlaces como en el receptor. Los errores pequeños se pueden corregir en recepción mediante técnicas FEC (*Forward Error Correction*), sin percepción por parte del usuario, pero los errores de ráfaga afectan a varios paquetes consecutivos y el corte es perceptible por el usuario, ya que típicamente es necesario esperar uno o varios segundos hasta conseguir una trama I (*intra*) correcta del vídeo codificado.

El estudio de los errores en la red conduce a introducir un modelo de pérdidas en los enlaces que no se ha abordado en este capítulo. Este aspecto de la pérdida de paquetes y su efecto en la reproducción del vídeo se aborda en la implementación de la CDN con soporte de streaming (sección 4.3).

4.2.4.3. Tiempo medio de respuesta vs. Porcentaje de replicación

Otro aspecto interesante a considerar es la evolución del tiempo medio de respuesta conforme varía el porcentaje de replicación. Para evitar posibles situaciones de congestión que no son objeto de estudio en esta sección, se ha tomado un escenario con 20 servidores y 400 clientes, donde se asegura que hay suficiente ancho de banda disponible en la red como para que no se produzca congestión (véase Tabla 42). Tal como se indicaba en la Tabla 40, el modo de funcionamiento es *static*.

La Figura 118 muestra la evolución del tiempo medio de respuesta frente al porcentaje de replicación de objetos en el rango [0%, 100%] con incrementos del 10%, y para diferentes tipos de estrategia de replicación (*random, all*, *mostPopular*) en la CDN.

Se observa una disminución del retardo experimentado por los clientes a medida que se incrementa el porcentaje de replicación. Este comportamiento es coherente, ya que conforme los surrogates disponen de más objetos en su caché, se consigue una mayor tasa de acierto (*hit rate*) y la necesidad de acceder al servidor origen se reduce.

La Figura 118 presenta una comparación interesante entre la respuesta obtenida para las diferentes estrategias de replicación de objetos. Cuando el porcentaje de replicación es del 0% o del 100%, el retardo experimentado es independiente de la estrategia utilizada. Este hecho es debido a que, en el primer caso (replicación 0%) los clientes acceden directamente al servidor origen; en el segundo caso (replicación 100%) el servidor más próximo a cualquier cliente siempre dispone del objeto multimedia en memoria. Por otra parte, para valores de replicación intermedios, la estrategia que resulta más efectiva (menor retardo) es *all*, luego *random* y, finalmente, *mostPopular*. La explicación de este comportamiento radica en que el modo de funcionamiento de la CDN es *static*. Así, cuando se replican la totalidad de los objetos multimedia en algún *surrogate* (modo *all*) el cliente no necesita acceder al servidor origen (ya que accede a la réplica más cercana que contiene el objeto) y, de esta forma, se evita la penalización o aumento del retardo que esto supone. Por su parte, en el modo *mostPopular*, aunque los objetos más populares están disponibles en el servidor más próximo al cliente, para obtener el resto de objetos es preciso contactar con el servidor origen, lo cual incrementa el tiempo medio de respuesta. Por último, el modo *random*, debido a su carácter aleatorio, es el que presenta mayores fluctuaciones y variabilidad en su rendimiento. Cabe destacar que, con un modo *dynamic*, los resultados hubiesen resultado totalmente contrarios por su propia forma de funcionamiento, y la estrategia *mostPopular* obtendría menor retardo medio.

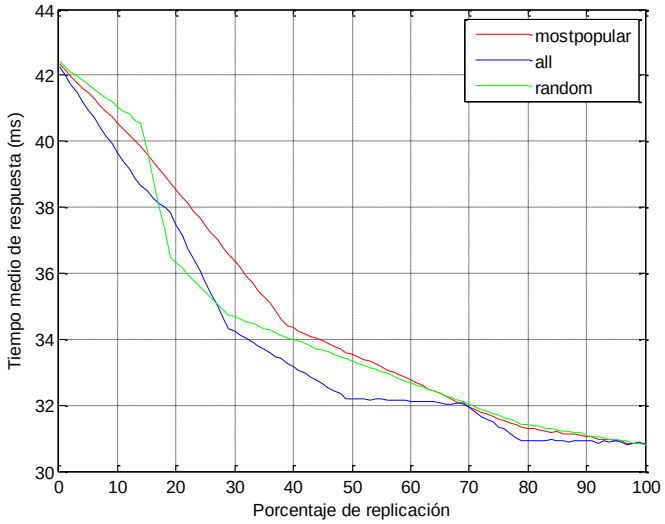


Figura 118. Tiempo de respuesta vs. Replicación.

En el modo *dynamic*, el cliente es redirigido al *surrogate* más cercano aunque este último no disponga del contenido. Si el servidor RTSP dispone del contenido, se le ofrece un flujo multimedia con retardos mínimos. Si el servidor RTSP no dispone del contenido, lo obtiene previamente del servidor origen y luego lo sirve. Esto representa un retardo mayor para el primer cliente, pero no para sucesivos clientes que soliciten el mismo flujo. En términos estadísticos, si se almacena el contenido *mostPopular* se obtienen unos valores de retardo inferiores que con otras estrategias (véase Tabla 43). Puesto que la simulación se ha realizado en un escenario de poca carga, los valores de retardo y *jitter* permiten ofrecer una sesión multimedia de calidad sin paradas.

El efecto de la replicación en las prestaciones es aún más significativo en condiciones de carga elevada en la red (20 servidores y 2000 clientes), como se observa en la Tabla 44. Conforme aumenta el porcentaje de replicación, el servidor más cercano sirve completamente el contenido y el número de enlaces a atravesar en la comunicación cliente-servidor es menor, con lo que la probabilidad de congestión se reduce. Para un porcentaje de replicación del 100% el efecto es más acusado porque no se establece ninguna comunicación con el servidor origen. Si el porcentaje de replicación es reducido, el retardo resulta excesivamente elevado.

	mostPopular		**all**		**random**	
Replicación	**Retardo medio**	**Jitter máximo**	**Retardo medio**	**Jitter máximo**	**Retardo medio**	**Jitter máximo**
20	**88**	**14**	**118**	**13**	**123**	**14**
40	**84**	**12**	**102**	**11**	**98**	**11**
60	**78**	**10**	**84**	**9**	**87**	**9**
80	**69**	**9**	**78**	**9**	**77**	**9**
100	**61**	**9**	**62**	**8**	**63**	**8**

Tabla 43. Retardo y jitter para estrategias mostPopular, all y random (modo dynamic) en baja carga.

	mostPopular		all		random	
Replicación	**Retardo medio**	**Jitter máximo**	**Retardo medio**	**Jitter máximo**	**Retardo medio**	**Jitter máximo**
20	**221**	**42**	**232**	**52**	**238**	**53**
40	**200**	**35**	**199**	**41**	**201**	**41**
60	**183**	**29**	**183**	**32**	**187**	**34**
80	**171**	**22**	**174**	**27**	**178**	**29**
100	**145**	**13**	**147**	**11**	**148**	**14**

Tabla 44. Retardo y jitter para estrategias mostPopular, all y random (modo dynamic) en alta carga.

4.2.4.4. Distribución de carga en los servidores vs. Porcentaje de replicación

El porcentaje de replicación también tiene un efecto en la carga que soportan los *surrogates* y el servidor origen. La Figura 119 representa este efecto para un modo de funcionamiento *static* y con los parámetros de simulación descritos en la Tabla 40. Al igual que en la sección anterior (4.2.4.3) se ha tomado un escenario con poca carga (20 servidores y 400 clientes). Se ha representado la distribución de carga, que se puede desglosar en la carga soportada por los *surrogates* y la carga soportada por el servidor origen. A medida que la carga de proceso se va acumulando en los *surrogates*, el servidor origen queda descargado. Esto puede resultar interesante en aquellos escenarios de CDNs donde se intenta minimizar el acceso al servidor origen.

Nuevamente, se ha tomado un porcentaje de replicación en el rango [0%, 100%] con incrementos del 10%, para diferentes tipos de estrategia de replicación (*random*, *all*, *mostPopular*) en la CDN.

Como se aprecia en la Figura 119, al incrementar el porcentaje de replicación existe un mayor número de peticiones gestionadas por los *surrogates* (y no por el servidor origen), lo cual contribuye a incrementar la carga asociada a los surrogates. En referencia a las diferentes estrategias de replicación empleadas, se observa una evolución coherente con el modo de funcionamiento *static*. En el modo *all*, en cuanto el porcentaje de replicación alcanza un valor suficiente para que todos los objetos multimedia estén replicados en algún servidor (20% en la Figura 119), todas las solicitudes son satisfechas por los *surrogates* y, en consecuencia, rápidamente se alcanza el valor máximo de carga. Por su parte, con la estrategia *random*, debido a su carácter aleatorio, este máximo se alcanza para valores mayores de replicación (50% en la Figura 119). Por último, con la estrategia *mostPopular*, dado que todos los *surrogates* almacenan los mismos objetos multimedia, hasta que no se dispone de un porcentaje de replicación del 100% no se consigue que todas las peticiones sean gestionadas por los *surrogates*.

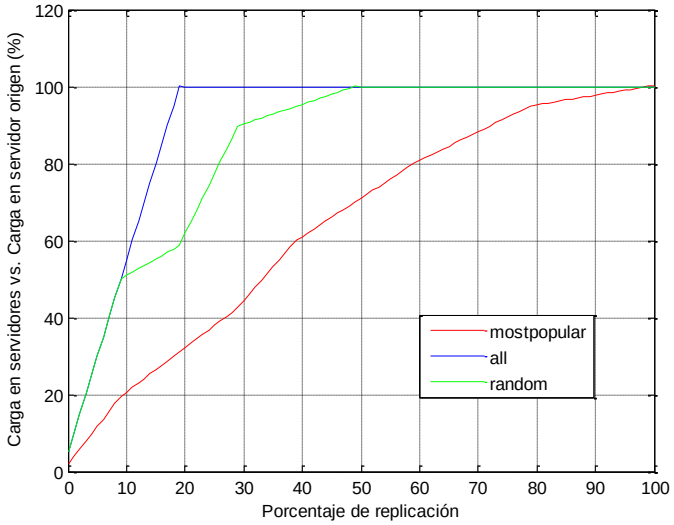


Figura 119. Distribución de carga en los servidores vs. Porcentaje de replicación.

4.2.4.5. Tiempo medio de respuesta vs modo de funcionamiento

El modo de funcionamiento (*static*, *dynamic*) influye en el tiempo medio de respuesta, que se compara en esta sección. La Figura 120 permite analizar el comportamiento de este tiempo medio de respuesta frente al incremento del número de *surrogates* en la CDN (entre 0 y 40 réplicas). Los parámetros de simulación empleados quedaban recogidos en la Tabla 40, con un porcentaje de replicación del 50% y una estrategia de replicación de objetos multimedia *random*.

Puede observarse cómo el retardo experimentado por los usuarios frente al aumento del número de servidores disminuye, como ya se había explicado en la sección 4.2.4.2. Sin embargo, el objetivo principal de esta sección es comparar ambos modos de funcionamiento (*static*, *dynamic*). De esta forma, cuando no hay *surrogates*, todos los clientes acceden directamente al servidor origen y, por tanto, resulta indiferente usar un método u otro. No obstante, para un número de *surrogates* no nulo, se observan mejores resultados (menor retardo) en el modo *dynamic* respecto del *static*, más concretamente en torno a un 11% de diferencia. La explicación de esta reducción radica en el propio modo de funcionamiento. En el modo *dynamic*, los usuarios acceden siempre al *surrogate* más próximo, independientemente de si éste dispone o no del objeto multimedia en caché. A partir de un determinado porcentaje de replicación (50%), el hecho de contactar siempre con el *surrogate* más cercano compensa la penalización de que éste carezca del objeto en caché y, por tanto, precise descargarlo del servidor origen.

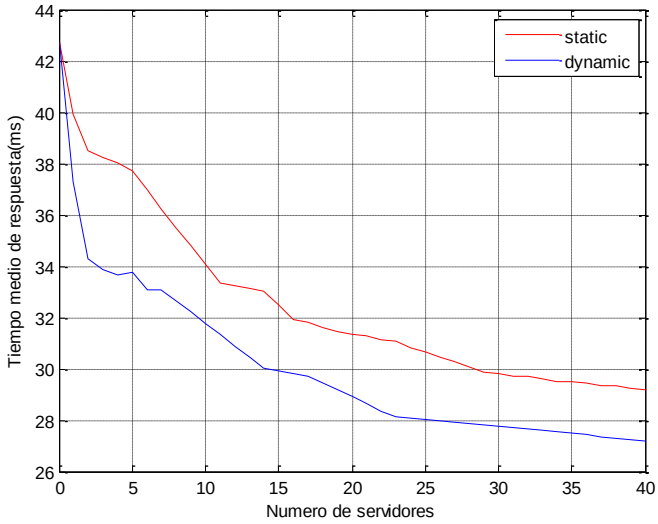


Figura 120. Tiempo medio de respuesta vs. Modo de funcionamiento.

4.2.4.6. Comparativa del tiempo medio de respuesta con y sin CDN

Finalmente y para completar el estudio del modelo de simulación, es importante comparar los escenarios evaluados con y sin CDN, para observar las ventajas de estos sistemas frente a los esquemas mono-servidor. En primer lugar se va a evaluar el tiempo medio de respuesta con respecto al número de *surrogates* disponibles en la CDN y para distintos valores en el porcentaje de replicación de objetos (100%, 70% y 50%).

El método seguido para llevar a cabo este análisis es el siguiente. Por cada cliente, se ha generado otro idéntico conectado al mismo nodo de la red y que ha solicitado los mismos recursos multimedia que el anterior a lo largo de la simulación. Sin embargo, mientras que con el primero se ha seguido el modelo de CDN tradicional, en el segundo cliente las peticiones se han efectuado directamente al servidor origen. Adicionalmente la simulación en cada caso (con y sin CDN) se ha generado con el mismo grafo de red pero en tiempos no solapados, para que no haya ningún tipo de interferencia en el tráfico de un enlace u otro.

Los resultados se ilustran en la Figura 121. Puede apreciarse que, en el modo sin CDN, hay una ligera variación de los tiempos de respuesta, pero su diferencia es mínima y puede deberse a las diferentes simulaciones efectuadas para cada escenario con diferente número de *surrogates*. En cuanto a la evolución del tiempo medio de respuesta con el número de servidores en el modelo con CDN, se aprecia una reducción notable del mismo ante un aumento del número de servidores, como se había indicado en la sección 4.2.4.2.

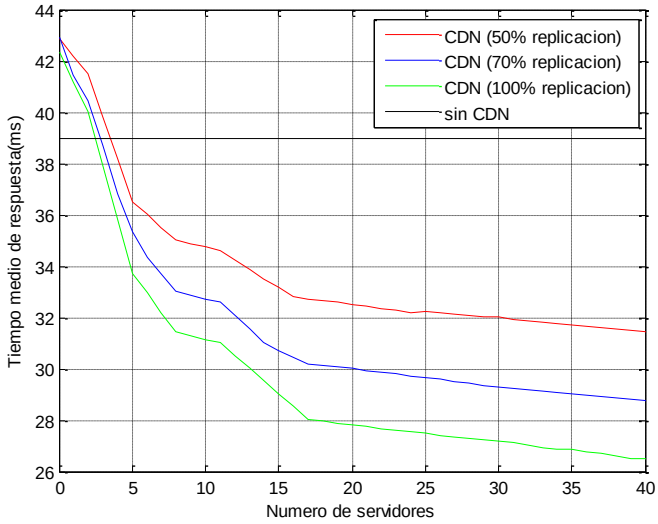


Figura 121. Tiempo medio de respuesta con y sin CDN.

Asimismo, el retardo es menor cuanto mayor es el porcentaje de replicación de objetos, aspecto que también se explicó en la sección 4.2.4.4. La Figura 121 muestra la mejora que se experimenta al emplear una CDN, así como la evolución de la misma con el número de servidores. Así, para el caso de 15 *surrogates* y un 100% de replicación de objetos, se obtiene una reducción del 25,6% en el tiempo de respuesta experimentado por los clientes con respecto a un caso sin CDN. Este valor representa la cota máxima de reducción, ya que típicamente la replicación no será del 100% en un entorno de producción real. Como se observa en la Figura 121, esta disminución es tanto más significativa cuanto mayor es el número de *surrogates* de los que se dispone. Así, para 40 servidores y un porcentaje de replicación del 100% la reducción en el tiempo de respuesta es del 30% (cota máxima). Para un porcentaje de replicación del 50% (entorno más realista) y en el caso de 40 servidores, la reducción es del 10%. Otro aspecto importante a destacar es que la reducción en el tiempo de respuesta es más acusada (mayor pendiente) cuando el número de *surrogates* es menor de 10. Cabe señalar, por último, que en un entorno de pocos servidores (rango [0 2]) el tiempo de respuesta con CDN es mayor que sin CDN. Esto es completamente lógico ya que una CDN con muy pocos servidores es poco útil: el tiempo en procesar la redirección penaliza el tiempo medio de respuesta, y el servidor al que es redirigido el cliente (al haber pocos *surrogates*) puede ser prácticamente equivalente (en términos de saltos y/o retardo) que la redirección al servidor origen.

Se ha evaluado el retardo medio y el *jitter* para dos escenarios con CDN con parámetros realistas (replicación oscilante entre un 30% y un 50%) y otro sin CDN, que se refleja en la Tabla 45. Se puede observar como una CDN reduce estos valores, incluso en escenarios con carga de red moderada, ya que se minimiza el tráfico por los enlaces troncales.

CDN y replicación (30%)		CDN y replicación (50%)		Sin CDN	
Retardo medio	Jitter máximo	Retardo medio	Jitter máximo	Retardo medio	Jitter máximo
200	35	183	29	238	53

Tabla 45. Comparación de retardos y jitter en un escenario con y sin CDN.

4.2.5. Comparación con otros modelos de simulación

Una vez realizado la evaluación de la CDN con soporte de streaming, es interesante compararla con otros modelos de simulación, de forma análoga al modelo de CDN para servicio web. En relación con el modelo analítico, no es posible establecer una comparación adecuada ya que éste está propuesto para un modelo de comunicación web. El modelado de una sesión de streaming (flujo continuo de datos) que permita obtener valores de retardo y *jitter* no se ha realizado en el modelo analítico, por lo que no es posible la comparación en este sentido.

Por otro lado, no existe en la literatura un modelo de CDN que incorpore servicios de streaming con soporte de RTSP/RTP. Si bien CDNsim incorpora como modalidad de tipo de contenido la capacidad de streaming (véase Tabla 22), la distribución propia de contenido se realiza típicamente mediante TCP. CDNsim incorpora una interfaz genérica que se puede modificar para soportar servicios de tiempo real como VoIP o media streaming [Sta_10], pero no existe ninguna implementación ni evaluación en este sentido. Es más, la documentación hace referencia al uso de sumideros TCP (*TCP dumps*) para obtener el tráfico TCP relativo al servicio de streaming, por lo que resulta complicado introducir el modelo RTP desarrollado en nuestro simulador dentro de CDNsim, sobre todo porque no está desarrollado sobre NS-2, sino en OMNeT++.

CDN Simulator, sí está implementado sobre NS-2, pero tal como se describía en la sección 3.3.6.3, el modelo de transferencia de ficheros está basado en HTTP, por lo que no se soporta el streaming sobre RTSP/RTP. Si bien es posible la realización de streaming sobre HTTP, la simulación de ello requiere cambios en el modelo de CDN Simulator, ya que no es posible calcular retardos medios y *jitter* a medida que van llegando los paquetes del servidor al cliente, sino que todo el contenido multimedia (video) sería tratado (simulado) como la descarga de un fichero, y la información proporcionada por RTCP sobre el control de flujo y la calidad de servicio habría que introducirla de alguna forma en el protocolo HTTP en CDN Simulator. Esta tarea no se ha incluido en la presente tesis, aunque sería un aspecto a tratar como ampliación.

4.2.6. Conclusiones parciales del modelo de simulación (streaming)

El modelo de simulación con capacidad de streaming permite ampliar el modelo web disponible introduciendo factores adicionales y nuevos modos de funcionamiento en la caracterización de una CDN, ya que se modelan sesiones de streaming con protocolos de tiempo real (RTSP, RTP). Esto permite evaluar más profundamente el modelo desde diversos puntos de vista, aunque fundamentalmente los parámetros más destacados son la carga media en los servidores (en el análisis de los *surrogates*) y el tiempo medio de respuesta (en el análisis de los clientes). En el caso de las sesiones de streaming, para evaluar el tiempo medio de respuesta hay que estudiar los retardos medios y el *jitter* durante toda la sesión, para evitar paradas en la visualización del contenido multimedia (vídeo).

En el modelo de simulación con soporte de streaming se ha estudiado la carga media de los *surrogates*; el tiempo medio percibido por los clientes y el porcentaje de replicación. El resultado más relevante se relaciona con la evaluación de los retardos medios y el *jitter* durante toda la sesión multimedia, para identificar y cuantificar si se realiza alguna parada. Del estudio se pueden derivar las siguientes conclusiones:

- El aumento de servidores en una CDN de streaming conduce a una disminución en la carga soportada por cada uno de los *surrogates* de forma individual, para un número dado de clientes o sesiones RTSP. Dado que el incremento de carga en los servidores no es lineal conforme aumenta el número de sesiones RTSP soportadas por un mismo servidor, es conveniente repartir la carga entre todos los *surrogates* para que ninguno de ellos llegue a saturarse o a unas condiciones elevadas de carga que no le permitan ofrecer un servicio de calidad (tiempos de respuesta excesivos o una calidad en la reproducción del flujo multimedia con paradas en el cliente).

- A medida que el número de servidores aumenta, se produce una disminución en la latencia percibida por los usuarios. Este hecho es significativo no sólo en el retardo inicial, sino en los retardos medios y *jitter* durante toda la sesión RTSP.

- Para valores de replicación intermedios, la estrategia que resulta más efectiva en términos de un menor retardo es *all*, luego *random* y, finalmente, *mostPopular*. Esto es así cuando el modo de funcionamiento es *static*. En el modo *dynamic*, los resultados son totalmente contrarios por su propia forma de funcionamiento, y la estrategia *mostPopular* presenta un menor retardo medio. Comparando ambos modos de funcionamiento, el modo *dynamic* presenta un menor retardo.

- Es interesante destacar como, en el caso de streaming, los resultados (retardo medio y *jitter*) varían más allá de 15 servidores, a diferencia del modelo de simulación para tráfico web. Esto es debido fundamentalmente a la diferencia en términos de tráfico generado entre ambos servicios (web y streaming), de forma

que la presencia de servidores adicionales tiene un impacto más acusado (beneficioso) en el caso de servicio de streaming que en el caso web.

Los resultados obtenidos en esta sección describen de una forma bastante clara el rendimiento de una CDN con soporte de streaming, también es importante centrarse en las limitaciones del modelo de simulación propuesto. Nuevamente el tiempo de respuesta del Redirector es inmediato, al igual que ocurría en el modelo de CDN para servicio web. La razón de esto es centrarnos en modelar adecuadamente la CDN en general y no en el proceso para la toma de decisión en el encaminamiento.

Otro aspecto susceptible de interés es el número de servidores empleados, que nuevamente llega hasta la cifra de 40. Este valor es adecuado para la topología de red propuesta en nuestras simulaciones, ya que alrededor de este valor los tiempos de respuesta prácticamente no varían, o lo hacen muy ligeramente. No obstante, cabría aumentar la topología de red y el número de servidores para evaluar otros aspectos que no tienen impacto en los resultados y objetivos de la tesis. Se han tomado estos valores en los nodos de la topología porque son los que típicamente se emplean en la literatura científica, y permiten una mejor comparación a posteriori.

Finalmente, un aspecto que no se ha considerado son las pérdidas en los enlaces de comunicación. Estas pérdidas ocasionan cortes en la reproducción de un flujo multimedia, tanto mayor (más perceptible) cuanto mayores sean las pérdidas o, en otros términos, según afecten a los paquetes de datos. En la sección de resultados se ha mostrado cómo resulta relativamente fácil saturar una red (o parte de ella) si se aumenta el número de sesiones RTSP por parte de los clientes. En este sentido, sería interesante estudiar y caracterizar las pérdidas para conocer los límites de una CDN en número de sesiones, para una cierta topología. El estudio de las pérdidas, sin embargo, se ha evaluado en la implementación de la CDN y un escenario real.

En esta sección se ha estudiado y evaluado el modelo de simulación de CDN con soporte de streaming, donde se han introducido nuevos parámetros que permiten caracterizar mejor el comportamiento de una CDN, como son los protocolos RTP y RTSP. Si bien algunos aspectos se han simplificado a efectos de simulación (tiempo de respuesta del Redirector), otros se han implementado de manera bastante realista (sesiones RTSP sobre RTP y control de flujo mediante RTCP) de tal forma que los resultados obtenidos son un punto de partida fiable para proceder a la ampliación de la CDN implementada para que soporte servicios de streaming. Este es el objetivo de la siguiente sección, donde se describirá el diseño de una CDN real y se estudiará su comportamiento de manera análoga a las secciones previas. Esto permitirá validar definitivamente el modelo. Adicionalmente, se comparará la CDN implementada con otras implementaciones similares disponibles, en la medida de lo posible.

4.3. Implementación de una CDN de streaming

4.3.1. Introducción

Si bien en el capítulo 3 se describían tres aproximaciones diferentes (modelo analítico, modelo de simulación e implementación) para el análisis de una CDN de servicios web, el capítulo 4 se centra en el análisis de una CDN que ofrece servicios de media streaming. En la sección 4.2 se ha presentado y evaluado un modelo de simulación en NS-2, que es una versión mejorada del modelo de simulación web (sección 3.3) e incorpora soporte para servicios de streaming mediante los protocolos RTP, RTCP y RTSP. En esta segunda parte, de manera análoga, se parte de la implementación desarrollada para servicios web (sección 3.4) y se amplia para soportar servicios de streaming. En este caso, también se han incorporado los protocolos RTP, RTCP y RTSP. Por otro lado, otra diferencia respecto a la CDN para servicios web es que no se puede considerar una replicación al 100% en todos los *surrogates*, ya que los objetos multimedia (vídeos) requieren un tamaño significativamente superior a los objetos web.

En primer lugar, se describirá brevemente la arquitectura de implementación propuesta. Dicha arquitectura está basada en la arquitectura de CDN descrita en la sección 3.4.2, por lo que simplemente se comentarán aquellos aspectos novedosos que contribuyan a comprender mejor el funcionamiento de la CDN. Posteriormente, se evaluará el rendimiento de dicha CDN para aplicaciones de media streaming, analizando especialmente el comportamiento del algoritmo de redirección implementado. Nótese que este algoritmo es básicamente el mismo que el descrito en la sección 3.4.2.3.3, ya que las variables de utilización de servidores y de la red son las mismas. Sin embargo, en este caso cambia la política de decisión, dando un mayor peso a unas variables que a otras. En última instancia, se comparará este modelo con otras implementaciones reales, como CoDeen [WWW_CoD], si bien cabe mencionar nuevamente que este tipo de comparaciones siempre son limitadas por las diferencias entre ambas implementaciones.

Los objetivos detallados de esta sección son los siguientes:

- Especificar, describir e implementar una arquitectura de CDN de media streaming.
- Diseñar un banco de pruebas para analizar el rendimiento de la CDN para servicios de media streaming.
- Evaluación del rendimiento (algoritmo de redirección) y comparación con otras CDNs de media streaming.

Los principales resultados de este capítulo se han descrito en dos publicaciones propias [Mol_04] [Mol_06].

4.3.2. Descripción de la arquitectura

Los componentes que definen la arquitectura de la CDN de streaming son los mismos que los descritos en la sección 3.4.2 y representados en la Figura 90, si bien se ha añadido el soporte para las comunicaciones RTP, RTCP y RTSP que emplea el servidor de streaming. Es por ello por lo que en esta sección se describirán básicamente las ampliaciones y/o diferencias con respecto a la arquitectura de la CDN para servicio web. Adicionalmente, se entrará con mayor detalle en algunos aspectos comunes a ambas arquitecturas, para una mejor compresión del funcionamiento.

4.3.2.1. Servidor de streaming (DSS)

Como se indicó en el capítulo 2, el concepto de streaming (media) hace referencia al envío continuo de datos en un flujo generado por un servidor y consumido (reproducido) por un cliente a medida que lo recibe. Los principales formatos de vídeo para streaming son: Quicktime, Macromedia Flash, MPEG 4, Windows Media y RealMedia, y el uso de uno u otro depende normalmente de las preferencias del usuario o de la aplicación concreta a usar, ya que la calidad del vídeo es similar en todos ellos.

Como servidor de streaming en la CDN implementada se ha empleado el Darwin Streaming Server (DSS) [WWW_Dar]. Se trata del primer servidor de streaming disponible como software libre por parte de Apple, basado en su equivalente comercial denominado QuickTime Streaming Server (actualmente denominado HTTP Live Streaming). DSS cuenta con soporte para RTP/RTSP y es capaz de realizar streaming en tiempo real sobre varios tipos de formatos multimedia, como H.264/MPEG-4 AVC, MPEG-4 Parte 2 y 3GP. DSS se puede configurar tanto para streaming en vivo (*live*) o bajo demanda, y es capaz de soportar distribución unicast y multicast. Cabe mencionar que Akamai empleó durante un tiempo DSS para distribuir contenido QuickTime y MPEG-4. Actualmente lo emplea para distribuir la versión móvil de Youtube en formato 3GP utilizando el códec H.263/AMR.

La Figura 122 muestra el escenario de interacción básico cuando un cliente desea acceder a un contenido multimedia ubicado en un servidor de streaming. El reproductor ubicado en la parte del cliente es típicamente denominado *media player* mientras que el servidor de streaming se suele denominar también *media server*.

- Inicialmente, el cliente emplea un navegador web para acceder al servidor web donde se listan los contenidos multimedia

- Una vez el usuario selecciona un contenido multimedia y obtiene los metadatos, es redirigido a un servidor de streaming donde interactúa con éste para obtener la información pertinente e iniciar una sesión multimedia. Los pasos típicos en esta sesión RTSP se ilustran en la Figura 122:

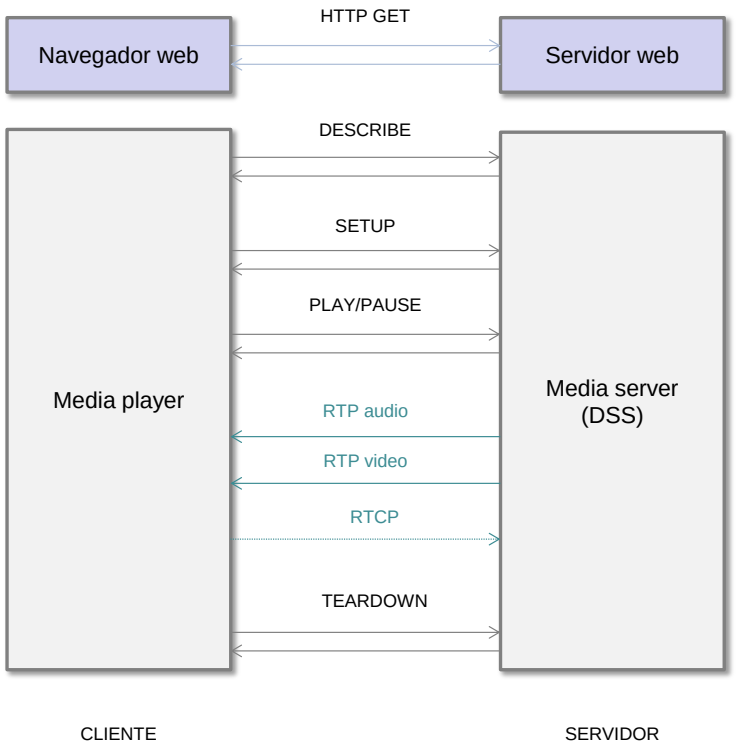


Figura 122. Escenario de interacción RTSP.

- Durante la fase de inicialización RTSP, el reproductor multimedia envía un mensaje DESCRIBE al servidor de streaming. Este método permite obtener una descripción del objeto multimedia (referenciado mediante una URL de la forma rtsp://cdn.upv.es/video.mp4 y empleando típicamente el puerto TCP 554). El servidor devuelve una descripción del recurso solicitado, que básicamente es una lista de los flujos multimedia necesarios para la reproducción, empleando codificación descrita en el protocolo SDP (*Session Description Protocol*), especificado en la RFC 4566.

- El cliente envía un mensaje SETUP donde indica el flujo o flujos multimedia solicitados, así como una especificación del protocolo de transporte, que típicamente incluye un puerto para recibir los flujos (audio y/o vídeo), y otro para los datos RTCP (metadatos). El servidor confirma los parámetros escogidos e indica sus puertos de comunicación. Es necesario configurar cada flujo con SETUP antes de proceder con una petición de PLAY.

- A partir de este momento el cliente ya puede solicitar el flujo mediante un mensaje PLAY. Esta petición provoca que el servidor envíe los flujos especificados a través de los puertos configurados con SETUP. Cada flujo (vídeo y audio) se transportan de manera independiente mediante el protocolo RTP. Adicionalmente, el protocolo RTCP ofrece un mecanismo para controlar cada flujo.

- El cliente puede para la reproducción del flujo multimedia en cualquier momento, mediante un mensaje PAUSE al servidor. Para reiniciar (proseguir) la reproducción, basta con enviar un nuevo mensaje PLAY al servidor.
- Para finalizar la sesión RTSP, el cliente envía un mensaje TEARDOWN al servidor, que le indica que puede liberar los recursos asignados a dicha sesión.

El navegador web debe disponer de un plugin capaz de soportar sesiones RTSP. Para la implementación de la CDN se han empleado dos: un plugin de QuickTime y otro de VLC. Por otro lado, cabe mencionar en la Figura 122 que el *surrogate* consta de dos elementos lógicos, el servidor web y el servidor de streaming (*media server*). Estos dos elementos suelen estar desplegados físicamente en equipos separados para evitar que la carga de uno influya en el otro. No obstante, en nuestra implementación, para minimizar equipamiento, se ha desplegado sobre el mismo equipo físico. Esto no es relevante ya que las pruebas del servicio web (sección 3.4) y de streaming (sección 4.3.5) se han implementado de forma independiente. Adicionalmente, DSS está soportado en varias plataformas (Windows, Linux, OSX), por lo que este aspecto tampoco representa un problema.

4.3.2.2. Comunicación entre módulos. Flujo de mensajes

Los componentes de la arquitectura, descrita ya en la sección 3.4.2 donde hay que añadir el correspondiente servidor de streaming, se comunican mediante una serie de mensajes. Los principales mensajes se ilustran en la Figura 123 y se describen a continuación:

- ***DNSRequest***: Mensaje generado por el servidor DNS solicitando al Redirector la dirección IP del portal óptimo desde el que proporcionar acceso a la CDN a un cliente. El Redirector tres modos de operar: (i) considerar el servidor DNS que realizó la petición para identificar la ubicación del cliente, (ii) considerar el estado de la red y de los portales y (iii) combinar los dos criterios anteriores. En el mensaje ***DNSRequest*** se envía como parámetro el identificador de la CDN a la que el cliente desea acceder.
- ***DNSResponse***: Mensaje enviado por el Redirector al servidor DNS. Este mensaje contiene típicamente una lista de direcciones IP de portales de la CDN, ordenada en función de la disponibilidad de los servidores, de tal forma que el primer servidor de la lista es el que ofrece un mejor servicio.

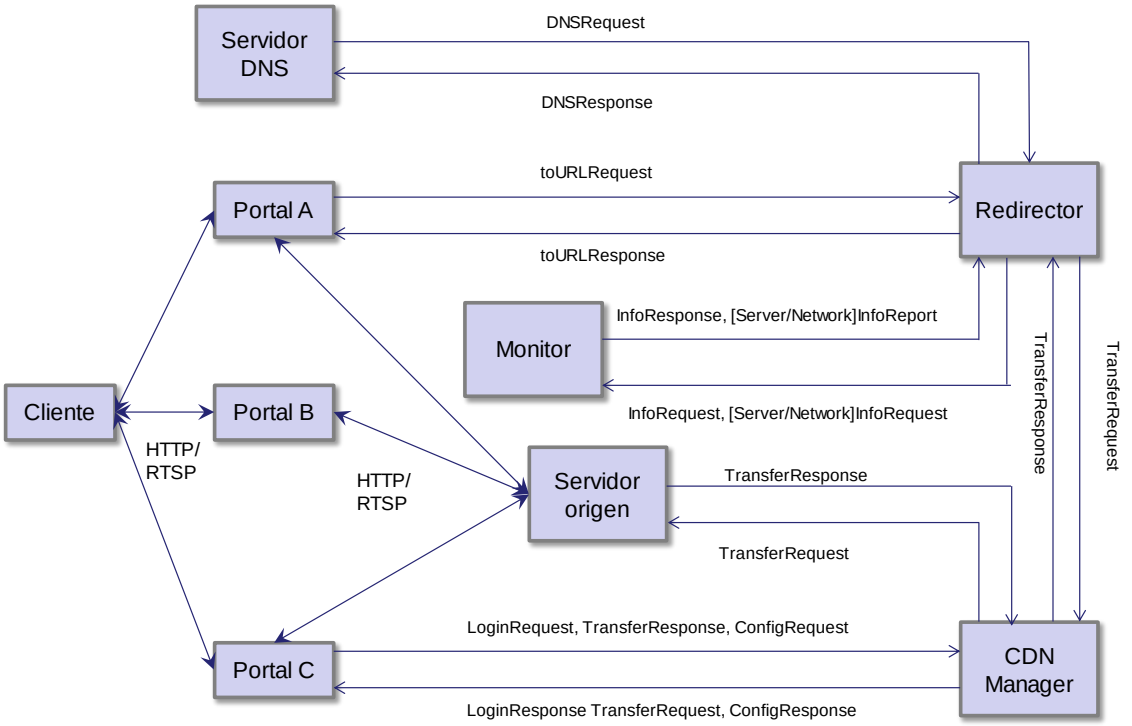


Figura 123. Flujo de mensajes entre módulos.

- ***ToURLResquest***: Mensaje enviado por un portal al Redirector cuando un usuario le pida un contenido. Los parámetros de este mensaje son: la dirección IP del cliente; el identificador del cliente; el identificador del contenido solicitado; el identificador del servidor (portal) que realiza la petición, y el número máximo de URLs que se devolverán en la respuesta (este último parámetro es opcional, y toma el valor 1 por defecto).

- ***ToURLResponse:*** Mensaje de respuesta desde el Redirector a un portal, donde se incluye una lista con las direcciones IP de los servidores que pueden proporcionar al cliente el contenido solicitado. Si no es posible ofrecer el contenido, se devuelve una lista vacía, y mediante un código de estado se indica la causa por la que no se puede ofrecer el servicio.

- ***InfoRequest:*** Mensaje enviado por el Redirector al módulo Monitor para obtener la información necesaria que determina el servidor óptimo para ofrecer servicio a un cliente. Los parámetros del mensaje son la dirección IP del cliente y una lista con los identificadores de los portales candidatos que disponen del contenido solicitado por el cliente. Si se quiere recoger información de todos los servidores de la CDN se utiliza como parámetro el identificador de la CDN.

- ***InfoResponse:*** Mensaje de respuesta del módulo Monitor para indicar al Redirector que la información solicitada ya está disponible en la BBDD. Al recibir este mensaje, el Redirector obtiene y procesa los datos recogidos, y calcula los coeficientes para los portales (*surrogates*) correspondientes.

- ***TransferRequest:*** Este mensaje es enviado por el Redirector para indicarle al Gestor de la CDN (*CDN Manager*) que es necesario realizar una transferencia de un contenido desde un portal a otro. Se envían como parámetros el identificador del contenido a transferir; el identificador del portal destino, y una lista con los identificadores de los portales origen (disponen del contenido), ordenada en función de la disponibilidad de los portales. Este mensaje se propaga entre otros módulos involucrados, como se observa en la Figura 123.

