UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE

INGENIERÍA DE TELECOMUNICACIÓN

TESIS DOCTORAL

ESTUDIO Y MEJORA DE LOS MECANISMOS DE CACHÉ EN REDES INALÁMBRICAS AD HOC

Autor: Francisco Javier González Cañete

Ingeniero en Informática

2011

Dr. EDUARDO CASILARI PÉREZ, PROFESOR TITULAR DEL DEPARTAMENTO DE TECNOLOGÍA ELECTRÓNICA DE LA UNIVERSIDAD DE MÁLAGA

INFORMO:

Que D. Francisco Javier González Cañete, Ingeniero en Informática, ha realizado en el Departamento de Tecnología Electrónica de la Universidad de Málaga, bajo mi dirección el trabajo de investigación correspondiente a su Tesis Doctoral titulado:

"ESTUDIO Y MEJORA DE LOS MECANISMOS DE CACHÉ EN REDES MANET"

Revisado el presente trabajo, estimo que puede ser presentado al Tribunal que ha de juzgarlo. Por tanto AUTORIZO y AVALO la presentación de esta Tesis en la Universidad de Málaga.

Málaga, a 28 de noviembre de 2011

Fdo. Eduardo Casilari Pérez

Profesor Titular del Dpto. de Tecnología Electrónica

Departamento de Tecnología Electrónica

E.T.S.I. Telecomunicación

Universidad de Málaga

TESIS DOCTORAL

ESTUDIO Y MEJORA DE LOS MECANISMOS DE CACHÉ EN REDES INALÁMBRICAS AD HOC

Autor: Francisco Javier González Cañete
Ingeniero en Informática

Director: D. Eduardo Casilari Pérez
Dr. Ingeniero de Telecomunicación

Resumen

El objetivo principal de esta tesis es el estudio y mejora de los mecanismos de caché en entornos de redes móviles ad hoc (MANET – Mobile Ad Hoc Networks).

En primer lugar se realiza una revisión detallada de las políticas de reemplazo en cachés Web con el ánimo de conocer los pormenores de su funcionamiento, así como los parámetros de tráfico a tener en cuenta en una caché. De igual forma, se estudian las diferentes políticas de control de admisión en cachés Web.

Una vez terminada esta primera fase, se propone una taxonomía de las políticas de reemplazo en caché Web y se evalúan y comparan tanto las políticas de reemplazo aleatorias como las aplicadas a tipos específicos de documentos. Para llevar a cabo dicho trabajo se requirió la implementación de un simulador de cachés que permitiera la cómoda evaluación tanto de las políticas de reemplazo como de las políticas de control de admisión.

Como resultado del estudio anterior, se constata que las métricas comúnmente utilizadas en la evaluación de cachés (*Hit Ratio* y *Byte Hit Ratio*) no eran adecuadas para la evaluación de cachés que implementaran políticas de control de acceso, por lo que se proponen unas nuevas (*Not Unique Hit Ratio* y *Not Unique Byte Hit Ratio*) más acordes. Por otro lado también se proponen unas métricas para evaluar la política de control de acceso (*Access Control Hit Rate* y el *Access Control Byte Hit Rate*). Para validar estas propuestas, se realiza un estudio para comprobar la adecuación de las métricas propuestas.

Basándose en los estudios comentados anteriormente, se pasa a estudiar los mecanismos de caché aplicados a las redes inalámbricas ad hoc y, más concretamente, a las redes MANET.

Se efectúa un análisis en detalle de los mecanismos de caché propuestos en la literatura, proponiéndose, además, una clasificación de los mismos. Basándose en este estudio se localizan las ventajas e inconvenientes de cada uno de los mecanismos de caché propuestos, lo que permite proponer un mecanismo de caché distribuido para redes inalámbricas ad hoc que mejore el rendimiento de los anteriormente propuestos. Dicho mecanismo de caché, denominado CLIR (*Cross Layer Interception and*

Redirection), utiliza un mecanismo de petición respuesta similar al de HTTP. CLIR se basa en cuatro pilares:

- Implantación de una caché local en cada uno de los nodos de la red inalámbrica, implementando la política de reemplazo LRU.
- Funcionamiento de los nodos como servidores de documentos para los demás nodos, lo que permite que se puedan interceptar las peticiones de documentos en su camino hacia el servidor.
- Utilización de los mensajes de creación de ruta del protocolo de encaminamiento (técnica *cross-layer*) para localizar los documentos en la red al mismo tiempo que se crean las rutas hacia el servidor.
- Implementación de una caché de redirecciones. Dicha caché está compuesta por información de la localización de los documentos en la red. Esta caché se nutre de la información que se extrae de los mensajes que los nodos retransmiten.

Previamente a la evaluación del mecanismo de caché propuesto se realiza un exhaustivo análisis del estado de la técnica de simulación en redes MANET para, de este modo, realizar unas simulaciones acordes a lo propuesto en la literatura. Además, se ha implementado CLIR en el simulador de redes NS2, así como los mecanismos de caché SimpleSearch, MobEye, HybridCache, DPIP y COOP con los que ha sido comparado. Para facilitar la depuración y comprobar el correcto funcionamiento de CLIR se ha implementado también un visualizador de redes MANET que permite observar la evolución de la red y el estado de las cachés en cada nodo.

Se ha estudiado la influencia que tienen la velocidad de los nodos en la red, el número de nodos, el tiempo entre peticiones, el tiempo de vida de los documentos, la popularidad de los mismos y el tamaño de las cachés, en el rendimiento de los esquemas de caché anteriormente mencionados así como de una red que no implementa ningún tipo de caché. Se comprueba que CLIR ofrece mejores prestaciones que los demás esquemas de caché evaluados.

Por último, se realiza una evaluación de CLIR y del resto de mecanismos de caché en redes malladas estáticas, obteniéndose similares resultados a los obtenidos en redes móviles, comprobándose que CLIR resulta adecuado con independencia del grado de movilidad de los nodos.

Abstract

The aim of this thesis is the study and improvement of the caching mechanisms for Mobile Ad hoc NEtworks (MANET).

Firstly, a detailed review of the replacement policies in Web caches is performed. This state of the art allows knowing how the replacement policies work as well as the traffic parameters to take into account when analysing the cache behaviour. Similarly, the admission control policies in Web caches are also revised.

Based on the abovementioned study, a taxonomy of the Web cache replacement policies is proposed. Additionally, the random replacement policies as well as some classic replacement policies applied to a specific type of document are evaluated and compared. In order to perform this work, a specific cache simulator is developed. This simulator enables the evaluation of the replacement policies as well as the admission control policies.

From the results of this evaluation, the metrics usually employed to characterize the cache performance (Hit Ratio and Byte Hit Ratio) were confirmed to be inadequate to assess the performance of a cache utilising an admission control policy. Thus, two new metrics are proposed (Not Unique Hit Ratio y Not Unique Byte Hit Ratio). On the other hand, two new metrics are also proposed in order to evaluate the admission control policy (Access Control Hit Rate y el Access Control Byte Hit Rate). A study is performed in order to validate the suitability of these new metrics.

After these analyses of general caching strategies, we present a review of the existing caching mechanisms for wireless ad hoc networks, especially those devised for MANETs.

A state of the art of the caching mechanisms existing in the literature is accomplished. Additionally, a taxonomy of them is proposed. This revision allows characterizing the advantages and drawbacks of the proposed caching mechanisms. Then, a new distributed caching mechanism for wireless ad hoc networks called CLIR (Cross Layer Interception and Redirection) is proposed. CLIR employs a request/reply mechanism similar to the used in HTTP. CLIR is based on the following pillars:

- The development of a local cache in each network node using the LRU (Least Recently Used) replacement policy.
- The nodes can behave as document server for the rest of the nodes. This allows the interception of request messages in their way to the server.
- The messages for the route creation of routing protocol are employed to locate the documents in the network (This can be considered as a cross-layer technique).
- The development of a redirection cache. This cache stores information about document location in the network. This information is updated using the forwarded messages.