- ***TranferResponse:*** Mensaje de respuesta desde el *CDN Manager* al Redirector para notificar que ya se ha iniciado la transferencia, y el contenido, o al menos un fragmento inicial de éste, ya está disponible en el portal destino. Este mensaje se propaga entre otros módulos involucrados, como se observa en la Figura 123.

- ***ServerInfoReport:*** Mensaje enviado por el módulo Monitor al Redirector cada vez que se obtiene nueva información sobre el estado de los *surrogates*. Al recibir este mensaje, el Redirector calcula de nuevo los coeficientes f y lo notifica con un mensaje de confirmación ***ServerInfoResponse.***

- ***NetworkInfoReport:*** Mensaje enviado por el módulo Monitor al Redirector cada vez que se realiza una nueva medición del estado de la red entre los distintos *surrogates*. Al recibir este mensaje, el Redirector calcula de nuevo los coeficientes g y lo notifica con un mensaje de confirmación ***NetworkInfoResponse.***

- ***ConfigRequest:*** Este mensaje es usado por los portales al iniciarse para solicitar al *CDN Manager* que le proporcione un identificador de servidor, así como un identificador de la CDN a la que pertenece. Como parámetro se envía el nombre de dominio de la CDN.

- ***ConfigResponse:*** Mensaje de respuesta a *ConfigRequest*, donde se incluye el identificador de la CDN (cdnID) a la que está asociado el portal y el identificador del servidor en el que se encuentra el portal (serverID). Este mensaje también puede ser utilizado para responder al *CDN Manager* cuando éste envía un mensaje ***ConfigReport***, utilizado para confirmar que se ha recibido y procesado correctamente el mensaje.

- ***ConfigReport:*** Mensaje enviado por el CDN Manager a cualquiera de los otros módulos para informar de un cambio en alguno de los parámetros de configuración del sistema (direcciones IP, identificadores, etc.).

- ***LoginRequest:*** Mensaje enviado por un portal al *CDN Manager* para autenticar un usuario que inicia una sesión en la CDN. Se incluye en el mensaje las credenciales del usuario.

- ***LoginResponse:*** Mensaje de respuesta a la petición *LoginRequest*, donde se indica si la autenticación es correcta o no mediante un código de estado. Si hay éxito, se inicia una sesión de usuario en el portal. En caso contrario, se muestra un mensaje de error al usuario.

4.3.2.3. Bases de datos del sistema

La información relevante para el correcto funcionamiento de la CDN se almacena en varias bases de datos, cuya estructura y organización se describe en esta sección. Esta información se encuentra distribuida en cuatro bases de datos fundamentales:

- ***Base de datos de usuarios***, donde se almacena información asociada a los usuarios.
- ***Base de datos de contenidos***, que contiene los metadatos vinculados a los contenidos ofrecidos por la CDN.
- ***Base de datos de monitorización***, donde se almacena la información obtenida por el módulo Monitor.
- ***Base de datos de Redirección***, que alberga toda la información necesaria para el módulo Redirector.

Además de estas cuatro bases de datos principales, en cada uno de los portales existe una copia reducida (no hay replicación total de contenidos multimedia) de la información de la base de datos de contenidos. La motivación de distribuir esta información sobre los contenidos en los portales evita que la base de datos principal de contenidos se sobrecargue por las consultas de los usuarios, y reduce el tiempo de respuesta en las consultas. Por otro lado, se requiere actualizar todas las bases de datos cada vez que se haga una modificación en la base de datos principal.

En la Figura 124 se muestra el diseño de la base de datos de CDNs, portales, contenidos y usuarios. Sin entrar en detalle en cada uno de los campos, el punto de partida es la creación de una CDN (tabla *cdns*), descrita mediante un fichero XML (*metafile*). A partir de este momento, se pueden ir agregando servidores (tabla servers), también descritos por su correspondiente fichero XML (*metafile*). Un servidor (*surrogate*) puede pertenecer a una o más CDNs. Por otro lado, conforme se van agregando contenidos (tabla *contents*) se indican una serie de parámetros descriptivos (título, sinopsis, año, etc.) de uso común por parte de un usuario en la búsqueda de contenidos. Nótese que estos datos corresponden parcialmente con la dimensión de contenido e instanciación del modelo de metadatos Dublin Core [WWW_Dcore]. Para la compatibilidad total con este modelo, se ha generado un fichero XML (*metafile*) con todos los campos correspondientes, aunque este aspecto se ha dejado como posible trabajo futuro.

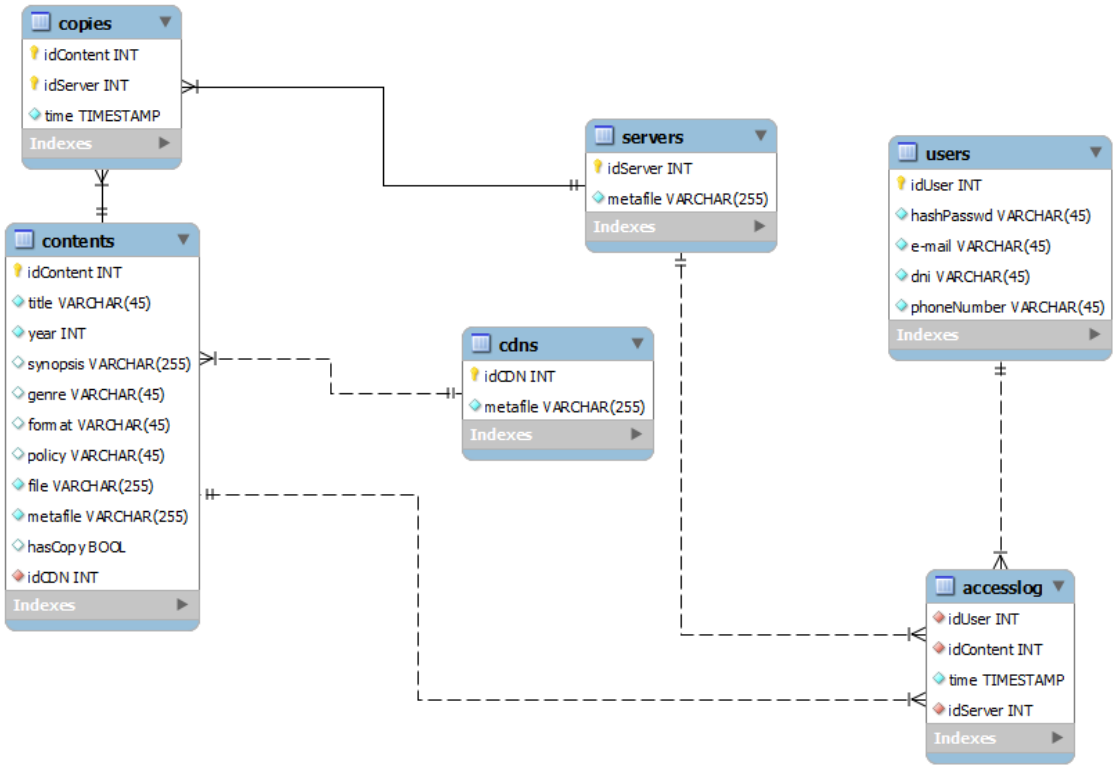


Figura 124. Base de datos de usuarios y contenidos.

La tabla de contenidos también tiene otros parámetros de interés, como son los campos *policy* y *hasCopy*. El primero de ellos permite indicar las políticas de gestión de un contenido (vídeo) en concreto. El segundo campo (*hasCopy*) es válido solamente cuando esta tabla se encuentra ubicada en cada *surrogate*, de forma que cada servidor es consciente si dispone de una copia local y válida de dicho contenido, lo que permitiría servir dicho contenido sin necesidad de solicitarlo al servidor origen o a otro *surrogate*. Otro aspecto importante desde el punto de vista de la gestión es conocer en qué servidores se encuentra una copia de los contenidos (vídeos), junto con una marca temporal vinculada a la actualización (tabla copies). En relación a los usuarios (tabla *users*), estos quedan identificados por una serie de credenciales y datos informativos. Para el seguimiento de los contenidos a los que acceden los usuarios se dispone de una tabla especializada (tabla *accesslog*). Esta tabla tiene una doble finalidad: (i) por un lado, permite conocer aquellos contenidos más demandados por los clientes y emplear políticas proactivas de distribución de contenido, (ii) por otro lado, permite obtener datos para tarificación, aunque este aspecto no es objetivo de esta tesis.

La Figura 125 muestra la base de datos del módulo Redirector, donde se calculan los coeficientes *x*, *y*, *f*, *g* y *h* para cada uno de los *surrogates*, CDNs y direcciones IP correspondientes. Nótese que, en este caso, sólo se están almacenando los últimos valores, para agilizar las búsquedas, pero sería posible almacenar todos los valores para hacer un seguimiento histórico de la carga de los servidores y la variabilidad de la red.

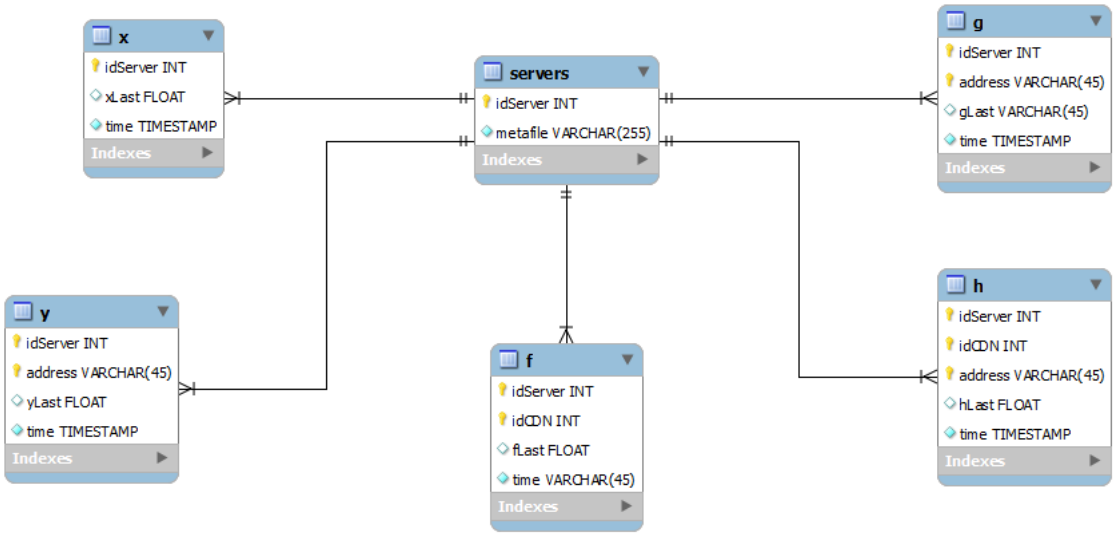


Figura 125. Base de datos del módulo Redirector.

En la Figura 126 se muestra la base de datos del módulo Monitor, que se encarga de obtener las medidas relativas al estado de los servidores y la red. Si bien ciertas medidas se pueden obtener parcialmente mediante la propia implementación de SMNP (ej. número de conexiones), los valores almacenados en estas tablas están normalizados con respecto a valores máximos (carga máxima, conexiones máximas, memoria total, almacenamiento máximo, distancia máxima, ping máximo). Por otro lado, existen otra serie de medidas que es necesario calcular explícitamente, como el número de saltos (tabla *hops*) o la distancia temporal (tabla *ping*), que se obtienen periódicamente. Las marcas temporales (*timestamp*) en cada una de las tablas permiten hacer un seguimiento continuo sobre cada uno de los servidores y conocer su disponibilidad.

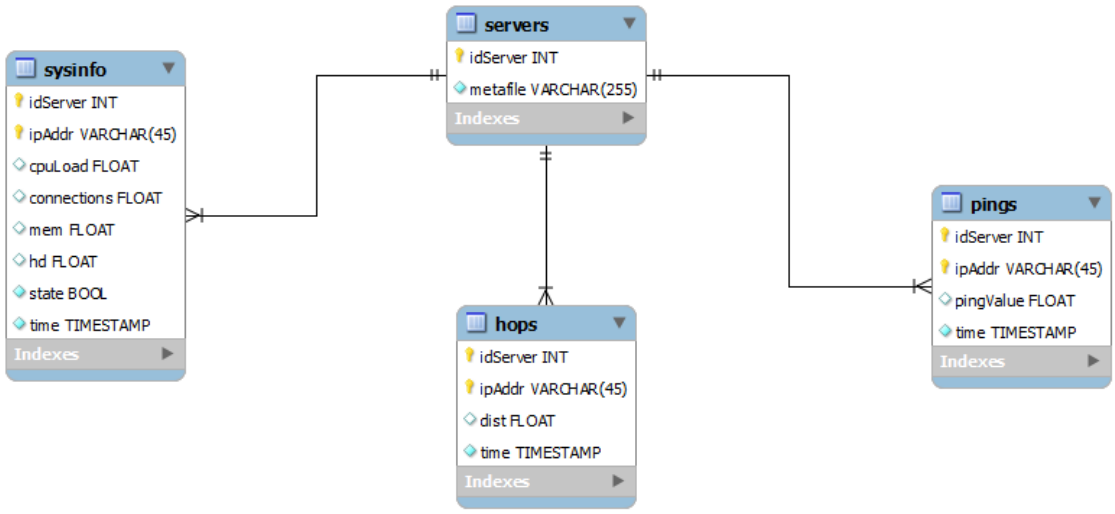


Figura 126. Base de datos del módulo Monitor.

4.3.2.4. Acciones y ejemplos de funcionamiento

En los apartados anteriores (junto con la sección 3.4.2) se han definido las funciones de cada uno de los módulos que forman la arquitectura de la CDN, así como los mensajes usados para la comunicación entre ellos. En esta sección se describen varios ejemplos de funcionamiento de la CDN para completar la explicación de su funcionamiento.

4.3.2.4.1. Acciones e interacciones del cliente con la CDN

Los clientes interactuarán con la CDN desarrollada mediante un navegador web donde seleccionan un contenido multimedia y lo reproducen mediante un plugin adecuado (QuickTime, VLC).

Durante el inicio de sesión en la CDN (primera vez que el usuario se conecta), el usuario proporciona la dirección de acceso a la CDN (por ej. http://cdn.upv.es) en el navegador. Las acciones seguidas se ilustran en la Figura 127. En el primer paso, el cliente manda una petición DNS a su servidor DNS local, que será uno de los servidores DNS de la CDN. El servidor DNS local procesa el mensaje recibido, comprueba que la URL solicitada pertenece a la CDN, y entonces genera un mensaje *DNSRequest* que envía al Redirector. El Redirector procesa el mensaje y selecciona uno de los servidores, priorizando el servidor más cercano al cliente, y generando un mensaje *DNSResponse*. A continuación, el servidor DNS manda al cliente la respuesta a su petición DNS. El cliente se conecta a la dirección IP proporcionada, y solicita la página principal (*index.html*). El servidor (portal A) devuelve al cliente la página solicitada.

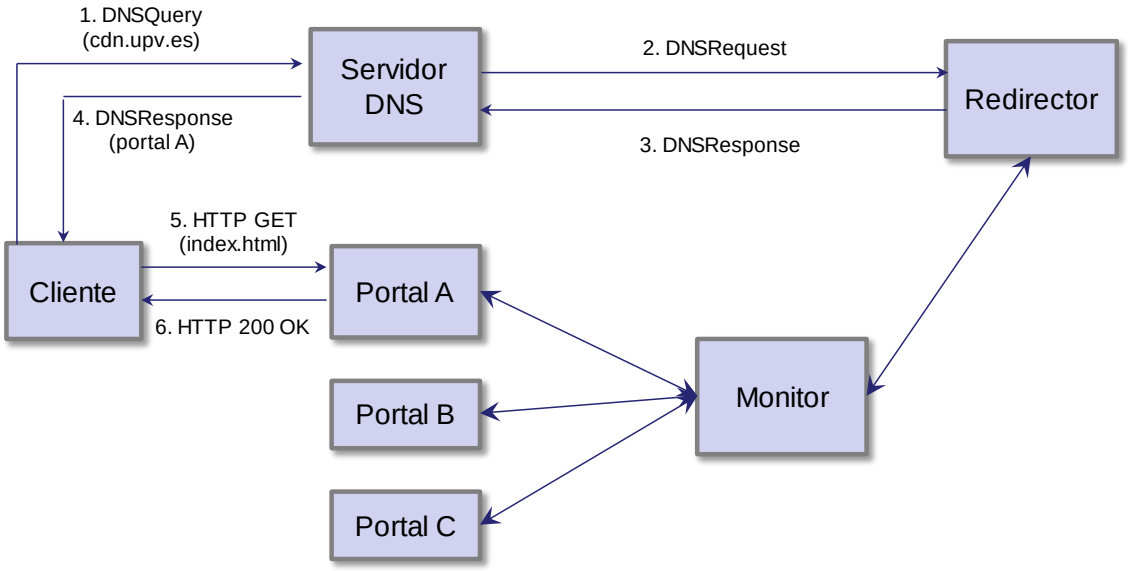


Figura 127. Inicio de sesión en la CDN.

Una vez se ha iniciado una sesión en la CDN, los usuarios tienen acceso a determinados contenidos. Si bien los contenidos abiertos son accesibles por todos los usuarios, hay ciertos contenidos de acceso restringido a usuarios registrados. El registro en la CDN se realiza mediante autenticación web, si bien se pueden introducir otros mecanismos de autenticación como SAML, que representa una futura línea de investigación y se explica en la sección 5.2.3.1.

Los usuarios registrados se organizan en diferentes subgrupos o perfiles para controlar el acceso a los recursos, de tal forma que sólo los miembros de un determinado subgrupo tienen acceso a una serie de recursos específicos. La configuración de estás políticas para cada contenido de la CDN quedaba reflejada mediante el campo *policy* de la tabla *contents* (véase Figura 124). Por ejemplo, en un entorno educativo, se puede hacer que sólo los alumnos matriculados en un determinado curso o seminario tengan acceso a los contenidos asociados a dicho curso (o seminario).

El proceso de autenticación en la CDN se refleja en la Figura 128. El portal realiza tareas de *proxy* en esta autenticación ya que las credenciales de los usuarios y su acceso a los contenidos se gestionan de forma centralizada desde el *CDN Manager*. El usuario rellena un formulario web insertando sus credenciales, y el portal correspondiente genera un mensaje *LoginRequest* al *CDN Manager* aportando esta información. El CDN Manager valida estos datos y responde con un mensaje *LoginResponse*. Si la autenticación es correcta, el portal crea una nueva sesión para el usuario, y se le permite el acceso a los contenidos protegidos, en base a los permisos de los que disponga. Si la autenticación es incorrecta, se presenta un mensaje de error, y se redirige al usuario nuevamente al formulario de autenticación.

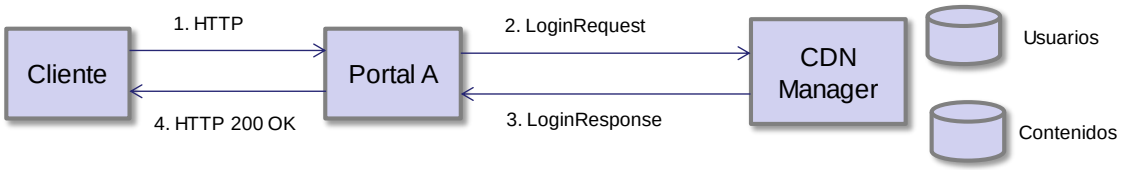


Figura 128. Proceso de autenticación en la CDN.

Cuando el usuario autenticado desea ver uno de los contenidos multimedia (videos), pincha en el enlace correspondiente de la página web. Ante esta petición, el portal genera un mensaje *ToURLRequest* hacia el Redirector, quien decide el servidor más adecuado para ofrecer el video al cliente. El Redirector responde al portal mediante un mensaje *ToURLResponse*.

4.3.2.4.2. Acciones de gestión

Desde el punto de vista de la gestión y administración, y para un correcto funcionamiento de la CDN, es necesario realizar una serie de acciones de carácter periódico. Estas tareas son, básicamente, cuatro:

- ***Inicialización y configuración de todos los módulos***: Antes de poner en funcionamiento el sistema, se deben realizar una serie de tareas, tanto en el *CDN Manager*, como en los *surrogates:*

    - *Definir los servidores que forman la CDN en el CDN Manager*. Esto se hace mediante un fichero de configuración que se carga al iniciar el *CDN Manager*.

    - *Introducir contenidos*. Existe un GUI que permite añadir contenidos y registrarlos en las bases de datos.

    - *Iniciar y configurar servidores*. Cada portal requiere de una cierta información inicial para empezar a funcionar: identificador de servidor, identificador de CDN, y las direcciones IP del Redirector y del *CDN Manager*. Existe un fichero de configuración para este propósito. No obstante, existe un intercambio de mensajes (*ConfigRequest*, *ConfigResponse*) mediante los cuales cada portal queda totalmente configurado (véase Figura 129)

    - *Distribución inicial de contenidos*. Cuando se añade un nuevo contenido en la CDN, las políticas de contenidos generan una serie de copias del fichero multimedia que distribuyen entre los servidores. Estas políticas se describen en un fichero de configuración. En primer lugar, el contenido se copia al servidor origen mediante un intercambio de mensajes (*TransferRequest*, *TransferResponse*), como se muestra en la Figura 129

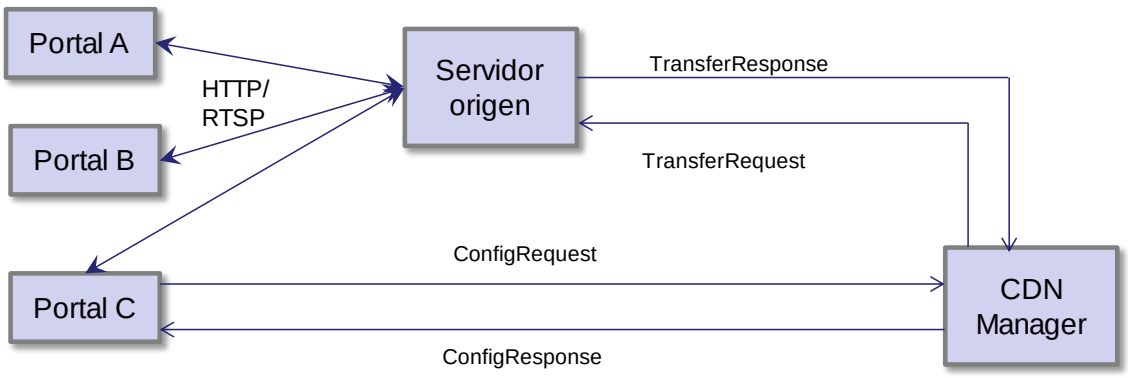


Figura 129. Configuración inicial de portales.

- ***Obtener información sobre el estado de los servidores y de la red***. Esta tarea es llevada a cabo por el módulo Monitor, quién ejecuta de manera continuada un proceso independiente, interrogando a los agentes SNMP instalados en los servidores de la CDN. Cada cierto intervalo de tiempo (fijado por el administrador), se consulta a los servidores (*surrogates*) de forma secuencial. Si un servidor no responde, se registra como inactivo en la base de datos del Monitor hasta que se vuelve a recibir respuesta. El período de monitorización de los *surrogates* depende del tipo de información que se solicita. Para la información sobre el estado de los servidores, el intervalo es menor que para recoger información sobre el estado de la red (comentado en el algoritmo de redirección). Cuando el módulo Monitor finaliza la consulta de todos los servidores, el módulo Redirector realiza una nueva iteración del cálculo de los coeficientes *f*. En la Figura 130 se representa este esquema de comunicación entre portales (*surrogates* RTSP) con el módulo Monitor, así como entre el Monitor y el módulo Redirector.

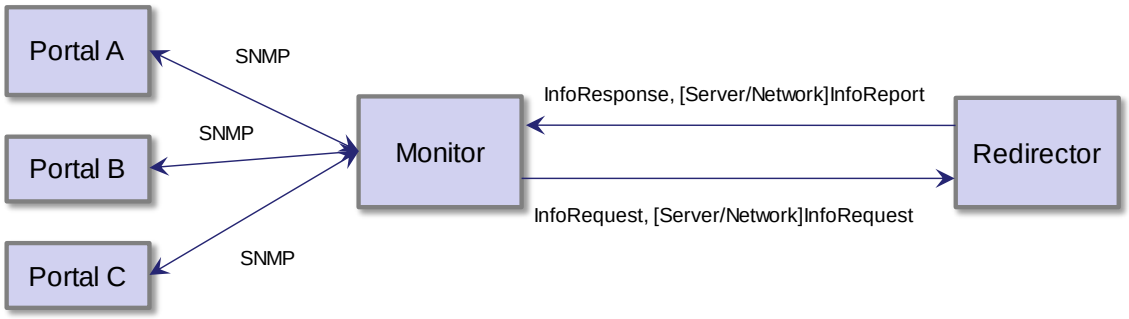


Figura 130. Monitorización de surrogates RTSP.

- ***Gestionar el tiempo de almacenamiento de los contenidos***. En cada *surrogate* se almacenan copias de contenidos (vídeos) en base a dos criterios. Las políticas de distribución inicial de los contenidos (inicialización) y las políticas de funcionamiento que indican el tiempo en que estos videos permanecen almacenados en cada *surrogate*. Se trata de un mecanismo equivalente a las políticas de caché para contenido web descritas en la sección 2.3.1.2, pero en esta ocasión se trata de objetos multimedia (o segmentos de ellos, como se describió en la sección 2.3.4.4). Para la aplicación de estas políticas de gestión, es necesario controlar el tiempo que permanecen almacenados los contenidos en los servidores. Para ello, en las bases de datos de los portales se incluye un campo *timestamp* en la tabla de contenidos, que se actualiza cada vez que un cliente solicita dicho contenido. En la CDN con capacidad de streaming se ha implementado una técnica de tipo *LRU* (*Least Recently Used*), de forma que cuando haya que eliminar un contenido (o un segmento) en uno de los servidores, se considerará el valor del campo *timestamp.*

- ***Registrar los accesos a los contenidos a lo largo de las sesiones de usuarios***. Cada vez que un usuario accede a un contenido, se crea un registro de acceso en la base de datos. A partir de esta información se generan informes y estadísticas, que permiten determinar los contenidos más demandados, en función de varios criterios (por ubicación, por franja horaria, etc.). Esta información de acceso sirve de realimentación para distribuir de una forma más eficiente los contenidos, según la demanda de los clientes. Esta distribución la realiza el *CDN Manager*, que tiene un conocimiento global del uso y acceso a los contenidos en todos los servidores. En la Figura 131 se puede observar como el *CDN Manager* solicita directamente al portal C su registro de acceso (en este caso el módulo Monitor no es necesario) mediante un intercambio de mensajes (*AccessRequest*, *AccesReport*). De forma similar, el *CDN Manager* puede solicitar el registro de acceso al servidor origen; se trata de un caso especial de portal donde los clientes son el resto de portales. El registro de acceso solicitado puede estar acotado en el tiempo, de forma que se obtiene dicho registro en un intervalo temporal determinado o a partir de un determinado instante de tiempo.

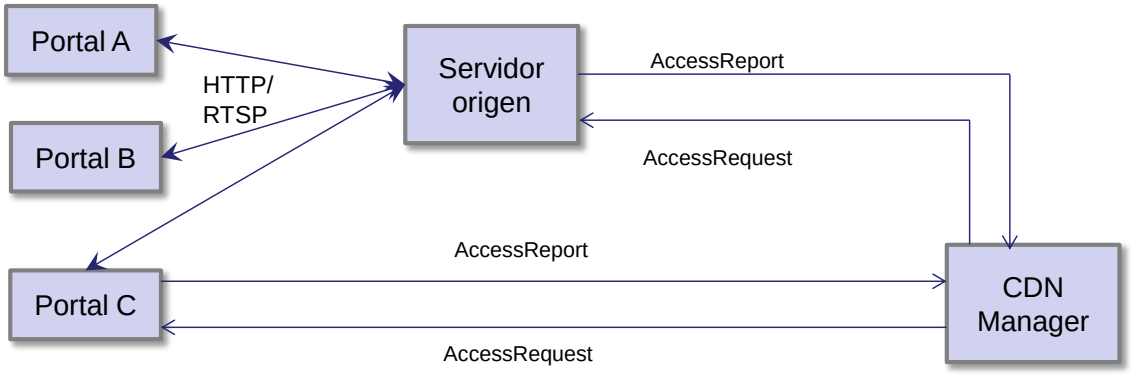


Figura 131. Acceso a los accesos de usuario.

4.3.2.4.3. Acciones iniciadas por el administrador

El administrador de la CDN es el responsable de configurar inicialmente la CDN y supervisar su correcto funcionamiento, evaluando los datos recogidos y modificando o proponiendo nuevas políticas de configuración que optimicen el funcionamiento. De una forma más concreta, las tareas o acciones más relevantes que realiza un administrador son las siguientes:

- ***Introducción y eliminación de servidores***: Una vez inicializada la CDN, se pueden añadir nuevos servidores (*surrogates*), y eliminar otros si se considera necesario. Cuando se agrega un servidor, se registra el servidor en la base de datos del módulo Monitor, y en la base de datos del *CDN Manager*. De forma

análoga, cuando se elimina un servidor se borra de estas bases de datos. Existe también un campo (*flag*) de actividad en estas bases de datos que permiten poner un *surrogate* en (temporalmente) inactivo. Esto permite realizar, por ejemplo, actividades de mantenimiento sin necesidad de eliminar y volver a añadir el mismo servidor ya previamente dado de alta. En la Figura 132 se ilustra un escenario de actuación, donde el administrador interactúa directamente con una aplicación ubicada en el *CDN Manager* para insertar un nuevo contenido (vídeo). Una vez insertado en su base de datos, esta información se propaga automáticamente al módulo Monitor mediante un intercambio de mensajes (*ConfigRequest*, *ConfigResponse*). De esta forma, se puede monitorizar.

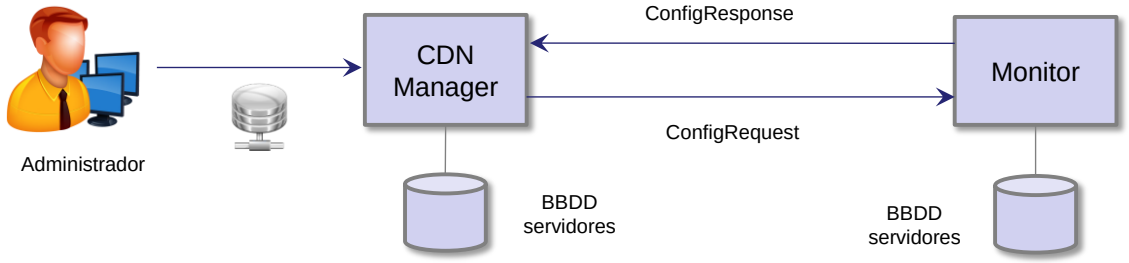


Figura 132. Insertar contenidos en la CDN.

- ***Introducción y eliminación de contenidos***: De forma similar a los servidores, también se pueden introducir y eliminar contenidos. Esta tarea se realiza siempre desde el *CDN Manager*. Una vez se introduce un nuevo contenido, en el primer paso se copia dicho contenido en el servidor origen. En algunos casos, puede ser necesario convertir el formato del contenido a un formato de streaming que permita ser transmitido por los servidores RTSP. Una vez copiado el contenido, se registra en la base de datos de contenidos. Este registro es distribuido y se actualiza automáticamente en todas las bases de datos de los portales. Durante el proceso de registro, el administrador debe especificar a qué grupo o perfil está vinculado el contenido, y en función de éste, se distribuye dicho contenido entre los portales. Se trata de una política inicial de distribución basada en perfiles. En referencia a la eliminación de contenidos, basta con borrar el contenido del servidor central para que se propague la eliminación al resto de *surrogates*. También es posible deshabilitar temporalmente un contenido mediante un campo (*flag*) de la base de datos. En la Figura 133 se visualiza un escenario de actuación donde el administrador introduce un nuevo contenido multimedia mediante la aplicación ubicada en el CDN Manager. En este caso, el vídeo se transfiere al servidor origen y se almacena en su base de datos de contenidos mediante el intercambio de mensajes (*TransferRequest*, *TransferResponse*). Para que todos los portales tengan conocimiento de este nuevo contenido, se envía un mensaje (*UpdateRequest*) desde el servidor origen al resto de portales, quienes actualizan sus respectivas bases de datos. Dependiendo de las políticas de administración y perfiles, algunos portales descargarán también el contenido.

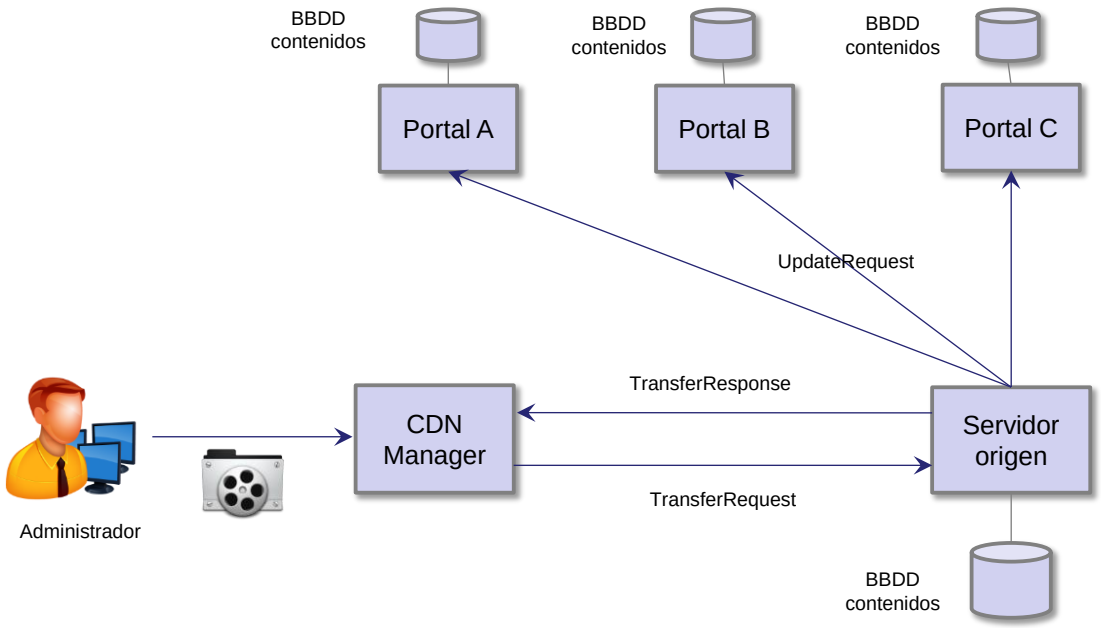


Figura 133. Insertar nuevo contenido en la CDN.

- ***Distribución de contenido forzada***: Si en algún momento crece puntualmente (*flash-crowd*) la demanda de un contenido en concreto, el administrador puede realizar de forma manual la distribución de nuevas copias de este contenido, según su criterio, con el fin de mejorar el servicio ofrecido a los clientes. Aunque existen políticas de distribución y actualización de contenido de forma automática que evitan su intervención, la intervención del administrador se considera imprescindible. La herramienta permite al administrador efectuar distribuciones forzadas y comprobar si el rendimiento del sistema mejora o empeora. Este ejemplo de actuación se observa en la Figura 134

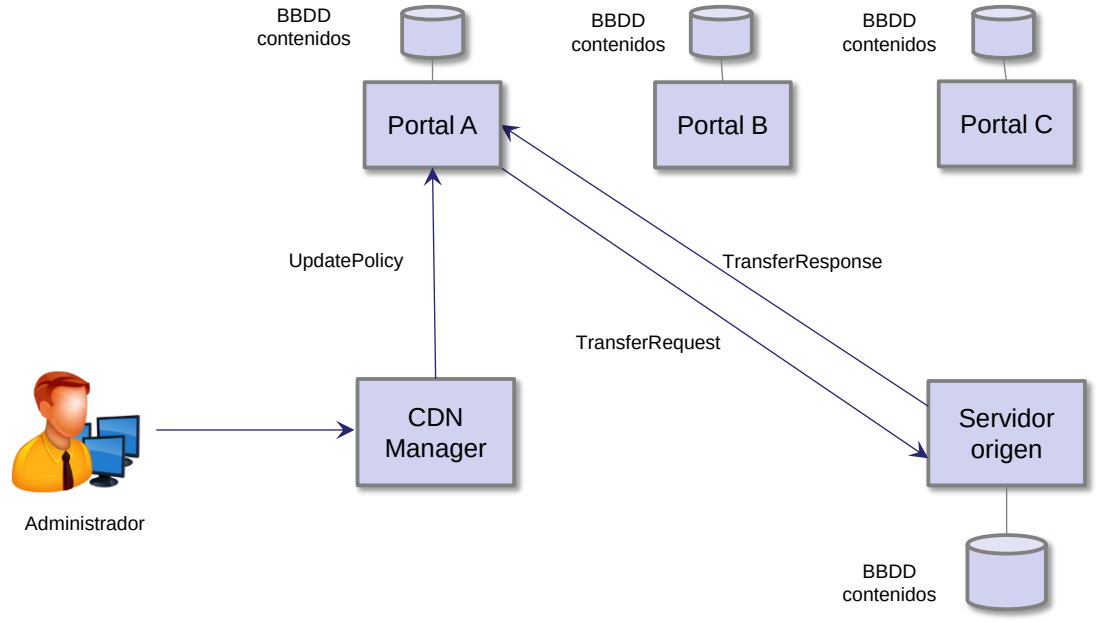


Figura 134. Forzar inserción de contenido en un portal de la CDN.

- ***Modificar políticas de gestión de contenidos:*** El administrador puede, basándose en los informes de rendimiento generados, modificar el grupo de gestión al que pertenecen los contenidos y, de esta forma, aumentar o disminuir la oferta de estos contenidos. También se pueden modificar los distintos parámetros de cada uno de los grupos de gestión, crear nuevos grupos o eliminar los no deseados. En la Figura 135 se ilustra un ejemplo de funcionamiento, donde el administrador actualiza una política de gestión de un cierto contenido (vídeo). Esto se refleja en el servidor origen, concretamente en la base de datos de contenido que tiene vinculada una política de gestión por contenido. Una vez actualizada esta base de datos, la actualización se propaga a los portales. El mensaje empleado en la comunicación es *UpdatePolicy*.

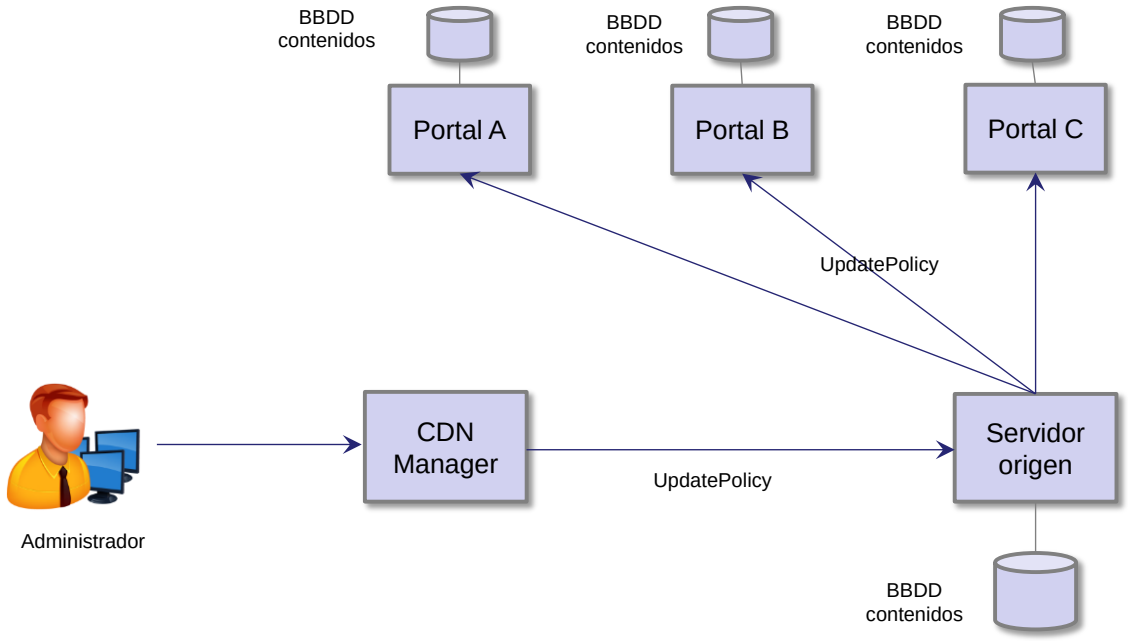


Figura 135. Actualización de políticas de gestión.

4.3.3. Tecnologías empleadas

Para la implementación de la CDN con soporte de streaming se han empleado las mismas tecnologías que las descritas en la CDN para servicio web (véase sección 3.4.3), si bien se han desarrollado las diferentes clases que amplían la arquitectura anterior para soportar el servicio de video streaming. Como servidor de streaming se ha empleado DSS, que se ha, descrito en la sección 4.3.2.1 y las nuevas clases se han desarrollado en Java (JSP) sobre Tomcat v6.X. Las bases de datos, se han desarrollado sobre MySQL v5.X, ampliándose para introducir los servidores de streaming, los contenidos de vídeo y las políticas de gestión. En los servidores de streaming también se ha desplegado un agente SNMP para la monitorización de su estado.

4.3.4. Escenarios de evaluación. Topologías de red y consideraciones

Los escenarios de simulación que se van a emplear en esta sección quedan ilustrados en la Figura 95 y en la Figura 96, que ya han sido empleados en la CDN para servicio web. En esta ocasión, si bien el escenario es el mismo, el tipo de contenido transferido es diferente en términos de ancho de banda (requisito para la red de comunicación) y capacidad de almacenamiento (requisito para los servidores RTSP). Es más, al tratarse de una sesión RTSP de varios minutos de duración, no sólo es necesario garantizar un ancho de banda medio en los enlaces, sino analizar la variabilidad del retardo en estos enlaces (*jitter*) para evaluar cómo afecta en la correcta reproducción del flujo multimedia.

Si bien la Figura 95 representa una red real (red de la UPV); la red representada en la Figura 96 permite un control total de todos los elementos de la red, por lo que se puede actuar de manera directa sobre unos componentes (*routers*, *switches*) u otros y evaluar su impacto en el rendimiento de la CDN. Tal es el caso, por ejemplo, de la influencia de las pérdidas en los enlaces en la reproducción de un flujo multimedia. En esta sección se va a introducir este efecto, y es conveniente tener un control total de la red de comunicación para generarlas y evaluarlas en el equipo de red adecuado y en el momento oportuno.

El escenario de la Figura 96 se ha ampliado para analizar parcialmente ciertos aspectos:

- *Introducción de redes inalámbricas*: este tipo de redes introducen retardos y pérdidas mayores que las redes cableadas, por lo que resulta interesante conocer su impacto en la distribución de contenido multimedia en formato streaming. Como redes inalámbricas se han empleado Wi-Fi y WiMAX, y el escenario de evaluación se describirá con mayor detalle en su sección de resultado correspondiente (sección 4.3.5.2). Este caso de uso ha sido posible gracias a la participación en el proyecto FP6 Multinet (IST 027437) [WWW_Multinet].
- *Introducción de fuentes de video por parte de usuarios*: en los entornos actuales, los usuarios no sólo consumen contenido, sino que también son capaces de crearlo y compartirlo a través de aplicaciones y redes sociales, incluso en tiempo real. En este caso, se propone un caso de uso donde el usuario comparte en tiempo real un video a través de una red inalámbrica. En este caso, se pretende caracterizar los requisitos de esta red de acceso para garantizar una distribución aceptable del contenido. El escenario se describirá con mayor detalle en su sección de resultado correspondiente (sección 4.3.5.3). Este caso de uso ha sido posible gracias a la participación en el proyecto ITEA2 Expeshare (06026) [WWW_Exp].

En la evaluación de esta CDN con capacidad de streaming es importante no solamente obtener parámetros objetivos (carga, tiempo de respuesta, *jitter*, etc.), sino también disponer de medidas subjetivas por parte de usuarios. En este contexto se suele emplear

el concepto de experiencia de usuario (QoE, *Quality of Experience*), empleada habitualmente en entornos de IPTV y definida por la organización de estándares de la industria ETSI TISPAN (*European Telecommunications Standards Institute Telecommunications and Internet converged Services and Protocols for Advanced Networking*) en su norma TR 102 479 [WWW_Tis]. Si bien la QoE se compone de varios parámetros (precio, seguridad, confiabilidad, etc.), en esta tesis nos centraremos en la percepción del vídeo por parte del usuario, y un método de evaluarlo es a través del MOS (*Mean Opinion Score*) [Ven_07]. Existen varios artículos en la literatura [Rub_06] [Can_07] [Pia_09] que proponen diferentes métodos para obtener y analizar el MOS. La relación entre el MOS con el retardo y el *jitter* proporciona una visión clara de la experiencia de usuario: un valor elevado del MOS indica una buena calidad en el video streaming. Para mantener la calidad percibida por el usuario a un nivel aceptable, es importante disponer de realimentación (*feedback*) por parte de estos y por ello se empleará el MOS. Se han realizado tests subjetivos para varios usuarios de diferente sexo y edad. Los valores de MOS se han obtenido para cada flujo de vídeo con variaciones en el retardo y *jitter*, y posteriormente se han analizado los resultados.

4.3.5. Resultados y análisis de la evaluación

A continuación se describirán las diferentes pruebas realizadas, mediante la modificación de uno o varios parámetros en la configuración y la evaluación de otros en el rendimiento de la CDN. Nuevamente, al emplearse la misma configuración de red que en el caso de la CDN para servicios web (véase sección 3.4.4) el número de *surrogates* está acotado. El número de clientes, por otro lado, sí puede ser emulado con programas capaces de generar sesiones RTSP. En la maqueta descrita en la Figura 96 los tres clientes incluyen el programa Videolan [WWW_VLC] y, mediante un sencillo script, es posible generar múltiples peticiones RTSP.

Nuevamente es importante evaluar si el algoritmo de redirección tiene un comportamiento adecuado (equivalente al servicio web) cuando las sesiones son de streaming. Es importante compararlo en ambos modos (web y streaming) y ver cuándo su comportamiento es igual, similar o diferente. Nótese que en el algoritmo se encuentran implícitos los valores de carga en los *surrogates* e incluso distancias de red, que también son dos parámetros de interés. En este sentido, se ha sometido a nuestro sistema CDN a tres tipos de pruebas, de igual forma que para el servicio web:

- Estudio de la redirección mediante DNS.
- Estudio de la redirección basada en contenidos.
- Estudio de los tiempos de respuesta (retardos y *jitter*).

4.3.5.1. Evaluación del algoritmo de redirección

4.3.5.1.1. Redirección DNS

El mecanismo de redirección DNS es el mismo independientemente del servicio final consumido por el cliente (web o streaming). En ambos casos, el cliente conecta con un servidor web, ya sea para visualizar páginas web o un listado de los vídeos disponibles. Es por ello por lo que el algoritmo de redirección se comporta exactamente igual en este caso que en la CDN para servicio web. No se prestará mayor atención a esta redirección puesto que el estudio ya se realizó en la sección 3.4.5.1.

4.3.5.1.2. Redirección basada en contenidos

El objetivo de este apartado es analizar el comportamiento de la CDN cuando un cliente solicita una sesión de streaming en la CDN. En este caso, se han tenido en cuenta las siguientes consideraciones:

- Los *surrogates* no poseen todo el contenido multimedia solicitado (replicación menor 100%). Esta simplificación podía ser válida para contenido web (véase sección 3.4.5.2) pero no para contenido multimedia. Ante esta situación, cabe evaluar si es mejor redirigir al cliente a un *surrogate* que posea el contenido o mandar el contenido al *surrogate* que se encuentra en mejores condiciones para servirlo al cliente, pero que no disponga de dicha copia y debe conseguirla. Esto corresponde, básicamente, a los modos de funcionamiento estático y dinámico que se describían en la simulación de la CDN de streaming (véase sección 4.2). Esta evaluación se realizará en base a los tiempos de respuesta, que se aborda en la siguiente sección (4.3.5.1.3). Se decidió emplear un mecanismo híbrido basado en un umbral de popularidad para cada contenido multimedia que determina el modo de funcionamiento: si el contenido es poco popular, se emplea el modo estático. Si el contenido es muy popular, se emplea el modo dinámico. La forma de determinar si un contenido es popular o no se determina a continuación.

- Los parámetros calculados para seleccionar el *surrogate* óptimo dependen del estado de los servidores y del cliente que lo ha solicitado (su ubicación de red en referencia a los *surrogates*). De este modo, por cada cliente tendremos una gráfica de los coeficientes *y*, *g* y *h* (véase sección 3.4.2.3) de cada *surrogate*. Dependiendo del modo de funcionamiento (estático o dinámico), el criterio para seleccionar el surrogate óptimo variará. En el modo dinámico, se tienen en cuenta todos los *surrogates*. En el modo estático, por el contrario, se consideran únicamente aquellos surrogates que disponen del contenido solicitado. Este es un aspecto que varía con respecto a la CDN para servicio web.

La popularidad de un objeto multimedia (ej. vídeo) está determinada por su patrón de acceso por parte de los usuarios. El análisis de este acceso para valorar o estimar la popularidad es complejo por varios motivos:

- ***Variabilidad temporal***: un contenido que es popular durante un tiempo de observación *T* puede dejar de serlo en el siguiente instante de tiempo *T+1*. Es importante fijar este tiempo de observación T, y típicamente depende de la variabilidad con la que los usuarios generan peticiones a contenidos diferentes.
- ***Variabilidad espacial***: este aspecto hace referencia a la clusterización de clientes. Un contenido popular en una zona puede no serlo en otra. Dicho en otros términos, si el volumen de usuarios es muy elevado, la popularidad a nivel global de un contenido está muy poco correlada con la popularidad del mismo contenido a nivel local. De la misma forma que se clusterizan clientes también se pueden clusterizar *surrogates* para optimizar el rendimiento de la CDN a nivel local en base a un criterio de popularidad local.

Existen numerosos estudios en referencia a la popularidad de objetos para servicios web. Hay casos donde se propone que el algoritmo de reemplazo de los proxy caches incorpore la popularidad de los objetos como parámetro en su política [Shu_00]. En otras ocasiones se proponen modelos de popularidad web basado en objetos de alta y baja frecuencia [Shi_05].

El estudio detallado de la caracterización del contenido y su grado de popularidad no es objeto en esta tesis, por lo que se va a emplear un criterio sencillo y se van a evaluar únicamente aquellos escenarios donde dicho criterio es válido. Se va a suponer una distribución de popularidad inicial que no variará durante la simulación. Esta distribución de popularidad de tipo Zipf, que es válida para servicios web y para servicios de video bajo demanda (VoD) [Shi_04] [Kar_02b] [Ser_00] [Bre_99b]. Se va a partir de los 10 ficheros descritos en la Tabla 39. Empleando Zipf, la probabilidad de ocurrencia en el rango k para una población de N viene determinada por la expresión:

$$f(k,s,N) = \frac{1/k^s}{\sum_{n=1}^{N} \frac{1}{n^s}} \qquad \text{(E 4.4)}$$

Siendo *s* un parámetro que caracteriza a la distribución. En nuestro caso, por simplicidad, se tomará un valor s=1. En cuanto al resto de valores, N corresponde a los 10 vídeos disponibles y k corresponde al rango de aparición. Así pues, se dispone de dos vectores iniciales de tamaño 1x10: el vector de vídeos y el vector de probabilidad de petición de cada uno de ellos determinado por la expresión (E4.4.).

Tras definir la popularidad, hay que definir la distribución inicial de objetos en los *surrogates* RTSP. El proceso se describe en la Figura 136.

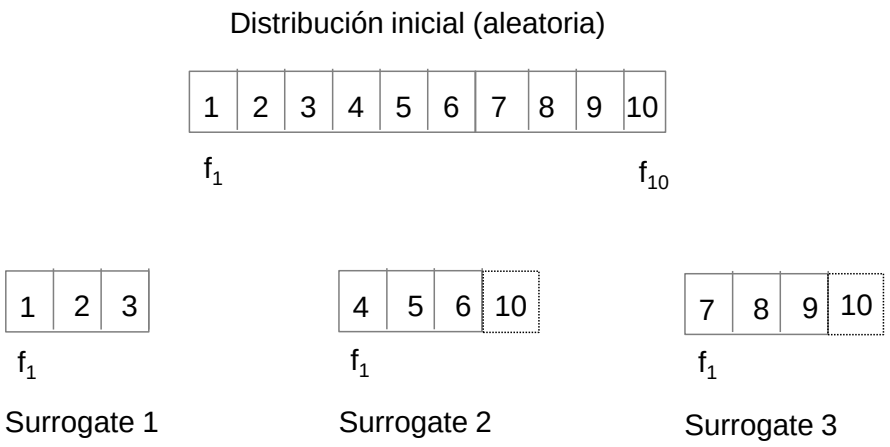


Figura 136. Proceso de inicialización de servidores.

Para evitar la homogeneidad, se emplearán popularidades diferentes en cada una de tres subredes de la arquitectura de pruebas. Para ello, se parte de una distribución inicial de 10 vídeos ordenados de forma aleatoria, como se muestra en la Figura 136. Si establecemos una ley Zipf en este vector de vídeos, el primer elemento será el más popular, con una probabilidad de ser solicitado f_1. El ultimo (décimo vídeo) será el que tenga una probabilidad menor (f_{10}). A partir de este vector inicial, se van a crear los vectores de los vídeos almacenados en cada uno de los *surrogates*. Dado que la replicación es parcial, se va a suponer que cada *surrogate* sólo almacena 3 vídeos. Adicionalmente, el *surrogate* 2 y 3 también almacenan el vídeo 10; su utilidad se mostrará posteriormente en la evaluación.

Evidentemente, una CDN real almacena una mayor cantidad de vídeos, y un *surrogate* también tiene capacidad para más vídeos. Sin embargo, para las pruebas realizadas (evaluar el comportamiento del algoritmo de redirección cuando un contenido es popular y está previamente almacenado o cuando no lo es), se ha considerado suficiente. Adicionalmente, si el número de vídeos es mayor, la probabilidad de los últimos vídeos de la lista es muy reducida (debido a la ley Zipf), y con un tamaño de 10 vídeos es más probable que se soliciten todos los vídeos en algún momento por parte de los clientes durante las sesiones RTSP.

La distribución de los objetos en los tres *surrogates* de la CDN experimental es la siguiente:

- El *surrogate* 1 tomará los tres primeros videos, que almacenará localmente. Estos tres videos corresponden con los vídeos más populares para su subred. El patrón de acceso de los clientes de la subred 1 corresponde en este caso con la distribución inicial.
- El *surrogate* 2 almacenará localmente los siguientes vídeos (4, 5, 6) de la distribución inicial, de forma que se produce un desplazamiento correlativo. Estos vídeos serán los más populares para la subred 2, de forma que el video 4

será el más demandado con una probabilidad f_1. El video menos demandado será el vídeo 3.

- El *surrogate* 3 almacenará localmente los siguientes vídeos (7,8 y 9) de la distribución inicial. Estos vídeos serán los más populares para la subred 3, de forma que el video 7 será el más demandado con una probabilidad f_1. El video menos demando será el vídeo 6.
- Adicionalmente, los *surrogates* 2 y 3 también almacenan el contenido multimedia 10, por motivos que se mostrarán a continuación en el análisis de las gráficas. Básicamente, permite evaluar el correcto funcionamiento del algoritmo de redirección en una situación donde un objeto demandado se encuentra en dos *surrogate*s disponibles.

En esta ocasión los valores de los coeficientes *y*, *g* y *h* se calculan cuando algún *surrogate* está por debajo del umbral inferior de *f* (al igual que ocurría con el servicio web) o bien cuando se opera en el modo estático. Cuando el *surrogate* se encuentra por encima de dicho umbral y se opera en el modo dinámico no se realizan estos cálculos. Es por ello por lo que, en las gráficas que se mostrarán, existen intervalos de tiempo donde estos coeficientes (*y*, *g* y *h*) no se muestran. Téngase en cuenta que por encima de este umbral (modo dinámico) las peticiones se redirigen al *surrogate* local, que será el servidor óptimo. Por debajo del umbral, hay que estimar aquella subred no local que ofrece unas mejores garantías de servicio.

En la Figura 137 se representan las gráficas obtenidas al realizar las peticiones desde un cliente ubicado en la subred 1. Este cliente realiza una serie de sesiones RTSP a lo largo del tiempo, de 20 segundos de duración. En cada nueva sesión el cliente solicita un contenido diferente, de forma que, a los 200 segundos, ha realizado una sesión para los 10 vídeos disponibles en la CDN. Aunque una sesión RTSP es más larga, se toma una duración de 20 segundos por simplicidad para mostrar los resultados, ya que los efectos a analizar en la sesión (retardo, *jitter*, cortes, etc.) ya son peceptibles a los 20 segundos.

Para las tres primeras sesiones (hasta 100 s), el cliente es redirigido al *surrogate 1*, ya que se trata de un contenido popular en la subred 1, y dicho *surrogate* se encuentra descargado (parámetro *f* por encima del umbral). En las siguientes sesiones (a partir de 100s hasta 160 s) se inician sesiones con los objetos 4,5 y 6. Al tratarse de un objeto no popular en la subred 1, se entra en el modo de funcionamiento estático, y el cliente es redirigido al *surrogate* 2. Nótese que, en esta situación, los valores de red (parámetro *y*, *g*) tampoco serían necesarios ya que el *surrogate* 2 es el único servidor que dispone del contenido solicitado. Durante este periodo de tiempo f_2 está por encima de su umbral, por lo que la sesión RTSP puede ser realizada por el *surrogate* 2. El siguiente grupo de sesiones comienza en el instante t=160s hasta t=220, cuando el cliente solicita los contenido 7,8 y 9 de forma consecutiva. En este caso, sucede un caso análogo al anterior (modo estático), pero el cliente es redirigido al *surrogate 3*, pues es el único que dispone de este contenido.

En el instante t=220, el cliente solicita el contenido 10. Este contenido no está disponible en el *surrogate* 1, pero sí está disponible en los *surrogates* 2 y 3. En este caso, sí es necesario recurrir a los valores de red (parámetros *y*, *g*) para estimar aquel que está más próximo al cliente. En este caso, el *surrogate* 2 ofrece mejores resultados y el cliente es redirigido a éste (*g2>g3*). Nótese, no obstante, que los valores de red son similares (aunque con ciertas oscilaciones) entre *y2* y *y3*. En el instante t=240, se vuelve a iniciar una sesión con el contenido 10, pero en este caso es redirigido al *surrogate* 3, ya que sus parámetros de red son mejores (*g3> g2*).

El parámetro *f* también influye en la decisión cuando se excede un umbral. En el instante t=260 se selecciona el objeto multimedia 4. Inicialmente, el cliente debería ser redirigido al *surrogate 2*, pues es el único que dispone del contenido. Sin embargo, este servidor se encuentra sobrecargado (f_2 por debajo de un umbral), por lo que debe ser otro servidor el que atienda la petición. En esta situación, cuando no hay ningún servidor disponible con el contenido solicitado, el cliente es redirigido a su servidor local (si no está sobrecargado), y este obtiene el objeto del servidor origen.

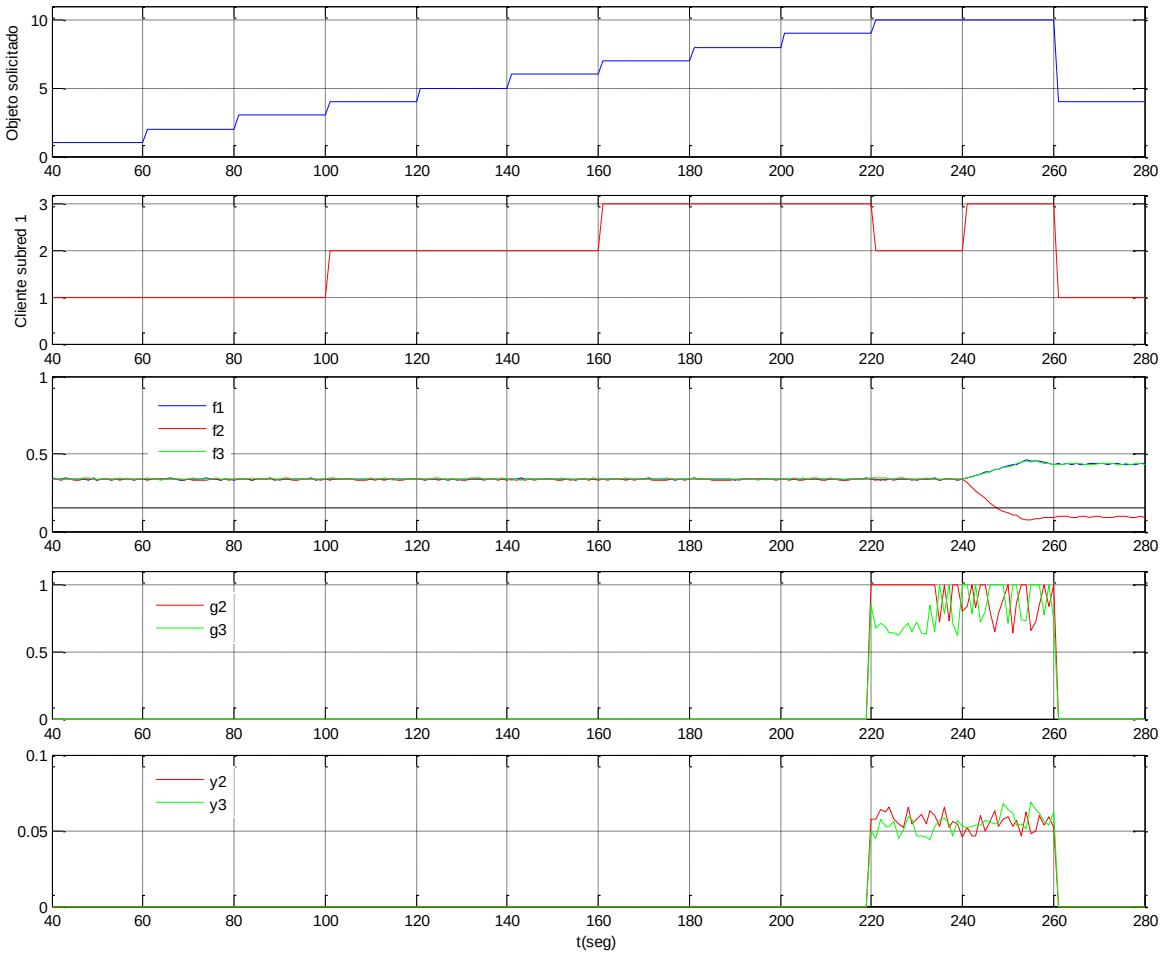


Figura 137. Redirección basada en contenidos. Modos estático y dinámico.

Esto último no implica que el *surrogate* 1 deba eliminar alguno de los objetos populares que ya tiene almacenados (como se podría intuir a partir de la Figura 136); esto implicaría una pérdida de su efectividad. El nuevo contenido obtenido del servidor origen es almacenado adicionalmente en el *surrogate*. Esto quiere decir que existe una memoria caché semi-permanente con los objetos más populares y otra memoria caché para el resto de objetos no necesariamente populares. La naturaleza semi-permanente de los objetos populares está motivada a su posible variación en cada intervalo de monitorización de la popularidad.

4.3.5.1.3. Estudio de los tiempos de respuesta

Uno de los factores críticos en el rendimiento de una sesión de streaming es el efecto del retardo y, principalmente, del *jitter* en la reproducción fluida de un flujo multimedia. La reproducción puede sufrir paradas y cortes debido a que los paquetes de datos en la sesión RTSP no llegan a tiempo al receptor, con lo que la fluidez en la reproducción se pierde. El hecho de que los paquetes de datos no lleguen al receptor puede ser por un motivo de tiempo excesivo (retardo y *jitter*) o simplemente por la aparición de pérdidas en el canal de comunicación.

Aunque el proceso de selección del *surrogate* introduce un retardo adicional, el propio hecho de seleccionar el *surrogate* más adecuado asegura que el cliente correspondiente va a estar conectado al servidor que menores tiempos de respuesta le proporcionará durante la sesión RTSP. Típicamente será aquel topológicamente más cercano, pues al minimizar las distancias en términos de saltos se minimiza el retardo, el *jitter* y la probabilidad de pérdidas.

En esta sección se mostrarán los incrementos en los tiempos (*retardo, jitter*) al introducir los mecanismos de selección de *surrogate*. Estos valores hay que compararlos con una situación en la que no existe redirección, para evaluar su impacto en la calidad del vídeo. Como se podrá comprobar, el aumento del retardo es relativamente pequeño si se considera las mejoras que se introducen al reducir el *jitter*.

El estudio del tiempo medio de respuesta en las consultas DNS y en las consultas a bases de datos ya se realizó en la secciones 3.4.5.3.1 y 3.4.5.3.2. El resultado es el mismo que para una sesión de streaming, por los que nos centraremos en los casos de uso particulares de la CDN con soporte de streaming.

Para calcular los valores de retardo medio y *jitter* medio se va a tomar una situación similar a la de la sección 4.3.5.1.3 y se calcularán los valores que se obtendrían para los tres *surrogates* que conforman la CDN para sesiones de 20 s. De esta forma, se puede ver el beneficio en la elección del *surrogate* óptimo. Este escenario se ilustra en la Figura 138 donde en cada momento analizado en la sección anterior (4.3.5.1.3) se obtienen los valores de retardo medio (r_1, r_2, r_3) y *jitter* medio (j_1, j_2, j_3).

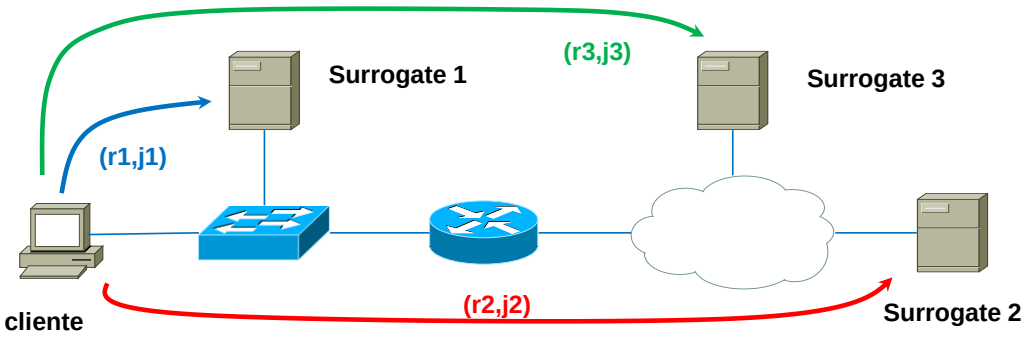


Figura 138. Escenario de evaluación de retardo y jitter.

Los resultados para el retardo se ilustran en la Figura 139. A partir de estos valores, se pueden calcular los valores medios del retardo y del *jitter*, reflejados en la Tabla 46.Evaluación del retardo y jitter medio. Número de pruebas: 20.

En el período de tiempo 40-100s, el cliente solicita el objeto multimedia 1 y es redirigido al *surrogate* 1. Puede apreciarse que los valores medios del retardo y *jitter* son menores que al contactar con los *surrogates* 2 y 3. La diferencia sustancial en estos valores es debida a que los *surrogates* 2 y 3 no disponen del objeto multimedia 1, por lo que deben obtener previamente del servidor origen y esto supone un retardo adicional. En las dos siguientes sesiones (60-100s), se observa como los valores son ligeramente mayores en los *surrogates* 2 y 3 (véase Figura 139). Esto es debido a que los tres servidores se encuentran en subredes próximas. En una CDN real, las subredes pueden estar alejadas y la diferencia puede ser considerable. Otro aspecto que se observa en la Figura 139 es que el *jitter* es mayor durante la primera sesión para los *surrogates* 2 y 3; es decir, el contacta con el servidor origen también afecta al *jitter*.

En el período de tiempo 100-160, el cliente solicita el objeto multimedia 4 y es redirigido al *surrogate* 2. Puede apreciarse nuevamente que los valores medios del retardo y *jitter* son menores que al contactar con los *surrogates* 1 y 3. La diferencia está motivada porque los *surrogates* 1 y 3 no disponen del objeto multimedia 4, por lo que deben obtener previamente del servidor origen y esto supone un retardo adicional. En las dos siguientes sesiones (120-160s), se observa como los valores son ya parecidos a los valores del *surrogate* 2. Es más, el *surrogate* 1 ofrece durante este intervalo valores menores de retardo que el propio *surrogate* 2 (véase Figura 139).

Periodo	r_1 (ms)	j_1 (ms)	r_2 (ms)	j_2 (ms)	r_3 (ms)	j_3 (ms)
40s-100s	42.1	4.2	90.3	10.2	90.3	10.4
100s-160s	87.7	7.7	48.9	5.1	94.1	8.4
160s-220s	89.3	7.4	93.2	8.3	50.1	5.2
220s-260s	87.4	7.4	54.2	5.2	52.1	4.9
260s-280s	42.9	4.3	80.8	9.7	51.8	5.2

Tabla 46.Evaluación del retardo y jitter medio. Número de pruebas: 20.

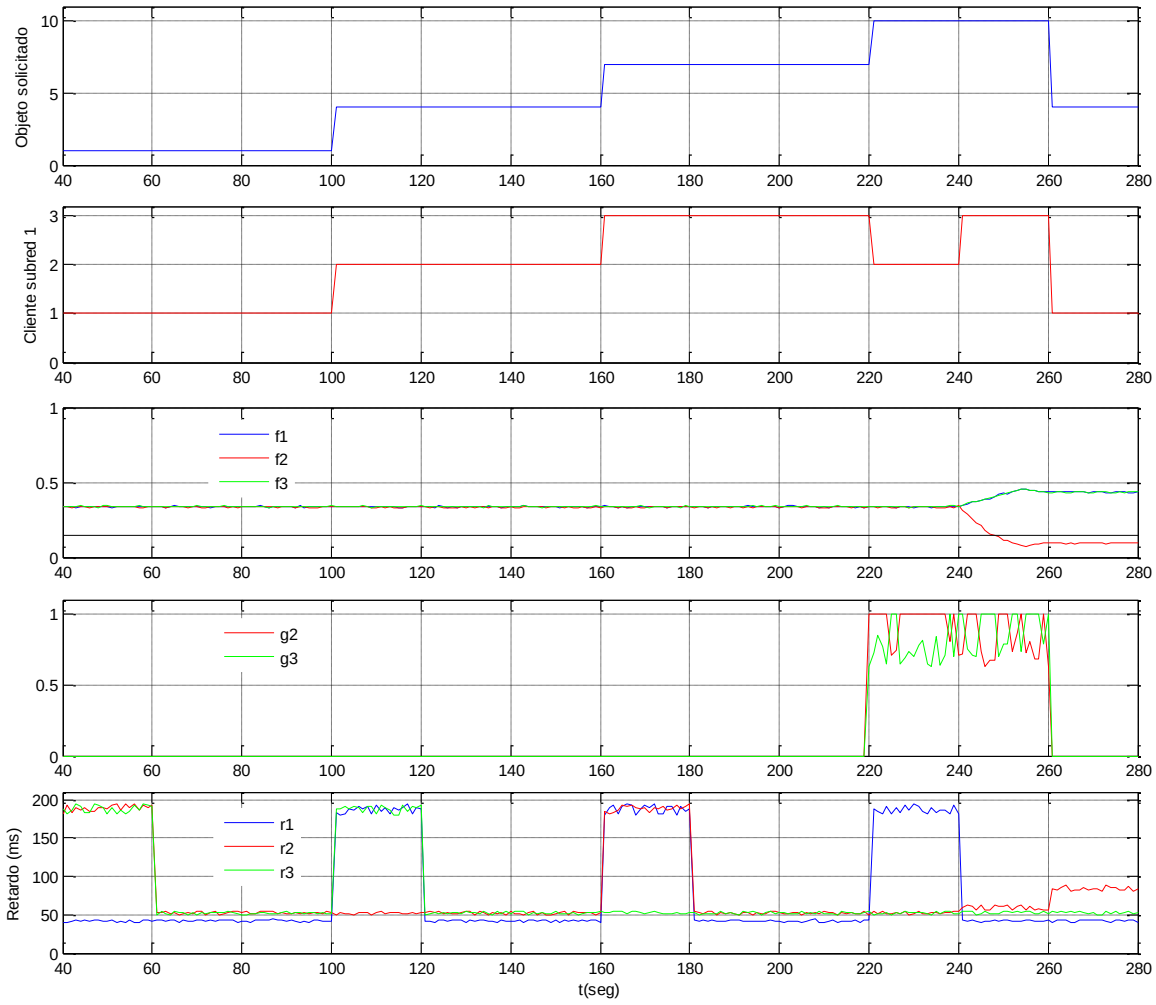


Figura 139. Medidas de retardo (CDN de streaming).

Esto es debido a que el *surrogate* 1 está ubicado en la propia subred que el cliente 1 y ya dispone del objeto multimedia 4. Lo que se pretende demostrar con este hecho no es que el algoritmo de redirección funcione de forma incorrecta, sino que sólo puede conseguir resultados *subóptimos*. Nótese que el contenido 4 no es un contenido popular para el clúster de clientes 1, por lo que posiblemente será descartado en un cierto intervalo de tiempo de la caché del *surrogate* 1 para ir llenándola con contenido popular. Por otro lado, si bien para el cliente 1 se hubiera podido conseguir un mejor tiempo de respuesta para las sesiones en el intervalo 120-160s, se aumenta el retardo para otros clientes ubicados en el mismo clúster de clientes que estén solicitando un contenido más popular localmente que el contenido 4 y no esté disponible en ese momento porque se ha almacenado el objeto multimedia 4. Es decir, el algoritmo de redirección optimiza los valores en media para todos los clientes, no para uno en particular.

En el período de tiempo 160-220, el cliente solicita el objeto multimedia 7 y es redirigido al *surrogate* 3. Los resultados en este caso son análogos al caso anterior, intercambiando los valores entre los *surrogates* 2 y 3.

En el período de tiempo 220-260s, el cliente solicita el objeto multimedia 10 y es redirigido en primera instancia al *surrogate* 2, al presentar unas condiciones de red ligeramente mejores que el *surrogate* 3 (*g2>g3*). Puede apreciarse que los valores medios del retardo y *jitter* son similares en los *surrogates* 2 y 3. El *surrogate* 1 no dispone del objeto multimedia 10, por lo que debe obtenerlo previamente del servidor origen y esto supone un incremento en el retardo significativo. Por otro lado, en el período de tiempo 240-260s, el cliente vuelve a solicitar el objeto multimedia 10 y es redirigido en esta ocasión al *surrogate* 3, al presentar éste unas condiciones de red mejores que el *surrogate* 2 (*g3>g2*). El *surrogate* 1 ya dispone del objeto multimedia 10, por lo que su valor de retardo es incluso menor que el de los surrogates 2 y 3. Sin embargo, en valores medios (contando la sesión anterior) los valores son peores (véase Tabla 46). Otro aspecto a destacar es que, en esta última sesión, el valor de *r2* y *j2* aumenta con respecto a *r1* y *j1* debido a la carga experimentada por el surrogate 2 (véase Figura 139).

En el período de tiempo 260-280s, el cliente solicita el objeto multimedia 4 y es redirigido al *surrogate* 1. Esto es debido a que el *surrogate* 1 ya dispone de dicho contenido en su caché por la sesiones previas. Por otro lado, nótese que, en principio, la redirección debería ser hacia el *surrogate* 2 (quien dispone de dicho contenido por ser popular), pero como se encuentra cargado (*f2* por debajo del umbral en la Figura 139), el cliente 1 sería redirigido al *surrogate* 1. Se observa en la Tabla 46.Evaluación del retardo y jitter medio. Número de pruebas: 20.

que el *surrogate* 1 ofrece los mejores resultados. El *surrogate* 2 ofrece tiempos elevados pese a disponer del contenido, precisamente por estar cargado. Si los *surrogates* 1 y 3 no dispusieran del objeto multimedia 4 en el momento de la solicitud (por ejemplo, porque la política de reemplazo hubiera descartado dicho objeto al no ser localmente popular) los retardos obtenidos hubieran sido mayores, similares a los casos anteriores donde era necesario contactar con el servidor origen.

4.3.5.2. Tiempo medio de respuesta con redes inalámbricas

En la arquitectura de CDN que se ha desplegado, se han empleado redes cableadas (véase Figura 96), concretamente Ethernet, por lo que el rendimiento es bastante bueno en términos de retardo, a no ser que la propia red se sature. En este caso (saturación),

cualquier red sin mecanismos de reserva de recursos funcionará mal. Sin embargo, resulta interesante introducir las comunicaciones inalámbricas (Wi-Fi, WiMAX, 3G), ya que su rendimiento en términos de retardo (y *jitter*) es muy variable incluso en condiciones de red moderadas (por la naturaleza radio del canal de comunicación), y puede afectar seriamente a la percepción del usuario en una sesión de streaming. Es por ello por lo que se ha introducido dos redes de acceso Wi-Fi en la arquitectura implementada. La utilidad y finalidad de estas dos redes es doble:

- Por un lado, permite evaluar cómo el canal de comunicación radio afecta a la calidad del vídeo en la sesión de streaming, ya que el retardo y el *jitter* pueden variar considerablemente con respecto a un medio cableado (Ethernet). Adicionalmente, se puede establecer una transición (*handover*) de un cliente entre dos redes Wi-Fi y estudiar cómo afecta a la sesión.
- Por otro lado, al disponer de dos redes Wi-Fi, es posible establecer un criterio para seleccionar la red Wi-Fi que mejor rendimiento efectivo ofrezca en ese momento. El estudio en este aspecto forma parte de una adaptación a esta tesis del trabajo realizado dentro del proyecto de investigación Multinet [WWW_Multinet].

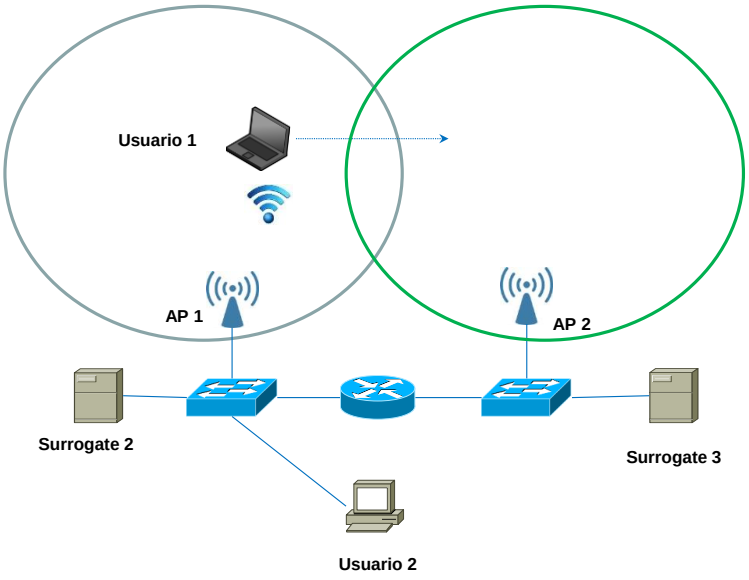


Figura 140. Introducción de redes inalámbricas (caso de uso 1).