Prior to the evaluation of the proposed caching mechanism, a detailed review of the state of the technique about simulation in MANETs is performed in order to develop simulations according to those proposed in the associated literature. For the comparison of our proposal to other existing mechanisms, CLIR has been implemented using the network simulator NS-2, as well as the SimpleSearch, MobEye, HybridCache, DPIP and COOP caching mechanisms. In order to facilitate the simulation debugging, a MANET visualization tool was also developed. This tool allows observing the network evolution as well as the cache states at every node of the MANET.

In the performance simulations, the influence on the performance of the network of the speed of the nodes in the network, the number of nodes, the time between requests, the time to live of the documents, their popularity and the cache size have been considered. The evaluation compares the performance of the abovementioned caching mechanisms as well as the performance of a network that does not implement any caching method. Results show that CLIR obtains a better performance than the rest of the evaluated caching mechanisms under different circumstances.

Finally, an evaluation of CLIR and the rest of caching mechanisms is performed in static grid networks. The results obtained are similar to those achieved in the MANET environment. As a consequence, we can conclude that CLIR is adequate independently of the nodes' mobility in the network.

AGRADECIMIENTOS

Escribir unos agradecimientos siempre es complicado, ya que se corre el riesgo de omitir a alguien o de mencionar a una persona antes que a otra. De modo que espero no olvidar a nadie y, para solucionar el problema del orden, seguiré, como en el cine, el denominado "orden de aparición".

Quisiera agradecer a mis padres que, con su apoyo, trabajo, esfuerzo y sacrificio consiguieron darme y costearme una educación más allá de sus posibilidades.

A los maestros y profesores de todos mis años de estudio, con especial mención a Antonio Sevilla, Cristóbal Martín y Antonio Morales, que inculcaron en mí el gusanillo de las matemáticas, la literatura y el voleibol respectivamente.

A mis compañeros de voleibol que tan buenos (y no tan buenos) momentos hemos pasado recorriendo la geografía andaluza, jugándonos la vida en las carreteras para llegar a un partido, con especial mención a mi compañero de vóley playa Víctor Ruz.

A mis esposa Isabela, que lleva sufriendo esta tesis tanto tiempo como llevamos casados, por su apoyo y cariño. También quisiera agradecerle su paciencia al haber corregido el inglés de todos mis artículos.

A mi director Eduardo Casilari Pérez por su infinita paciencia, apoyo e ideas ofrecidas durante todos estos años en los que se ha estado rumiando esta tesis.

A Alicia Triviño Cabrera por sus valiosas ideas y críticas en el desarrollo de esta tesis, sobre todo su ayuda a la hora de empezar a trabajar con NS-2.

Finalmente, quisiera agradecer a Duane Wessels, creador del *proxy* Squid, por darme acceso a las muestras de tráfico de los *proxies* del proyecto NLANR.

Quisiera mostrar mi agradecimiento a los siguientes proyectos del Plan Nacional I+D con los cuales ha sido parcialmente subvencionada esta tesis:

- Aplicaciones Integradas en Redes Multiesfera (AIRES) (TIC2003-07953-C02-01)
- Aplicaciones Biomédicas en Redes Inalámbricas Heterogéneas (ABRIL) (TEC2006-12211-C02-01)
- Redes de Arquitectura Mallada para Aplicaciones Sociosanitarias (RAMAS) (TEC2009-13763-C02-01)

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ACRÓNIMOS

ABR	*Associative-Based Routing Protocol*
ACBHR	*Access Control Byte Hit Ratio*
ACHR	*Access Control Hit Ratio*
ACK	*Acknowledgement*
Aggarwal DF	*Aggarwal Dynamic Frequency*
ALRFU	*Adaptive Least Recently Frequency Used*
A-swLFU	*Aged swLFU*
AODV	*Ad Hoc On-Demand Distance Vector*
ASM	*Access Similarity Matrix*
BHR	*Byte Hit Ratio*
CA	*Caching Agent*
CAR	*Cache Access Rate*
CACK	*Cache Acknowledgement Packet*
CC	*Cluster Cooperative*
CCS	*Cluster Cache State*
CGI	*Common Gateway Interface*
CGSR	*Cluster-head Gateway Switch Routing Protocol*
CH	*Cluster Head*
CIP	*COACS Information Packet*
CLIR	*Cross Layer Interception and Redirection*
CMM	*Clustered Mobility Model*
CN	*Caching Nodes*
CPU	*Central Processing Unit*

COACS	*COoperative and Adaptive Caching System*
COCA	*COoperative CAching*
CSN	*Cache State Node*
CSP	*COACS Score Packet*
CSS	*Cubic Selection Scheme*
DF	*Dynamic Frequency*
DGA	*Distributed Greedy Algorithm*
DPIP	*Data Pull/Index Push*
DREP	*Data Reply Packet*
DRP	*Data Request Packet*
DS	*Data Source, Data Server, Document Server*
DSDV	*Destination-Sequenced Distance-Vector Routing Protocol*
EDP	*Entry Deletion Packet*
EXAMS	*EXtensible Animator for Mobile Simulations*
Exp1	*Exponential smoothing*
FBR	*Frequency Based Replacement*
FIFO	*First Input First Output*
GA	*Genetic Algorithm caching model*
GDFS	*Greedy-Dual Frequency Size*
GDSF	*Greedy-Dual Size with Frequency*
g-GDFS	*generalized-GDFS*
GM	*Gauss-Markov mobility model*
GMT	*Greenwich Mean Time*
GPS	*Global Positioning System*
GPSCE	*Global Positioning System Conectivity Estimation*

GROCOCA	*GROup-based COoperative CAching*
HAM	*High Activity Mobile host*
HCRP	*Heuristic-Cache Replacement Policy*
HTTP	*HyperText Transfer Protocol*
HR	*Hit Ratio*
IANA	*Internet Assisted Numbers Authority*
ICP	*Internet Caching Protocol*
IDM-IM	*Intelligent Driver Model with Intersection Management*
IFH	*Ignore First Hit*
iNSpect	*interactive NS2 protocol and environment confirmation tool*
HLRU	*History LRU*
HTTP	*Hypertext Transfer Protocol*
IP	*Internet Protocol*
IV	*Index Vector*
IXP	*IndeX Push*
LAM	*Low Activity Mobile host*
LAT	*Latency Estimation Algorithm*
LDF	*Least Dynamic Frequency*
LFF	*Largest File First*
LFU	*Least Frequently Used*
LFU-Aging	*Least Fequently Used with Aging*
LFU-DA	*Least Fequently Used with Dynamic-Aging*
LLDF	*Localized Least Dynamic Frequency*
LLF	*Lowest Latency First*
LNC-R-W3	*Least Normalized Cost Replacement WWW*

LNC-W3-U	*Least Normalized Cost Replacement WWW Unified*
LRFU	*Least Recently/Frequently Used*
LRU	*Least Recently Used*
LPPB-R	*Least-Popularity-per-Byte Replacement*
LRV	*Lowest Relative Value*
LUV	*Least Utility Value*
LUV	*Least Unified Value*
LUV-Mi	*Least Utility Value with Migration*
LR	*Logistic Regression*
LR	*Latency Reduction*
LRFU	*Least Recently-Frequency Used*
LRU	*Least Recently Used*
LRU-QoS	*LRU Quality of Service*
LRU-SLFR	*LRU Small Latency First Replacement*
LRU-SP	*LRU Size-Adjusted and Popularity-aware*
MAC	*Media Access Control*
MANET	*Mobile Ad Hoc Network*
MCW-n	*Media Characteristic Weighted*
MDPF	*Minimum Distance Packet Forwarding*
MF	*Membership Funcitions*
MH	*Mobile Host*
ML-LRU	*MultiLevel LRU*
MobEye	*Mobile intErcepting proxY cachE*
MQ	*Multi-Queue*
MTTR	*Mean Time-To-Reaccess*