En el primer caso de uso, se va a evaluar el retardo y el *jitter* en el escenario presentado en la Figura 140. El usuario 1, ubicado en la zona de cobertura del punto de acceso 1 (AP 1) solicitará el objeto multimedia 10, disponible tanto en el *surrogate* 2 como en el 3. Inicialmente, es redirigido al *surrogate* 2. Posteriormente, el usuario 1 comenzará a moverse penetrando en el área de cobertura del punto de acceso 2 (AP 2), iniciándose un *handover*. Tras este proceso, que típicamente representa una interrupción, el usuario 1 recupera el flujo de vídeo con el *surrogate* 2. No hay conmutación al *surrogate* 3 hasta que no se inicie una nueva sesión. Para poder comparar los retardos y el *jitter*, se

ha ubicado en el escenario de pruebas de la Figura 140 el usuario 2, que también solicita el objeto multimedia 10. De esta forma, se puede apreciar mejor el efecto de la red inalámbrica con una red cableada. Adicionalmente, el usuario 2 genera una carga variable hacia el usuario 1 con la finalidad de ir saturando la red Wi-Fi 1 de forma incremental.

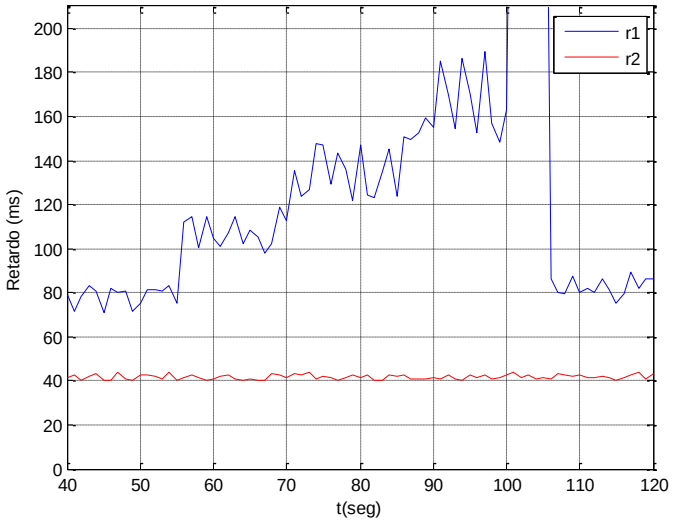


Figura 141. Retardo obtenido en la evaluación Wi-Fi (caso de uso 1)

Los resultados obtenidos se pueden apreciar en la Figura 141. Para el caso del usuario 2, el retardo no varía a lo largo de todo el experimento, y se obtienen valores similares a los ya obtenidos en la sección 4.3.5.1.3. Para el usuario 1, se observa que tanto el retardo como el *jitter* son crecientes a medida que aumenta la carga en la red Wi-Fi 1. En el instante t=100s se produce el *handover*, y durante unos 5 segundos el flujo multimedia se corta. El cliente 1 tiene que obtener una nueva dirección IP (ya que ha cambiado de red) y tiene que volver a conectarse al servidor 1. En este caso, el cliente ya conoce la dirección IP del servidor RTSP a contactar, por lo que no hay redirección al *surrogate* 3. Puesto que la red Wi-Fi 2 no está cargada, el retardo durante los tiempos 105-120 s es similar a los obtenidos al principio de la sesión (40-55 s), aunque ligeramente superiores porque se ha de atravesar una red adicional.

Un aspecto que no se ha tratado anteriormente es el aspecto de las pérdidas y su efecto en la calidad del video. Cuando se produce la pérdida de un paquete UDP durante una sesión RTP, el paquete típicamente no es retransmitido y puede afectar perceptiblemente a la reproducción del flujo de vídeo, dependiendo del tipo de trama (I, P o B) a la que corresponda la pérdida. El efecto puede ser desde algunos macrobloques de la imagen claramente borrosos o distorsionados o directamente el corte de la imagen. En los canales de comunicación radio también se producen pérdidas de ráfaga *(burst loss*) lo cual siempre implica un corte en la reproducción del flujo porque afecta a varias tramas de la secuencia de vídeo. En las pruebas realizadas en el escenario de la Figura

140, la tasa de pérdidas percibida por el usuario 2 era del 0 %, mientras que la de la red Wi-Fi oscilaba entre un 2% y un 5%, dependiendo de la carga de la red.

Para minimizar el retardo y el *jitter* en el escenario anterior, se pueden aplicar dos tipos de soluciones:

- *Introducción de QoS en la gestión Wi-Fi*: Si bien una red radio donde exista mecanismo de contienda introducirá un retardo mayor que una red cableada, existen mecanismos de marcado de paquetes en el punto de acceso para priorizar unos flujos sobre otros, de forma que se pueden identificar aquellos con requerimientos de tiempo real y darles una prioridad. Esto es lo que se denominó Wi-Fi Multimedia (WMM) o WME (*Wireless Multimedia Extensions*), basado en el estándar 802.11e, incorporado al estándar 802.11 en 2007.
- *Introducción de multihoming*: otra forma de reducir el retardo y el *jitter* es disponer de varias redes inalámbricas y seleccionar en cada momento aquella que se encuentre más descargada y pueda ofrecer un mejor ancho de banda efectivo. El trabajo realizado está basado en el proyecto de investigación FP6-IST Multinet [WWW_Multinet]. Un esquema de su empleo se puede apreciar en la Figura 142.

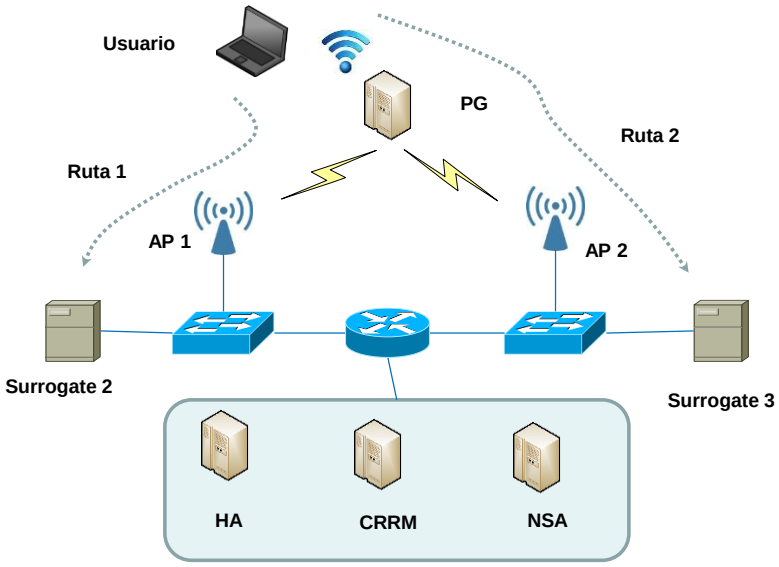


Figura 142. Mejora del retardo en redes inalámbricas con multihoming.

En el escenario representado en la Figura 142 se dispone de varios componentes adicionales en la arquitectura para gestionar el correcto funcionamiento. El PG (*Personal Gateway*) es un dispositivo multimodo que da soporte a una red de usuarios, ya sea de forma inalámbrica o cableada. Este dispositivo permite registrar múltiples direcciones IP (*MCoA, Multiple Care-of-Address registration*) sobre un HA (*Home Agent*). Se trata de una arquitectura extendida de Mobile IPv6 denominada NEMO (*NEtwork MObility*) descrita en la RFC 3963 del IETF. Otros componentes de la arquitectura lo constituyen:

- ***Módulo CRRM (Common Radio Resource Management)***: encargado de gestionar todos los recursos radios disponibles de forma jerarquizada
- ***Módulo NSA (Network Selection Algorithm)***: encargado de decidir la mejor ruta para un flujo de comunicación determinado. Este elemento formaría parte en nuestro sistema del algoritmo de redirección.

En este experimento se ha simplificado el algoritmo NSA y se han tomado como criterio ciertos umbrales de retardo para ejecutar la conmutación entre redes, desde la red Wi-Fi 1 a la red Wi-Fi 2. Al estar basado en IPv6, se ha empleado VLC tanto en la parte servidora como en la cliente para establecer las sesiones RTP. Los resultados se muestran en la Tabla 47. Se observa que el tiempo de conmutación es muy reducido (inferior a 1 s) y el retardo medio se ve mejorado durante la sesión multimedia. Si la sesión tiene una mayor duración, la reducción es mayor, al ser valores medios.

	Th (r>80ms)	**Th (r>120 ms)**	**Th (r>160 ms)**
r (previo)	77 ms	103 ms	132 ms
r (posterior)	76 ms	87 ms	121 ms
T (conmutación)	0.7s	0.6s	0.7s

Tabla 47. Mejora del retardo con multihoming inteligente. Número de pruebas: 20.

4.3.5.3. Introducción de fuentes de vídeo en los clientes

En la CDN desplegada con soporte de streaming se ha considerado en todo momento contenido multimedia previamente almacenado (bajo demanda). Sin embargo, también es posible introducir contenido en vivo (*live streaming*) a partir de eventos en tiempo real (véase sección 2.3.4.1). Para ello se requiere un elemento capaz de capturar vídeo en tiempo real y distribuirlo al servidor origen para que sea, a su vez, difundido a los *surrogates*, si fuera necesario. Aunque existen varias formas de llevar a cabo este proceso, consideramos que uno de los casos más prácticos y atractivos es permitir que el propio usuario capture video a través de su móvil y lo comparta con una comunidad social en tiempo real. El uso de redes sociales otorga una mayor relevancia a los usuarios, quienes se convierten en productores y consumidores al mismo tiempo. En ocasiones es común la denominación *prosumers* (contracción en inglés de los términos *producer* y *consumer*) para referirse a los usuarios.

Desde el punto de vista de los requisitos de red, se corresponde con un caso bastante desfavorable, ya que afecta a la transmisión de vídeo en el acceso del cliente quien, haciendo uso de su móvil, típicamente compartirá el vídeo mediante conexión inalámbrica (Wi-Fi o 3G). Este video debe alcanzar a un *surrogate* de la CDN, que pueda distribuirlo al resto de *surrogates*, si fuera necesario. Este servidor (o servidores) RTSP serán los que finalmente distribuyan el vídeo al usuario final, que posiblemente

use otro móvil con conexión inalámbrica. Este escenario permite caracterizar los requisitos y limitaciones de la red. Cabe mencionar que, si bien la sesión de streaming puede ser encapsulada bajo RTSP, tampoco tiene un sentido completo ya que, al ser contenido en vivo, no es posible avanzar ni retroceder.

Desde el punto de vista de los requisitos de la CDN, generar un contenido en vivo requiere actualizar las tablas que gestionan los contenidos para que soporten esta modalidad. Nótese que, en este caso, no es el administrador quien añade un nuevo contenido, sino el propio usuario. Es por ello, por lo que el usuario debe disponer de ciertos permisos (perfil en la CDN) para añadir contenido en tiempo real. En realidad, al tratarse de *live streaming*, realmente lo que se añaden son los metadatos de dicho flujo, ya que el vídeo no se almacena en la CDN, simplemente se requiere, a nivel técnico, un punto de entrada en la CDN por parte del usuario productor y un punto de acceso en la CDN por parte del usuario consumidor. La descripción del vídeo (metadatos) sí debe ser introducido y almacenado en la CDN, al menos mientras dure la sesión en vivo.

Desde el punto de vista de la propia innovación, resulta novedosa la capacidad de compartir contenido de vídeo en tiempo real con los usuarios. Las redes sociales, como Facebook, permiten compartir un vídeo una vez grabado, pero no en tiempo real. Curiosamente, este vídeo es almacenado en una CDN operada por Akamai (puede verse la URL *akamaihd.net* tanto en los vídeos como en las imágenes/fotos de Facebook). La transmisión de video en vivo sólo está accesible mediante herramientas de chat (Skype, GoogleTalk) pero suele ser una comunicación *one-to-one*.

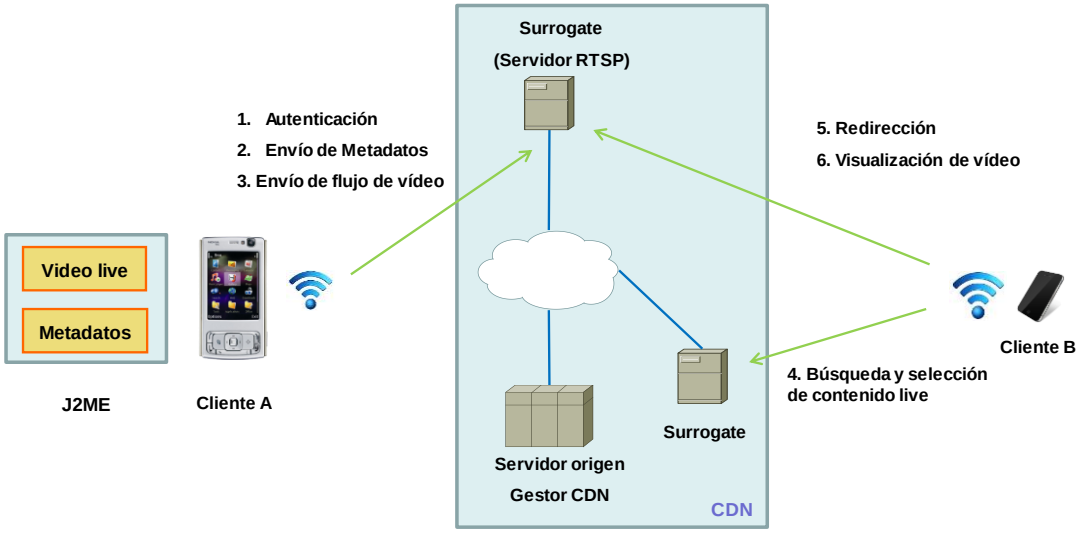


Figura 143. Distribución de video en tiempo real por un cliente.

La arquitectura para el soporte de streaming por parte de los usuarios se muestra en la Figura 143. El proceso seguido es el siguiente:

- El usuario A ejecuta una aplicación determinada en su móvil. Esta aplicación ha sido desarrollada por el tesitando dentro del proyecto de investigación *Expeshare* [WWW_Exp] y ha sido adaptada a los requisitos de la CDN.

- Durante el inicio de la aplicación, el usuario A se registra en la CDN como un usuario autorizado para generar sesiones en vivo (véase Figura 144).
- El usuario A rellena un pequeño formulario antes de proceder a capturar y transmitir el vídeo (nombre, ubicación geográfica, comentarios, etc.). La mayoría de ellos son optativos, pero son almacenados en la CDN para describir la sesión, de tal forma que un usuario remoto B tendrá un mejor conocimiento de dicho vídeo antes de visualizarlo.
- El dispositivo móvil del usuario A empieza a capturar vídeo a través de la aplicación desarrollada, que tiene acceso a la cámara. Dicho vídeo es transmitido en tiempo real a un servidor concreto de la CDN (servidor RTSP), que sirve de punto de entrada (véase Figura 144).
- Un usuario remoto B accede a la CDN (véase Figura 144) mediante su móvil y observa que hay un vídeo en vivo de un conocido (usuario A). Accede a los metadatos asociados de dicha sesión y le resultan de interés, por lo que decide iniciar una sesión RTSP para visualizar dicho vídeo. El usuario B es redirigido a un *surrogate* capaz de soportar dicha sesión.

Nótese cómo la CDN desarrollada no sólo daría soporte a una aplicación de video bajo demanda, como se ha descrito en secciones previas, sino que puede dar soporte a una red social para distribuir contenido (*live media*) entre usuarios; en esta ocasión, el usuario B accedería a una red social y, a través de ella, a la sesión en vivo.

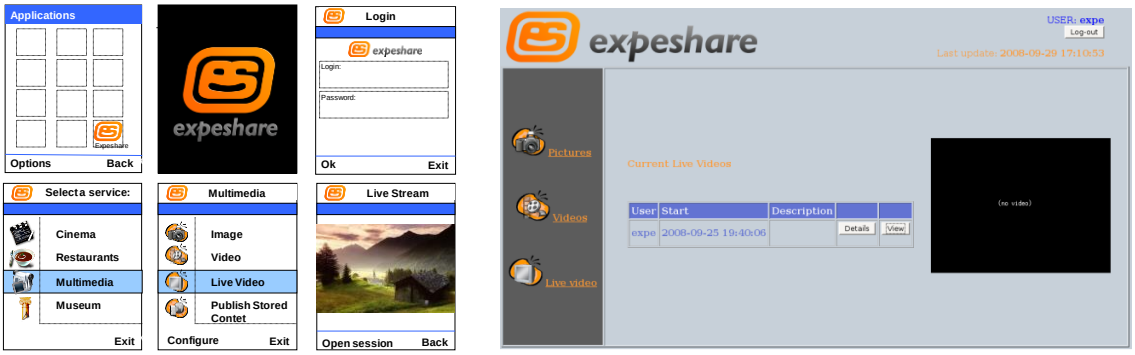


Figura 144. Aplicación móvil del productor y aplicación web para acceder al flujo en vivo.

El experimento en este escenario consiste en evaluar el retardo introducido, fundamentalmente por el efecto de las redes inalámbricas. Para ello, se han ubicado dos sondas en el escenario, tal como se ilustra en la Figura 145. La red inalámbrica empleada fue IEEE 802.11g, soportada por la mayoría de móviles. Se han empleado dos redes Wi-Fi diferentes para productor y consumidor con la finalidad de no causar interferencias. La primera sonda (sonda A) está ubicada justo en la red de acceso del usuario productor, de tal forma que se puede evaluar el degradado inicial que sufre el flujo de vídeo; cualquier usuario de la CDN podrá visualizar, en el mejor de los casos,

el flujo con esta calidad. Si en la sonda A la calidad es deficiente, lo será para todos los usuarios de la CDN, ya que el resto de equipos (*surrogates*, *switches*, *routers*, etc.) sólo pueden contribuir a aumentar el retardo y el *jitter*. La segunda sonda (sonda B) está ubicada en la red de acceso del usuario consumidor. En este caso, se pretende evaluar el degradado introducido por el servidor RTSP y la red cableada de la CDN. Finalmente, en el cliente también se ha evaluado el retardo y el *jitter*, pues corresponde a la calidad recibida en el extremo e incorpora además el efecto de una red inalámbrica adicional. Tanto la sonda A como la sonda B se han aplicado a través de los switches Catalyst 2950 ya documentados en la Tabla 31. Estos switches disponen de una herramienta para monitorizar puertos (*monitor session*) de forma que es posible obtener el flujo de video sin interferir en el tráfico de red.

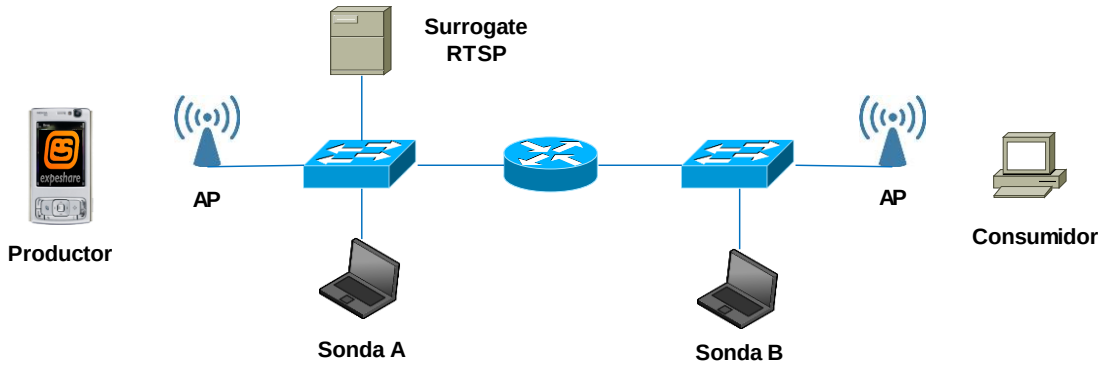


Figura 145. Escenario de evaluación para streaming desde el cliente.

Para este experimento se han generado tres secuencias de vídeo con diferentes requisitos en términos de ancho de banda (véase Tabla), de forma que se puede evaluar su impacto al atravesar la primera red Wi-Fi desde la sonda A. Por otro lado, dada la variabilidad de las comunicaciones inalámbricas, se ha generado la sesión para cada vídeo en tres instantes de tiempo diferentes: el vídeo 1 en los instantes t_1, t_2 y t_3; el vídeo 2 en los instantes t_4, t_5 y t_6 y el vídeo 3 en los instantes t_7, t_8 y t_9.

Video	Resolución	Cuadros/seg	Ancho de banda	Formato
1	320x240	30	1.2 Mbps	MJPEG
2	320x340	20	800 Kbps	MJPEG
3	320x240	10	400 Kbps	MJPEG

Tabla 48. Características de los vídeos generados por el cliente.

Los resultados de este escenario se muestran en la Tabla 49. La sonda A evalúa el flujo entre el productor y el *surrogate*, mientras que la sonda B evalúa el flujo de vídeo del *surrogate* al consumidor. Se trata de flujos diferentes. Las principales observaciones son las siguientes:

- El ancho de banda teórico de IEEE 802.11g es de 54 Mbps sin embargo el ancho de banda efectivo obtenido es mucho menor y capaz de afectar el envío de tráfico de vídeo en tiempo real. Si bien el retardo medio varía, el efecto más

perjudicial es el *jitter*. Los vídeos de mayor calidad, si bien presentan un retardo parecido a los vídeos de menos calidad, sufren un mayor *jitter*.

- La red cableada apenas introduce pérdida de la calidad en el vídeo (sonda B). Si bien el retardo es elevado, esto es debido al servidor de streaming, que necesita transcodificar el flujo del productor y generar un nuevo flujo para el usuario consumidor. Esto requiere el uso de buffers que incrementan el retardo. Sin embargo, el *jitter* es bastante reducido.

- El consumidor obtiene un vídeo aparentemente degradado en base a los valores de retardo medio y *jitter*. La red Wi-Fi afecta de una forma similar que en el caso del productor. Los vídeos de mayor calidad sufren una mayor degradación.

	Sonda A		**Sonda B**		**Consumidor**	
Video	**Retardo medio (ms)**	**Jitter medio**	**Retardo medio (ms)**	**Jitter medio**	**Retardo medio (ms)**	**Jitter medio**
Video 1 (t_1)	**92**	**10**	**72**	**6**	**143**	**14**
Video 1 (t_2)	**94**	**12**	**69**	**4**	**139**	**12**
Video 1 (t_3)	**91**	**9**	**68**	**5**	**149**	**14**
Video 2 (t_4)	**84**	**7**	**70**	**3**	**138**	**9**
Video 2 (t_5)	**88**	**9**	**71**	**4**	**131**	**9**
Video 2 (t_6)	**87**	**8**	**69**	**5**	**134**	**8**
Video 3 (t_6)	**77**	**7**	**67**	**5**	**129**	**8**
Video 3 (t_7)	**72**	**7**	**72**	**6**	**130**	**8**
Video 3 (t_8)	**74**	**8**	**70**	**4**	**131**	**8**

Tabla 49. Evaluación del retardo y jitter (streaming en el cliente). Número de pruebas: 20.

Para finalizar el estudio, conviene destacar de qué forma una CDN ayuda o mejora la calidad en un sistema de difusión de vídeo por parte de los clientes, con relación a un escenario sin CDN. Las mejoras son básicamente dos:

- *Reducción del retardo y jitter*: al seleccionar el servidor RTSP que recibe el flujo del productor desde un punto cercano en términos de topología de red, se reduce el retardo (y el *jitter*). En un escenario sin CDN, el productor tendrá que atravesar un mayor número de subredes hasta alcanzar un único servidor, aumentando el retardo e incluso la probabilidad de pérdidas. Esto se ha evaluado mediante el escenario representado en la Figura 146, donde se propone que el productor envíe el flujo al *surrogate* A (caso de una CDN) o a un servidor remoto (en este caso *surrogate* B). Para un mayor realismo, se ha introducido un retardo variable en el router desde la interfaz 1 al 2, de forma que se emula el comportamiento de Internet, y se ha evaluado el desfase de tiempos entre el video generado y el visualizado, así como el número de cortes detectado producidos por el *jitter* para una sesión de 20 segundos. En este caso, no existe retardo variable desde la interfaz 2 a la 1 del router, para que el consumidor perciba un retardo similar para ambos *surrogates*. Los resultados se pueden observar en la Tabla 50.

- *Aumento de la escalabilidad*: una vez el vídeo ya se ha insertado en la CDN, su propia capacidad de escalabilidad permite soportar un mayor número de clientes mediante la difusión del vídeo en directo (flujo) a otros *surrogates* RTSP.

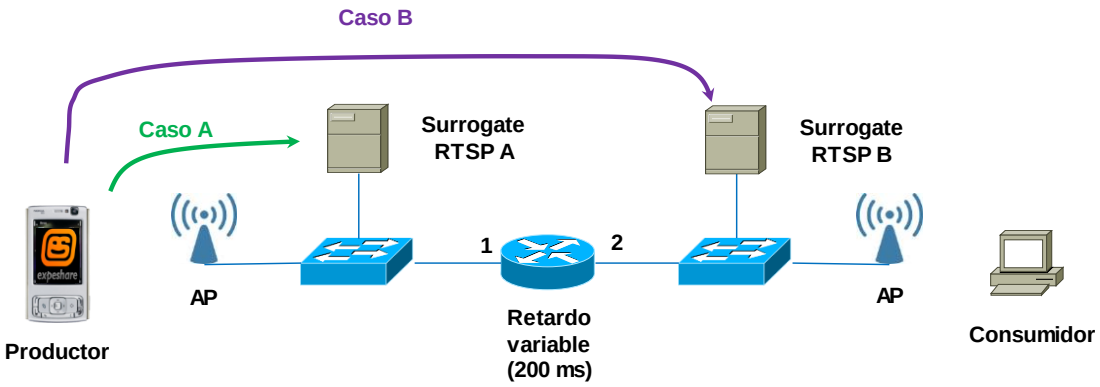


Figura 146. Mejora del retardo mediante CDN (usuario productor).

	Surrogate A		Surrogate B	
Video	**Desfase (s)**	**Nº cortes**	**Desfase (s)**	**Nº cortes**
Video 1	**3.5**	**10**	**5.7**	**17**
Video 2	**3.1**	**7**	**5.9**	**12**
Video 3	**3.2**	**5**	**5.5**	**8**

Tabla 50. Resultados del retardo en el consumidor. Número de pruebas: 20.

4.3.5.4. Evaluación subjetiva de la calidad (QoE)

La gran mayoría de los mecanismos de reserva de recursos para aplicaciones multimedia están basados en parámetros de QoS (*Quality of Service*) y SLAs (*Service Level Agreements*). La decisión de los valores de cada uno de estos parámentros está basada en el QoE (*Quality of Experience*), que es capaz de evaluar la percepción subjetiva del vídeo por parte de los clientes. Este aspecto es muy importante para los proveedores de servicios.

Básicamente, existen dos métodos para estimar el QoE:

- ***Evaluación objetiva***: las medidas de calidad objetivas representan modelos matemáticos que estiman el comportamiento perceptual del ser humano. Estos modelos son fuertemente dependientes del tipo de codificación empleada y los parámetros usados en dicha codificación. Dentro de estos modelos destacan los denominados modelos paramétricos, que estiman la calidad percibida del video

tomando en cuenta los valores de un conjunto reducido de parámetros, como por ejemplo la tasa de bits (*bit rate*), la tasa de imágenes (*frame rate*), el porcentaje de pérdida de paquetes, y/o algún parámetro relacionado con el propio contenido del video [Jos_12]. Típicamente resulta complicado modelar el comportamiento humano en todas las situaciones, por lo que se suele emplear una evaluación subjetiva.

- ***Evaluación subjetiva***: consiste en medir la percepción del usuario, lo cual puede resultar complicado. Para obtener resultados adecuados y significativos, los tests deben repetirse varias veces y para un número extenso de usuarios. Las medidas subjetivas de calidad están típicamente basadas en el análisis MOS (*Mean Opinion Score*) proporcionado por los evaluadores (aquellas personas que visualizarán y puntuarán los vídeos) y basado en una recomendación de la UIT [Itu_99]. Esto implica ofrecer a estos evaluadores un gran número de secuencias de vídeo con diferentes características de retardo y/o *jitter*. Para cada escenario el evaluador puntúa cada vídeo con un nivel de calidad representado en la Tabla 51. Los valores medios obtenidos serán los que permitan juzgar la calidad del vídeo. Este es el método empleado en esta tesis.

MOS	Calidad	Percepción
5	Excelente	Imperceptible
4	Buena	Perceptible
3	Mediocre	Ligeramente molesto
2	Baja	Molesto
1	Pobre	Muy molesto

Tabla 51. MOS (Mean Opinion Score).

Las medidas subjetivas de la calidad se han realizado para un número limitado de videos (3), un número limitado de evaluadores (15) y unas condiciones de red idénticas para los evaluadores.

Con la finalidad de limitar el tiempo de evaluación y así maximizar el interés de los usuarios evaluadores, se han tomado tres vídeos diferentes a partir de los vídeos disponibles en la CDN descritos ya en la Tabla 39. Según recomendaciones de la UIT-R [Itu_02], la longitud de cada vídeo objeto de evaluación debe ser, al menos, de 5 segundos. Para esta evaluación, se ha tomado una duración de 15 segundos para cada vídeo (una duración mayor suele despertar el desinterés del evaluador cuando hay un elevado número de secuencias de vídeo). Se han tomado tres tipos de vídeos diferentes en base a sus características de movimiento: elevado, medio y bajo (véase Tabla 52).

Fichero	cars_15sec.mp4	lion_king_15sec.mp4	dumbo_15sec.mp4
Movilidad	Elevada	Moderada	Baja
Resolución	640x480 (VGA)	640x480 (VGA)	640x480 (VGA)
Duración	15 s	15 s	15 s
Cuadros/s (FPS)	25	25	25

Tabla 52. Características de los vídeos en la evaluación de la QoE.

Las condiciones de cada escenario (red y servidores) deben ser las mismas para poder luego comparar los resultados y establecer medias; de lo contrario, los resultados no serían equiparables. Para controlar las condiciones de los servidores se van a emplear sesiones dedicadas para cada escenario de evaluación, de tal forma que el servidor RTSP sólo servirá el flujo a un usuario evaluador en cada ocasión, sin interferencia de otras peticiones. Por otro lado, el ordenador del cliente sólo ejecutará una sesión RTSP con el servidor, y su propia carga no afectará a la reproducción del vídeo. En referencia a las condiciones de la red, se requiere tener un control total sobre ella para garantizar el mismo funcionamiento en todos los escenarios. Es por ello por lo que se ha partido de la topología de red mostrada en la Figura 96. Sin embargo, el equipamiento de red (*routers* y *switches*) disponible no permite controlar los retardos introducidos en la red. Es por ello por lo que se introduce un ordenador como *router* de tal forma que permite hacer control de tráfico (*traffic shaping*) en cada uno de sus interfaces red. Este escenario para la evaluación de la QoE se ilustra en la Figura 147. En ella se dispone de un cliente que funcionará de evaluador y contactará con su servidor RTSP asociado. El ubicar este servidor RTSP es una subred diferente permite introducir un retardo variable en el canal de comunicación cliente-*surrogate*, que si estuvieran en la misma subred no sería posible (sería necesario un *switch* con capacidad de *traffic shaping* del que no se dispone; los switches Cisco Catalyst 2950 de la Figura 147 no soportan esta funcionalidad, sino clasificación, *policing* y marcado de paquetes). Dependiendo de los valores de retardo escogidos, equivale a ubicar al *surrogate* de la Figura 147 en la misma subred o en otra diferente. Con respecto al servidor origen se puede hacer una analogía similar: si el cliente solicita un contenido al *surrogate* y éste no dispone de dicho contenido, entonces debe obtenerlo del servidor origen.

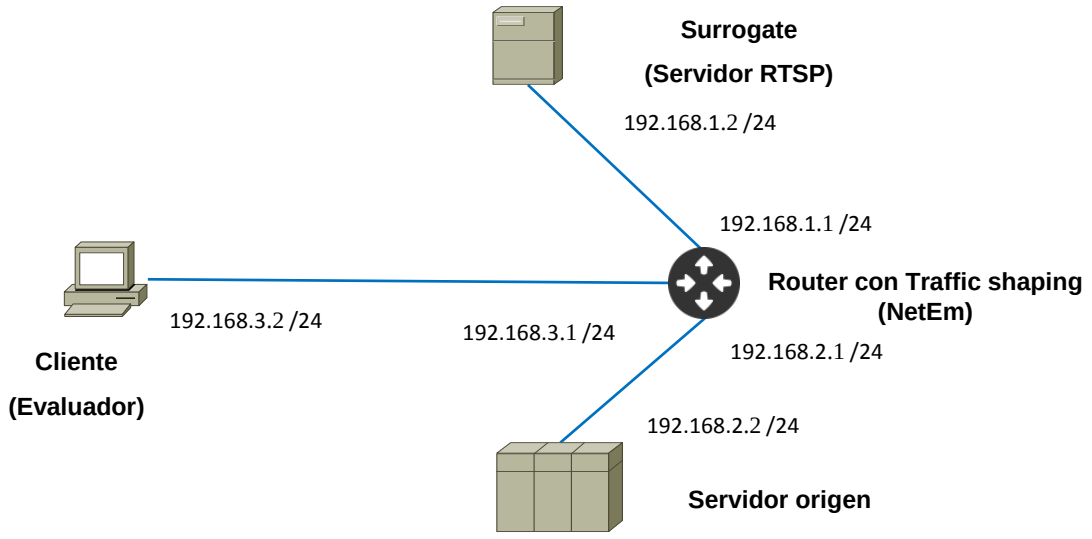


Figura 147. Escenario general para la evaluación de la QoE.

En este canal de comunicación también se puede introducir un retardo variable. El control de tráfico (*traffic shaping*) se ha realizado mediante NetEm [WWW_Neem], aunque existen otros emuladores de red válidos como DummyNet [WWW_Dnet] o NISTNet [WWW_Nnet].

En este escenario se introducirá un retardo variable en todos los paquetes salientes hacia la red del cliente (pues son los que afectan al flujo RTP desde el servidor RTSP al cliente) en la Figura 147.

Como se ha comentado anteriormente, se han utilizado tres videos diferentes en el experimento, y cada uno de estos vídeos se ha distribuido en formato streaming empleando diferentes valores de retardo y *jitter* (r+Δr) mediante NetEm. El retardo (fijo), teóricamente, apenas tiene impacto en la QoE porque simplemente introduce un retardo constante en todos los paquetes, pero su variabilidad sí repercute negativamente en la calidad del video percibido. En la Tabla 53 se reflejan los valores de retardo y *jitter* escogidos para los tres vídeos en el experimento.

Retardo (ms)	50, 100, 150, 200
Jitter (ms)	0, 2, 4, 8, 12,16, 20

Tabla 53. Valores de retardo y jitter en NetEm para la evaluación de la QoE.

Para determinar este impacto mínimo del retardo en la calidad percibida se ha tomado un grupo reducido de usuarios (5), un solo vídeo (*cars.mp4*) y se ha reproducido para todas las combinaciones posibles de retardo y *jitter*. Los resultados se muestran en la Figura 148. Como se puede comprobar, los resultados (medios) de los usuarios son muy parecidos y apenas presentan variación con el cambio en el retardo. Es por ello por lo que se ha tomado el caso aparentemente más desfavorable de 200 ms de retardo.

Con este valor de 200 ms de retardo se ha procedido a evaluar los tres ficheros de la Tabla 52 para los 15 usuarios. Para la visualización de los flujos de vídeo, se ha tratado de respetar las condiciones indicadas en las recomendaciones ITU-R Rec. BT.5005 [Itu_02] and ITU-T Rec. P.910 [Itu_96]. Los usuarios no eran conscientes del retardo y *jitter* introducidos, simplemente debían evaluar cada uno de los vídeos. Evidentemente, antes de proceder a la evaluación se les mostró a los usuarios el video original. Cada sesión tuvo una duración de unos 15 minutos aproximadamente y los vídeos no se mostraron ni en orden ascendente ni descendente. Una vez visualizado cada vídeo, el usuario debía evaluarlo mediante una puntuación en la escala MOS.

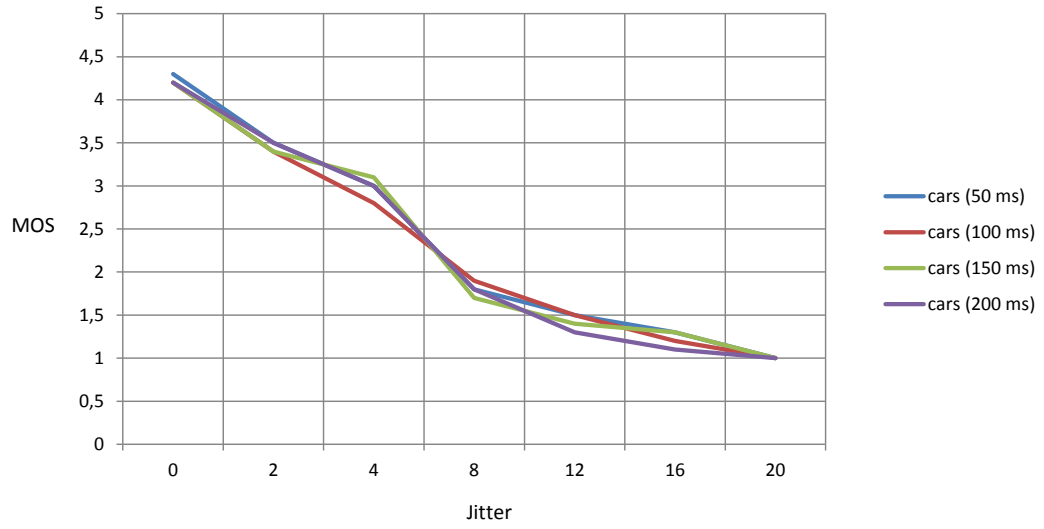


Figura 148. Evaluación de la QoE para diferentes valores de retardo.

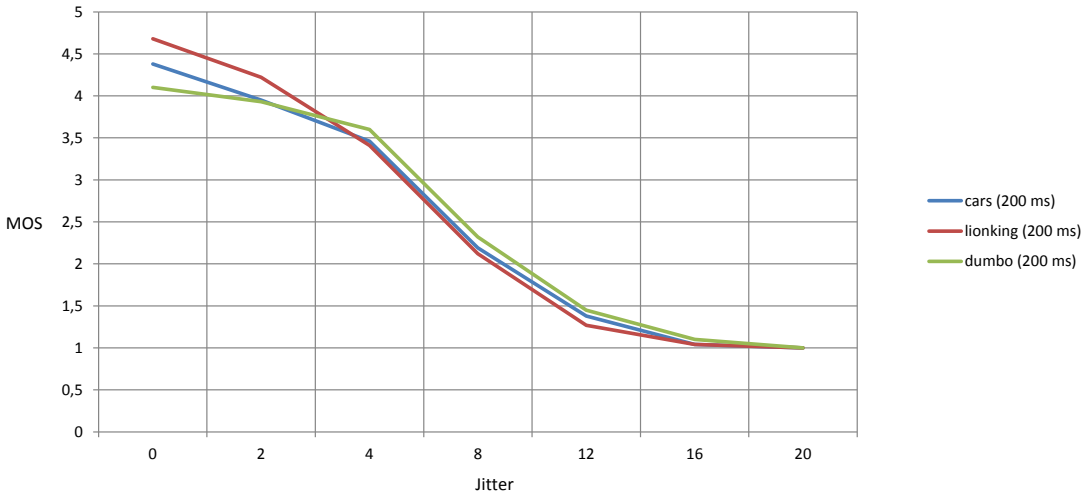


Figura 149. Evaluación del QoE para tres tipos de video diferentes.

Los resultados de la evaluación se muestran en la Figura 149. Se puede observar que un *jitter* de 2 ms ofrece una calidad percibida similar al video original. Entre 2 y 8 ms la calidad disminuye linealmente a medida que aumenta el *jitter*. A partir de 8 ms la calidad percibida es baja, y a partir de 16 ms la calidad es pobre. Resulta relevante el hecho de que para un valor reducido de retardo (2-4ms), los usuarios son más sensibles al vídeo de *dumbo*, cuando éste tiene precisamente un nivel de movimiento menor, y los cuadros (*frames*) son más estáticos. Precisamente el mayor movimiento de los otros vídeos enmascara los efectos negativos del *jitter*. Sin embargo, a partir de 8 ms, los efectos perjudiciales del *jitter* son percibidos con mayor claridad en las películas *cars* y *lionking*.

4.3.6. Comparación con otros modelos

Una vez realizado el análisis de la implementación, se va a proceder a comparar los resultados obtenidos con el modelo de simulación para streaming (sección 4.2). Adicionalmente, se realizará una comparación de la implementación desarrollada con otras implementaciones disponibles. La finalidad es doble:

- Por un lado, permite validar el modelo de simulación propuesto, y establecer en qué medida las hipótesis asumidas en este modelo eran válidas o razonables.
- Por otro lado, la comparación con otras implementaciones permite validar nuestro modelo, detectar las propiedades de unos y otros y establecer en qué medida se complementan.

4.3.6.1. Comparación con el modelo de simulación

En esta sección se realiza una comparación de la implementación de la CDN desarrollada con el modelo de simulación analizado en la sección 4.2. En dicho modelo de simulación se ha propuesto una arquitectura con soporte para sesiones RTSP, así como escenarios de red variables mediante generadores de topologías. El modelo de simulación ha permitido obtener un gran número de resultados y gráficas para el análisis, como: la estimación de la carga media en los *surrogates*; el tiempo medio de respuesta; el *jitter*; el modo de funcionamiento y el porcentaje de replicación (véase sección 4.2.4).

Por otro lado, el esquema de comunicación tanto en el modelo de simulación como en la implementación es similar:

- El cliente contacta con el Redirector y con el *surrogate* correspondiente.
- El cliente no contacta con el servidor origen, sino con un *surrogate*, quien actúa de servidor *proxy cache* para objetos multimedia si no dispone del contenido solicitado.

Si consideramos el esquema de replicación parcial, es posible comparar ambos modelos. Ya que la topología de red en la implementación no es fácilmente modificable, se ha cambiado la topología de red en el modelo de simulación, generando dos topologías equivalentes desde el punto de vista de los clientes, los *surrogates* y los enlaces de red entre ellos. Nótese que, en el modelo de simulación, los coeficientes *f*, *g* y *h* no están implementados (ya que no se implementaron los agentes), por lo que hay que establecer unos escenarios en cierto modo similares. Para ello, se pueden reproducir los mismos escenarios en ambos casos donde se conoce a priori el *surrogate* óptimo para un cliente

dado. De igual forma, se tomarán unos patrones de tráfico iguales en ambos casos. Lo importante en este caso no es tanto estudiar la carga generada, sino los valores de retardo medio y *jitter* en ambos modelos, por lo que no es necesario generar un gran número de sesiones RTSP. Los resultados de los tiempos medios de respuesta y *jitter* obtenidos se ilustran en la Tabla 54 y en la Tabla 55 respectivamente, donde se han incorporado los resultados obtenidos para la implementación correspondientes a la Figura 138.

Se puede apreciar como los valores son similares, si bien en el modelo de simulación son ligeramente inferiores ya que se asume una serie de simplificaciones dentro del funcionamiento de NS-2. El efecto de la carga sobre los *surrogates* en la simulación también es diferente, ya que el uso de CPU y su efecto en el retardo no está simulado. Tampoco está simulado el hardware del *surrogate*, por lo que los valores de simulación son más parecidos entre servidores. Por otro lado, los valores de *jitter* para la simulación son mayores. Esto es debido a las diferencias de implementación al calcular estos tiempos, y podría explicar también los valores de *jitter* excesivamente elevados en la simulación de CDN de streaming cuando la red estaba cargada (véase sección 4.2.4).

Periodo	r_1 (real)	r_1 (sim)	r_2 (real)	r_2 (sim)	r_3 (real)	r_3 (sim)
40s-100s	42.1	32.4	90.3	82.3	90.3	84.4
100s-160s	87.7	74.3	48.9	37.2	94.1	88.2
160s-220s	89.3	81.2	93.2	84.3	50.1	44.4
220s-260s	87.4	80.1	54.2	45.3	52.1	47.6
260s-280s	42.9	35.2	80.8	78.3	51.8	47.2

Tabla 54. Comparación de retardos (modelo real y simulado).

Periodo	j_1 (real)	j_1 (sim)	j_2 (real)	j_2 (sim)	j_3 (real)	j_3 (sim)
40s-100s	4.2	5.2	10.2	11.2	10.4	13.8
100s-160s	7.7	7.9	5.1	5.8	8.4	9.3
160s-220s	7.4	7.5	8.3	9.1	5.2	6.1
220s-260s	7.4	7.6	5.2	5.9	4.9	5.7
260s-280s	4.3	4.7	9.7	10.6	5.2	5.5

Tabla 55. Comparación de jitter (modelo real y simulado)

4.3.6.2. Comparación con otras implementaciones

En esta sección se realizará una breve comparación de la implementación de la CDN desarrollada con otras implementaciones reales disponibles. Tal y como se mencionaba en la introducción de esta tesis, existen actualmente escasas implementaciones disponibles que ofrezcan distribución de contenido en formato streaming, como Globule [Pie_01] [Pie_03], o Prism [Whi_11].

Globule es una CDN cuya descripción ya fue realizada de forma comparativa con la CDN para servicio web (véase sección 3.4.6.3). El mecanismo de distribución de contenido es mediante HTTP, por lo que no soporta RTP ni RTSP. Sin embargo, si es posible realizar streaming sobre HTTP. De hecho, existen implementaciones como HLS (*HTTP Live Streaming*) de Apple que permiten su uso en CDNs comerciales. Otro aspecto diferenciador entre Globule y la CDN implementada es el mecanismo de redirección; en el caso de Globule, el criterio de redirección sería el mismo que para objetos web, ya que simplemente se trata como un objeto web de elevado tamaño. En nuestra CDN, el algoritmo de redirección tiene en cuenta la popularidad local y el estado de la red para seleccionar un *surrogate* óptimo.

La arquitectura básica de Prism está compuesta básicamente por un codificador y un cliente, y se trata de encontrar la mejor forma de distribuir el contenido entre ambos. La codificación es tal que permite el acceso al flujo por parte de clientes con ancho de banda limitado. El cliente Prism procesa el flujo entrante y produce un flujo saliente estándar (H.264 o MPEG-2), haciéndolo compatible con los principales reproductores disponibles. La principal ventaja de este sistema es que está orientado a flujos adaptativos (*ABR, Adaptive Bit Rate*) con ancho de banda variables, soportando dispositivos móviles. En realidad, Prism no es exactamente una CDN, sino que se emplea sobre una infraestructura de CDN para optimizar la distribución de vídeos. La arquitectura contempla la existencia de unos elementos intermedios a modo de proxy denominados *Prism edge*. Estos elementos son capaces de obtener un flujo de datos del codificador y transformarlo en diferentes formatos (transcodificador), de forma que sólo circula un formato de flujo multimedia por la CDN.

4.3.7. Conclusiones parciales de la implementación multimedia

La implementación de CDN con soporte streaming descrita y analizada en esta sección permite abordar los principales aspectos de funcionamiento de una CDN real, aunque fundamentalmente el aspecto que se ha estudiado con mayor profundidad es el correcto funcionamiento del algoritmo de redirección, teniendo en cuenta parámetros como la carga media en los servidores (monitorización de *surrogates*) y el tiempo medio de respuesta y *jitter* (monitorización de clientes). Adicionalmente, la implementación propuesta también se ha comparado con el modelo de simulación previamente analizado, así como con otras implementaciones (Globule y Prism). Los principales resultados de este capítulo se han descrito en dos publicaciones propias [Mol_04] [Mol_06].

En nuestro modelo de simulación se ha evaluado el algoritmo de redirección estudiado la redirección DNS, la redirección basada en contenidos y, finalmente, el estudio de los tiempos de respuesta y *jitter*. Posteriormente, se han introducido redes inalámbricas en la arquitectura de pruebas real; la capacidad de introducir flujos por parte del usuario y un estudio de evaluación subjetiva de la calidad en los flujos recibidos. Las principales conclusiones son las siguientes:

- El algoritmo de redirección está basado en la captura inteligente de parámetros de funcionamiento obtenidos por agentes distribuidos en los *surrogates*. A partir de ellos se generan las funciones *f, g* y *h* que determinan la decisión de encaminamiento. La popularidad de los objetos también afecta en la decisión, de tal forma que los objetos son servidos por aquellos *surrogates* que lo almacenan al ser popular para ellos.

- La popularidad de los objetos determina el modo de funcionamiento en la redirección: dinámico o estático. En el modo dinámico, el cliente es redirigido al *surrogate* de su misma subred y el retardo y *jitter* experimentado es mínimo. En el modo estático, se evalúa aquel *surrogate* más cercano (óptimo) para el cliente, siempre que disponga del contenido.

- El algoritmo de redirección se adapta de forma dinámica a tres parámetros: el estado de carga de los *surrogates*, el estado de la red, y la variación en el patrón de acceso a los objetos multimedia (popularidad).

- En la redirección basada en contenidos y el modo estático los valores de los coeficientes *y, g* y *h* únicamente se calculan para aquellos servidores que disponen del contenido. Si bien en una replicación total (caso web) había que calcularlos todos, para replicación parcial (caso streaming) existe un filtrado según disponibilidad de contenido, lo que simplifica el algoritmo en este sentido.

- En cuanto a los tiempos de respuesta obtenidos, se observa que es adecuado la redirección del cliente a un *surrogate* cercano, ya que el retardo medio y el *jitter* son menores en comparación con los otros servidores, lo que repercute en una mejora de la calidad percibida.

- Las redes inalámbricas representan un canal con elevado retardo y *jitter* en comparación con las redes cableadas. Estos valores son mayores conforme mayor es el ancho de banda requerido, por lo que es conveniente enviar formatos de menor calidad a dispositivos móviles si se desea garantizar una reproducción fluida del flujo multimedia. Otra alternativa es emplear multihoming con varias redes inalámbricas y establecer en cada momento la red más adecuada para un flujo. Este mecanismo de selección puede ser indicado y gestionado por el cliente o bien por el servidor RTSP.

- En las aplicaciones de vídeo es fundamental establecer un criterio subjetivo de calidad percibida por los usuarios (QoE). Una herramienta útil para esto es el MOS (*Mean Opinion Score*) que permite vincular valores subjetivos de calidad percibida (buena, mala, regular, etc.) en valores objetivos de medida (retardo y *jitter*). Se puede comprobar que para valores de *jitter* superiores 8 ms el usuario percibe un flujo de mala calidad, por lo que habría que garantizar que se está por debajo de este valor durante la sesión multimedia.

Una vez presentadas las principales conclusiones a partir de los resultados obtenidos, es importante comentar algunas limitaciones o posibles mejores de la implementación propuesta. Por ejemplo, en la sección 4.3.5.3 se podría aumentar el número de flujos multimedia hasta saturar la red Wi-Fi e ir analizando el impacto gradual mediante una evaluación subjetiva con MOS. A partir de ahí, se podría repetir el experimento y ver el beneficio de multihoming. Otro aspecto que no se ha tratado consiste en el uso de descripción múltiple (*MDC, Multiple Description Coding*) [Xu_13] para mejorar la calidad de la experiencia; en lugar de emplear un mismo vídeo con diferentes tasas de codificación, con MD se trata de aprovechar múltiples canales diferentes entre el emisor y el receptor.

5. CONCLUSIONES Y LÍNEAS FUTURAS

5.1. Conclusiones generales

En esta tesis se ha presentado, estudiado, implementado y evaluado un modelo de CDN que permite ofrecer tanto contenido web como streaming de vídeo. Las conclusiones generales son las siguientes:

Estado del Arte

- El crecimiento de tráfico web y multimedia en Internet justifica la necesidad de escalar en las grandes redes para proporcionar mecanismos eficaces y eficientes en la distribución de contenido, de tal forma que se ofrezca una mejor calidad de servicio (QoS) a los usuarios de Internet.

- Aunque se dispone de mecanismos de QoS de nivel de red (*IP IntServ, IP DiffServ*, etc.), únicamente se despliegan en entornos intra-AS y no en Internet (inter-AS). En este sentido, se entiende por 'calidad' en Internet a la capacidad de ofrecer un tiempo reducido de espera (retardo y *jitter*) a los clientes cuando contactan con un servidor.

- Algunos tipos de aplicaciones desplegadas en servidores centralizados no son capaces de ofrecer este tipo de calidad por diversos motivos, y se han ido produciendo e incorporando novedades tecnológicas a lo largo de los últimos años.

- Una de las primeras tecnologías aplicadas, aunque en constante evolución, ha sido el *caching*, aplicado al servicio web y, más recientemente, a objetos multimedia, y es capaz de reducir el tiempo de latencia percibido por los usuarios. Estos sistemas de *caching* disponen de varias configuraciones (*forward*, *reverse* e *interception proxy*) y pueden formar redes malladas o jerárquicas, especificándose varios protocolos de comunicación (ICP, cache digest, etc.).

- Por otro lado, los sistemas de réplicas, *mirrors* o espejos duplican la información almacenada en un servidor origen en otros servidores remotos. El objetivo principal es descargar al servidor origen de un exceso de peticiones, además de aumentar la disponibilidad de los datos.

- Las redes de distribución de contenidos (CDNs) son una evolución dentro de las redes de caché y los espejos, integrando ambas tecnologías de una forma eficiente dentro del concepto de *content networking*. El principal interés estriba en reducir la latencia percibida por el usuario mediante la redirección a un servidor cercano, denominado *surrogate*. En Internet, el número de *surrogates* a desplegar debe ser significativo para minimizar el efecto de la latencia WAN, que es impredecible.

- Las CDNs pueden ser clasificadas atendiendo a aspectos de composición, distribución de contenido y gestión, encaminamiento y redirección, etc., y cada uno de ellos puede ser subdivido en otros. Una CDN es un sistema distribuido y altamente complejo, por lo que se requiere un cierto conocimiento de sus diferentes formas de interactuar antes de implementar cualquier módulo o servicio dentro de la CDN.

- El algoritmo de redirección de usuarios es crucial y definitivo para caracterizar la efectividad de una CDN. Para la implementación de un buen algoritmo se debe conocer detalladamente su estructura de funcionamiento: cómo está formada, cómo está monitorizada, cómo se externaliza el contenido, etc. En definitiva, es necesario un correcto análisis de la CDN para poder especificar e implementar un algoritmo de redirección efectivo y eficiente.

Modelo analítico

- El modelo analítico presentado y analizado permite caracterizar el funcionamiento básico del esquema de distribución de una CDN, con un marcado interés sobre el retardo medio experimentado por los clientes. Se han realizado múltiples simulaciones para varios valores de M (número de clientes), P (número de *surrogates*), N (número de paquetes), etc. para verificar la coherencia de los resultados.

- Considerando el parámetro RTT (*Round Trip Time*), es deseable descargar todo el contenido a los *surrogates*. El parámetro $\overline{R}_t$ analizado en el modelo se corresponde con una gráfica decreciente conforme aumenta la probabilidad de acierto *p* o, en otros términos, el *hit ratio*. Esto es debido a que el número de enlaces entre un cliente y un *surrogate* siempre es menor que entre el cliente y el servidor origen, por lo que el retardo disminuye.

- Considerando el tiempo de proceso (parámetro $\overline{R}_p$), éste es independiente (despreciable) de la probabilidad de acierto sólo en condiciones de baja carga. Para situaciones de elevada carga en los servidores es necesario introducir en el modelo un nuevo parámetro variable (*k*) capaz de reproducir efectos del tipo *flash-crowd*.

- A partir de los resultados obtenidos en las simulaciones, se pueden obtener otros parámetros de interés, como la carga del sistema. Ésta se puede obtener fácilmente a partir del tiempo de respuesta del servidor origen y de los *surrogates*. Si bien la carga media del sistema es la misma en las simulaciones, lo interesante es la variabilidad de las cargas entre *surrogates* y también respecto al servidor origen para identificar un adecuado balanceo de carga.

- El algoritmo de redirección no se ha introducido en el modelo analítico para disponer de un modelo sencillo y simplificado de estudio. La complejidad de la redirección introduce numerosos parámetros, por lo que se ha estudiado en el modelo de simulación y la implementación real.

- Los principales resultados del modelo analítico han originado dos publicaciones propias [Mol_04b] [Mol_12].

Modelo de simulación

- El modelo de simulación representa una ampliación significativa del modelo analítico, al introducir numerosos factores adicionales en la caracterización de una CDN. Esto permite analizar mejor el comportamiento de la CDN desde varias perspectivas.

- El modelo de simulación permite evaluar la carga media de los *surrogates*, el tiempo medio percibido por los clientes, la distribución de acceso, el porcentaje de replicación, el retardo de procesamiento, el hit ratio, el byte hit ratio y el efecto *flash-crowd*. Adicionalmente, la integración de replicación y caching en los *surrogates*, permite obtener un rendimiento mejorado en términos de tiempo de respuesta percibido, hit ratio y byte hit ratio.

- El modelo integrado (replicación y caching) ofrece unos tiempos de respuesta hasta un 70% mejores que en el caso de únicamente replicación.

- El modelo integrado ofrece tiempos de respuesta similares al caso de solamente caching en condiciones normales; sin embargo, en situaciones de *flash-crowd*, los tiempos de respuesta son hasta un 15% mejores (80% replicación, 20% caching) que únicamente con caching.

- Considerando el hit-ratio (HR) y byte-hit–ratio (BHR), la replicación conduce a un rendimiento deficiente, aunque dependiente del tamaño del disco. El empleo combinado de caching y replicación mejora sustancialmente el hit ratio. Con un reducido porcentaje de caching (20%) ya se obtienen valores similares de hit ratio y byte hit ratio que con un porcentaje de caching del 100%.

- La relación 80-20 en la replicación y caching ofrece unos valores muy convenientes en el estudio del rendimiento, pero debido a las diferentes configuraciones posibles no se puede fijar un porcentaje de ambos valores que proporcione un máximo de rendimiento en todos los casos.

- El modelo de simulación dispone de ciertas limitaciones o simplificaciones. En primer lugar el Redirector tiene un tiempo de proceso prácticamente nulo, por lo

que habría que caracterizarlo de una forma más realista. En segundo lugar, el modelo se ha simulado para un tipo de grafo en concreto (usados típicamente en la literatura científica), y los resultados de la simulación apenas varían a partir de 15 *surrogates*. Si se amplía el grafo, es posible extender la variabilidad a un mayor número de *surrogates*.

- Por otro lado, el modelo de simulación se ha comparado en su modalidad web con otros modelos de simulación existentes (de forma limitada), como CDNsim y CDN Simulator. Desde el punto de vista de la comparación entre modelos, puede apreciarse una cierta eficiencia en memoria de CDNsim con respecto a nuestro modelo de simulación, en torno al 10- 15%. En relación al consumo de CPU, los resultados son muy similares, si bien en CDNsim es ligeramente inferior. En relación con CDN Simulator, el tiempo medio de respuesta en nuestro modelo es claramente mejor. Adicionalmente, este tiempo de respuesta es ligeramente mejor con el algoritmo R2C (*Two Random Choices*) que LL (*Least Loaded*).

- En el modelo de la CDN de media streaming, el aumento en el número de servidores produce una disminución no sólo en el retardo inicial, sino en los retardos medios y *jitter* durante toda la sesión RTSP.

- Se han definido dos modos de funcionamiento: estático (*static*) y dinámico (*dynamic*). El primero de ellos considera la disponibilidad de contenido como criterio principal en la redirección, mientras que el segundo selecciona el servidor con menor retardo, independientemente de la disponibilidad e contenido. Asimismo, se han definido varias estrategias de distribución inicial de contenido disponible en el servidor origen. En el modo *all* se distribuye todo el contenido, en el modo *random* se distribuye el contenido de forma aleatoria, mientras que en el modo *mostPopular* se distribuye el contenido más popular. Para valores de replicación intermedios, la estrategia de distribución de contenidos que menores retardos proporciona es *all*, luego *random* y, finalmente, *mostPopular*, siempre que el modo de funcionamiento sea estático. En el modo dinámico, los resultados son totalmente opuestos por su propia forma de funcionamiento, y la estrategia *mostPopular* presenta un menor retardo medio. Si se comparan ambos modos de funcionamiento, el modo dinámico presenta un menor retardo.

- Es interesante destacar como, en el caso de streaming, el retardo medio y *jitter* varían cuando el número de servidores es mayor de 15, a diferencia del modelo de simulación para tráfico web. Esto se debe a la diferencia en términos de tráfico generado entre ambos servicios (web y streaming), y la presencia de servidores adicionales en sesiones de streaming tiene un mayor impacto en el rendimiento general.

Implementación de la CDN

- La implementación de la CDN considera todos los factores reales que caracterizan una CDN, y permite analizar su comportamiento desde diversos puntos de vista. El estudio se ha centrado principalmente en el correcto funcionamiento del algoritmo de redirección, teniendo en cuenta parámetros como la carga media en los servidores (monitorización de *surrogates*) y el tiempo medio de respuesta (monitorización de clientes).

- Los principales análisis llevados a cabo en la implementación han sido el estudio de la redirección DNS bajo un funcionamiento normal, la caída y recuperación de un *surrogate*, así como variaciones abruptas en la carga de dichos *surrogates*. Adicionalmente se ha analizado la redirección basada en contenidos y el estudio de los tiempos de respuesta en las consultas DNS, consultas a base de datos y redirecciones HTTP.

- El algoritmo de redirección está basado en agentes distribuidos en los *surrogates*, capaces de monitorizar el sistema. A partir de esta información se realizan una serie de cálculos que permiten obtener una serie de coeficientes, que se tienen en cuenta en la toma de decisiones del algoritmo. Mediante los coeficientes *f* se conoce el estado de los servidores para determinar sus condiciones de carga (o si están caídos). A través de los coeficientes *g* se tiene conocimiento del estado de la red para determinar las rutas menos cargadas. Finalmente, los coeficientes *h* combinan los coeficientes <*f,g*> asignando mayor peso a uno u a otro dependiendo de la configuración o servicio ofrecido.

- El algoritmo de redirección implementado se adapta dinámicamente al estado de carga de los *surrogates*, y la decisión de encaminamiento no depende del estado de carga de un *surrogat*e de forma independiente, sino de forma relativa al resto de *surrogates*.

- La CDN implementada permite fijar umbrales superiores e inferiores parametrizables en las condiciones de carga de un *surrogate* (función *k[]*). El umbral superior limita las peticiones (posibles implosiones), mientras que el umbral inferior permite un rápido crecimiento de nuevas conexiones para situar al *surrogate* en su punto óptimo de funcionamiento.

- En la redirección basada en contenidos (redirección HTTP) las condiciones de la red (coeficientes *y, g* y *h*) sólo se calculan cuando el servidor local correspondiente supera un umbral inferior de carga. Por encima de dicho umbral las peticiones se redirigen directamente al *surrogate* local, lo que facilita y simplifica la operativa del algoritmo.

- Considerando los tiempos de respuesta en el servicio web, la mayor parte del retardo está motivada por el algoritmo de redirección, cuyo procesado requiere más de 10 veces el tiempo de envío de una página web. No obstante el tiempo global es inferior al medio segundo y es asumible por el cliente.

- El algoritmo de redirección emplea un esquema con memoria a partir de la iteración (intervalo de monitorización) anterior. Dado que el histórico de valores de monitorización se guarda, sería posible considerar más iteraciones previas, aunque esto complica el algoritmo.

- El algoritmo de redirección contempla a cada cliente de una forma independiente, aunque sería posible correlar clientes topológicamente cercanos e inferir el estado de la red para optimizar y acelerar la ejecución del algoritmo.

- La CDN implementada para servicio de streaming se ha comparado con el modelo analítico y el modelo de simulación previamente analizados, así como con otra implementación real (Globule).

- El algoritmo de redirección para sesiones de streaming también considera la popularidad de objetos multimedia como criterio en la toma de decisiones. Esta popularidad determina el modo de funcionamiento en la redirección: dinámico y estático. En el modo dinámico, el cliente es redirigido al *surrogate* de su misma subred y el retardo y *jitter* experimentado es mínimo. En el modo estático, se se redirige al cliente al surrogate más cercano que disponga del contenido. La disponibilidad de contenido representa un filtrado inicial antes de calcular los parámetros de red (coeficientes *y*, *g* y *h*). En términos de tiempo de respuesta y *jitter* para el servicio de streaming, se observa que es adecuada la redirección del cliente a un surrogate cercano, y repercute en una mejora de la calidad percibida.

- Los principales resultados de la implementación de la CDN se han descrito en dos publicaciones propias [Mol_04] [Mol_06].

5.2. Líneas futuras de investigación

Una CDN es una red compleja donde intervienen múltiples tecnologías que dan soporte a uno o varios módulos que conforman la arquitectura. Si bien es posible ampliar el estudio realizado en esta tesis en múltiples direcciones, en esta sección se ofrecerán algunas opciones tanto en el ámbito del modelo analítico, el modelo de simulación y la implementación real, por analogía con el esquema de esta tesis. Adicionalmente, se introducirá un capítulo con una visión futura de las CDNs y su relevancia estratégica dentro de la Internet del Futuro (*Future Internet*).

5.2.1. Modelo analítico

En el modelo analítico presentado en esta tesis se partía de una configuración mostrada en la Figura 53, donde el servidor origen constituye el punto central del modelo, alrededor del cual se sitúan clientes y *surrogates*. Este modelo, si bien ofrecía un punto de partida para resultados básicos, no contemplaba una serie de aspectos que pueden ser abordados en un modelo ampliado para un enfoque más realista. A continuación se analizarán estas limitaciones para posteriormente establecer un posible modelo mejorado.

- ***Dificultad en la escalabilidad:*** el modelo establece dos posibles niveles de acceso, a través del servidor origen o bien a través de los *surrogates*. Estos *surrogates* se encuentran configurados bajo una misma estructura plana. Este tipo de estructura es útil con un número pequeño de nodos, pero las labores de gestión y localización de errores es muy costosa si no se establece algún tipo de agrupamiento que permita asociar a los *surrogates* y aislar los posibles fallos que puedan aparecer. Un proceso similar sucede con el servidor origen, que es capaz de servir contenido a todos los clientes, lo que dificulta su rendimiento conforme aumenta el número de peticiones. Esta situación, que conducía a aumentos repentinos e inesperados de la carga en el sistema (*flash-crowds*), se representaba en el modelo propuesto con un factor de capacidad de proceso. Sin embargo, su utilidad estaba limitada en cuanto a la incapacidad de establecer una relación directa entre una situación de carga en un servidor concreto con respecto al tráfico que estaba procesando. Una configuración jerárquica ofrece ciertas ventajas en términos de escalabilidad y flexibilidad. Por un lado, conforme aumenta el tráfico en el sistema, la jerarquía establecida permite aumentar la capacidad del sistema en aquellas zonas que experimentan dicho incremento. Por otro lado, el servidor origen queda descargado seriamente de solicitudes que pueden ser satisfechas en varios niveles de la jerarquía. El acceso al servidor origen correspondería a la situación en que un objeto solicitado no se hubiera localizado por otra ruta de acceso más rápido.

- ***Concepto de clúster:*** el modelo analítico propuesto agrupaba a los clientes en forma de clústeres (véase sección 3.2); sin embargo, no lo hace así en el caso de los *surrogates*. Para alcanzar una estructura jerárquica, es necesario la agrupación de los servidores, normalmente, aquellos topológicamente cercanos. Recuérdese que en el modelo estudiado cada clúster de clientes contactaba con un número determinado de *surrogates*, dependiendo de los valores de latencia que los distanciaba (esto constituía un parámetro del modelo). Esta agrupación de *surrogates* no puede ser caracterizada como un clúster debido a un motivo fundamental: cada cliente percibe diferentes clústeres de *surrogates* o, desde otro punto de vista, un mismo *surrogate* puede pertenecer a varios clústeres. La agrupación de los clústeres de *surrogates* debe responder a un criterio unívoco, por ejemplo, que el retardo intracluster sea reducido. De esta forma, un clúster concreto de *surrogates* dará servicio a los clientes de sus cercanías topológicas. Todos estos clientes pueden ser agrupados en uno o varios clústeres. El criterio de agrupación en clúster de clientes no sólo depende de la ubicación de dichos clientes, sino también del perfil de tráfico generado. De esta forma, puede haber un clúster de clientes de navegación web, mientras que otro puede ser de tipo streaming.
- ***Concepto de caching:*** la forma de trabajar en el modelo anterior contemplaba la replicación de contenido en todos los *surrogates*. Si no es así, se introducía una probabilidad de acierto (hit rate) para modelar el caching, pero hacía difícil estudiar el comportamiento por separado en cada uno de los *surrogates*. Si se desea introducir una configuración jerárquica, resulta útil la posibilidad de caracterizar cada uno de los clústeres de clientes de diferente forma en términos de caching. Los parámetros básicos de caching son: la cantidad de objetos almacenados, la política de actualización de cache y la comunicación intracluster, intercluster y con el servidor origen a la hora de obtener un contenido determinado.
- ***Modelo de redirección:*** actualmente la mayoría de CDNs actuales basan su modelo de redirección en el propio DNS. Esta comunicación se ha obviado en el modelo anterior, de tal forma que se establecía una probabilidad de contacto con los *surrogates* y con el servidor origen. De esta forma, un cierto número de peticiones eran encaminadas a algunos de los *surrogates*, mientras que otro se dirigía al servidor origen, dependiendo de la probabilidad de acierto. Así, la comunicación podía adoptar dos caminos (retardos) distintos: cliente-*surrogate* o cliente-origen. En el nuevo modelo jerárquico, existen más alternativas a tener en cuenta, como la comunicación intercluster e intraclúster. Sin introducir un modelo analítico de la redirección DNS, se puede asumir que el cliente será redirigido a su clúster de *surrogates* más cercano. Evidentemente, esto implica un tiempo de proceso a tener en cuenta. En el caso de objetos web, este tiempo es representativo en relación con el tiempo de respuesta una vez se ha

contactado con el servidor que dispone del contenido. En el caso de objetos de tipo streaming, suponemos que el posible retardo inicial debido a la redirección justifican sobradamente la necesidad de servir contenido desde el clúster de *surrogates* óptimo, puesto que el retardo y *jitter* percibido por el cliente será reducido.

La ampliación del modelo propuesto en la sección 3.2 se muestra en la Figura 150. Se observa la estructura jerárquica centrada en el servidor origen. Este servidor origen está en comunicación directa con un grupo de clústeres primarios de *surrogates* (primer nivel), quienes, a su vez, controlan en cierta forma un pequeño grupo de clústeres de *surrogates* secundarios (segundo nivel). El modelo podría extenderse fácilmente a un número mayor de niveles, pero se partirá inicialmente de esta situación de dos niveles.

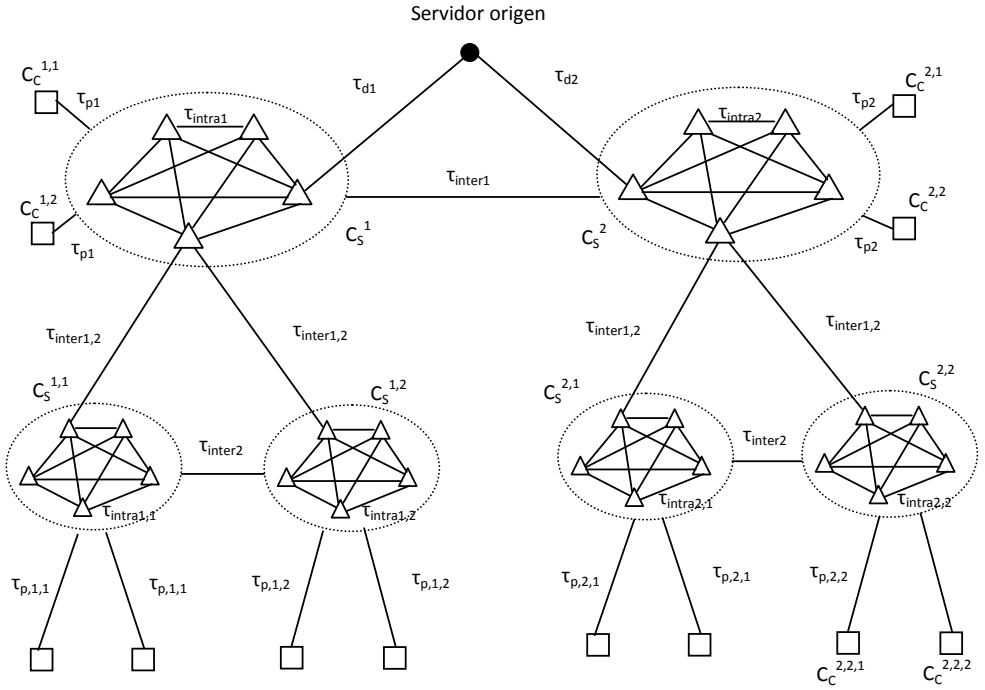


Figura 150. Modelo jerárquico de una CDN.

Este nuevo modelo ampliado conduce a la necesidad de expresar cada uno de los parámetros que conforman el sistema completo de una forma unívoca. Las diferencias con respecto al modelo anterior dificultan la opción de conservar la nomenclatura de los objetos que constituyen la infraestructura de la CDN. A continuación se definen cada uno de estos parámetros:

- $\tau_{d,i}$ representa el RTT medio entre el servidor origen y el clúster de *surrogates* i-ésimo de primer nivel (este nivel es el que se conecta directamente con el servidor origen).

- C_S^i representa el i-ésimo clúster de *surrogates* de primer nivel, mientras que $C_S^{i,j}$ denota al j-ésimo clúster de *surrogates* de segundo nivel asociado al i-ésimo clúster de primer nivel.

- Cada clúster de *surrogates*, tanto de primer (C_S^i) como de segundo nivel ($C_S^{i,j}$) está formado por un conjunto determinado de *surrogates*, $N_S^{C_S^i}$ (primer nivel) y $N_S^{C_S^{i,j}}$ (segundo nivel).

- De manera individual, cada *surrogate* (k-ésimo) dentro de un clúster se designa por $S_k^{C_S^i}$ *y* $S_k^{C_S^{i,j}}$ para el primer y segundo nivel respectivamente.

- Cada uno de los *surrogates* que conforman el clúster se encuentran interconectados en forma de malla, y lo suficientemente cercanos topológicamente como para poder emplear un solo parámetro, de forma que se define $\tau_{intra,i}$ como el RTT medio entre dos *surrogates* cualesquiera dentro del i-ésimo clúster (primer nivel). De manera análoga para el segundo nivel, $\tau_{intra,i,j}$ representa el RTT medio entre dos *surrogates* del clúster $C_S^{i,j}$.

- Los clústeres de *surrogates* tienen asociados a ellos (debido a un sistema inteligente de redirección) un conjunto de clientes que se agrupan en forma de clústeres; $C_C^{i,j}$ representa el j-ésimo clúster de clientes del i-ésimo clúster de *surrogates* de primer nivel, C_S^i, mientras que $C_C^{i,j,k}$ representa el k-ésimo clúster de clientes asociado al clúster de *surrogates* $C_S^{i,j}$.

- Los clústeres de clientes asociados al mismo clúster de *surrogates* tienen en común el retardo medio de acceso; de esta forma, $\tau_{p,i}$ denota el RTT medio de acceso al clúster C_S^i desde cualquier clúster de clientes ($C_C^{i,j}$ o $C_C^{i,j,k}$) asociado a éste.

- Los clústeres de *surrogates* también interactúan entre ellos, por lo que se define el parámetro $\tau_{inter,1}$ como el RTT medio intercluster de nivel 1, mientras que $\tau_{inter,2}$ representa el RTT medio intercluster de nivel 2; la interacción entre clústeres de *surrogates* de diferente nivel se define como $\tau_{inter,1,2}$. Estos factores no se han definido de manera individualizada por cada uno de los clústeres con el fin de simplificar el modelo, sin perder generalidad alguna. Esta forma de proceder se puede considerar como consecuencia de la política de ampliación de la infraestructura de CDN multinivel.

- En cuanto al número de clústeres de clientes y *surrogates*, suponemos que existe un total de N_{P1} clústeres primarios de *surrogates*, N_{P2} clústeres secundarios y un total de N_C clústeres de clientes.

Una vez se han descrito los principales parámetros del modelo analítico ampliado, se puede abordar su impacto (sus beneficios para caracterizar la CDN de forma más realista) en los esquemas de modelado de contenido y comunicación.

5.2.2. Modelo de simulación

El modelo de simulación, tanto en su modalidad web como de streaming, se pretende ampliar desde diferentes perspectivas:

- ***Introducción de nuevas métricas***: el modelo de simulación utiliza el número de páginas web servidas o el número de conexiones como indicador de la carga en los *surrogates*. Sería interesante analizar e introducir nuevas métricas que permitan calcular fielmente el estado de carga de cada agente (*surrogate*). Por ejemplo, la evaluación de los posibles retardos o congestiones en las colas de cada uno de estos agentes de nivel de aplicación.
- ***Integración con otros simuladores***: aunque no existen simuladores de redes para contenido streaming, en la presente tesis se ha realizado una comparación de simuladores para contenido web, como son CDNsim y CDN Simulator. En el primer caso, no se puede hacer propiamente una integración de código, ya que CDNsim está desarrollado sobre OMNeT++ mientras que nuestro modelo lo está sobre NS-2. Sin embargo, sí se pueden portar ciertos aspectos para hacer un mejor estudio de las CDNs. Tal es el caso del método de externalización de contenido, donde CDNsim soporta cuatro configuraciones distintas (véase sección 3.3.6.2). Incorporar estos cuatro tipos de configuración podría resultar interesante en el modelo de simulación. En relación con CDN Simulator, la integración o portabilidad es relativamente sencilla al estar desarrollado sobre NS-2. De especial interés resulta incorporar las diferentes políticas de balanceo de carga (LL, RR, RAND, R2C) así como su modelo de flexible de programación en la redirección a nivel de *surrogate*. Este esquema de programación modular se puede incorporar en nuestro modelo de simulación para la redirección de contenido cuando el *surrogate* correspondiente se encuentra cargado por encima de un umbral. Esto conduce también a la integración del algoritmo de redirección del modelo real en la propia simulación, que se describe a continuación.
- ***Integración con el modelo real***: en el modelo de simulación se han asumido ciertas simplificaciones que se pueden suprimir (al menos parte de ellas) si ya se dispone de un modelo de implementación. El código de dicha implementación puede ser portado al modelo, teniendo en cuenta las características propias de NS-2 y su propio modelo de programación. Los dos aspectos más importantes en la integración de ambos modelos son:

- *Integración del algoritmo de redirección:* en el modelo de simulación se asumía (de forma simplificada) un tiempo de proceso en la redirección nulo, ya que los cálculos se realizaban de antemano. Si se introduce el código del algoritmo de implementación en el modelo de simulación, el tiempo de proceso en la redirección es completamente realista. La introducción del algoritmo tiene bastantes implicaciones nada triviales, pues también implica introducir la monitorización de servidores y red de forma periódica y aperiódica respectivamente.
- *Introducción de datos reales:* una de las ventajas del simulador NS-2 es que permite introducir datos reales en los nodos que conforman la red de comunicaciones. Dicho de otro modo, es posible disponer de instancias reales de *surrogates* y emplear NS-2 como red de comunicaciones para recrear entornos LAN y WAN.

- ***Actualización***: si bien el modelo de simulación está desarrollado en NS-2, este simulador de redes dejará de ser empleado en poco tiempo al ser reemplazado por la nueva versión denominada NS-3. Un aspecto importante para disponer de herramientas actualizadas consiste en la portabilidad del código generado en NS-2 a NS-3. Esta portabilidad no es inmediata, ni existe un método automático para portar código de un entorno a otro, ya que su diseño y modo de funcionamiento son distintos.