MRU	*Most Recently Used*
MWP	*Markovian Way Point mobility model*
NACK	*Negative ACKnowledgement*
NAM	*Network AniMator*
NDLM	*Neighbor-Dependent Link Model*
NNC	*Nearest Neighbor Classifier*
NNPCR	*Neural Network Proxy Cache Replacement*
NUBHR	*Not Unique Byte Hit Ratio*
NUHR	*Not Unique Hit Ratio*
OLSR	*Optimized Link State Routing Protocol*
OM	*Obstacle Model*
ORION	*Optimized Routing Independent Overlay Network*
OSI	*Open Systems Interconnection*
PDA	*Personal Digital Assistant*
PDF	*Portable Document Format*
PM	*Prefetch Module*
POACH	*POware Aware Caching Heusistic*
PSS	*Pyramidal Selection Scheme*
QCRP	*Query Caching Request Packet*
QD	*Query Devices*
QDA	*Query Device Assigner*
QDAP	*QD Assignment Packet*
RD	*Random Direction*
RPGM	*Reference Point Group Mobility*
RREP	*Route REPly*

RREQ	*Route REQuest*
RRGVF	*Randomized Replacement with General Value Functions*
RRT	*Recent Request Table*
RUT	*Recently Used Time*
RWP	*Random Way Point*
SCGW	*Same Conditional Group Window*
SE	*Simple and Efficient replacement policy*
SLAW	*Self-similar Least Action Walk*
SLRU	*Segmented LRU*
SLRU	*Size-adjusted LRU*
SSR	*Signal Stability Routing Protocol*
SSW-LFU	*Size-adjusted Sliding Window LFU*
SXO	*Size*Order*
SWL	*Sliding Window Length*
swLFU	*server-weighted LFU*
TCL	*Tool Command Language*
TCP	*Transmission Control Protocol*
TDS_D	*Time and Distance Sensitive – Distance*
TDS_N	*Time and Distance Sensitive – Neutral*
TDS_T	*Time and Distance Sensitive – Time*
TORA	*Temporary Ordered Routing Algorithm*
TSP	*Taylor Series Predictor*
TTL	*Time To Live*
TVCM	*Time-Variant Community Model*
URL	*Uniform Resource Locator*

WADM	*Weighted Average Distance Matrix*
WCD	*Worst Cacheable Document*
WebCASE	*Web Caching Algorithm Simulation Environment*
WLAN	*Wireless Local Area Network*
WRP	*Wireless Routing Protocol*
WWW	*World Wide Web*
ZC	*Zone Cooperative Caching*
ZRP	*Zone Routing Protocol*

Capítulo 1

Introducción

El Diccionario Panhispánico de Dudas [DPD, 2006] de la Real Academia Española define el término "caché", en su segunda acepción, como:

"*Se usa en informática, como adjetivo invariable, para referirse a la memoria de rápido acceso, situada entre el procesador y la memoria principal: «Existen dos tipos de memoria caché: primaria y secundaria» (Pimentel Multimedia [Perú 1997]). También se emplea como sustantivo femenino: «Un genuino Pentium Pro, a 200 MHz de velocidad, con [...] caché interna» (Mundo [Esp.] 13.4.97). En este caso, es voz tomada del inglés cache (memory), con acentuación aguda por influjo del galicismo caché. En español se usan también, con este sentido, las expresiones antememoria o memoria intermedia.*"

Aunque esta definición únicamente hace referencia al uso de la memoria caché como aquélla situada entre microprocesador y memoria principal en un ordenador, el concepto de memoria caché ha sido adoptado en otros tipos de tecnologías aunque con similar filosofía. Entre estas tecnologías se pueden mencionar las bases de datos, los sistemas de ficheros en red o el acceso a recursos en Internet a través de la Web. En todas ellas, la idea del uso de cachés reside en almacenar en una memoria intermedia aquellos datos que se estima que van a ser utilizados en un futuro próximo. Al situarse dichas cachés más cerca que el lugar donde están situados los recursos originales a los que se pretende acceder, se consigue que el tiempo de acceso a los mismos se reduzca notablemente.

Esta tesis estudia el uso de cachés tanto en redes cableadas como en redes inalámbricas ad hoc. Este capítulo se organiza como sigue: en la Sección 1.1 se presenta la motivación para la realización del trabajo desarrollado, mientras que los objetivos de esta tesis, así como la metodología empleada, se describe en la Sección 1.2. Finalmente, la Sección 1.3 resume la estructura de esta tesis.

1.1 Motivación

Internet se ha convertido, en relativamente poco tiempo, en un fenómeno de masas. Se ha pasado en menos de veinte años de un uso de Internet prácticamente exclusivo de entidades universitarias, al acceso por parte de gran parte de la población no solo desde el

ordenador de su casa, sino también desde dispositivos portátiles que son capaces de conectarse desde cualquier sitio, siempre que se disponga de cobertura de comunicaciones.

Este enorme crecimiento de Internet y, más concretamente, de la *World Wide Web* (WWW) ha conllevado, inexorablemente, un aumento proporcional en el tráfico telemático que deben soportar tanto las redes de comunicaciones como los servidores que ofrecen los contenidos. Para paliar este crecimiento se hizo necesario el uso de cachés que minimizaran el efecto de la carga del tráfico generado sobre la calidad del servicio prestado. Por ello, se realizaron propuestas para implementar cachés en los navegadores de los usuarios Web y en los propios servidores, así como para crear un nuevo tipo de caché, denominado *proxy*, situado entre los usuarios y los servidores, cuya función consiste en servir de intermediario entre ellos, entregando los documentos que los usuarios reclaman desde la propia caché del *proxy*.

Ya que la función de una caché es almacenar aquellos documentos que se espera que sean solicitados con mayor probabilidad en un futuro cercano, se hace necesaria la presencia de algún tipo de algoritmo que discierna qué documentos deben ser almacenados en la caché y cuáles deben ser eliminados de ella para hacer sitio a un nuevo documento.

Las políticas de control de admisión son las encargadas de decidir, basándose en algún heurístico, si un documento debe ser almacenado en la caché o no, dependiendo de si se estima que lleve o no aparejado beneficio para la el rendimiento del servicio. Por otro lado, las políticas de reemplazo son las encargadas de decidir cuál o cuáles son los documentos que deben ser eliminados de la caché para hacer sitio a un documento nuevo. Esta decisión se puede realizar en función de algunos parámetros de los documentos almacenados como pueden ser su tamaño, tiempo de vida, etc.

Como primer paso en la realización de esta tesis se estimó oportuno realizar un estudio exhaustivo de la literatura asociada tanto a las políticas de reemplazo, como a las políticas de control de admisión en cachés Web. Este estudio previo permitiría aplicar los conocimientos adquiridos a la adaptación de dichos mecanismos en redes inalámbricas ad hoc.

Las redes inalámbricas ad hoc están compuestas por dispositivos que se comunican entre ellos sin ningún tipo de infraestructura. Para conseguirlo, los dispositivos colaboran retransmitiendo y encaminando los paquetes de los demás dispositivos de forma que éstos puedan alcanzar su destino. Dentro de las redes ad hoc se encuentran las redes móviles ad hoc (Mobile Ad Hoc Networks – MANET), que fueron concebidas inicialmente para casos de emergencias o campos de batalla en los que no es posible el despliegue de una red de comunicaciones. Sin embargo, el éxito de las comunicaciones inalámbricas ha ampliado el uso de las redes MANET a aplicaciones comerciales como a conferencias, museos o redes vehiculares. En todos estos escenarios, los usuarios pueden demandar el acceso a redes externas, especialmente a Internet.