5.2.3. Implementación de la CDN

Una CDN se puede percibir como una sistema de sistemas, por lo que cada uno de los subsistemas son susceptibles de ser ampliados y mejorados para ofrecer un mejor servicio, capacidad operativa, rentabilidad, etc. En esta sección se describirán dos líneas de investigación en la implementación, abarcando aspectos diferentes:

- ***Aspecto tecnológico***: Si bien una CDN es altamente escalable y su capacidad puede crecer agregando más servidores (*surrogates*), esto puede suponer problemas operativos (plazos de tiempo lentos en el despliegue, dificultad en la gestión, etc.). En ocasiones, resulta útil la interconexión de CDNs, como se describe en la sección 2.5.5, para abarcar rápidamente nuevas áreas geográficas, o incluso para aumentar la densidad de servidores un una misma zona. En estos casos, la autenticación de usuarios entre CDNs diferentes es un aspecto relevante para el correcto funcionamiento en la interoperabilidad de ambos sistemas, de tal forma que el usuario sólo necesite autenticarse una sola vez (SSO, *Single Sign On*). La sección 5.2.3.1 describe el uso de SAML (*Security Assertion Markup Language*) como mecanismo de autenticación federada entre CDNs.

- ***Aspecto de servicio***: si bien las CDNs se emplean típicamente para ofrecer servicios web y de streaming, existen otro tipo de servicios a través de Internet de fuerte crecimiento y gran demanda, como son los juegos masivos en red multijugador (MMOG, *Massive Multiplayer Online Game*). El uso de una CDN como infraestructura de soporte de este tipo de juegos se analiza en la sección 5.2.3.2.

5.2.3.1. Federación de CDNs mediante SAML

Un aspecto relevante y perceptible por el usuario cuando éste accede a un conjunto de redes y sistemas diferentes es la autenticación unificada o federada, en ocasiones denominada también SSO (*Single Sign On*). Este mecanismo consiste en que el usuario se autentica solamente una vez cuando accede al sistema y, si es redirigido a otro sistema que forma parte de la federación, no es necesaria una nueva autenticación o, de producirse, se realiza de forma totalmente transparente para el usuario.

En el contexto de los sistemas federados, esto se traduce en relaciones de confianza entre dichos sistemas, que abarcan un número concreto de usuarios y un conjunto de acciones que estos usuarios pueden realizar en todo el sistema federado. Típicamente un usuario pertenece a un único sistema de forma nativa (se ha registrado en dicho sistema), pero es capaz de acceder al resto de sistemas que conforman la federación. La autenticación federada garantiza el acceso mediante SSO, pero el conjunto de reglas para los recursos de cada sistema de la federación (autorización) dependerá de las relaciones de confianza entre sistemas. En las federaciones se distinguen dos entidades fundamentales:

- ***Proveedor de Identidad (IdP, Identity Provider)***: es la entidad responsable de identificar y autenticar al usuario.
- ***Proveedor de Servicio (SP, Service Provicer):*** es la entidad a la que accede el usuario y confía en los datos que dicho usuario le proporciona a través del IdP y, en función de los mismos, autoriza al usuario a usar los recursos.

En Internet han surgido una serie de protocolos de autenticación y autorización en Internet, que se describen brevemente a continuación:

- ***OpenID [WWW_Open]***: se trata de un protocolo abierto, y su primera versión fue definida en 2005 para su uso en el sitio web LiveJournal. Es un protocolo de autenticación federada, y si se utiliza siempre un mismo proveedor de OpenID, el usuario sólo requiere recordar sus credenciales OpenID para autenticarse en todos los servicios que acepten OpenID como mecanismo de autenticación. Muchos proveedores de servicios actuales en Internet soportan ser utilizados como

proveedores de OpenID: Google, Yahoo!, flickr, etc.; también existen proveedores independientes de OpenID, como myOpenID. El usuario incluso tiene la opción de implementar su propio proveedor de OpenID.

- ***OAuth [WWW_OAut]***: a diferencia de OpenID, es un protocolo (abierto) de autorización o, más exactamente, de delegación de acceso: permite definir cómo un tercero va a acceder a los recursos propios. Se trabajó en él a partir de 2006 debido a ciertas carencias del protocolo OpenID, y en 2007 se publicó la primera versión oficial. Mediante este protocolo un usuario que tiene determinados recursos en un servidor (proveedor OAuth) pueda dar acceso total o parcial a un tercero (consumidor), sin que este tercero conozca su usuario y contraseña (con esos datos tendría el control total de la cuenta). OAuth 2.0 es una versión mejorada (simplicidad en la implementación, arquitectura más robusta) de OAuth, y está soportada por grandes compañías como Facebook, Twitter, Yahoo!, Google o Microsoft.
- ***SAML (Security Assertion Markup Language) [WWW_Saml]:*** Se trata de un lenguaje abierto especificado por OASIS basado en XML; su principal propósito es servir de marco para protocolos de autenticación federada. Este protocolo sirve de base para algunos sistemas propietarios de *Single-Sign-On* [WWW_Pin], aunque no es utilizado por los grandes proveedores de servicios en Internet.

En la actualidad se perciben dos tendencias en los esquemas de autenticación [San_11]. Por un lado, los grandes servidores de redes sociales (como Facebook, Twitter o Google) aspiran a ser elegidos por los usuarios como su gestor principal de identidad, lo que les convierte en el repositorio donde los usuarios almacenan sus datos personales (y actualizados) Esto constituye un gran valor para las compañías. Por otro lado, protocolos como OpenID suponen un esfuerzo no sólo hacia la definición de protocolos abiertos, sino hacia la descentralización de la gestión de identidad y de datos personales. Si el interés de los proveedores de redes sociales es convertirse en el punto donde los usuarios tienen su identidad principal para atraer y gestionar sus datos, la propuesta de OpenID es que se pueda elegir entre diferentes servidores para ser utilizados como proveedores de identidad, y evitar esa centralización que puede provocar problemas de privacidad. En entornos cerrados o controlados, como puede ser una CDN, se pueden emplear otros mecanismos de autenticación como SAML.

En el grupo de investigación SATRD ya se han realizado despliegues de SAML dentro del marco de varios proyectos europeos y nacionales [WWW_Fasys] [WWW_F4all]. La Figura 151 muestra un flujo de paquetes entre un cliente, un proveedor de servicios (SP) y un proveedor de identidad (IdP) desplegados en la UPV. Para la implementación tanto de SP como de IdP se ha empleado SimpleSAMLphp [WWW_Sim]. Desde el punto de vista del cliente, el proceso es el siguiente:

- El cliente contacto con el SP, solicitando un servicio concreto.
- El SP delega la autenticación en un IdP, por lo que redirige al cliente a éste.

- El IdP comprueba si el usuario ya ha sido autenticado previamente y, en caso contrario (Figura 151) redirige al usuario a su página por defecto de autenticación.
- Una vez el usuario introduce sus credenciales y se autentica satisfactoriamente en el IdP, éste le envía un *token* y le redirige nuevamente al SP.
- El cliente contacta nuevamente con el SP, adjuntando el *token* obtenido del IdP. El SP valida dicho *token* y permite el acceso a los recursos. Este *token* se envía de una forma cifrada, de forma que la información sensible sólo es legible por el SP y el IdP (emplean claves públicas y privadas). No obstante, para una mayor seguridad, durante las redirecciones se puede emplear HTTPS en vez de HTTP.

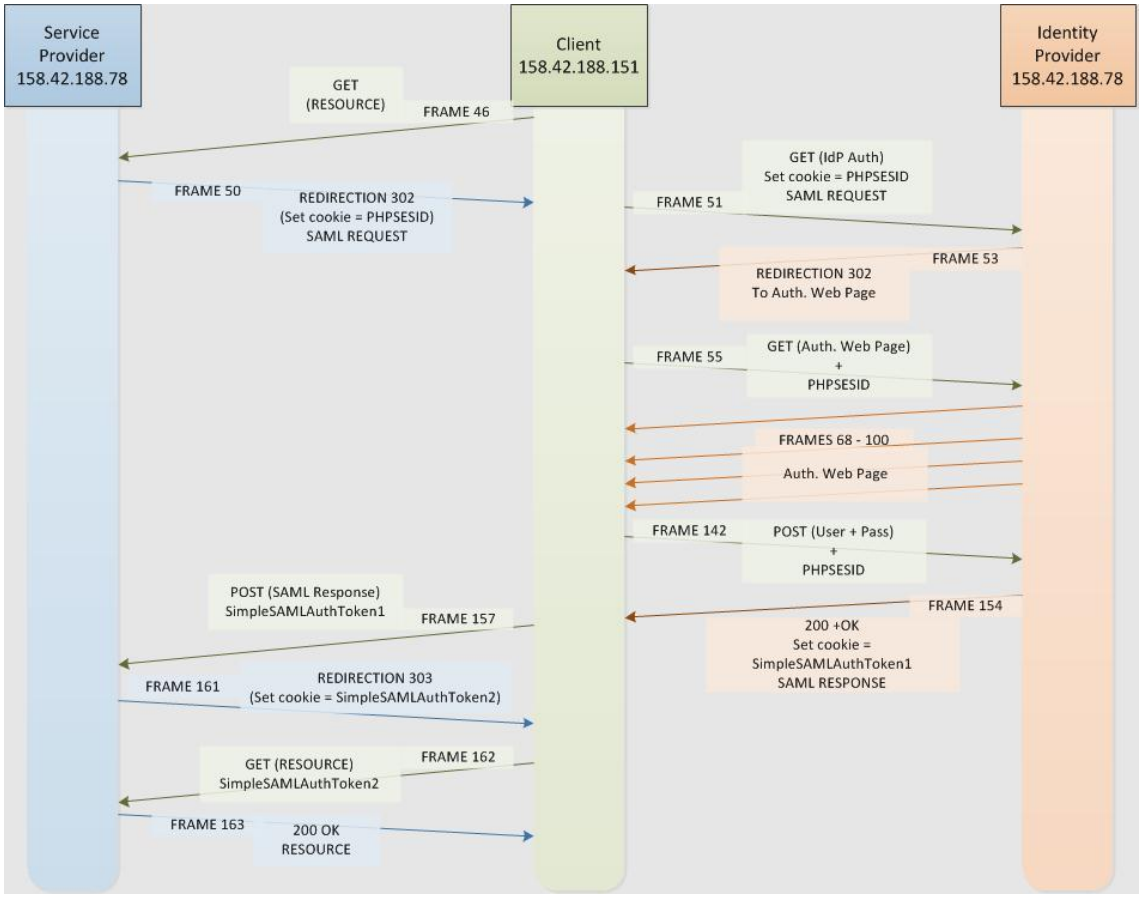


Figura 151. Implementación SAML con SimpleSAMLphp (flujo de tramas).

Para implantar un esquema de autenticación federada entre CDNs, se requiere desplegar dos o más CDNs. En este caso, cada uno de los *surrogates* de cada CDN desempeñará un rol de SP. Sin embargo, en cuanto a la implementación del IdP, se deben establecer relaciones de confianza entre todos ellos con los SPs. Es decir, dado un SP (*surrogate*), este debe ser capaz de delegar la autenticación a un conjunto de IdPs: bien el propio de la CDN, o bien cada uno de los IdPs del resto de CDNs que conforman la federación. Durante este proceso existe un mecanismo de descubrimiento donde el usuario es capaz de seleccionar el IdP que desea emplear (típicamente aquel donde se haya registrado).

5.2.3.2. MMOG

Los juegos en red multijugador (MMOG, *Massive Multiplayer Online Game*), en sus múltiples variantes, MMOGRPG (MMOG *Real-Role Playing Game*), MMOGFPS (MMOG *First-Person Shooter*), MMOGRTS (MMOG *Real-Time Strategy*), BBMMOG (*Browser Based* MMOG) y MMMOG (*Mobile* MMOG), han experimentado un continuo y elevado crecimiento en Internet en los últimos años. Posiblemente el ejemplo más claro y notorio lo ha constituido el WOW (*World of Warcraft*) [WWW_WoW]. Las principales características de este tipo de juegos son:

- ***Uso de Internet***. Se juega con conexión a Internet ya que la información relevante se almacena en los servidores de la empresa que produce el videojuego.
- ***Suscripción***. Hay que pagar una cuota (mensual) para jugar.
- ***Multijugador***. El jugador no es el centro de la historia, y todos pueden ser protagonistas. La popularidad de un jugador depende de su habilidad para hacerse un nombre en la comunidad de jugadores.
- ***Masivos***. Están diseñados para acoger a miles de personas al mismo tiempo en un mundo virtual, y todos los personajes coexisten y se relacionan en tiempo real.
- ***Persistencia***. El mundo virtual del juego es independiente del jugador, tiene su propia historia y no desaparece ni se detiene cuando el usuario se desconecta.

Tradicionalmente los MMOG se han desplegado sobre arquitecturas cliente-servidor en Internet, donde los clientes se conectan con un servidor de gran capacidad de proceso y elevado ancho de banda. El aspecto de la escalabilidad es fundamental porque el número de clientes (suscriptores) puede ser masivo. Existen dos tipos de soluciones en este aspecto:

- Crear un clúster de servidores dinámico. Se trataría de un servicio tipo *cloud* que se describió en la sección 2.5.5.
- Distribuir la carga en varios servidores distribuidos. En este caso, cada servidor controla una parte del mundo virtual, que queda dividido en zonas. Los principales problemas en esta situación son la gestión de zonas fronterizas y los saltos entre zonas (conmutación de un jugador a otro servidor), así como la posibilidad de zonas superpobladas con un impacto sobre el rendimiento del servidor correspondiente.

Cabe destacar que los juegos MMOG, principalmente MMOGFPS y MMOGRTS, tienen características de tiempo real [Cla_06] [Hob_12], lo cual representa un problema si se desea ofrecer calidad de servicio sobre Internet. Típicamente el ancho de banda promedio empleado por los usuarios (jugadores suscriptores) era cercano a los 40 Kbps, y los primeros servidores limitaban el número de usuarios concurrentes para poder

garantizar un buen servicio [Fen_02]. Estos juegos de elevada acción o interactividad son muy sensibles a la latencia de red, y el retraso tolerado por un jugador debe ser inferior de 100 ms [Jar_11] [Cla_10], aunque también depende del tipo de juego. Los juegos dinámicos y de mucha acción sí requieren una latencia aproximada de 100 ms, mientras que en otros puede ser suficiente un retardo de 150 ms [Jar_11].

En la actualidad se utiliza el concepto de *cloud gaming* para referenciar a aquellos juegos bajo demanda soportados por una infraestructura *cloud*, y son capaces incluso de generar los gráficos para que los requisitos de usuario sean mínimos (por otro lado, los requisitos de red se ven aumentados). Si bien esto representa una solución escalable y aparentemente válida, los elevados requisitos de tiempo real descritos anteriormente hacen que la infraestructura actual de *cloud computing* resulte insuficiente (ya que suele estar orientada más a minimizar costes de electricidad y enfriamiento que retardo), y se sugieren propuestas y soluciones que emplean CDNs como mecanismo de distribución [Cho_12]. Algunos proveedores de plataformas de juegos *cloud* como Gaikai [WWW_Gai] y Onlive [WWW_Onl] disponen de varios centros de datos en EEUU, de forma que garantizan un mejor servicio a sus clientes. Evidentemente, dado que un solo y potente centro de datos es económicamente más eficiente que un centro de datos pequeño, los proveedores de servicios *cloud* deben determinar adecuadamente la rentabilidad de la inversión.

El uso de CDNs en el despliegue y gestión de juegos MMOG pueden ser objeto de estudio por sus potenciales beneficios en varios aspectos, entre otros:

- ***Cercanía a los clientes***. Al contrario que una solución *cloud* ubicada en un centro de datos, con una CDN se dispone de múltiples servidores en varias zonas, de forma que el retardo entre clientes y servidores es menor. La gestión entre *surrogates* está garantizada, de forma que es posible el intercambio de información y mapas (contextos de cada juego) para preservar la continuidad y la interactividad de cada sesión de usuario.
- ***Gestión de usuarios***. En ocasiones, dependiendo del tipo de juego, al jugador se le ofrece una lista de servidores que puede seleccionar para iniciar una sesión. Mediante el empleo de una CDN, el cliente puede ser redirigido de forma transparente a aquel servidor con menor latencia y/o carga. Es más, dado que la CDN monitoriza el estado de los servidores en todo momento, es posible también la conmutación de una sesión de jugador de un *surrogate* a otro de manera transparente si lo criterios de calidad (exceso de carga o de retardo) no se están cumpliendo.
- ***Gestión de grupos***. Dependiendo del tipo de juego, es posible solicitar el acceso al juego de manera conjunta (grupo de amigos). En este caso, se les puede asignar un mapa ubicado en un servidor cuya latencia media sea mínima con el dicho grupo. Con el uso de una CDN, se dispone de mecanismos para estimar la

latencia de acceso de los clientes, de tal forma que se puede hacer una asignación automática a todo el grupo.

Como futura línea de investigación, se plantea el diseño de una implementación real de un escenario MMOG sobre una CDN, teniendo en cuenta sus mecanismos de redirección. De esta forma se puede estudiar la calidad de la experiencia percibida por los jugadores, midiendo adicionalmente parámetros objetivos como la latencia. De una manera más concreta, se sugiere el empleo del mecanismo de redirección propuesto en esta tesis y cómo debe parametrizarse sus variables para adaptarse a cada tipo de juego MMOG.

5.2.4. CDNs en el contexto de Future Internet. Visión estratégica

El concepto *Future Internet* hace referencia a todas aquellas actividades que estudian y desarrollan nuevas arquitecturas para Internet, no sólo en su dimensión puramente tecnológica (seguridad, distribución, adaptación, etc.) sino en su dimensión económica y social (modelos de negocio, modelos de inclusión, etc.). Las principales diferencias entre el actual y futuro Internet se pueden resumir, básicamente, en dos aspectos:

- ***Elevado y verdadero ancho de banda***. Las redes serán capaces de proporcionar velocidades de Gbps, lo que permitirá introducir nuevos servicios como video 3D en redes cableadas e inalámbricas.
- ***Control compartido***. Si bien la primera generación de Internet permitía compartir información (e-mail, web, streaming, etc.) la segunda generación debe permitir compartir el control para gestionar el aprovisionamiento de servicios por mecanismos de usabilidad universales y de una forma ubicua, fiable y segura.

Estos aspectos tienen ciertas implicaciones o impacto en los agentes de la cadena de distribución:

- Desde el punto de vista del ***proveedor del servicio***, debe ser capaz de ofrecer al usuario una comunicación más natural, lo que típicamente conducirá a un enriquecimiento multimedia de los servicios, con las últimas generaciones de formatos HD y 3D y codificadores de vídeo con soporte H.264 AVC (*Advanced Video Coding*), SVC (*Scalable Video Coding*) y MVC (*Multiview Video Coding*) sobre redes heterogéneas.
- Desde el punto de vista de los ***operadores de red***, estos deben ofrecer una serie de características a sus clientes, como son:

- *Elevada capacidad de decisión*: el usuario debería poder elegir el rendimiento (precio) que necesita dependiendo de sus requerimientos de QoS.
- *Acceso real de banda ancha*: esto se traduce en el empleo de VSDL o FTTH para permitir mejor calidad en los servicios de streaming.
- *Fiabilidad, resiliencia y disponibilidad*: de esta forma, se dotará a la red de una inteligencia especial que permitirá:
  - Detectar errores y cambiar rutas o codificaciones para diferentes dispositivos (*network awareness*).
  - Emplear diferentes mecanismo de corrección de errores dependiendo del tipo de medio intercambiado (*media awareness*).
  - Emplear herramientas autónomas y automáticas para solucionar los problemas de congestión (*autonomic network awareness*).
- *Seguridad y control de acceso*: proporcionará un amplio espectro de diferentes productos de seguridad por parte de los operadores de red capaces de interoperar, así como facilitar a los usuarios su propia capacidad para definir sus políticas de seguridad.
- *Confianza*: es importante proveer un marco de confianza a través del cual los proveedores de contenido multimedia puedan definir sus propias reglas de DRM (*Digital Rights Management*), compartición e identificación de contenido.
- *Conectividad de usuario*: mediante el uso de contenido 3D multimedia con alta capacidad de personalización, sesiones multi-usuario en tiempo real, mundos virtuales 3D para uso profesional y aplicaciones de juegos y comunidades.
- *Networking*: proporcionará requisitos de red de una forma más balanceada para la distribución de contenido, control distribuido, caching, P2P y streaming de contenido multi-fuente y multi-red.
- *Calidad de servicio*: se ofrecerá verdadera calidad de servicio con diferentes precios y rendimientos para atraer a la industria multimedia en la provisión de redes y servicios.

Como se puede apreciar, las aplicaciones del *Future Internet* requieren un despliegue de ancho de banda en todo el mundo sobre infraestructuras convergentes de voz, vídeo y datos. En este punto cabe destacar que la distribución de streaming multimedia será diferente a la tradicional por cable y satélite, lo que representa nuevos retos para los operadores de redes de telecomunicaciones y el despliegue de sus infraestructuras MDP (*Media Delivery Platform*). Estos retos se deben analizar en términos de capacidad, eficiencia del ancho de banda, escalabilidad, calidad de la experiencia y *responsividad*

en los modelos de negocios y modelos de interacción soportados (*Internet of Things*, movilidad, personalización, etc.)

Uno de los mayores desafíos en la implementación exitosa del concepto de *Future Media Internet*, desde el punto de vista de las MDPs, es la convergencia entre las arquitecturas de red y de transporte. Tal y como están desplegadas actualmente, la arquitectura convergente alámbrica e inalámbrica soporta tráfico de Internet y VoIP (*Voice Over IP*), pero la calidad de experiencia percibida (PQoE, *Perceived Quality of Experience*) para estos servicios es limitada. Si, por ejemplo, se quiere garantizar el éxito de IPTV, los proveedores de servicio y de transporte deben ofrecer un valor de PQoE igual o mejor del ofrecido actualmente por TV por cable o satélite. La clave para conseguir este objetivo es el despliegue de redes conscientes del contenido (*content aware network*) o, al menos, con dispositivos conscientes de dicho contenido en los extremos de la red (*content aware edge devices*), capaces de monitorizar, gestionar y priorizar los flujos de datos a través de los extremos de la red.

En la actualidad, desde el punto de vista del nivel de red, los protocolos de encaminamiento más empleados en Internet son OSPF (*Open Shortest Path First*) y BGP (*Border Gateway Protocol*). Estos protocolos son capaces de encaminar dependiendo de las direcciones IP de los paquetes IP que conforman el flujo de comunicación, pero carecen del conocimiento necesario para encaminar estos paquetes adecuadamente al servidor más apropiado para conseguir un mejor valor de PQoE.

La visión a medio y largo plazo consiste en que las redes de distribución serán capaces de encaminar diferentes tipos de contenido a través de varias rutas y emplear mecanismos de reserva de recursos sin necesidad de señalización por parte del usuario o el nivel de aplicación. En este sentido, se requieren nuevas arquitecturas y tecnologías para servicios convergentes y escalables. Una propuesta reciente la constituye ICN (*Information Centric Networking*) [Pav_11] [Sir_12], en ocasiones también denominado *networking* consciente del contenido (*content-aware*), centrado en el contenido (*content-centric*) u orientado a los datos (*data oriented*), que trata de unificar los espacios de direcciones y de contenidos [Bar_12]. La tecnología ICIN es tema de interés en la FIA (*Future Internet Assembly*) [WWW_Icin] y en ciertos congresos recientes como SIGCOMM 2012 [WWW_Sig].

La visión a corto y medio plazo sigue pasando por el uso de CDNs como mecanismo de gestión de grandes redes orientadas a balancear la carga y reducir considerablemente los tiempos de latencia percibidos por los usuarios. Dependiendo del ámbito de actuación, se requerirá en mayor o menor medida mecanismos de interconexión entre estas redes, como puede ser CDNI (descrito en la sección 2.5.5.2). Es posible que las propias CDNs evolucionen e integren paulatinamente tecnologías similares a ICIN. Nótese que se trata de aproximaciones diferentes: si bien ICIN trata de abordar la distribución de contenido desde los niveles inferiores (red y transporte), las CDNs lo hacen fundamentalmente desde el nivel de aplicación, lo que confiere mayor independencia y flexibilidad, al no requerir modificaciones en la estructura básica desplegada en Internet.

6. GLOSARIO DE TÉRMINOS

Anycast	Mecanismo de transmisión similar al multicast, con la peculiaridad de que sólo participa uno de los nodos de la sesión multicast a la vez.
AWK	Lenguaje de programación diseñado para procesar y manipular datos basados en texto, típicamente trazas de datos producidas durante la ejecución de un programa previo, como una simulación.
Best effort	Servicio que intenta ofrecer el mejor servicio posible sin garantías ni reserva de recursos.
BGP	Border Gateway Protocol. Protocolo que permite el intercambio de información de encaminamiento entre redes o dominios administrativos (AS) distintos. Este protocolo se suele ejecutar entre los routers externos de dichas redes o dominios.
BIND	Berkeley Internet Name Daemon. Estándar de facto como servidor DNS junto con librerías cliente de acceso.
Broadcast	Mecanismo de transmisión (difusión) donde los paquetes se transmiten una sola vez, y son escuchados por todos los nodos de la red en cuestión.
CARP	Cache Array Routing Protocol. Técnica o protocolo que permite distribuir peticiones a un array de proxy cachés.
CCSP	Cloud Delivery Service Provider. Proveedor de servicios de cloud computing
CDI	Content Delivery Interconnection. Antiguo grupo de trabajo (2001-2003) del IETF que especificaba estándares para la interconexión de CDNs.
CDN	Content Distribution Network. Básicamente se trata de un servicio ofrecido por compañías en Internet que permite replicar (parcial o totalmente) servidores origen en diferentes ubicaciones de Internet para ofrecer un menor tiempo de latencia o una mayor escalabilidad.
CDNI	Content Delivery Network Interconnection. Grupo reciente de

	trabajo orientado a la especificación de interfaces para permitir la interconexión de CDNs.
CDSP	Content Delivery Service Provider. Empresa que proporciona servicios de CDN
Carrier	Un carrier es una empresa que ofrece servicios de tránsito en Internet, representado una red 'troncal' en Internet que permite llegar a cualquier punto. Hace años se distinguía entre ISP (red de acceso) y carrier (red troncal) dentro de Internet, aunque debido a las fusiones y acuerdos entre ambos la diferencia entre estos conceptos es actualmente difusa, pues una misma empresa puede ser ISP y carrier a la vez.
Cloud computing	La computación en nube es un paradigma actual donde los recursos y servicios se ofrecen desde una nube (típicamente Internet) sin que el usuario o nodo cliente sea consciente de la ubicación física de estos recursos o servicios. Incluye varios tipos como Saas (Software as a Service), PaaS (Platform as a Service) e IaaS (Infrastructure as a Service)
DARPA	Defense Advanced Research Projects Agency. Se trata de una agencia del Departamento de Defensa de EEUU orientada al desarrollo de nuevas tecnologías militares. De esta agencia surgió ARPANET, red que dio posteriormente origen a Internet.
DNS	Domain Name System. Se trata de una base de datos distribuida que contiene diferentes tipos de registros para los nombres de dominio de Internet. Típicamente se emplea para convertir nombres de host en direcciones IP.
DRM	Digital Rights Management. Se refiere a las tecnologías de control de acceso usadas por editoriales y dueños de derechos de autor para limitar el uso de medios o dispositivos digitales. También se puede referir a las restricciones asociadas a instancias específicas de obras digitales.
Escalabilidad	Capacidad de un sistema para mantener un nivel de prestaciones aceptable conforme aumenta el número de participantes.
Forward proxy	Proxy cache ubicado en la subred del cliente o usuario

FTP	File Transfer Protocol. Protocolo de nivel de aplicación que permite intercambiar ficheros entre sistemas conectados mediante un modelo cliente-servidor.
GDS	Greedy Dual Size. Algoritmo de sustitución para proxy cachés
Green computing	Este concepto hace referencia a todas las tecnologías *verdes* que pretenden minimizar el impacto medioambiental (huella sobre el CO_2) junto con el consumo energético. Las principales tecnologías son cloud computing, grid computing y virtualización.
HTCP	Hypertext Caching Protocol. Segunda generación de protocolo Intercache, similar a ICP y documentado en la RFC 2756.
HTML	Hypertext Markup Language. Lenguaje de marcado que permite representar la información a clientes y servidores web.
HTTP	Hypertext Transport Protocol. Protocolo de nivel de aplicación empleado en interacciones web. La versión actual es la v1.1.
HTTPS	Hypertext Transport Protocol Secure. Se trata de HTTP sobre SSL/TLS. Protocolo para introducir seguridad en los datos sobre el protocolo HTTP, documentado en la RFC 281y 2818.
IANA	Internet Assigned Numbers Authority. Se trataba de la autoridad encargada de mantener y delegar los espacios de nombres y su numeración, como direcciones IP, dominios DNS y puertos TCP/UDP. En 1998 fue sustituido por el ICANN.
ICANN	Internet Corporation for Assigned Names and Numbers. Organización de ámbito internacional encargada de asignar las direcciones del protocolo IP, los identificadores de protocolo, las funciones de gestión del sistema de dominio y la administración del sistema de servidores raíz.
ICMP	Internet Control Message Protocol. Forma parte del protocolo IP y permite el control y la notificación de errores. Está descrito en la RFC 792. El protocolo se encuentra disponible tanto para IPv4 como IPv6.
ICP	Internet Cache Protocol. Primer protocolo intercache, y permite a los proxy caches ubicar URLs específicas en sus vecinos. Se

	encuentra documentado en las RFCs 2186 y 2187.
IGMP	Internet Group Management Protocol. Protocolo de nivel de red que permite gestionar la adhesión y abandono a grupos multicast. La última versión disponible de este protocolo es la IGMPv3 descrita en la RFC 3376.
IETF	Internet Engineering Task Force. Organismo responsable del estudio y publicación de estándares de Internet.
Interception proxy	Proxy cache ubicado en una subred intermedia entre cliente y servidor
IP multicast	Mecanismo de envío de paquetes IP desde un emisor a múltiples nodos destinos. Esto implica el empleo de un rango concreto de direcciones IP, junto con el uso de protocolos de gestión de grupo, como IGMP, que no están soportados de manera nativa en la infraestructura de Internet.
ISP	Internet Service Provider. Proveedor de acceso a Internet, bien sea mediante acceso fijo o móvil.
IXP	Internet Exchange Point. Se trata de un punto de interconexión entre dos ISPs para permitir el flujo de tráfico de Internet.
Jitter	Variabilidad del retardo experimentado por los paquetes al atravesar una red de comunicaciones y que afecta seriamente a la calidad subjetiva percibida por el usuario en aplicaciones de tiempo real.
LFU	Least Frequently Used. Se trata de un algoritmo de sustitución para cachés que básicamente elimina el objeto menos frecuentemente usado de memoria antes de insertar un nuevo objeto.
LRU	Least Recently Used. Se trata de un algoritmo de sustitución para cachés que básicamente elimina el objeto menos recientemente usado de memoria antes de insertar un nuevo objeto.
MD5	Message-Digest Algorithm 5. Algoritmo del tipo Message Digest empleado para crear una marca unívoca de un mensaje o fichero mediante una reducción criptográfica de 128 bits. Se encuentra documentado en la RFC 1321.

MIB	Management Information Base. Se trata de una base de datos que permite gestionar variables (equipos) en una red de comunicaciones y es típicamente empleada por el protocolo SNMP.
MMOG	Massive Multiplayer Online Game. Se trata de aquellos juegos en red multijugador que son masivos por la gran afluencia de usuarios.
MPLS	Multi-Protocol Label Switching. Se trata de un mecanismo de conmutación de alto rendimiento que encamina los paquetes entre nodos de la red en base a unas etiquetas (nivel 2), evitando recurrir a la tabla de enrutamiento (nivel 3).
MRTG	Multi-Router Traffic Grapher. Aplicación muy popular que permite visualizar tráfico de dispositivos de red a través de SNMP.
Multicast	Mecanismo de transmisión donde los paquetes se transmiten una sola vez, y sólo son escuchados por los nodos que pertenecen a la sesión multicast en curso, sin interferir con el resto de estaciones.
NAM	Network Animator. Componente. Herramienta de animación para visualizar trazas generadas durante simulaciones (NS-2) o incluso entornos reales.
NAP	Network Access Point. Se trata de un término histórico para designar los actuales IXP.
NSP	Network Service Provider. Proveedor de servicios de red, típicamente un operador de telecomunicaciones (ISP).
NTP	Network Time Protocol. Protocolo que permite sincronizar relojes de varios dispositivos de red tomando como referencia una fuente.
P2P	Peer to Peer. Modelo de comunicación entre dos entidades de red donde ambas pueden actuar simultáneamente como clientes y servidores.
Peering	Acuerdo entre dos redes de Internet administrativamente independientes, típicamente ISPs y carriers, para permitir el flujo de tráfico de red entre ambas. El protocolo empleado

	suele ser BGP.
POP	Point Of Presence. Hace referencia a aquellos lugares donde un ISP despliega equipamiento para dar cobertura o servicio a un área geográfica.
QoE	Quality of Experience. Calidad de la experiencia del usuario. Se trata de una métrica subjetiva que pretende medir la percepción del usuario final en la entrega de un servicio.
QoS	Quality of Service. Calidad del servicio. Se trata de una métrica que pretende medir la calidad de un servicio mediante parámetros objetivos y medibles, típicamente mediante métricas de red como retardos, *jitter*, pérdida de paquetes, etc.
RAS	Remote Access Server/Service. Se refiere a una combinación de hardware y software que permiten acceder remotamente a recursos de una red. Se trata de una configuración típica empleada por ISPs para ofrecer una conexión física a sus clientes, aunque también es posible el empleo en redes corporativas.
Reverse proxy	Proxy cache ubicado en la subred del servidor. A veces también se le llama acelerador web.
RFC	Request For Comments. Informes técnicos que proponen estándares de protocolos telemáticos. Son estudiados y publicados por el IETF.
RSVP	Resource ReSerVation Protocol. Protocolo de nivel de transporte que permite reservar recursos en la red para una comunicación punto a punto, dentro del marco de la arquitectura de servicios integrados (IP IntServ). El protocolo se describe inicialmente en la RFC 2205, pero ha tenido bastantes revisiones y mejores en muchas otras RFCs (2210,2211,2212,2750,3936,4495)
RTCP	Real-Time Control Protocol. Protocolo que proporciona información de estadísticas y de control de un flujo RTP a modo de realimentación para estimar la calidad del servicio. Está especificado en la RFC 1889 y, posteriormente en la RFC 3550.
RTP	Real-Time Transport Protocol. Protocolo de nivel de aplicación

	que permite la distribución de vídeo en formato streaming (típicamente a través de UDP). Se suele emplear típicamente junto con RTCP. Está especificado en la RFC 1889 y, posteriormente en la RFC 3550.
RTSP	Real-Time Streaming Protocol. Protocolo que permite funcionalidades de reproducción, pausa, posicionamiento (avance y retroceso), etc. en la visualización de un flujo de vídeo. Esta especificado en la RFC 2326.
RTT	Round Trip Time. Tiempo de ida y vuelta transcurrido entre la emisión de un mensaje (paquete de red) y la llegada de su acuse de recibo.
UMD	Ultra Mobile Device. Dispositivo portátil con capacidad de conexión a Internet (PDA o Tablet)
SGML	Standard Generalized Markup Language. Es un sistema para la organización y etiquetado de documentos. La Organización Internacional de Estándares (ISO) normalizó este lenguaje en 1986.
SLA	Service Level Agreement. Acuerdo del nivel de servicio. Se trata de una especie de contrato entre dos entidades que pretende describir el tipo de servicio proporcionado por una entidad a la otra. Normalmente se emplean métricas objetivas relativas al QoS.
SNMP	Simple Network Management Protocol. Es un protocolo de aplicación orientado al intercambio de información de administración entre dispositivos de red. El protocolo dispone de tres versiones.
SOAP	Simple Object Access Protocol. Protocolo basado en HTTP y XML para invocar métodos y servicios en servidores.
SSH	Secure Shell. Aplicación que permite el acceso remoto seguro mediante encriptación de datos.
TCP	Transmission Control Protocol. Protocolo de nivel 4 que proporciona envío de datos seguro mediante reconocimientos y retransmisiones. Se encuentra documentado en la RFC 793.
TTL	Time-To-Live. Representa el tiempo de vida de un paquete o

	mensaje antes de ser (o poder ser) eliminado, ya sea en la red (IP) o en memoria (DNS).
UDP	User Datagram Protocol. Protocolo de nivel 4 que proporciona entrega de datos no fiable (en contraste con TCP). Se encuentra documentado en la RFC 768.
URI	Uniform Resource Identifier. Se trata de un esquema de nombrado estándar para recursos web. Se encuentra documentado en la RFC 2396.
URL	Uniform Resouce Locator. Subconjunto de URIs que hacen referencia a ubicaciones específicas (basadas en nombres de equipos o hosts)
URN	Uniform Resource Name. Subconjunto de URIs relativas a recursos web que no están vinculadas a ubicaciones específicas, como las URLs.
UTC	Universal Time Coordinates. Es el tiempo de la zona horaria de referencia respecto a la cual se calculan todas las otras zonas del mundo (equivalente a GMT).
XML	eXtensible Markup Language (lenguaje de marcas extensible). Se trata de un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Ejemplos de XML lo constituyen SVG, XTHML y SOAP.
W3C	World Wide Web Consortium. Consorcio internacional que produce recomendaciones para la World Wide Web. Está dirigida por el creador del WWW, Tim Berners-Lee.
WCCP	Web Cache Coordination Protocol. Es un protocolo definido por Cisco que se emplea para definir la manera de redirigir el tráfico IP desde un router a un proxy-caché
WWW	World Wide Web. Sistema de distribución de información basado en hipertexto enlazado y accesible a través de Internet mediante un navegador web.

7. BIBLIOGRAFÍA

[Abe_02] Aberer, K. Hauswirth, M. (2002). *An overview on peer-to-peer information systems*. In Proc. Of the Workshop on Distributed Data and Structures (WDAS), France.

[Adl_99] Adler, S (1999). *The slashdot effect: an analysis of three Internet publications*. Linux Gazette Issue,38

[Agg_98] Aggarwal A., Rabinovich, M. (1998). *Performance of dynamic replication schemes for an Internet hosting service.* Technical Report, HA6177000-981030-01-TM, AT&T Research Labs, Florham Park, NJ, USA.

[Agr_01] Agrawal, D. et al (2001). On the performance of Content Distribution Networks, International Symposium on Performance Evaluation of Computer and Telecommunication Systems, Orlando (USA)

[And_02] Andrews M., Shepherd, B. et al (2002). *Clustering and server selection using passive monitoring*. INFOCOM 2002. Proceedings IEEE Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies., vol.3, no., pp. 1717-1725 vol.3, doi: 10.1109/INFCOM.2002.1019425

[Ant_97] Antonioletti M. (1997). *Load sharing across networked computers*, Edinburgh Parallel Computing Centre. The University of Edinburgh

[Ard_01] Ardaiz O., Freitag, F. et al (2001). *Improving the service time of Web clients using server redirection*. ACM SIGMETRICS Performance Evaluation Review, 29(2), ACM Press, NY, USA, pp. 39–44.

[Ari_03] Ari I., Hong B., et al. (2003). *Managing flash crowds on the Internet*. 11th Modeling, Anal & Simulation of Computation and Telecommunication Systems, pp. 246-249.

[Arl_00] Arlitt M., Jin, T (2000). *A workload characterization study of 1998 world cup Web site*. IEEE Network, pp. 30–37

[Asa_04] Asano T., Tsuno A. et al (2004). *Quality measuring system for network provisioning and customer service in content delivery*. IEEE/IFIP Network Operations and Management Symposium, 2004, vol.1, no., pp.875-876, doi: 10.1109/NOMS.2004.1317776

[Ata_07] Atajanov M, Shimokawa T, et al (2007). *Autonomic Multi-Server Distribution in Flash Crowds Alleviation Network*. In Proc. IFIP 3rd Int. Symp. on Network Centric Ubiquitous Systems (LNCS 4809, Springer), 309–320

[Aya_11] Ayadi, I.; Serhrouchni, A. et al (2011). *HTTP Session Management: Architecture and Cookies Security*. Conference on Network and Information Systems Security (SAR-SSI), vol., no., pp.1-7, doi: 10.1109/SAR-SSI.2011.5931364

[Bae_97] Baentsch M., Baum L., et al (1997). World Wide Web Caching: The Application Level View of the Internet. IEEE Communications Magazine Vol. 35, No. 6.