En lo que respecta al acceso a Internet, algunas tecnologías Web pueden requerir adaptarse, tal y como se comentó anteriormente, a las redes MANET debido a las restricciones que éstas presentan:

- Capacidad de procesamiento limitada. Algunos dispositivos móviles presentan restricciones en su capacidad de cómputo y en la visualización de información.
- Capacidad de las baterías limitada. Los dispositivos móviles deben operar con baterías, con lo que la cantidad de mensajes que deben generar y procesar debe limitarse en lo posible.
- Escaso ancho de banda. El medio inalámbrico tienen un ancho de banda más restringido que el cableado, por lo que el tráfico de señalización y de datos debe ser reducido.
- Conexión temporal a Internet. La integración de las MANET con redes externas se realiza a través de un dispositivo denominado pasarela o *Gateway*. Sin embargo, la movilidad de los dispositivos en la red puede provocar que el *Gateway* no esté disponible temporalmente. Bajo estas circunstancias, no es posible el acceso a los servidores, por lo que la red debería adaptarse a esta condición de desconexión transitoria.

Dadas todas estas limitaciones, se estima oportuno estudiar la posible aplicación de los mecanismos de caché propios de la Web en las redes ad hoc. Para ello se realiza un estudio de las propuestas de esquemas de caché en redes ad hoc existentes en la literatura, analizando sus ventajas e inconvenientes, para así poder proponer una nueva solución que mejore el funcionamiento global de este tipo de redes.

1.2 Objetivos y contribuciones

El objetivo principal de esta tesis es el estudio y mejora de los mecanismos de caché en entornos de redes MANET.

Las principales contribuciones de esta tesis se resumen a continuación:

- **Estudio y clasificación de las políticas de reemplazo en cachés Web**. Como un primer paso se realiza un estudio pormenorizado de la literatura científica asociada a las políticas de reemplazo en cachés Web. Este estudio permite conocer los pormenores del funcionamiento y los parámetros a tener en cuenta en una caché. Hasta donde nosotros sabemos, el estado de la técnica que se realiza en esta tesis es el más amplio realizado hasta la fecha.

- **Estudio de las políticas de control de admisión en cachés Web**. Se ha realizado, también, un estudio detallado y extenso de las políticas de control de admisión propuestas en la literatura técnica asociada.

- **Comparación de políticas de reemplazo en cachés Web**. Con el objetivo de estudiar la aplicabilidad de las políticas de reemplazo, se ha comparado el rendimiento de varias políticas de reemplazo. Primeramente se han comparado las políticas aleatorias y, más tarde, algunas políticas basadas en valoración y frecuencia aplicadas a cada tipo de documento Web.

- **Propuesta de métricas de rendimiento para cachés con políticas de control de admisión**. Dado que la implementación de una política de control de admisión origina que algunos documentos no sean almacenados en la caché, las métricas de rendimiento clásicas de cachés no son útiles, ya que ofrecen resultados distorsionados. Se proponen, por tanto, unas métricas nuevas y se evalúa su adecuación a la hora de caracterizar el funcionamiento de la caché.

- **Estudio y clasificación de esquemas de caché en redes inalámbricas ad hoc**. Se realiza un estudio pormenorizado de los esquemas de caché propuestos hasta el momento para redes inalámbricas ad hoc. Se realiza además una clasificación de los mismos que, hasta la fecha, no había sido realizada.

- **Propuesta de un nuevo esquema de caché para redes inalámbricas ad hoc**. Basándose en el objetivo anterior, se propone un esquema de caché que aprovecha las ventajas de los esquemas propuestos pero evitando sus inconvenientes. Dicha propuesta se compara con otros esquemas de caché comprobando la bondad del algoritmo propuesto tanto en redes inalámbricas móviles como estáticas.

Estos dos últimos apartados entendemos que constituyen el núcleo de la tesis.

1.3 Estructura de la Tesis

El resto de esta tesis se organiza con la estructura que a continuación se describe:

- El capítulo 2 realiza un estudio del uso de las cachés en Internet. En primer lugar se estudian los principales parámetros del tráfico Web que influyen en el rendimiento en una caché, así como diferentes métricas de rendimiento utilizadas para evaluar la eficiencia de las políticas de reemplazo en cachés Web. A continuación, se realiza un estudio pormenorizado del estado de la técnica de las políticas de reemplazo y políticas de control de admisión. Finalmente, se compara el rendimiento de varias políticas de reemplazo en cachés Web y se proponen nuevas métricas para el rendimiento de cachés con control de admisión.

- El capítulo 3 estudia el funcionamiento de las redes ad hoc comentando los diferentes protocolos de encaminamiento y modelos de movilidad usualmente empleados. A continuación se repasan los diferentes esquemas de caché propuestos en la literatura, realizando una clasificación de los mismos, además de enumerar las métricas de rendimiento comúnmente usadas para evaluarlas. Añadidamente, se realiza un estudio del estado de la técnica de los parámetros de simulación comúnmente empleados para evaluar los esquemas de caché propuestos en la literatura. Finalmente, se propone un nuevo esquema de caché, analizando su rendimiento tanto en redes móviles como en redes estáticas inalámbricas.

- El capítulo 4 resume las principales conclusiones obtenidas del desarrollo de esta tesis así como las líneas futuras de desarrollo del presente trabajo.

- El apéndice A describe la implementación y funcionamiento del software que fue necesario desarrollar para la consecución de esta tesis. Este software incluye: un simulador de políticas de reemplazo, un generador de muestras de tráfico sintético Web, una ampliación del simulador de redes NS-2 y un software para visualizar la evolución de las interacciones que se producen en redes ad hoc con caché.

Capítulo 2

Políticas de reemplazo y control de admisión en cachés Web

El presente capítulo se estructura como sigue. En la sección 2.1 se comenta la estructura de cachés presente en Internet. En la sección 2.2 se estudian los principales parámetros del tráfico Web que influyen en el rendimiento en una caché. La sección 2.3 repasa las diferentes métricas de rendimiento utilizadas para evaluar la eficiencia de las políticas de reemplazo en cachés Web. Además, se efectúa un estudio pormenorizado del estado de la técnica de las políticas de reemplazo, sugiriéndose una taxonomía de las mismas. En la sección 2.4 se describen las políticas de control de admisión propuestas en la literatura. Por último, la sección 2.5 compara el rendimiento de varias políticas de reemplazo en cachés Web, proponiéndose nuevas métricas para el rendimiento de cachés con control de admisión.

2.1 Introducción

La Figura 2.1 ilustra la arquitectura de la World Wide Web. En esta arquitectura nos encontramos, por un lado, a los usuarios que realizan peticiones de documentos hacia los servidores Web, entendiéndose como documento Web a todo recurso (fichero) que tiene asociado una URL (*Uniform Resource Locator*). De esta forma, cuando un usuario solicita desde su navegador una página Web, no solo se descargará el documento de la propia página, sino todos los documentos asociados a la misma, como imágenes o código de *script*.

Para mejorar el tiempo de respuesta, es decir, el tiempo que tardan los documentos en ser obtenidos por el usuario, se han propuesto, prácticamente desde que existe la Web, el uso de cachés a varios niveles. Por un lado, los usuarios pueden implementar una caché en sus propios navegadores. De esta forma, y debido a que, normalmente, los usuarios suelen acceder con frecuencia a los mismos sitios Web, se reduce el tiempo de respuesta al tener los documentos comúnmente accedidos almacenados en caché local. Por otro lado, los servidores Web también pueden implementar una caché en memoria local con aquellos documentos que son más solicitados. De esta forma se ahorra el tiempo de acceso a disco y se consigue un menor tiempo de respuesta. Finalmente, los *proxy* son servidores de documentos que se sitúan entre los usuarios y los servidores Web. Estos servidores reciben

las peticiones de los usuarios y las reenvían hacia los servidores Web, de modo que, cuando los servidores Web devuelven los documentos, éstos son almacenados en la caché del *proxy*. Sucesivas peticiones de los usuarios a estos documentos harán que el *proxy* pueda responder a las peticiones directamente desde su caché sin necesidad de redireccionarlas hacia los servidores Web. Los *proxies*, por tanto, reducen el tiempo de respuesta de las peticiones, ya que se encuentran situados más cerca de los usuarios que los servidores Web y, por otro lado, reducen la carga de los servidores Web ya que las peticiones no tienen que ser servidas por ellos.

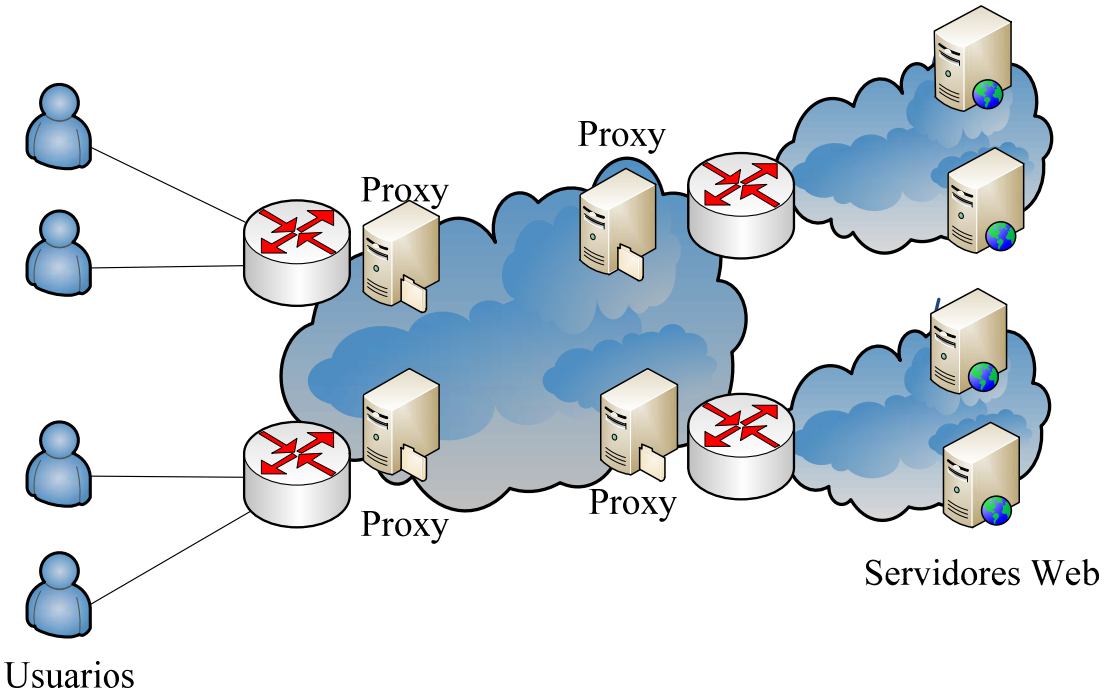


Figura 2.1. Arquitectura Web

Una caché Web, debido a que tiene un espacio finito e independientemente de dónde se implemente, requiere de un mecanismo que decida qué documentos deben ser eliminados cuando la caché se encuentra llena y un nuevo documento debe ser almacenado. La política de reemplazo es la responsable de esta decisión con la finalidad de mantener en caché aquellos documentos con mayor probabilidad de ser referenciados en el futuro próximo. Sin embargo, este objetivo no es sencillo de cumplir, ya que existen numerosos heurísticos a la hora de tomar dicha decisión que dependen de las características del tráfico en cada momento. Por otro lado, la caché Web también puede implementar un control de admisión para rechazar aquellos documentos que no deben almacenarse en la caché ya que se considera que su almacenamiento no supone ningún beneficio para el rendimiento de la misma. Por la misma razón anteriormente comentada, dicha decisión es difícil de tomar al no conocerse con certeza el comportamiento de las futuras peticiones a los documentos.

2.2 Parámetros del tráfico a considerar en una caché Web

En este apartado se describen los parámetros del tráfico Web más influyentes en el comportamiento de una caché Web.

2.2.1 Popularidad

Una característica común en el tráfico Web es la irregular distribución de las referencias a los distintos documentos [Arlitt,1999] [Roadknight,1999]. En muchos casos se aplica la ley de Zipf [Zipf,2011] para modelar la popularidad de los documentos. La ley de Zipf expresa una relación hiperbólica entre la popularidad *P(i)* (número de veces que se repite la petición del documento *i*) y su *ranking r*. Este parámetro *r* se define en función de la ordenación de los documentos según su popularidad. Esta relación se expresa en la siguiente ecuación:

$$P(i) = \frac{\beta}{r^{\alpha}} \tag{2.1}$$

donde α caracteriza la pendiente de la representación logaritmo/logaritmo del número de referencias al documento *i* en función de su ranking de popularidad *r*, mientras que β es una constante que determina el desplazamiento de la función. Cuanto mayor sea el valor de α, mayor será la probabilidad de que los documentos más frecuentes sean solicitados.

Algunos investigadores han encontrado que el valor de α es cercano a la unidad [Almeida,1996] [Crovella,1997], precisamente siguiendo la ley de Zipf. Otros autores [Breslau,1999] [Heegaard,2000] [Busari,2002] [Adamic,2002] sugieren que el valor de α es menor que la unidad, y que la distribución puede ser descrita como la de Zipf con un valor de α que varía dependiendo del tráfico. Este comportamiento típico puede describirse como una línea recta de pendiente negativa ($-\alpha$) si se representa en ejes logarítmicos *P(i)* frente a *r*. Este ajuste lineal suele ser casi perfecto para la parte principal del cuerpo de la distribución. Sin embargo, suele desajustarse tanto para los elementos más populares como para los menos populares debido a los *one-timers* o elementos que sólo se solicitan una vez [Mahanti,1999].

2.2.2 Tamaño de los documentos

Estudios previos de tráfico Web [Park,1996] [Crovella,1996] [Mahanti,1999] [Mahanti,1999b] [Hernández,2002] [Hernández,2002b] han demostrado que la distribución de los tamaños de los documentos de tráfico Web sigue una función (al menos frente a modelos exponenciales) de "cola pesada" (*heavy-tail*). Este tipo de distribuciones implica que relativamente pocos documentos de tamaño muy grande acumulan un porcentaje no despreciable del volumen total de datos del tráfico Web. Por otro lado, la distribución del tamaño de los documentos puede afectar significativamente en el diseño de estrategias de

caché. Almacenando en caché sólo documentos pequeños se puede reducir el número de peticiones enviadas a los servidores Web, aunque esta política también se traduce en que los documentos grandes serán los que tengan que ser servidos por los servidores. En el otro extremo, almacenar en caché documentos de gran tamaño provoca que los documentos grandes sean entregados por la caché, mientras que los pequeños, que son mayoría, tengan que ser servidos por los servidores Web.

Estudios de tráfico Web en *proxies* muestran que muchos de los documentos transferidos en la Web tienen un tamaño pequeño [Abdulla,1997] [Mahanti,1999] [Breslau,1999]. Una cuestión natural que se plantea al respecto es si existe alguna correlación estadística entre la frecuencia de acceso a un determinado documento y su tamaño. Algunos estudios [Crovella,1997] [Mahanti,2000] han mostrado que hay una muy pequeña correlación entre la frecuencia de acceso y el tamaño, siendo más accedidos los documentos con menor tamaño.

2.2.3 Localidad temporal

La localidad temporal hace referencia a la tendencia de documentos referenciados en el pasado a volver a ser nuevamente referenciados en el futuro. El modelo de la localidad temporal está basada en el modelo de pila finita LRU (*Least Recently Used*). Una pila LRU es una lista de todos los documentos ordenados según la proximidad del momento en el que hayan sido referenciados por última vez [Almeida,1996] [Busari,2002], es decir, el último documento que haya sido referenciado estará situado en primer lugar de la pila y el documento que fue referenciado hace más tiempo estará en la última posición. La pila se actualiza dinámicamente cada vez que se procesa una petición de documento. En muchos casos, esta actualización implica el tener que añadir un nuevo elemento en la cima de la pila empujando el resto hacia el fondo, en otros casos (cuando se resuelve a solicitar un documento ya existente en la pila), implica extraer un elemento existente en el interior de la pila y trasladarla a la cima de la misma, desplazando al resto de los elementos hacia el fondo. El aspecto más importante en una pila LRU es que cada posición en la pila tiene asociada una probabilidad de referencia. Las probabilidades son asociadas a la posición de la pila y no a los documentos. Las probabilidades de las posiciones de la pila pueden ser proporcionadas por modelos u obtenidas al analizar muestras de tráfico reales.