[Bah_09] Bhatia R., Narlikar, et al. (2009). *UNAP: User-Centric Network-Aware Push for Mobile Content Delivery*. IEEE InfoCom 2009, DOI: 10.1109/INFCOM.2009.5062126, pp.: 2034 – 2042.

[Bak_05] Bakiras S., Loukopoulos T. (2005). *Combining replica placement and caching techniques in content distribution networks*. Computer Communications, 28(9), pp. 1062–1073.

[Bal_03] Balakrishnan H., Kaashoek, M. et al (2003). *Looking up data in P2P systems*. Communications of the ACM, 46(2), ACM Press, NY, USA, pp. 43–48.

[Bal_04] Balamash, A., Krunz, M. (2004). *An overview of web caching replacement algorithms*. IEEE Communications Surveys & Tutorials, vol.6, no.2, pp.44-56, doi: 10.1109/COMST.2004.5342239.

[Bar_96] Bartal, Y. (1996). *Probabilistic approximation of metric space and its algorithmic applications*. In Proc. of 37th Annual IEEE Symposium on Foundations of Computer Science.

[Bar_01] Barbir A., Cain B., et. Al (2001). *Known CDN Request-Routing Mechanisms* – draft-cain-cdnp-known-request-routing-01.txt.

[Bar_03] Barbir A., Cain, B. et al (2003). *Known content network request-routing mechanisms*. Internet Engineering Task Force RFC 3568.

[Bar_04] Barbir A., Batuner O., et al (2004). Policy, authorization, and enforcement requirements of the open pluggable edge services (OPES). Internet Engineering Task Force RFC 3838.

[Bar_04b] Barbir A., Penno R., et al (2004). *An architecture for open pluggable edge services (OPES).* Internet Engineering Task Force RFC 3835.

[Bar_05] Baryshnikov Y., Coffman E.G et al (2005). *Predictability of web-server traffic congestion*, Proc. 10th Intl. Workshop on Web Content Caching and Distribution), pp. 97-103, Sophia Antipolis, France.

[Bar_12] Bari M.F., Chowdhury S. et al (2012). *A survey of naming and routing in information-centric networks*. IEEE Communications Magazine, vol.50, no.12, pp.44-53, doi: 10.1109/MCOM.2012.6384450

[Bek_08] Bektas T., Ouveysi I. (2008). *Mathematical models for resource management and allocation in CDNs*. Lecture Notes in Electrical Engineering, vol. 9, Springer, Berlin, pp. 225–250.

[Ber_92] Berners-Lee T (1992). *Re: Is there a paper which describes the www protocol," WWW-Talk Mailing List,* lists.w3.org/Archives/Public/www-talk/1992JanFeb/0000.html

[Ber_11] Bertrand G., Emile S. et all (2011). *Use Cases for Content Delivery Network Interconnection*. Internet Draft. http://tools.ietf.org/html/draft-ietf-cdni-use-cases-00.

[Ber_11b] Bertrand G., Faucheur F. L. et al (2011). *Content Distribution Network Interconnection (CDNI) Experiments*. Internet Draft. (draft-bertrand-cdni-experiments-02).

[Bla_98] Blake S, Black D., et al. (1998), *An architecture for Differentiated Services*, RFC 2475.

[Bol_85] B. Bollobás. Random Graphs. Academic Press, Inc., Orlando, Florida, 1985

[Bra_94] Braden R., Clark D., et al. (1994). *Integrated Services in the Internet: an Overview*, RFC 1633

[Bra_97] Braden R., Zhang. L, et al. (1997). *Resource Reservarion Protocol (RSVP): Version 1 Functional Specification . RFC 2205*.

[Bre_99] Breslau, L., Cao P. et al (1999). *Web Caching Algorithms*. Proceedings of the USENIX Symposium on Internet Technologies and Systems.

[Bre_99b] Breslau, L.; Cao P at all (1999). *Web caching and Zipf-like distributions: evidence and implications*. INFOCOM '99. 18th Annual Joint Conference of the IEEE Computer and Communications Societies. IEE Proceedings, vol.1, pp.126-134 vol.1, doi: 10.1109/INFCOM.1999.749260.

[Bru_01] R. Brussee, H. Eertink, et al. (2001). *Content Distribution Network. State of the art*, Telematica Instituut (Netherlands).

[Bu_02] Bu T.; Towsley, D. (2002). *On distinguishing between Internet power law topology generators.* IEEE Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies, vol.2, no., pp. 638- 647, doi: 10.1109/INFCOM.2002.1019309

[Buf_09] Buford, J., Yu, H. at al (2009*). P2P Networking and Applications*, Morgan Kaufmann Series in Networking.

[Bye_03] Byers J., Considine, J. et al (2003). *Simple load balancing for distributed hash tables*. In Proc. of 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03), pp. 31–35.

[Buy_06] Buyya R., Pathan A.K., et al (2006). *A Case for Peering of Content Delivery Networks*. IEEE Distributed Systems Online, vol.7, no.10, pp.3, doi: 10.1109/MDSO.2006.57

[Cal_97] Calvert, K.L., Doar, M.B. et al (1997). *Modeling Internet topology*. IEEE Communications Magazine, vol.35, no.6, pp.160-163, doi: 10.1109/35.587723

[Cal_02] Calo S.B., Verma, D.C., et al (2002)., On the effectiveness of Content Distribution Networks, International Symposium on Performance Evaluation of Computer and Telecommunication Systems, San Diego (USA)

[Cam_02] Cameron C., Low, S.H., et al (2002). *High-density model of content distribution network.* Proc. Information, Decision and Control.

[Can_07] Cancela H., Rodriguez-Bocca P. et al (2007). *Perceptual quality in p2p multi-source video streaming policies*. IEEE Global Telecommunications Conference (GLOBECOM '07), pp. 2780–2785.

[Cao_97] Cao P., Irani S. (1997). Cost-Aware WWW Proxy Caching Algorithms. Proceedings of the 1997 USENIX Symposium on Internet Technology and Systems, pages 193-206.

[Car_97] Carter R.L., Crovella, M.E. (1997), *Server selection using dynamic path characterization in wide-area networks*, IEEE Infocom'97.

[Car_01] Cardellini V., Casalicchio E., et al. (2001). *The state of the art in locally distributed Web-server systems*, IBM Research Report RC 22209, Yorktown Heights, NY.

[Cas_99] Castro M., Dwyer M., et al. (1999). *Load balancing and control for distributed World Wide Web servers*. Proceedings of the 1999 IEEE International Conference on Control Applications, vol.2, no., pp.1614-1619, doi: 10.1109/CCA.1999.801213

[Cas_01] Casalicchio E., Colajanni M. (2001). *A client-aware dispatching algorithm for Web clusters providing multiple services*. Proceedings of the 10th International World Wide Web Conference, pages 535–544, Hong Kong.

[Cat_92] Cate Y. (1992). *Alex-A Global Filesytem*. Proceedings of the 1992 Usenix File System Workshop, Ann Arbor, MI, May 1992,1-12.

[Cec_10] Cece F., Formicola V. et al (2010). *An extended ns-2 for validation of load balancing algorithms in Content Delivery Networks*. Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, Article No. 32, ISBN: 978-963-9799-87-5, doi:10.4108/ICST.SIMUTOOLS2010.8716

[Cic_03] Ciciani B.; Quaglia F. et al (2003). *Analysis of design alternatives for reverse proxy cache providers. Modeling*, 11th IEEE/ACM International Symposium on Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003., vol., no., pp. 316- 323, doi: 10.1109/MASCOT.2003.1240676

[Cie_00] Cieslak M., Foster D. et al (2000).*Web cache coordination protocol versión 2*. Internet Draft.

[Cla_06] Claypool M., Claypool K. (2006). *Latency and player actions in online games*. Communications of the ACM, vol. 49, pp. 40–45, doi>10.1145/1167838.1167860.

[Cla_10] Claypool M., Claypool K. (2010). *Latency can kill: precision and deadline in online games*. Proceedings of the first annual ACM SIGMM conference on Multimedia systems (MMSys '10), pp. 215-222, ACM New York, NY, ISBN: 978-1-60558-914-5, doi>10.1145/1730836.1730863

[Cof_98] Coffman K.G., Odlyzko A.M. (1998). *The size and growth rate of the Internet*. First Monday (Journal on the Internet), Vol. 3, nº 10.

[Cof_02] Coffman K. G., Odlyzko A.M. (2001) *Internet growth: Is there a "Moore's Law" for data traffic?*, Handbook of Massive Data Sets.

[Coo_01] Cooper I., Melve I., et al (2001). *Internet Web replication and caching taxonomy*. Internet Engineering Task Force RFC 3040.

[Cop_05] Coppens J., Wauters T. et al (2005). *Evaluation of a monitoring-based architecture for delivery of high quality multimedia content*. 10th IEEE Symposium on Computers and Communications, 2005. ISCC 2005, vol., no., pp. 611- 616, doi: 10.1109/ISCC.2005.68

[Cra_00] Cranor C.D, Green M. et al (2000).PRISM, an IP-based architecture for broadband access to TV and other streaming media." Proceedings of NOSSDAV 2000

[Cra_01] Craig J. (2001) *Caching Hierarchies: Understanding Content Distribution/Delivery Networks*, Dell Power Solutions, Issue 1.

[Cha_96] Chankhunthod, A., Danzig, P. B., et al. (1996). *A Hierarchical Internet Object Cache*, Usenix Technical Conference.

[Che_00] Cheng K., Kambayashi, Y. (2000). *LRU-SP: a size-adjusted and popularity-aware LRU replacement algorithm for web caching*. The 24th Annual International Computer Software and Applications Conference, COMPSAC 2000, pp.48-53, 2000, doi: 10.1109/CMPSAC.2000.884690

[Che_02] Che H., Tung Y. et al (2002). *Hierarchical Web caching systems: modeling, design and experimental results*. IEEE Journal on Selected Areas in Communications, vol.20, no.7, pp. 1305- 1314, doi: 10.1109/JSAC.2002.801752

[Che_02b] Chen Y., Katz, R. H. et al (2002). *Dynamic replica placement for scalable content delivery*. In Proc. of International Workshop on Peer-to-Peer Systems (IPTPS 02), LNCS 2429, Springer-Verlag, pp. 306–318.

[Che_03] Chen Y., Qiu L. et al. (2003). *Efficient and adaptive Web replication using content clustering*. IEEE Journal on Selected Areas in Communications, vol.21, no.6, pp. 979- 994,doi: 10.1109/JSAC.2003.814608

[Che_05] Chen C., Ling Y. et al (2005. *Scalable request routing with next-neighbor load sharing in multi-server environments*. 19th International Conference on Advanced Information Networking and Applications, vol.1, no., pp. 441- 446, doi: 10.1109/AINA.2005.303

[Che_07] Chen, S., Shen, B., et al. (2007). SProxy: A Caching Infrastructure to Support Internet Streaming. IEEE Transactions on Multimedia, Volume 9, Issue 5, pp. 1062-1072, DOI: 10.1109/TMM.2007.898943.

[Chi_09] Chiang, M.L., Wu, C.H, et al (2009). *New content-aware request distribution policies in web clusters providing multiple services*. Proceedings of the 2009 ACM symposium on Applied Computing.

[Cho_12] Choy S., Wong B.et al. (2012). *The brewing storm in cloud gaming: A measurement study on cloud to end-user latency*. 11th Annual Workshop on Network and Systems Support for Games (NetGames), vol., no., pp.1-6, 22-23, doi: 10.1109/NetGames.2012.6404024

[Dan_97] Dan, A., Sitaram, D. (1997). *Multimedia Caching Strategies for Heterogeneous Applications and Server Environments*, Multimedia Tools and Applications, vol.4, no.3.

[Dat_01] A. Datta, K. Dutta, et al. (2001) . *Dynamic Content Acceleration: A Caching Solution to Enable Scalable Dynamic WebPage Generation*. Proceedings of the ACM International Conference on Management of Data (SIGMOD).

[Day_03] Day M., Cain B. et al (2003), A Model for Content Internetworking (CDI), RFC 3466.

[Dee_95] Deering S. (1995). *IP Multicast and the MBOne: Enabling Live, Multiparty, Multimedia Communication on the Internet*. CERN

[Dev_12] Devi, U., Polavarapu, R. et al (2012). *On the partial caching of streaming video*. 2012 IEEE 20th International Workshop on Quality of Service (IWQoS), vol., no., pp.1-9, doi: 10.1109/IWQoS.2012.6245982

[Dil_02] Dilley, B. Maggs, et al. (2002). *Globally distributed content delivery*, IEEE Internet Computing, pp. 50-58.

[Dio_00] Diot C., Levine B.L, et al. (2000). *Deployment Issues for the IP multicast Service and Architecture*. IEEE Network , pp. 78-88.

[Doa_96] Doar, M.B. (1996). *A better model for generating test networks*. Global Telecommunications Conference Communications: The Key to Global Prosperity , vol., no., pp.86-93, doi: 10.1109/GLOCOM.1996.586131

[Dou_01] Douglis F., Kaashoek M. F. (2001). *Scalable Internet services*. IEEE Internet Computing, 5(4), pp. 36–37.

[Doy_02] Doyle R.P, Chase, J.S., et al. (2002). *The trickle-down effect: web caching and server request distribution*, Computer Communications, vol 25, pp. 345-356

[Du_02] Du, X., Yang, Z. et al. (2002). *An active caching scheme*. IEEE 2002 International Conference on Communications, Circuits and Systems and West Sino Expositions, Volume 1, pp. 705-709, DOI: 10.1109/ICCCAS.2002.1180713

[Dus_97] Duska, B. Marwood D., et al (1997). The measured Access Characteristics of the World-Wide-Web Client Proxy Caches. USENIX Symposium on Internet Technologies and Systems

[Emt_92] Emtage, A., Deutch, P. (1992). A*rchie - An Electronic Directory Service for the Internet*, Proceedings of the Winter USENIX Conference, San Francisco, CA.

[Etsi_12] ETSI TS 102 990 V1.1.1 (2012-11). *Media Content Distribution (MCD); CDN Interconnection, use cases and requirements*. Technical Specification.

[Fal_99] Faloutsos C., Faloutsos P. et al (1999). *On Power-Law Relationships of the Internet Topology*. In Proceedings of the ACM SIGCOMM.

[Fan_98] Fan, L., Cao, P., et al (1998*). Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol*. Proceedings of ACM SIGCOMM, 1998

[Fan_11] Fan K.,Wang Y. et al (2011). Forward Secure Proxy Blind Signature Scheme. Third International Conference on Intelligent Networking and Collaborative Systems (INCoS), vol., no., pp.263-267, doi: 10.1109/INCoS.2011.95

[Fei_98] Fei Z.M., Bhattacharjee, S. et al (1998). *A novel server selection technique for improving the response time of a replicated service*. INFOCOM '98. Proceedings. IEEE

Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, vol.2, no., pp.783-791.

doi: 10.1109/INFCOM.1998.665101

[Fei_01] Fei Z. (2001). *A novel approach to Managing Consistency in Content Distribution Networks*, Proceedings of Web Caching and Content Distribution Workshop (WCW'01), Boston, MA.

[Fen_97] Feng W., Mishra P.P. et al (1997). *A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video*. INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, vol.1, no., pp.58-66 vol.1, doi: 10.1109/INFCOM.1997.635114

[Fen_02] Feng W., Chang W. et al. (2002). *Provisioning On-line Games: A Traffic Analysis of a Busy Counter-Strike Server*. ACM SIGCOMM Internet Measurement Workshop, pp. 151-156.

[Fie_99] Fielding R., Gettys J., et al. (1999). *Hypertext Transfer Protocol HTTP/1.1*, RFC 2616.

[For_03] Fortino, G., Nigro, L. (2003). *Collaborative Learning on-Demand on the Internet MBone*. In Ghaoui C (ed) Usability Evaluation of Online Learning Programs. Idea Group Publishing, Hershey (PA), USA, pp 40–68

[For_07] Fortino, G., Russo, W. et al. (2007). *CDN-supported Collaborative Media Streaming Control*. IEEE Multimedia, 14(2):60–71

[For_09] Fortino G., Mastroianni M. et al (2009). *A hierarchical control protocol for group-oriented playbacks supported by content distribution networks*. Journal of Network and Computer Applications, Volume 32 Issue 1, pp. 135-157

[Fran_01] Francis P., Jamin S. et al (2001). *IDMaps: a global Internet host distance estimation service*. IEEE/ACM Transactions on Networking, vol.9, no.5, pp.525-540, doi: 10.1109/90.958323

[Fre_03] Freedman, M.J., Mazières D. (2003). *Sloppy Hashing and Self-Organizing Clusters*. In Proc. 2nd Intl. Workshop on Peer-to-Peer Systems (IPTPS '03) Berkeley, CA.

[Fre_04] Freedman M.J., Freudenthal E.,et al (2004). *Democratizing Content Publication with Coral*. Proc. 1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04), San Francisco, CA.

[Fre_06] Freedman, M. J., Lakshminarayanan, K. et al (2006). *OASIS: anycast for any service*. Proc. of 3rd Symposium of Networked Systems Design and Implementation (NSDI'06), Boston, MA, USA.

[Fre_10] Freedman M.J. (2010). *Experiences with CoralCDN: A Five-Year Operational View*. Proc. 7th USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '10) San Jose, CA.

[Fuj_04] Fujita N., Ishikawa Y. et al (2004). *Coarse-grain replica management strategies for dynamic replication of Web contents*. Computer Networks: The International Journal of Computer and Telecom. Networking, 45(1), pp. 19–34.

[Fuj_12] Fujiwara K., Sato A. et al (2012). *DNS Traffic Analysis: Issues of IPv6 and CDN*. 2012 IEEE/IPSJ 12th International Symposium on Applications and the Internet (SAINT), vol., no., pp.129-137, doi: 10.1109/SAINT.2012.26

[Gad_97] Gadde S., Rabinovich M., et al (1997). Reduce, reuse, recycle: an approach to building large Internet caches. In Proc. of 6th Workshop on Hot Topics in Operating Systems, pp. 93–98.

[Gad_00] Gadde S., Chase, et al. (2000). *Web Caching and Content Distribution: a view from the interior*, In Proc. of the Fifth Int. Web Caching and Content Delivery Workshop.

[Gar_09] Garcia, L.; Arnaiz, L. et al. (2009). Protected seamless content delivery in P2P wireless and wired networks. IEEE Wireless Communications, Volume: 16, Issue: 5, DOI: 10.1109/MWC.2009.5300302, pp. 50 – 57

[Gay_04] Gayek P., Nesbitt R. et al (2004). *A Web content serving utility*. IBM Systems Journal, 43(1), pp. 43–63.

[Gra_89] Gray G., Cheriton D (1989). *Leases: An Efficient Fault-Tolerant Mechansim for Distributed File Cache Consistency*. In Proceedings of the Twelfth ACM Symposium on Operating Systems Principles, pages 202-210,1989

[Gre_01] M., Kalmanek, C., Schur, D., et al (2001), C.J., *Enhanced Streaming services in a content distribution network*, IEEE Internet Computing, pp 66-75

[Gri_01] Gritter M., Cheriton D.R (2001). *An Architecture for Content Routing Support in the Internet*. In the USENIX Symposium on Internet Technologies and Systems.

[Gho_95] Ghosh B., Leighton, F.T., et al. (1995). *Tight Analyses of two local load balancing algorithms*, Proceedings of the 27th Annual ACM Symposium on Theory of Computing, pp. 548-558.

[Had_98] Hadar 0., Cohen R. (1998). *PCRTT Enhancement for off-line video smoothing*. Proc. SPIE, Multimedia Systems and Applications, vol 3528, pp. 89-100, Boston, MA.

[Had_00] Hadar, O., Greenberg, S. (2000). *Statistical multiplexing and admission control policy for smoothed video streams using e-PCRTT algorithm*. International Conference on Information Technology: Coding and Computing, 2000, Proceedings, vol., no., pp.272-277, doi: 10.1109/ITCC.2000.844237

[Ham_98] Hamilton, M., Rousskov, A. et al (1998). *Cache digest specification – version 5*.

[Har_02] Harren M., Hellerstein, J. M. et al (2002). *Complex queries in DHT-based peer-to-peer networks*. Proc. of 1st International Workshop on Peerto-Peer Systems (IPTPS'02).

[Has_09] Hassan, O.A.-H., Ramaswamy, L. et al (2009). *MACE: A Dynamic Caching Framework for Mashups*. IEEE International Conference on Web Services, 2009. ICWS 2009, vol., no., pp.75-82, doi: 10.1109/ICWS.2009.119

[Ho_07] Ho, K.-M., Poon, W.-F., Lo, K.-T. (2007). *Performance Study of Large-Scale Video Streaming Services in Highly Heterogeneous Environment*. IEEE Transactions on Broadcasting, Volume 53 , Issue 4, pp. 763-773, DOI: 10.1109/TBC.2007.908326.

[Ho_08] Ho, K.M., Poon, W.F., Lo, K.T. (2008). *Investigating the Performance of Hierarchical Video-on-Demand System in Heterogeneous Environment*. International Conference on Information Networking, 2008 (ICOIN 2008), pp. 1-5, DOI: 10.1109/ICOIN.2008.4472804.

[Hof_00] Hofmann, M., Eugene Ng, T.S., et al. (2000). *Caching Techniques for Streaming Multimedia over the Internet*. Bell Labs Technical Report.

[Hof_05] Hofmann M., Beaumont, L. R (2005). *Content Networking: Architecture, Protocols, and Practice*. Morgan Kaufmann Publishers, San Francisco, CA, USA, pp. 129–134.

[Hos_06] Hosangar, K., Chuang, J. et al (2006). *Service adoption and pricing of content delivery network (CDN) services*. Technical Report, Social Science Research Network.

[Hob_12] Hobfeld T., Schatz R. et al. (2012). *Challenges of QoE management for cloud applications*. IEEE Communications Magazine, vol.50, no.4, pp.28-36, doi: 10.1109/MCOM.2012.6178831

[Hua_11] Huang R., Nong Q. (2011). *Security Analysis of a Proxy Blind Signature and Its Improved Scheme*. International Conference on Information Technology, Computer

Engineering and Management Sciences (ICM), vol.2, no., pp.179-182, doi: 10.1109/ICM.2011.58

[Huf_02] Huffaker B., Fomenkov M. et al (2002). *Distance metrics in the Internet*. Proc. of IEEE International Telecommunications Symposium, IEEE CS Press, Los Alamitos, CA, USA.

[Itu_96] ITU-T Recommendation P.910 (1996). *Subjective video quality assessment methods for multimedia applications*. International telecommunication Union, Geneva, Switzerland.

[Itu_99] International Telecommunication Union (1999). *Subjective video quality assessment methods for multimedia applications*, Rec. ITU-T P.910.

[Itu_02] ITU-R Recommendation BT.500-11 (2002). *Methodology for the subjective assessment of the quality of television pictures*. International Telecommunication Union, Geneva, Switzerland.

[Jam_01] Jamin, S., Jin C. et al (2001). *Constrained mirror placement on the Internet*. INFOCOM 2001. 20th Annual Joint Conference of the IEEE Computer and Comm. Societies. Proceedings, vol.1, pp.31-40, doi: 10.1109/INFCOM.2001.916684

[Jan_00] Jannoti J, Gifford D.K, et al. (2000). *Overcast: Reliable multicasting with an Overlay Network*. Proc. 4th Symposium on Operating Systems Design and Implementation, Usenix.

[Jam_00b] Jamin S., Jin C. et al (2000). *On the placement of Internet instrumentation*. In Proc. of IEEE INFOCOM, Tel-Aviv, Israel, pp. 295–304.

[Jar_11] Jarschel M., Schlosser, D.et al (2011). *An Evaluation of QoE in Cloud Gaming Based on Subjective Tests*. Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), vol., no., pp.330-335, doi: 10.1109/IMIS.2011.92

[Jia_03] Jian N., Tsang, D.H.K. et al (2003). *Hierarchical content routing in large-scale multimedia content delivery network*. IEEE International Conference on Communications, vol.2, no., pp. 854- 859, doi: 10.1109/ICC.2003.1204454

[Jia_05] Jian N., Tsang, D.H.K (2005). *Large-scale cooperative caching and application-level multicast in multimedia content delivery networks*. IEEE Communications Magazine, vol.43, no.5, pp. 98-105, doi: 10.1109/MCOM.2005.1453429

[Jin_99] Jin S., Bestavros A.(1999). *Popularity-Aware GreedyDual-Size Web Caching Algorithms*. Technical Report TR-99/09, Computer Science Department, Boston University.

[Jin_00] Jin C., Chen Q. et al (2000). *Inet: Internet Topology Generator*. Technical Report CSE-TR-433-00, EECS Department, University of Michigan.

[Jin_09] Jing C. (2009). *Application and Research on Weighted Bloom Filter and Bloom Filter in Web Cache*. Second Pacific-Asia Conference on Web Mining and Web-based Application, WMWA '09., vol., no., pp.187-191, doi: 10.1109/WMWA.2009.51

[Joh_00] Johnson K.L., Carr, J.F., et al. (2000). *The measured performance of Content Distribution Networks*, In 5th International Workshop on Web Caching and Content Distribution, Lisbon (Portugal).

[Jos_12] Joskowicz, J. (2012). *Hacia un Modelo Perceptual de Video. Desarrollo de un Modelo Paramétrico General de Estimación de la Calidad Percibida de Video*. Tesis Doctoral, Universidad de Vigo.

[Jun_02] Jung J., Krishnamurthy B., et al (2002). *Flash crowds and denial of service attacks: characterization and implications for CDNs andWeb sites*. In Proc. of the International World Wide Web Conference, pp. 252–262.

[Kah_91] Kahle, B., Medlar, A. (1991). *An Information System for Corporate Users: Wide Area Information Servers*, ConneXions—The Interoperability Report, 5(11).

[Kan_02] Kangasharju J., Roberts J., et al (2002). *Object replication strategies in content distribution networks.* Computer Communications, 25(4), pp. 367–383, 2002

[Kar_99] Karger D., Sherman A. et al (1999). *Web caching with consistent hashing*. Computer Networks, 31(11–16), pp. 1203–1213.

[Kar_94] Karedla R., Love J.S. et al. (1994). *Caching Strategies to Improve Disk System Performance.* IEEE Computer, 27(3):38-46,

[Kar_02] Karlsson M., Karamanolis C., et al. (2002). *A framework for evaluating replica placement algorithms*, Technical Report, HP Laboratories.

[Kar_02b] Karakostas G, Serpanos D.N (2002). *Exploitation of different types of locality for Web caches*. Proceeding of ISCC'02, pp. 207-212.

[Kas_07] Kastaniotis, G.; Maragos, E et al. (2007). *Web Proxy Caching Object Replacement: Frontier Analysis to Discover the Good-Enough Algorithms*. 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '07), pp. 132-137, DOI: 10.1109/MASCOTS.2007.68

[Kat_08] Katsaros D., Pallis G. et al (2009). *CDNs Content Outsourcing via Generalized Communities*. IEEE Transactions on Knowledge ands Data Engineering, vol. 21, n. 1.

[Kim_09] Dae-young K., Jinsung C. (2009). Active caching: a transmission method to guarantee desired communication reliability in wireless sensor networks. IEEE Communications Letters, Volume 13, Issue 6, pp.: 378–380, doi: 10.1109/LCOMM.2009.090023

[Kle_96] Kleinrock L., Gail R. *Queueing Systems: Problems and Solutions*. John Wiley & Sons

[Kri_97] Krishnamurthy B., Wills C.E (1997). *Study of piggyback cache validation for proxy caches in the World Wide Web.* Proceeding of the 1997 USENIX Symposium on Internet Technology and Systems, pp.I-12

[Kri_98] Krishnamurthy B., Wills C.E. (1998). *Piggyback server invalidation for proxy cache coherency*. Computer Networks and ISDN Systems, v.30 n.I-7, p.185-193.

[Kri_00] Kristol D., Montulli L. (2000), *HTTP State Management Mechanism*, RFC 2965.

[Kri_00b] Krishnan P., Raz D. et al (2000). *The cache location problem*. IEEE/ACM Transaction on Networking, 8(5).

[Kri_01] Krishnamurthy B., Willis, C. et al (2001). *On the use and performance of content distribution network*s. Proc. of 1st International Internet Measurement Workshop, ACM Press, pp. 169–182.

[Kru_95] Krunz M., Hughs H (1995). *A traffic model for MPEG-coded VBR streams*. ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, Ottawa.

[Lao_05] Laoutaris N., Zissimopouluos V. et al (2005). *On the optimization of storage capacity allocation for content distribution*. The International Journal of Computer and Telecommunications Networking archive, Volume 47, Issue 3, pp. 409-428 , Elsevier, doi:10.1016/j.comnet.2004.07.020

[Laz_01] Lazar I., Terrill, W (2001). *Exploring content delivery networking*. IT Professional, 3(4), pp. 47–49.

[Leg_91] Le Gall D. (1991). *MPEG: a video compression standard for multimedia applications*. Communications of the ACM, 34, pp. 47-58.

[Li_98] Li B., Deng X. et al (1998*). On the optimal placement of Web proxies in the Internet: The linear topology*. In Proceedings of the IFIP TC-6 8th International Conference on High Performance Networking (HPN'98), Kluwer, Amsterdam, The Netherlands, pp.485–495.

[Li_99] Li B., Golin, M. J. et al (1999). On the optimal placement of Web proxies in the Internet. In Proc. of IEEE INFOCOM, NY, USA, pp. 1282–1290.

[Liu_10] Liu H., Chen M. (2010); Evaluation of web caching consistency, 3rd Int. Conference on Advanced Computer Theory and Engineering (ICACTE), 2010 , vol.5, pp.V5-130-V5-132, doi: 10.1109/ICACTE.2010.5579113.

[Loh_11] Lohmar T, Einarsson T. et al (2011). *Dynamic adaptive HTTP streaming of live content*. 2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), vol., no., pp.1-8, doi: 10.1109/WoWMoM.2011.5986186

[Lou_11] Louedec Y.L., Timmerer C. et al (2011). *Catalogue of Advanced Use Cases for Content Delivery Network Interconnection*. Internet Draft.

[Lu_11] Lu Z., Gao X at al (2011). *Scalable and Reliable Live Streaming Service through Coordinating CDN and P2P*. 2011 IEEE 17th International Conference on Parallel and Distributed Systems (ICPADS), vol., no., pp.581-588, doi: 10.1109/ICPADS.2011.113

[Lun_10] Lungaro, P., Segall, Z., Zander, J. (2010). *Predictive and Context-Aware Multimedia Content Delivery for Future Cellular Networks*. IEEE Vehicular Technology Conference (VTC 2010-Spring), 2010,DOI: 10.1109/VETECS.2010.5493664, pp.: 1 – 5

[Luo_97] Luotonen, A. (1997). *Web proxy Servers*, Prentice Hall, ISBN 0-13-680612-0

[Ma_04] Ma, W.H, Du, D.H.C. (2004). *Design a progressive video caching policy for video proxy servers*. IEEE Transactions on Multimedia, Volume: 6 , Issue: 4, pp. 599-610, DOI: 10.1109/TMM.2004.830819.

[Mal_93] Malkin G. (1993). *Traceroute using an IP option*. RFC 1393

[Man_10] Manikandan, C.V., Manimozhi, P., et al. (2010). *Efficient load reduction and congestion control in Internet through multilevel Border Gateway Proxy Caching*. IEEE International Conference on Computational Intelligence and Computing Research (ICCIC 2010), pp. 1-4, DOI: 10.1109/ICCIC.2010.5705859.

[Mao_02] Mao Z. M., Cranor, C. D. et al (2002). *A precise and efficient evaluation of the proximity between Web clients and their Local DNS servers*. Proc. of the USENIX 2002 Annual Technical Conference, Monterey, CA, USA, pp. 229–242.

[Mar_03] Kind A., Meztler B (2003). *Rate based Active Queue Management with Token Buckets*. High-Speed Networks and Multimedia Communications, Lecture Notes in Computer Science Volume 2720, pp. 176-187

[Mar_04] Martinez O., Palau S (2004). *Introducción a la programación de protocolos de comunicaciones con Network Simulator 2*, ISBN 9788484543480.

[Mas_03] Mirco M., Parravicini, E. (2003) *Impact of Request Routing Algorithms on the Delivery Performance of Content Delivery Networks*, 22nd IEEE International Performance Computing and Communications Conference, Phoenix (USA).

[Med_01] Medina A., Lakhina A. et al (2001). *BRITE: An Approach to Universal Topology Generation*. In Proceedings of MASCOTS 2001, Cincinnati, OH,

[Men_09] Menai, M.F., Fieau, F., et al. (2009). *Demonstration of Standard IPTV Content Delivery Network Architecture Interfaces: Prototype of Standardized IPTV Unicast Content Delivery Server Selection Mechanisms*. IEEE Consumer Communications and Networking Conference, 2009. CCNC DOI: 10.1109/CCNC.2009.4785012, pp. 1 - 2

[Mer_01] Merit Network, Inc.: "NSFNET History of Usage by Service." ftp://nic.merit.edu/nsfnet/statistics/history.ports) [Accessed: April, 05, 2008].

[Mia_02] Miao, Z.; Ortega, A. (2002). *Scalable proxy caching of video under storage constraints*. IEEE Journal on Selected Areas in Communications, Volume: 20, Issue: 7, pp. 1315-1327, DOI: 10.1109/JSAC.2002.802061.

[Mil_02] Milojicic D. S., Kalogeraki, V. et al (2002). *Peer-to-peer computing*. Technical Report, HP Laboratories, Palo Alto, CA, HPL-2002-57.

[Mit_01] Mitzenmacher, M. (2001). *The power of two choices in randomized load balancing*. IEEE Transactions on Parallel and Distributed Systems, vol.12, no.10, pp.1094-1104, doi: 10.1109/71.963420

[Moc_87] Mockapetris, P. (1987). Domain names- concepts and facilities, RFC 1034.

[Moc_87b] Mockapetris, P. (1987) Domain names – implementation and specification, RFC 1035.

[Mol_04] Molina B., Ruiz V. et al (2004). *A closer look at a content delivery network implementation*. Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference, MELECON 2004, vol.2, no., pp. 685- 688, doi: 10.1109/MELCON.2004.1347022

[Mol_04b] Molina B., C.E. Palau et al (2004). *Modeling content delivery networks and their performance*. Journal on Computer Communications, vol. 27 Issue 15, pp 1401-1411, doi:10.1016/j.comcom.2004.05.012

[Mol_06] Molina B., C.E. Palau et al (2006). *On Content Delivery Network Implementation*. Journal on Computer Communications, Volume 29 Issue 12, pp. 2396-2412 ,Elsevier, doi>10.1016/j.comcom.2006.02.016

[Mol_12] Molina B., C.E. Palau et al (2012). *CDN Modeling and Performance*. Next Generation Content Delivery Infrastructures: Emerging Paradigms and Technologies, chapter 1, pp 1-28, doi: 10.4018/978-1-4666-1794-0.ch001

[Mon_10] Montagud, M. Boronat, F.(2010). *A new network simulator 2 (NS-2) module based on RTP/RTCP protocols to achieve multimedia group synchronization*. In Proceedings of the 3rd international ICST Conference on Simulation Tools and Techniques, Torremolinos, Malaga, Spain.

[Ngu_05] Nguyen T.V., Safaeli F. et al (2005). *Provisioning overlay distribution* networks. The International Journal of Computer and Telecommunications Networking archive, Volume 49, Issue 1, pp. 103-118, Elsevier , doi:10.1016/j.comnet.2005.04.001

[Ni_03] Jian Ni., Tsang, D.H.K. et al (2003). *Hierarchical content routing in large-scale multimedia content delivery network*. IEEE International Conference on Communications, vol.2, no., pp. 854- 859, doi: 10.1109/ICC.2003.1204454

[Ni_05] Ni J., Tsang, D.H.K. (2005). *Large-scale cooperative caching and application-level multicast in multimedia content delivery networks*. IEEE Communications Magazine, vol.43, no.5, pp. 98- 105, doi: 10.1109/MCOM.2005.1453429

[Nist_11] Mell P., Grance T. (2011). The NIST Definition of Cloud Computing. NIST Special Publication 800-145.

[Niu_09] Niu B., Wang L. (2009). *Reduced Cache Digests*. WASE International Conference on Information Engineering, 2009. ICIE '09., vol.2, no., pp.210-213, doi: 10.1109/ICIE.2009.38

[Niv_11] Niven-Jenkins B., Faucheur F. L. et al (2011). Content Distribution Network Interconnection (CDNI) Problem Statement. Internet Draft. http://tools.ietf.org/html/draft-ietf-cdni-problem-statement-01.

[Oli_05] Oliveira C.AS, Pardalos P.M (2005). *A survey of combinatorial optimization problems in multicast routing*. Computers and Operations Research, Volume 32 Issue 8, pp. 1953 – 1981, Elsevier Science Ltd. Oxford, UK, ISSN: 0305-0548, doi:10.1016/j.cor.2003.12.007

[One_93] O'Neil E.J., O'Neil P.E et al. (1993). *The LRU-K Page Replacement Algorithm for Database Disk Buffering*. In Proceedings of ACM SIGMOD International Conference on Management of Data, pages 297-306.

[One_03] O'Neill E. T., Lavoie B. F., et al. (2003). Tends in the Evolution of the Public Web, OCLC Office of Research, D-Lib Magazine.

[Pai_98] Pai V. S., Aron M. et al (1998). *Locality-aware request distribution in cluster-based network servers*. ACM SIGPLAN Notices, 33(11), ACM Press, NY, USA, pp. 205–216.

[Pal_00] Palmer, C.R., Steffan, J.G. (2000). *Generating network topologies that obey power laws*. IEEE Global Telecommunications Conference, vol.1, no., pp.434-438, doi: 10.1109/GLOCOM.2000.892042

[Pal_05] Pallis, G., Vakali, A. et al. (2005). *A latency-based object placement approach in content distribution networks*. Third Latin American Web Congress, 2005 , vol., no., pp. 8, doi: 10.1109/LAWEB.2005.3

[Pal_06] Pallis, G., Vakali, A (2006). *Insight and perspectives for content delivery networks*. Communications of the ACM, 49(1), ACM Press, NY, USA, pp. 101–106.

[Pal_06b] G. Pallis, K. Stamos et al. (2006). *Replication Based on Objects Load under a Content Distribution Network*. 22nd International Conference on Data Engineering Workshops, vol., no., pp.53, doi: 10.1109/ICDEW.2006.127

[Pan_02] Pantel L., Wolf L. (2002). On the impact of delay on real-time multiplayer games. Proceedings of the International Workshop on Network and Operating System Support for Digital and Audio (NOSSDAV), pp. 23-29

[Pan_06] Pan C, Atajanov M, et al (2006). *FCAN: Flash Crowds Alleviation Network Using Adaptive P2P Overlay of Cache Proxies*. In IEICE Trans. On Communications, E89-B (4):1119–1126.

[Pad_02] Padmanabhan, V.N., Wang, H.J., et al. (2002). Distributing Streaming Media content using cooperative networking, In ACM/IEEE NOSSDAV, Miami (USA).

[Par_93] Partridge C., Mendez, T. et al (1993). *Host anycasting service*. Internet Engineering, Task Force RFC 1546.

[Par_08] Park Y.; Lee Y. et al (2008). *Hybrid Segment-Based Transcoding Proxy Caching of Multimedia Streams*. IEEE 8th International Conference on Computer and Information Technology Workshops, 2008. CIT Workshops 2008., vol., no., pp.319-324, doi: 10.1109/CIT.2008.Workshops.67

[Pat_07] Pathan**,** A. M. K., Buyya, R. (2007). *A Taxonomy and Survey of CDNs*, Technical Report, GRIDS-TR-2007-4, University of Melbourne, Australia.

[Pat_07b] Pathan M., Broberg, J. et al (2007). An Architecture for Virtual Organization (VO)-Based Effective Peering of Content Delivery Networks, UPGRADECN' 07. Proc. of the 16th IEEE International Symposium on High Performance Distributed Computing (HPDC), Monterey, California, USA.