2.2.4 *One-timers*

Diferentes estudios de tráfico Web en servidores y *proxies* han mostrado que una parte importante de las peticiones hacia un servidor Web o un *proxy* son realizadas una sola vez, independientemente de la duración del acceso [Mahanti,1999] [Arlitt,1997]. A este tipo de documentos se les denomina *one-timers*. Es evidente, que no tiene ningún beneficio almacenar en caché este tipo de documentos puesto que nunca más van a ser requeridos. De hecho, sería deseable la existencia de algoritmos ubicados en las cachés para discriminar a estos documentos de modo que no inunden la caché reduciendo así su efectividad.

2.2.5 Tipos de documento

Los documentos existentes en la Web se pueden clasificar en función de su tipo en: aplicaciones, audio, imágenes, mensajes, texto y video. Cada tipo de documento presenta unas características particulares [Buchholz,2002] que los pueden hacer más o menos adecuados para ser almacenados en caché o para aplicar un tipo de política de reemplazo concreto.

2.3 Políticas de reemplazo en cachés Web

En este apartado se describen las métricas de rendimiento utilizadas para medir la bondad de las políticas de reemplazo. A continuación se hace un estudio pormenorizado de las políticas de reemplazo más importantes propuestas en la literatura. Finalmente, se realiza una clasificación de las mismas en función de los parámetros que tienen en cuenta para su funcionamiento.

2.3.1 Métricas de rendimiento

Para evaluar y comparar el rendimiento de las políticas de reemplazo en cachés Web se han propuesto en la literatura asociada una serie de métricas de rendimiento que pasan a resumirse a continuación.

- HR (*Hit Ratio*) – Se define como la cantidad de documentos que han sido servidos desde la caché (aciertos en caché) en relación con el número de peticiones que ha recibido la misma. La ecuación 2.2 define este parámetro, donde h_i es la cantidad aciertos en caché que ha producido el documento i, f_i es la cantidad de peticiones a dicho documento y R es el conjunto de todas las peticiones.

$$HR = \frac{\sum_{i \in R} h_i}{\sum_{i \in R} f_i} \tag{2.2}$$

- BHR (*Byte Hit Ratio*) – Se define como la cantidad de información que ha sido servida desde la caché en relación con la cantidad de información que ha sido procesada por la misma. La ecuación 2.3 caracteriza el BHR, donde s_i indica el tamaño en bytes del documento i.

$$BHR = \frac{\sum_{i \in R} s_i \cdot h_i}{\sum_{i \in R} s_i \cdot f_i} \tag{2.3}$$

- MR (*Miss Ratio*) – Se define de forma similar al HR pero midiendo la cantidad de documentos que no han podido ser servidos desde la caché al producirse un fallo de caché en función del número total de peticiones.
- DSR (*Delay Saving Ratio*) – Se define como la fracción del retardo ahorrado por satisfacer la petición desde la caché en lugar del servidor Web (ecuación 2.4).

$$DSR = \frac{\sum_{i \in R} (d_i \cdot h_i - c_i \cdot v_i)}{\sum_{i \in R} d_i \cdot f_i} \tag{2.4}$$

Donde d_i representa el retardo en obtener el documento i desde el correspondiente servidor Web, c_i es el tiempo necesario para realizar una validación del documento i y comprobar que la versión que se encuentra almacenada en la caché está actualizada y v_i es el número de validaciones realizadas sobre el documento i.

Para el caso en el que no se considere el tiempo de validación la ecuación 2.4 queda como sigue:

$$DSR = \frac{\sum_{i \in R} d_i \cdot h_i}{\sum_{i \in R} d_i \cdot f_i} \tag{2.5}$$

- Tiempo medio de descarga (*Average Download Time - ADT*) – Se define como se muestra en la ecuación 2.6, donde $||R||$ es el tamaño de la caché.

$$ADT = \frac{\sum_{i \in R} d_i \cdot (1 - h_i / f_i)}{\|R\|} \tag{2.6}$$

- Reducción de latencia (*Latency Reduction - LR*) – Es la reducción del tiempo de espera de los usuarios desde que realizan una petición hasta que el documento llega al terminal (latencia de descarga) en función de la suma de todas las latencias de descarga.
- Uso de CPU– Mide el porcentaje de uso de CPU (*Central Processing Unit* – Unidad de Proceso Central) del servidor Web o del *proxy*, computando el ahorro que implica el uso de la caché. Para servidores muy cargados es deseable que el uso de CPU se reduzca para que el procesador pueda gestionar más peticiones simultáneamente.

2.3.2 Estado de la técnica de políticas de reemplazo

En el presente apartado se va a realizar una descripción pormenorizada de las políticas de reemplazo en cachés Web que han sido propuestas en la literatura. Por regla general, estas políticas de reemplazo han sido concebidas para servidores *proxy*, aunque algunas de ellas pueden ser válidas también para clientes Web. En la descripción de cada

uno de los algoritmos se seguirá la nomenclatura matemática de cada autor, de ahí su heterogeneidad.

2.3.2.1 *First Input First Output* (FIFO)

FIFO es la política de reemplazo más simple, ya que no tiene en cuenta ningún parámetro de los documentos a insertar o eliminar. Cuando un documento debe ser insertado en una caché FIFO, éste se coloca en la última posición de la misma, siendo eliminado, si fuera necesario, el documento o documentos que se encuentren en la primera posición de la caché.

2.3.2.2 *Least Recently Used* (LRU)

La política de reemplazo LRU elimina de la caché aquellos documentos que hace más tiempo que fueron referenciados. Para ello, mantiene una lista de la que se eliminan, cuando se produce un reemplazo, los documentos de la primera posición de la lista. En el caso de que se referencie un documento que ya se encuentre almacenado en la caché, este documento pasa a situarse al final de la caché.

2.3.2.3 *Climb*

El funcionamiento de *Climb* es similar al de LRU con la salvedad de que, cuando se referencia un documento que se encuentra en la caché, éste se mueve una posición hacia el final de la lista, en lugar de colocarse en la última posición como se hace en LRU.

2.3.2.4 *Class-based LRU* (C-LRU)

La política de reemplazo C-LRU [Havercort,2003] propone dividir el espacio disponible en I listas o clases que se gestionan mediante LRU. En cada una de las listas se almacenan los documentos cuyos tamaños se encuentran entre el rango de valores r_i y r_{i+1}. Estos rangos se calculan mediante la ecuación:

$$r_i = \frac{\ln(c_i\lambda_i) - \ln(c_{i+1}\lambda_{i+1})}{\lambda_i - \lambda_{i+1}} \quad \text{para } i = 1,\ldots,I-1 \tag{2.7}$$

donde c_i indica la frecuencia de ocurrencias de los documentos de la lista i y $1/\lambda_i$ es el tamaño medio de los documentos de la lista i. I suele ser relativamente pequeño, en el rango de cuatro a ocho.

El tamaño en bytes reservados para cada una de las I listas se puede calcular de dos formas dependiendo del parámetro de rendimiento que se quiera optimizar:

1. Para optimizar el HR se toma el tamaño de la partición p_i de la lista i como proporcional a la probabilidad de que una petición sea a un documento de la lista i.

$$p_i = c_i \tag{2.8}$$

2. Para optimizar el BHR, se tiene en cuenta la cantidad esperada de bytes en la lista i en relación con la cantidad media de bytes.

$$p_i = \frac{c_i / \lambda_i}{\sum_{j=1}^{I} c_j / \lambda_j} \tag{2.9}$$

Los parámetros r_i y p_i pueden ser calculados de tres formas diferentes:

1. Una única vez en función de muestras de tráfico anteriores y suponiendo que las características del tráfico se van a mantener invariables.
2. Periódicamente se recalculan los parámetros y se reasignan los documentos almacenados.
3. Bajo demanda, observando los parámetros de rendimiento y, al observarse un descenso en el mismo, recalculando los valores.

2.3.2.5 *PARTition* (PART)

La política de reemplazo PART [Murta,1998] propone dividir el espacio de caché en tres listas gestionadas mediante LRU en las que se almacenan los documentos dependiendo de su tamaño. De esta forma, los documentos con un tamaño entre cero y 2 kB se almacenan en una lista, los que tienen más de 2 kB y menos de 6 kB en otra y, finalmente, los que tienen un tamaño mayor que 6 kB en otra. No se especifica el tamaño que tiene asignado cada una de estas listas.

2.3.2.6 *Segmented LRU* (SLRU)

SLRU [Karedla,1994] divide el espacio de caché en dos zonas: protegida y no protegida. Los documentos se insertan y eliminan en la zona no protegida usando una política LRU. Cuando hay una referencia a un documento que se encuentra en la zona no protegida, el documento es desplazado a la zona protegida, que también se gestiona mediante LRU. Si no hay espacio en la zona protegida, el documento más antiguo (usando LRU) de la zona protegida se traslada a la zona no protegida. Si no hay espacio en la zona no protegida, el documento más antiguo (usando LRU) se elimina de la caché.

2.3.2.7 *Size-adjusted LRU* (SLRU)

La política de reemplazo *Size-adjusted LRU* [Aggarwal,1999] ordena la lista de documentos en función del producto $S_i x \Delta T_i$, siendo S_i el tamaño del documento i, y ΔT_i el

número de accesos desde la última vez que fue referenciado el documento *i*. Se eliminan de la caché los documentos con mayor valor de este producto hasta que haya sitio para el nuevo documento.

2.3.2.8 Log(Size)+LRU

Esta política de reemplazo [Williams,1996] elimina primero los documentos cuyo tamaño es mayor que el logaritmo del tamaño del documento a introducir, y emplea LRU para desempatar entre documentos que cumplen tal condición.

2.3.2.9 *Pyramidal Selection Scheme* (PSS)

PSS [Aggarwal,1999] es una versión más implementable de SLRU (*Size-adjusted LRU*). En concreto, propone una separación de los tamaños de los objetos en $N=\log(M+1)$ clases, siendo M el tamaño de la caché. En cada cola i (con i entre 1 y N) se almacenan los documentos con tamaños entre (2^{i-1}) y (2^i-1) bytes. Cada cola se gestiona mediante LRU. Cuando hay que reemplazar, se comparan los valores del producto $S_i x \Delta T_i$ como ocurría en SLRU, aunque únicamente en el caso de los más antiguos de cada clase.

Para tener en cuenta los costes, el algoritmo propone que la función de decisión sea $S_i x \Delta T_i / c_i$, siendo c_i el coste de obtener el documento i. Además, propone tener en cuenta el factor de expiración, por lo que se define la función:

$$\frac{S_i \cdot \Delta T_i}{c_i \cdot (1 - \min(1, \delta_{ti1} / \delta_{ti2}))} \tag{2.10}$$

siendo δ_{ti1} la diferencia entre t (momento actual) y el instante de tiempo en que fue accedido por última vez el documento i y δ_{ti2} la diferencia entre el tiempo de expiración del documento i y t. En la fórmula anterior, el valor mínimo entre 1 y $\delta_{ti1} / \delta_{ti2}$ se encuentra necesariamente entre cero y uno. Se eliminan de la caché los documentos con mayor valor asociado.

2.3.2.10 LRU-k

LRU-k [O'Neil,1993] es una extensión de LRU que mantiene el instante de tiempo de las últimas k referencias a cada documento. LRU-k elimina el documento con la *k-ésima* petición menos reciente. Si hay documentos en la caché que han sido referenciados menos de k veces, se elimina el menos recientemente usado.

En [Boyar,2010] se demuestra teóricamente que LRU-2 es más eficiente que LRU.

2.3.2.11 LRU-MIN

La política de reemplazo LRU-MIN [Williams,1996] selecciona para eliminar los documentos con un tamaño mayor o igual que el que va a ser insertado. Si existen, se reemplaza usando LRU. Si no existen, se buscan aquéllos que son mayores que la mitad del tamaño del documento. Si existen, se reemplazan usando LRU. Si no existen, se sigue comprobando con un cuarto del tamaño del documento, iterándose el algoritmo con un octavo, decimosexto, etc. del tamaño del documento hasta que se eliminan los documentos necesarios para hacer sitio suficiente al nuevo documento.

2.3.2.12 *MultiLevel LRU* (ML-LRU)

La política de reemplazo ML-LRU [Feldman,2002] divide el espacio de caché en M colas LRU interconectadas. Existen i niveles, con i=(0, …,$M-1$). Cada cola aplica LRU de forma local. Las colas están interconectadas de forma que cuando un documento es eliminado del nivel i, es insertado al final de la cola de nivel $i-1$. Solo los documentos eliminados de la cola del nivel 0 son eliminados de la caché.

Cuando un documento nuevo llega a la caché, se clasifica en uno de los M niveles de acuerdo con alguna política de clasificación y se coloca en la cabeza de la cola correspondiente. Con cada acierto, el documento es transferido al final de la cola de su nivel original independientemente de su situación actual.

La política de reemplazo propone varias políticas de clasificación en clases:

- Basándose en el origen en función del nombre o dirección IP del servidor.
- Basándose en el tipo del documento.
- En función del tamaño del documento.
- En función de la popularidad del documento.
- Basándose en aspectos económicos, como lo que haya pagado el propietario del documento por el almacenamiento del mismo.

2.3.2.13 LRU-LSC

LRU-LSC [Hosseini-Khayat,1997] emplea una lista LRU para determinar la actividad de los documentos. Durante el reemplazo, los documentos con menor actividad se colocan en una segunda lista siempre que el tamaño total de esta nueva lista sea menor que ΘxB, siendo B es el tamaño total de la caché y Θ ($0<\Theta<1$) un parámetro umbral, que puede ser fijo o dinámico, que determina la fracción de la lista que se mueve a la nueva lista. La nueva lista se ordena de acuerdo a un valor $spc_i=c_i/s_i$ asignado a cada documento, siendo c_i el coste de obtener el documento i y s_i su tamaño. En caso de igualdad entre las métricas de los documentos se usa la actividad que presenta el documento como criterio de selección. Finalmente, los documentos se eliminan de esta nueva lista hasta que su tamaño acumulado sustraído del tamaño de todos los objetos en caché es menor que un valor específico.

2.3.2.14 *History LRU* (HLRU)

HLRU [Vakali,2000] define r_1, r_2,…, r_n como las peticiones a documentos en la caché en los instantes t_1, t_2,…, t_n. Se define la función de historia como:

$$hist(x,h) = \begin{cases} t_i & \text{si hay } h-1 \text{ referencias entre } t_i \text{ y } t_n \\ 0 & \text{en otro caso} \end{cases} \quad (2.11)$$

El parámetro *hist(x,h)* define, por tanto, el instante de la *h-ésima* referencia en el pasado del documento *x*. Se eliminan los documentos con mayor valor de *hist(x,h)*. Si hay muchos documentos con un valor nulo del parámetro *hist* (*hist*=0), se considera la política de reemplazo LRU.

2.3.2.15 LRU*

LRU* [Chang,1999] mantiene una lista ordenada basada en LRU. Cuando hay un acierto en un documento se incrementa su contador de aciertos y se mueve al final de la lista. Cuando no hay espacio para un nuevo documento, se comprueba el número de aciertos del documento a eliminar por LRU. Si es cero se elimina, en otro caso se decrementa su contador de aciertos y se pasa al final de la lista ordenada. Este proceso se repite hasta que se eliminen los documentos necesarios para almacenar el nuevo.

Para que los documentos no acumulen muchas referencias se limita a un valor máximo, proponiéndose un valor por defecto de cinco.