[Pat_08] Pathan, M., Buyya R. Vakali A. (2008). *Content Delivery Networks: State of the Art, Insights, and Imperatives*. Lecture Notes in Electrical Engineering, 2008, Volume 9, Part I, 3-32, DOI: 10.1007/978-3-540-77887-5_1

[Pav_11] Pavlou G. (2011). *Keynote 2: Information-centric networking: Overview, current state and key challenges*. 2011 IEEE Symposium on Computers and Communications (ISCC), vol., no., pp.1, doi: 10.1109/ISCC.2011.5984767

[Pen_03] Peng, G (2003). *CDN: Content distribution network*. Technical Report TR-125, Experimental Computer Systems Lab, Department of Computer Science, State University of New York, StonyBrook, NY.

[Per_12] Pérez, I., Palau C.E et al (2012). *Wireless Multimedia Content Distribution Architecture*. Next Generation Content Delivery Infrastructures: Emerging Paradigms and Technologies, chapter 4, pp. 78-104, doi: 10.4018/978-1-4666-1794-0.ch004.

[Pia_09] Piamrat, K., Viho, C. et al (2009). *Quality of Experience Measurements for Video Streaming over Wireless Networks*. Sixth International Conference on Information Technology: New Generations (ITNG '09), pp: 1184 – 1189.

[Pie_01] Pierre, G., van Steen, M. (2001). *Globule: a platform for self-replicating Web Documents*, Proceedings of the International Conference on Protocols for Multimedia Systems.

[Pie_02] Pierre, G., van Steen, M. et al. (2002). *Dynamically selecting optimal distribution strategies for Web documents*, IEEE Transactions on Computers 51, no. 6, 637-651.

[Pie_03] Pierre, G., and van Steen (2003. M., *Design and Implementation of a User-Centered Content Distribution Network*, Proceedings of the Third IEEE Workshop on Internet Applications.

[Pie_06] Pierre, G., van Steen, M. (2006). Globule: a collaborative content delivery network. IEEE Communications Magazine, vol.44, no.8, pp.127-133, doi: 10.1109/MCOM.2006.1678120

[Plag_06] Plagemann T., Goebel Vera et al (2006). *From content distribution networks to content networks - issues and challenges*. Elsevier Computer Communications, Volume 29 Issue 5, pp. 551-562, ISSN: 0140-3664, doi:10.1016/j.comcom.2005.06.006

[Pod_02] Podlipnig, S.; Boszormenyi, L. (2002). *Replacement strategies for quality based video caching*. Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '02), Volume: 2, pp. 49-52, DOI 10.1109/ICME.2002.1035395

[Pra_08] Papagianni, C., Tselikas, N.D. et al. (2008). A complete content production and delivery system in a controlled multimedia network. Computers and Communications, 2008. ISCC 2008. DOI: 10.1109/ISCC.2008.4625773, pp.: 44 - 49

[Pres_08] Lo Presti, F.; Petrioli, C. et al (2007). *Distributed Dynamic Replica Placement and Request Redirection in Content Delivery Networks*. Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 07), Istambul, Turkey.

[Pu_12] Pu W., Zou z. et al (2012). *Video adaptation proxy for wireless Dynamic Adaptive Streaming over HTTP*. 2012 19th International Packet Video Workshop (PV), vol., no., pp.65-70, doi: 10.1109/PV.2012.6229745

[Puj_09] Pujolle, G.; Serhrouchni, A et al (2009). *Secure session management with cookies*. 7th International Conference on Information, Communications and Signal Processing, ICICS 2009., vol., no., pp.1-6, doi: 10.1109/ICICS.2009.5397550

[Qiu_01] Qiu L., Padmanabhan, V. N. et al (2001). On the placement of Web server replicas. In Proc. of IEEE INFOCOM, Anchorage, Alaska, USA, pp. 1587–1596.

[Rab_02] Rabinovich M., Spatscheck, O (2002). *Web Caching and Replication.* Addison Wesley, USA.

[Rad_01] Radoslavov P., Govindan R. et al (2001). *Topology-informed Internet replica placement*. In Proc. of Sixth International Workshop on Web Caching and Content Distribution, Boston, Massachusetts.

[Raj_08] Rajkumar B., Mukkadim P. et al (2008). *Content Delivery Networks.* Springer, ISBN: 978-3-540-77876-8

[Ran_03] Rangarajan S., Mukheriee S., et al. (2003). *A Technique for user specific request redirection in a Content Delivery Network,* In Proceedings of the 8th Workshop on Web Content Caching and Distribution, New York.

[Rej_99] Rejaie, R., Handley, M., et al. (1999). Proxy Caching Mechanisms for Multimedia Playback Streams in the Internet, Proc. 4th International Web Caching Workshop.

[Rek_95] Rekhter Y., Li T. (1995). *A border gateway protocol 4*. RFC 1771.

[Riz_98] Rizzo L., Vicisano L. (1998). *Replacement Policies for a Proxy Cache*. Technical report m/98/13, University College London, Department of Computer Science, Cower Street, London WClE 6BT, UK, 1998.

[Rob_90] Robinson J.T., Devarakonda M.V. (1990). *Data Cache Management Using Frequency-based Replacement*. Performance Evaluation Review, 18(1):134-142.

[Rou_98] Rousskov A., Wessels D (1998). *Cache Digests*. Computer Networks and ISDN System, 30(22-23), pp. 2155-2168

[Rub_06] Rubino G., Varela M. et al (2006). Controlling Multimedia QoS in the Future Home Network Using the PSQA Metric. Computer Journal, vol. 49, no. 2, pp. 137–155.

[Sah_00] Sahasrabuddhe L.H., Mukherjee B. (2000). Multicast Routing Algorithms and Protocols: A tutorial. IEEE Network , pp. 90-102.

[San_11] Sánchez J. (2011). Computación distribuida. Sistemas de autenticación y autorización en Internet. Universitat de Lleida.

[Sar_02] Saroiu S., Gummadi, K.P, et al. (2002). *An Analysis of Internet Content Delivery Systems*. Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI), Boston, MA.

[Sar_06] Sarat S., Pappas V. et al (2006). *On the Use of Anycast in DNS.* Proceedings.15th International Conference on Computer Communications and Networks, 2006., vol., no., pp.71-78, doi: 10.1109/ICCCN.2006.286248

[Sat_12] Sathiyamoorthi, V.; Bhaskaran, V.M. (2012). *Web caching through modified cache replacement algorithm*. International Conference on Recent Trends In Information Technology (ICRTIT), pp.483-487, doi: 10.1109/ICRTIT.2012.6206749

[Sav_98] Savetz K., Randal N. et al (1998)*MBONE: Multicasting Tomorrow's Internet.* ISBN: 1-56884-723-8

[Sax_08] Saxena M, Sharan U. et al (2008). *Video Services in Web 2.0: A Global Perspective*. Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV ’08), pp. 39–44.

[Say_98] Sayal M., Breitbart, Y., et al. (1998). *Selection algorithms for replicated web servers,* Proceedings of the Workshopon Internet Server Performance '98

[Sch_96] Schulzrinne H., Casner S. et al (1996). *RTP: A Transport Protocol for Real-Time Applications.* RFC 1889.

[Sch_98] Schulzrinne H., Rao A. et al (1998). *Real Time Streaming Protocol (RTSP).* RFC 2326.

[Sen_97] Sen S., Dey J. et al (1997). *CBR transmission of VBR stored video*. SPIE Symposium on Voice, Video and Data communications, Dallas.

[Sen_99] Sen S., Rexford J., et al. (1999). Proxy Prefix Caching for Multimedia Streams, Proc. IEEE INFOCOM, IEEE Press.

[Ser_00] Serpanos, D.N., Karakostas, G. et al (2000). *Effective caching of Web objects using Zipf's law*. IEEE International Conference on Multimedia and Expo (ICME 2000), vol.2, no., pp.727-730, doi: 10.1109/ICME.2000.871464

[Set_10] Setton, E., Girod, B. (2010). Peer-to-Peer Video Streaming, Ed Springer.

[Sha_49] Shannon C. E. (1949), *Communication in the presence of noise*, Proc. Institute of Radio Engineers, vol. 37, no.1, pp. 10-21

[Sha_01] Shaikh A., Tewari, R. et al (2001*). On the effectiveness of DNS-based server selection*. INFOCOM 2001. Proceedings of the IEEE 20th Annual Joint Conference of the IEEE Computer and Communications Societies, vol.3, no., pp.1801-1810 vol.3,doi: 10.1109/INFCOM.2001.916678

[She_03] Sheldon N., Girard E. et al. (2003). The effect of latency on user performance in Warcraft 3. Proceedings of the 2nd Workshop on Network and System Support for Games, pp. 3-14

[She_11] Sheu S-T., Huang C-H. (2011). *Mixed P2P-CDN system for media streaming in mobile environment*. 2011 7th International Wireless Communications and Mobile Computing Conference (IWCMC), vol., no., pp.657-660, doi: 10.1109/IWCMC.2011.5982624

[Shi_00] Shimokawa T, Yoshida N, et al (2000). *Flexible Server Selection Using DNS*. In Proc. Int.Workshop on Internet 2000, in conjunction with IEEE-CS 20th Int. Conf. on Distributed Computing Systems, A76–A81.

[Shi_04] Shi L., Gu Z. et al (2004). Popularity-based selective Markov model. Proc. of IEEE/WIC/ACM International Conference on Web Intelligence, pp. 504-507.

[Shi_05] Shi L., Gu Z. et al (2005). *Modeling Web objects' popularity*. Proceedings of 2005 International Conference on Machine Learning and Cybernetics, 2005, vol.4, no., pp.2320-2324, doi: 10.1109/ICMLC.2005.1527331

[Shi_06] Shimokawa T, Yoshida N, et al (2006). *Server Selection Mechanism with Pluggable Selection Policies.* In: Electronics and Communications in Japan, III, 89(8):53–61.

[Shu_00] Shudong J., Bestavros A. (2000). *Popularity-aware greedy dual-size Web proxy caching algorithms*. 20th International Conference on Distributed Computing Systems, 2000. Proceedings., vol., no., pp.254,261, doi: 10.1109/ICDCS.2000.840936

[Sid_08] Sidiropoulos A., Pallis G. et al (2008). Prefetching in Content Distribution Networks via Web Communities Identification and Outsourcing. World Wide Web Journal, vol. 11, Issue 1, pp. 39-70, Springer US, doi:10.1007/s11280-007-0027-8

[Sir_12] Siris V.A., Ververidis C.N. (2012). *Information-Centric Networking (ICN) architectures for integration of satellites into the Future Internet*. IEEE First AESS European Conference on Satellite Telecommunications (ESTEL), vol., no., pp.1-6, doi: 10.1109/ESTEL.2012.6400134

[Siv_04] Sivasubramanian S., Szymaniak, M. et al (2004). *Replication of Web hosting.* ACM Computing Surveys systems, 36(3), ACM Press, NY, USA.

[Siv_05] Sivasubramanian S., Pierre G. et al (2005). *GlobeCBC: Content-blind result caching for dynamic web applications*, Technical report IR-CS-022, Vrije Universiteit.

[Siv_05b] Sivasubramanian S., Pierre G. et al (2005). *GlobeDB: Autonomic Data Replication for Web Applications*, Proceedings of the 14th International World-Wide Web Conference.

[Siv_07] Sivasubramanian S., Pierre G. et al (2007). *Analysis of caching and replication strategies for Web applications*. IEEE Internet Computing, 11(1), pp. 60–66.

[Siv_10] Sivakoumar, R., Anandhakumar, P.et al (2010). *Dynamic Caching and Replacing Mechanism for Multiple Videos over Quality of Service Networks*. 2010 First International Conference on Integrated Intelligent Computing (ICIIC), vol., no., pp.296-301, doi: 10.1109/ICIIC.2010.35

[Sta_00] Stardust.com (2000), *The ins and outs of Content Delivery Networks*, White Book

[Sta_01] Stardust.com (2001). *Content Networking and Edge Services: Leveraging the Internet for profit*, White Book.

[Sta_06] Stamos K., Pallis G. et al (2006). *Integrating caching techniques on a content distribution network*. In Proc. of 10th East-European Conference on Advances in Databases and Information Systems (ADBIS 2006), Springer-Verlag, Thessaloniki, Greece.

[Sta_10] Stamos K., Pallis G et al (2010). *CDNsim: A simulation tool for content distribution networks*. ACM Transactions on Modeling and Computer Simulation (TOMACS), Volume 20 Issue 2, Article No. 10, ACM New York, doi:10.1145/1734222.1734226

[Sto_03] Stoica I., Morris, R. ety all (2003). Chord: a scalable peer-to-peer lookup protocol for Internet applications. IEEE/ACM Transactions on Networking (TON), 11(1), ACM Press, NY, USA, pp. 17–32.

[Suk_09] Sukhov A.M., Kuznetsova N. (2009). *What type of distribution for packet delay in a global network should be used in the control theory?* Computing Research Repository (CoRR), Vol. abs/0907.4468.

[Sun_09] Sunitha, N.R.; Amberker, B.B (2009). *Forward-Secure Proxy Signature Scheme for Multiple Proxy Signers using DSA with Proxy Revocation*. IEEE International Advance Computing Conference, 2009. IACC 2009, vol 1, pp.681-686, doi: 10.1109/IADCC.2009.4809094

[Szy_03] Szymaniak M., Pierre, G. et al (2003). *Netairt: a DNS-based redirection system for apache*. Proc. of International Conference WWW/Internet, Algrave, Portugal.

[Szy_04] Szymaniak M., Pierre, G. et al (2004). *Scalable cooperative latency estimation,* Proceedings of the 10th International Conference on Parallel and Distributed Systems (ICPADS),Newport Beach, CA, USA.

[Szy_05] Szymaniak M., Pierre, G. et al (2005). *Latency-driven replica placement*, Proceedings of the International Symposium on Applications and the Internet (SAINT), Trento, Italy.

[Tak_10] Takekawa, A., Imaeda, N.et al. (2010). *The Study of How to Reduce the Content Delivery Time and the System Load in P2P System*. International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), DOI: 10.1109/CISIS.2010.158

[Tam_98] Tam, E.S., Rivers, J.A., et al. (1998). *Evaluating the performance of active cache management schemes*. Proceedings. International Conference on Computer Design: VLSI in Computers and Processors, 1998. ICCD '98, pp. 368-375, DOI: 10.1109/ICCD.1998.727076.

[Tan_02] Tangmunarunkit H., Govindan R. et al (2002). *Network topology generators: degree-based vs. structural*. Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 147-159, ACM New York, ISBN:1-58113-570-X, doi:10.1145/633025.633040

[Tse_05] Tse, S.S.H. (2004). *Approximate algorithms for document placement in distributed Web servers.* 7th International Symposium on Parallel Architectures, Algorithms and Networks, vol., no., pp. 61- 66, doi: 10.1109/ISPAN.2004.1300458

[Tur_04] Turini E. (2004). An architecture for Content Distribution Internetworking. Technical Report UBLCS-2004-2, Department of Computer Science University of Bologna

[Vak_03] Vakali A., Pallis G (2003). *Content delivery networks: status and trends*. IEEE Internet Computing, 7(6), IEEE Computer Society, pp. 68–74.

[Val_98] Valloppillil V., Ross K. W. (1997). *Cache Array Routing Protocol V1.0*, Internet Draft, <draft-vinod-carp-v1-03.txt>

[Ven_07] Venkataraman, M., Sengupta, S. et al (2007). *Towards a Video QoE Definition in Converged Networks*. Second International Conference on Digital Telecommunications, 2007 (ICDT '07), pp: 16 – 16.

[Ver_01] Verma D.C. (2001). *Content Distriution Networks. An engineering approach*. Wiley-Interscience Publication, ISBN 0-471-44341-7

[Ver_02] Verma D.C., Calo, S., et al. (2002). *Policy based management of content distribution networks*, IEEE Network Magazine.

[Viv_98] Vivek S. P., Mohit A, et al. (1998) Locality-aware request distribution in cluster-based network servers. Architectural Support for Programming Languages and Operating Systems, pp. 205–216.

[Vix_00] Vixie, P. andWessels, D. (2000). *Hyper text caching protocol (HTCP/0.0)*. Internet Engineering Task Force, RFC 2756.

[Wan_99] Wang, J. (1999). *A survey of Web caching schemes for the Internet*. SIGCOMM Computer Communication Review, 29(5), ACM Press, NY, USA, pp. 36–46.

[Wan_02] Wang L., Pai, V. S. et al (2002). *The effectiveness of request redirection on CDN robustness*. In Proc. of 5th Symposium on Operating Systems Design and Implementation, Boston, MA, USA, pp. 345–360.

[Wan_04] Wang L., Park K. et al (2004). *Reliability and security in CoDeeN content distribution network*. Proceedings of the USENIX 2004 Annual Technical Conference, Boston, MA.

[Wan_07] Wang Y., Yun X. et al (2007). *Analyzing the Characteristics of Gnutella Overlays*. Fourth International Conference on Information Technology (ITNG '07), vol., no., pp.1095-1100, doi: 10.1109/ITNG.2007.39

[Wax_88] Waxman, B.M. (1988). *Routing of multipoint connections*. IEEE Journal on Selected Areas in Communications, vol.6, no.9, pp.1617-1622, doi: 10.1109/49.12889

[Wes_97] Wessels D., Claffy K. (1997). Internet Cache Protocol (ICP), version 2. RFC 2186.

[Wes_97b] Wessels D., Claffy K. (1997). Application of Internet Cache Protocol (ICP), version 2. RFC 2187.

[Wes_01] Wessels D (2001). *Web caching*. O'Reilly, ISBN 1-56592-536-X

[Whi_11] White D. (2011). Prism- optimised delivery of video on the Internet, White Paper.

[Wie_02] Wierzbicki A. (2002), Models for Internet Cache Location. In Proceedings of the 7th International Workshop on Web Content Caching and Distribution**,** Boulder, Colorado

[Wil_96] Williams S., Abrams M., et al. (1996). Removal Policies in Network Caches for World Wide Web Dcouments, in Proc. Of ACM SIGCOMM, ACM Press.

[Wu_01] Wu, K.L., Yu, P.S., et al. (2001). Segment-Based Proxy Caching of Multimedia Streams, Proc. 10th Int. World Wide Web Conference

[Wu_04] Wu K.L., Yu, P.S. et al (2004) .Segmentation of multimedia streams for proxy caching. *IEEE Transactions on Multimedia*, vol.6, no.5, pp. 770-780, doi: 10.1109/TMM.2004.834870

[Wu_06] Wu B., Kshemkalyani A. D. (2006). *Objective-optimal algorithms for long-term Web Prefetching*. IEEE Transactions on Computers, 55(1), pp. 2–17.

[Xie_09] Xie F., Hua K.A. (2009). *A caching-based video-on-demand service in wireless relay networks*. International Conference on Wireless Communications & Signal Processing (WCSP 2009), pp. 1-5, DOI: 10.1109/WCSP.2009.5371560.

[Xu_13] Xu Y., Zhu C. (2013). *End-to-End Rate-Distortion Optimized Description Generation for H.264 Multiple Description Video Coding*. IEEE Transactions on Circuits and Systems for Video Technology, vol., no.99, doi: 10.1109/TCSVT.2013.2249018

[Yen_10] Yen C., Liaw H.T et al (2010). *Transparent Digital Rights Management System with Superdistribution*. 2010 International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA), vol., no., pp.435-440, doi: 10.1109/BWCCA.2010.110

[Yu_06] Yu, X.Z; Li, B. (2006). *A New Approach to Fragment Level Caching of Dynamically Generated Web Content in Forward Proxies*. International Conference on Machine Learning and Cybernetics, pp. 4495 – 4500, DOI: 10.1109/ICMLC.2006.259165.

[Zha_97] Zhang Z.L. (1997). *End-to-End support for statistical Quality of Service Guarantees in Multimedia Networks*. Ph.D. dissertation, Computer Science Department, University of Massachusetts, Amherst.

[Zhu_11] Zhuang Z.; Guo C. (2011). *Optimizing CDN Infrastructure for Live Streaming with Constrained Server Chaining*. 2011 IEEE 9th International Symposium on Parallel and Distributed Processing with Applications (ISPA), vol., no., pp.183-188, doi: 10.1109/ISPA.2011.44

[Zho_08] Zhou X., Wei P. (2008). Anonymous Proxy Authorization Signature Scheme with Forward Security. International Conference on Computer Science and Software Engineering, 2008, vol.3, pp.872-875, doi: 10.1109/CSSE.2008.528

8. REFERENCIAS WEB

[WWW_AIM] AIMC, Asociación para la Investigación de Medios de Comunicación, *Audiencia de Internet*, Noviembre/Diciembre 2009. (http://www.aimc.es/)

[WWW_Aka] Akamai, http://www.akamai.com/

[WWW_AmAWS] Amazon Web Services, http://aws.amazon.com/es/

[WWW_AmS3] Amazon Simple Storage Service, http://aws.amazon.com/es/s3/

[WWW_AND] A Collection of Modeling and Simulation Resources on the Internet, http://www.idsia.ch/~andrea/sim/simtools.html

[WWW_ARB] Arbor Networks, http://www.arbornetworks.com/

[WWW_ATT] AT&T Global Network Map (Internet Data Centers and Service Nodes) http://www.corp.att.com/globalnetworking/media/network_map.swf

[WWW_Brite] Brite, http://www.cs.bu.edu/brite/

[WWW_Bur] Burbuja punto com , http://es.wikipedia.org/wiki/Burbuja_punto_com

[WWW_CAST] Castalia, a simulator of Wireless Sensor Networks, http://castalia.npc.nicta.com.au/

[WWW_Cod] CodeeN, A Content Distribution Network for PlanetLab, http://codeen.cs.princeton.edu/

[WWW_Cor] The Coral Content Distribution Network, http://www.coralcdn.org/

[WWW_CDN1] CDNSimulator, http://cdnsimulator.sourceforge.net/

[WWW_CDN2] CDNSim, http://oswinds.csd.auth.gr/~cdnsim/

[WWW_Cha] Chandhok, N., *Web Distribution Systems: Caching and Replication*, http://www.cse.wustl.edu/~jain/cis788-99/web_caching/

[WWW_Dar] Darwin Streaming Server, http://dss.macosforge.org/

[WWW_Data] Rich Miller (2012). *Akamai Now Running 105,000 Servers*. http://www.datacenterknowledge.com/archives/2012/03/08/akamai-now-running-105000-servers/

[WWW_Dcore] Dublin Core Metadata Initiative, http://dublincore.org/

[WWW_Dnet] The DummyNet Project, http://info.iet.unipi.it/~luigi/dummynet/

[WWW_Edge] EdgeStream - Global HD Video Delivery Network, www.edgestream.com/

[WWW_Esi] ESI Developer Resources, http://www.akamai.com/html/support/esi.html

[WWW_Exp] Expeshare, Experience sharing in mobile peer communities. http://virtual.vtt.fi/virtual/expeshare/

[WWW_F4all] Proyecto Freight4All. A distributed and open FREIGHT transport ICT solution 4 ALL stakeholders in the Mediterranean area, http://www.med-freight4all.eu/

[WWW_Fasys] Proyecto FASyS. Fábrica Absolutamente Segura y Saludable, http://www.fasys.es/es/index.php

[WWW_F5] F5 Networks, Web Acceleration, www.f5.com/

[WWW_Fas] FastSoft Inc., Web Acceleration, www.fastsoft.com/

[WWW_Flash] Flash media services, http://www.adobe.com/es/products/flashmediaserver/fvss/

[WWW_Gai] Gaikai, http://www.gaikai.com

[WWW_Gar] Estudio Gartner, http://www.slideshare.net/rajeshdgr8/global-saa-s-2012

[WWW_Gea] Geant2, the high-bandwidth, academic Internet serving Europe's research and education community (http://www.geant2.net/)

[WWW_Glas] Glassfish Server, http://glassfish.java.net/

[WWW_Glob] Globule, Scalable Web application hosting, http://www.globule.org/

[WWW_GLO] GlomoSim, Global Mobile Information Systems Simulation Library, http://pcl.cs.ucla.edu/projects/glomosim/

[WWW_Gtitm] GT-ITM, http://www.cc.gatech.edu/projects/gtitm/

[WWW_Hperf] HTTPerf. http://www.hpl.hp.com/research/linux/httperf/

[WWW_Ica] ICAP Forum, http://www.icap-forum.org/

[WWW_Icin] ICIN in FIA, http://www.future-internet.eu/home/future-internet-assembly/budapest-may-2011/session-i1-information-centric-networking-icn.html

[WWW_Idm] IDMaps, Internet Distance Map Service, http://idmaps.eecs.umich.edu/

[WWW_Inet] Inte Topology Generator, http://topology.eecs.umich.edu/inet/

[WWW_Isma] Internet Streaming Media Alliance http://www.isma.tv [nota: actualmente este enlace ya no es válido]

[WWW_ISC] Internet Systems Consortium. ISC Internet Domain Survey. (http://www.isc.org/)

[WWW_ITR] The Internet Traffic Report (http://www.internettrafficreport.com/)

[WWW_ITU] International Telecommunication Union. (http://www.itu.int/home/index.html).

[WWW_IWS] Internet World Stats. Usage and Population Statistics. (http://www.internetworldstats.com/)

[WWW_Jbos] Jboss application server, http://www.jboss.org/

[WWW_JMe] Apache JMeter, http://jmeter.apache.org/

[WWW_Jonas] Jonas application server, http://jonas.ow2.org/xwiki/bin/view/Main/

[WWW_Key] Keynote Systems, http://www.keynote.com/

[WWW_Lime] Limelight Networks: Orchestrate Brilliant Digital Experiences, www.limelight.com/

[WWW_Lot] AppEx Networks, http://www.appexnetworks.com

[WWW_Meg] MegaUpload, http://www.megaupload.com/

[WWW_Mir] Mirror Image, http://www.mirror-image.com/site/

[WWW_MRT] MRTG,The Multi Router Traffic Grapher, (http://oss.oetiker.ch/mrtg/)

[WWW_Mul] Multinet, Enabler for Next Generation Services Delivery, FP6-IST 027437, http://www.ist-multinet.org/

[WWW_Mys] MySQL Community Server, http://dev.mysql.com/downloads/

[WWW_Nem] Iniciativa NEM (http://www.nem-initiative.org/)

[WWW_Neem] NetEm (Network Emulation), http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

[WWW_Net] Netcraft Ltd, March *2010 Web Server Survey*. (http://news.netcraft.com/)

[WWW_Netf] Netflix, https://signup.netflix.com/global

[WWW_Nie] Nielsen//NetRatings. Global Internet Trends. (http://www.nielsen-netratings.com/)

[WWW_NLA] The National Laboratory for Advanced Network Research (NLANR) (http://www.caida.org/projects/nlanr/)

[WWW_Nnet] NIST NET, http://snad.ncsl.nist.gov/itg/nistnet/

[WWW_NS2] UCB/LBNL/VINT, Network Simulator, http://www.isi.edu/nsnam/ns/,

[WWW_NS2b] Installation of video-streaming implementations for the ns-2 simulator http://gridnet.upc.es/~maguilar/ns2_code_victor/ns2tutorial/node1.html

[WWW_NS3] NS-3 Network simulator, http://www.nsnam.org/

[WWW_OAut] OAuth, http://oauth.net/

[WWW_OMN] OMNeT++, http://www.omnetpp.org/

[WWW_Onl] Onlive, http://www.onlive.com/

[WWW_Ope] OpenCDN, http://labtel.ing.uniroma1.it/opencdn/index.html

[WWW_Open] OpenID, http://openid.net/

[WWW_OPN] Opnet Modeler, http://opnet.com

[WWW_Pin] Ping Identity. The Identity Security Company, https://www.pingidentity.com/

[WWW_Pla] PlanetLab, an open platform for developing, deploying and accessing planetary-scale services (http://www.planet-lab.org/)

[WWW_Quick] Quicktime streaming products, http://www.apple.com/es/quicktime/extending/resources.html

[WWW_Rap] Rapidshare, http://www.rapidshare.com/

[WWW_Ray] Dan Rayburn, *Recent Analyst Research On The CDN Market Needs To Be Questioned*, http://blog.streamingmedia.com/the_business_of_online_vi/2009/08/recent-analyst-research-on-the-cdn-market-needs-to-be-questioned.html

[WWW_Real] Real Networks, http://www.realnetworks.com/

[WWW_Red] RedIris, red académica y de investigación española, http://www.rediris.es/

[WWW_Red2] Lista de instituciones adscritas a RedIRIS, http://www.rediris.es/rediris/instituciones/lista.php

[WWW_Sal] Salesforce, http://www.salesforce.com/

[WWW_Saml] Online Community for the SAML standard, http://saml.xml.org/

[WWW_Sig] SIGCOMM 2012 Helsinki, http://conferences.sigcomm.org/sigcomm/2012/icn.php

[WWW_Sim] Simuladores de redes, http://ee.lbl.gov/kfall/netsims.html

[WWW_Simple] SimpleSAMLphp, http://simplesamlphp.org/

[WWW_Sla] Slashdot effect, Wikipedia, http://en.wikipedia.org/wiki/Slashdot_effect

[WWW_Squ] Wessels, D.: "Squid Web Proxy Cache," 2001. www.squid-cache.org/

[WWW_Tel_Cdn] Telefonica Content Delivery Network, http://www.telefonica.com/cdn/es/index.shtml

[WWW_Tis] Definición de QoE en IPTV según la ETSI TISPAN. http://www.etsi.org/deliver/etsi_tr/102400_102499/102479/01.01.01_60/tr_102479v010101p.pdf

[WWW_Tom] Apache Tomcat, http://tomcat.apache.org/

[WWW_Tri] The TRAID project, http://www-dsg.stanford.edu/triad/

[WWW_UPV] Estructura de la red de la UPV, http://www.upv.es/entidades/ASIC/documentacion/386119normalc.html

[WWW_VLC] Videolan, http://www.videolan.org

[WWW_Wapt] WAPT, Web Application Load, Stress and Performance Testing http://www.loadtestingtool.com/

[WWW_Win] Windows Media Services, http://www.microsoft.com/windows/windowsmedia/forpros/server/server.aspx

[WWW_Web] Web caching and content delivery resources, http://www.web-caching.com/

[WWW_Wccp] Web Cache Communication Protocol v2 http://www.cisco.com/en/US/docs/ios/12_0t/12_0t3/feature/guide/wccp.html

[WWW_WoW] World of Warcraft, http://eu.battle.net/wow/es/

[WWW_You] Youtube, http://www.youtube.com/

[WWW_Yan] Yankee Group, http://www.yankeegroup.com/home.do

[WWW_Yan_SCR] Yankee Group Anywhere Scorecard: Content Delivery Networks, 2009, http://www.corp.att.com/awards/docs/scorecard_cdn_2009.pdf

[WWW_Zoo] Zooknic Internet Intelligence. (http://www.zooknic.com/)

9. APÉNDICE

9.1. Descripción de NS-2

Aunque existe una amplia bibliografía, referencias y tutoriales online acerca de NS-2, se dedicará un breve apartado a describir el funcionamiento de alto nivel de esta herramienta, que facilite posteriormente una mejor compresión del modelo de CDN desarrollado.

NS-2 es un simulador de eventos discretos que permite la simulación de entornos diversos, como:

- Encaminamiento unicast y multicast. El *routing* unicast puede configurarse de forma estática (empleo del algoritmo de *Djikstra*) o dinámica (algoritmo *Distance Vector*).
- Topologías simples o complejas, incluyendo la generación de dichas topologías. La topología de red viene caracterizada por los nodos y enlaces (objetos estáticos), así como de los agentes (objetos dinámicos).
- Agentes, definidos como puntos extremos donde los paquetes se crean o se consumen a nivel de red. Un agente es la entidad encargada del tratamiento de un paquete, y puede comportarse como emisor o receptor. Los agentes deben asociarse a los nodos que conforman la topología de red e interactúan con otros agentes.
- Simulación de aplicaciones y generación de tráfico. Las aplicaciones se implementan en NS-2 a través de los denominados agentes de aplicación. NS-2 dispone de varios de ellos, como son FTP, Telnet y Web. Para otro tipo de aplicaciones, existe la clase base *Traffic* a la que se le puede asociar una distribución estadística que describa la aplicación.
- Políticas de gestión de colas, asociado a la forma de tratar los mensajes o paquetes en cada uno de los nodos. NS-2 implementa las siguientes políticas: *Drop Tail, Fair Queuing* (FQ), *Stochastic Fair Queuing* (SFQ), *Deficit Round Robin Schedule* (DRRS), *Random Early Detection* (RED), *Class Based Queuing* (CBQ) y CBQ/WRR.
- Modelado de errores o pérdidas, redes de área local, redes inalámbricas, etc.

La idea original de NS-2 surge a finales de los años 80, a partir de REAL network simulator. A mediados de los años 90, recibe el apoyo de DARPA, a través del proyecto VINT (*Virtual InterNet Testbed*), lo cual le proporcionó un empuje definitivo.

NS-2 es un simulador gratuito que se suministra con el código fuente completo, y puede ejecutarse en múltiples plataformas (Unix, Linux, Windows). Aunque la herramienta se encuentra implementada en C++, se usa un segundo leguaje interpretado llamado OTCL, que es la versión orientada a objetos de TCL. Esto implica una interacción entre ambos lenguajes dentro de la propia herramienta, de tal forma que existe una correspondencia biunívoca entre una clase en C++ (jerarquía compilada de clases en

NS-2) y una clase en OTCL (jerarquía interpretada). Esta arquitectura dual de programación, pese a su complejidad inicial, permite una gran personalización de la simulación de las rutinas a nivel de paquete software (implementadas en C++), y una configuración y un control flexible de la simulación usando un lenguaje sencillo de interpretar, como OTCL.

La distribución de NS-2 para Linux se acompaña de un visualizador de las diferentes topologías creadas en NS-2, así como de la simulación de los flujos de tráfico (paquetes) que recorren dichas topologías. Esta utilidad se conoce como NetWork Animator (NAM), y está diseñada en OTCL. Esta herramienta simplemente requiere un fichero de traza, generado durante la simulación en NS-2, para poder visualizar gráficamente la simulación.

9.1.1. Componentes principales

NS-2 consta de varios componentes, algunos de los cuales son obligatorios (requeridos) y otros optativos (recomendados). Es por ello que la descarga puede realizarse por piezas o por completo (*ns-allinone*). Una vez descargado, descomprimido e instalado este último paquete, los paquetes fundamentales se ilustran en la Figura 152.

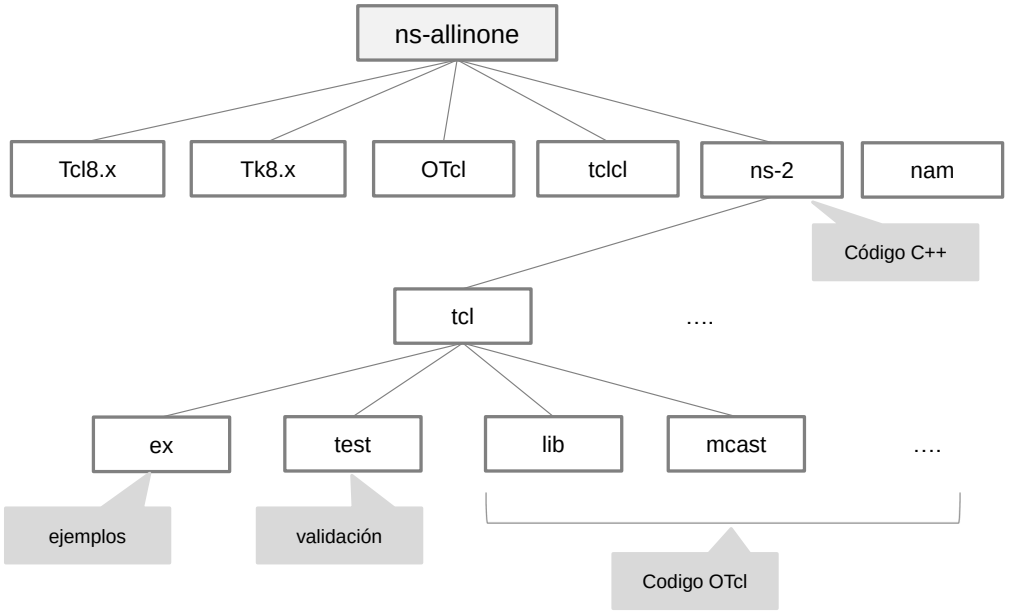


Figura 152. Paquetes principales de NS-2.

9.1.2. Funcionamiento de NS-2

NS-2 es un simulador de redes dirigido por eventos y el núcleo de la herramienta es el simulador *ns*.

NS-2 está diseñado para ser ejecutado en modo *batch*. De esta forma, el usuario define un script, generalmente un fichero con extensión *.tcl*, donde se especifica la información de control y configuración de la simulación. Los objetos de simulación (nodos, enlaces, fuentes de tráfico, etc.) son creados a través de instancias en el script, e inmediatamente se reflejan en la jerarquía compilada de C++. El script de entrada define la topología, construye los agentes (fuentes y destinos), especifica los ficheros de trazas y los tiempos de comienzo de los eventos iniciales en la simulación.

Este script es, más tarde, utilizado por el intérprete para ejecutar la simulación. Para hacer esto se debe invocar desde el *shell* (intérprete de órdenes) la orden

#ns <script>

A medida que avanza la simulación, se generan un conjunto de datos de salida que se almacenan en un fichero de traza. A partir de las trazas de simulación se pueden utilizar lenguajes como el *AWK* para filtrar la traza y obtener los valores de las variables que se desean evaluar.

Finalmente, herramientas tales como *Network Animator* (NAM) permiten realizar un análisis visual del envío y recepción de paquetes de datos y control a medida que avanza la simulación. La Figura 153 muestra un esquema del proceso de simulación.

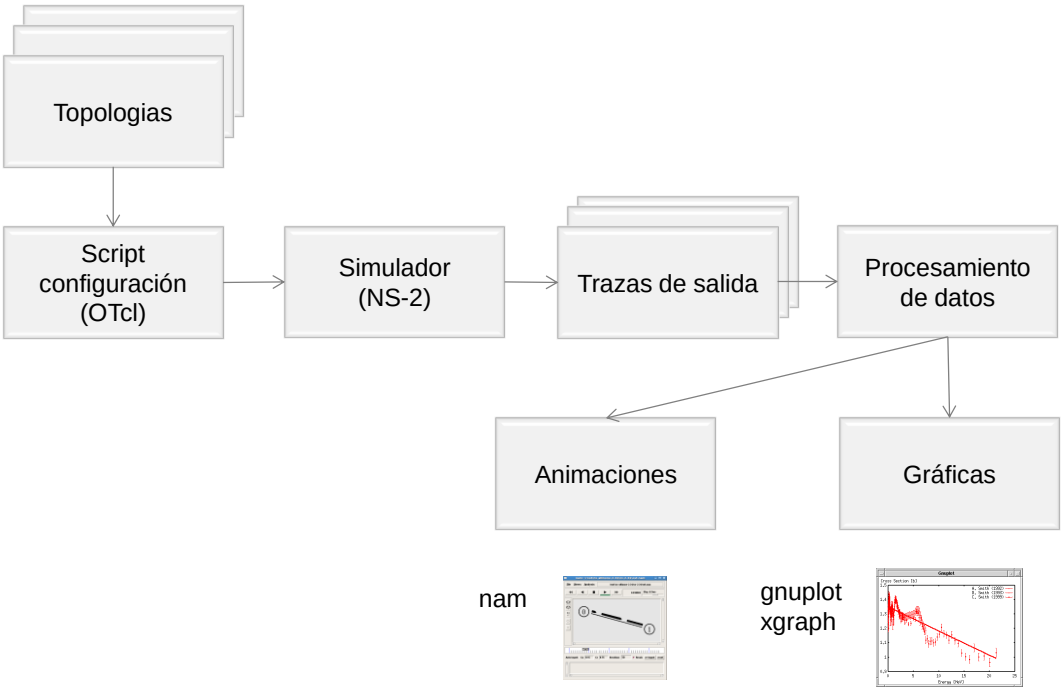


Figura 153. Proceso de simulación en NS-2.