2.3.2.16 *LRU Size-Adjusted and Popularity-aware* (LRU-SP)

LRU-SP [Cheng,2000] clasifica los documentos en grupos de acuerdo a $\lfloor \log_2(S_i / nref_i) \rfloor$, siendo S_i el tamaño del documento i y $nref_i$ su número de referencias. Cada grupo es una lista LRU. Cuando se produce un acierto, el documento puede pasar a otra lista de acuerdo con su nuevo valor de $\lfloor \log_2(S_i / nref_i) \rfloor$. Se eliminan los documentos que tienen un mayor valor de la función de coste $(\Delta T_{it} \cdot S_i / nref_i)$, siendo ΔT_{it} el tiempo transcurrido desde el último acceso hasta el momento actual. Se aconseja no usar más de 20 grupos.

2.3.2.17 LRU-Hot

La política de reemplazo LRU-Hot [Menaud,2000] mantiene dos listas: una de documentos "calientes" (*hot*) y otra de documentos "fríos" (*cold*). Esta clasificación la hace el servidor Web al servir los documentos, especificando si es de los más solicitados o no, respectivamente. Los documentos se ordenan en cada lista de acuerdo a su re-accesibilidad, que depende del momento en el que el documento fue accedido por última vez, de forma similar a LRU. La re-accesibilidad se va reduciendo periódicamente, siendo

este decremento mayor para los objetos fríos. Se usan dos contadores de referencia: un contador base y un contador para documentos calientes. Cuando se recibe una petición de un cliente, el contador base se incrementa en uno, mientras que el contador de objetos calientes se incrementa en uno cada dos peticiones.

Cuando un documento llega a la caché, se inserta en la cola que corresponda (*hot* o *cold*) y se le asigna como contador de acceso el contador base, colocándose al final de la lista. Cuando se accede de nuevo a un documento que está en la caché, se actualiza su contador de accesos con el valor actual del contador base y se coloca al final de la lista.

Cuando hay que extraer documentos, se calculan los valores de re-accesibilidad de los dos documentos que están en la cola de las dos listas y se calculan los valores *hot_value* y *cold_value* como:

$$\begin{aligned} hot\_value &= tail\_hot - hot\_reference\_counter \\ cold\_value &= tail\_cold - cold\_reference\_counter \end{aligned} \tag{2.12}$$

donde *tail_hot* y *tail_cold* son los valores de re-accesibilidad de los documentos que están en la cola de las listas *hot* y *cold*, y *hot_reference*_counter y *cold_reference*_counter son los contadores de referencia de dichos documentos, respectivamente.

Se elimina el documento que tiene el menor valor de re-accesibilidad. Se van eliminando documentos de esta forma hasta que haya sitio para insertar el nuevo documento.

2.3.2.18 *LRU Quality of Service* (LRU-QoS)

LRU-QoS [Menaud,2000] es similar a LRU-Hot, pero divide la caché en dos listas: lista de objetos degradables y no degradables. Se consideran documentos degradables aquéllos a los que se les puede bajar la calidad por recodificación, como imágenes o vídeos. Cuando se selecciona un documento degradable para eliminar, se le baja la calidad y se pasa a una lista de objetos degradados. Si se selecciona un documento degradable para eliminar, existe un segundo paso para seleccionar si realmente se elimina ese documento o el que fue accedido hace más tiempo de la lista de documento degradados. Este segundo criterio consiste en no permitir que haya más documentos degradados que degradables.

2.3.2.19 LRU-Hot+QoS

Propuesto en [Menaud,2000], combina las políticas de reemplazo LRU-Hot y LRU-QoS. Para ello, se dividen los documentos en degradables, no degradables y los degradados en dos sublistas correspondientes a objetos calientes y fríos.

2.3.2.20 *LRU Small Latency First Replacement* (LRU-SLFR)

LRU-SLFR [Shin,2003] mantiene una cola LRU y tiene en cuenta el número de accesos a cada documento así como su latencia. Sólo se eliminan los documentos que pertenecen al SCGW (*Same Conditional Group Window*) que son aquellos documentos del principio de la lista con igual número de accesos. Se elimina aquél que tenga menor latencia. El SCGW se propone que tenga un tamaño de once documentos.

2.3.2.21 *Multi-Queue* (MQ)

La política de reemplazo MQ [Zhou,2001] emplea *M* listas LRU, en cada lista *i* se almacenan los documentos que se han referenciado al menos 2^i veces, pero menos de $2^{i+1}-1$ veces recientemente. El algoritmo también mantiene una zona de memoria de historial. Cuando hay un acierto de un documento, se incrementa la frecuencia del mismo y se coloca el documento en la posición MRU (*Most Recently Used*) de la lista apropiada y a su *expireTime* se le asigna el valor *currentTime*+*lifeTime*, donde *currentTime* es el instante de tiempo actual y *lifeTime* es un parámetro que indica el tiempo de vida del documento. En cada acceso, se comprueba el valor de *expireTime* de los documentos de cada una de las colas y, si es menor que el valor de *currentTime*, el documento pasa a la zona MRU de la siguiente cola.

2.3.2.22 2Q

2Q [Johnson,1994] usa dos colas, la primera de ellas, denominada *A1*, se gestiona mediante FIFO y la segunda, denominada *Am*, se gestiona mediante LRU. En la primera referencia a un documento, éste se almacena en la cola *A1*. Si un documento es referenciado de nuevo cuando está en la cola *A1*, pasa a la cola *Am*. Si un documento no se vuelve a referenciar cuando está en la cola *A1*, se elimina. El tamaño de ambas colas es fijo.

2.3.2.23 Algoritmo de Pitkow/Recker

La política de reemplazo Pitkow/Recker [Pitkow,1994] emplea una cola LRU con la excepción de que si todos los documentos almacenados han sido accedidos en el día actual, se elimina el más grande en lugar del que fue accedido hace más tiempo.

2.3.2.24 *Least Dynamic Frequency* (LDF)

De acuerdo con esta política de reemplazo [Aggarwal,1997] cada documento *i*, en cada instante *t*, tiene una frecuencia dinámica (*DF – Dynamic Frequency*) asociada dada por:

$$DF(i,t) = \frac{1}{t - t_i} \tag{2.13}$$

donde t_i indica el instante de tiempo del último acceso. La frecuencia dinámica de un conjunto de documento adyacentes $S=\{1,...,K\}$ se define como:

$$\sum_{i \in S} DF(i,t) = \sum_{i=1}^{K} \frac{1}{t - t_i} \tag{2.14}$$

Se reemplaza el conjunto de documentos adyacentes con menor frecuencia dinámica que liberan el espacio suficiente para que quepa el documento que llega a la caché.

2.3.2.25 *Localized Least Dynamic Frequency* (LLDF)

LLDF [Aggarwal,1997] es una versión más implementable de LDF que considera sólo los grupos de documentos adyacentes que contienen uno de los *r* LRU documentos de la caché. Para *r* se toma un valor de 1 o 2 (*LLDF-1* o *LLDF-2*).

2.3.2.26 *Value Aging*

La política de reemplazo *Value Aging* [Zhang,1999b] asigna una valoración a cada documento existente en la caché, cada vez que es accedido, dada por:

$$V_{new} = V_{old} + C_t \sqrt{\frac{C_t + L_t}{2}} \tag{2.15}$$

donde V_{old} es la valoración que tenía el documento antes de ser reaccedido, C_t es el instante de tiempo actual y L_t es el instante de tiempo del último acceso al documento. En caso de reemplazo, se eliminan de la caché los documentos con menor valoración.

2.3.2.27 *Differential Aging*

Esta política de reemplazo [Zhang,1999b] emplea el nivel de llenado de la caché como evento para eliminar documentos de la misma. Cuando el espacio libre en la caché es inferior a *1/l* su capacidad y han transcurrido T_u segundos desde la última actualización, los documentos de la caché que no hayan sido accedidos en los últimos T_l segundos se eliminan de la caché. El proceso de eliminación de documentos se detiene cuando el espacio libre en la caché alcanza el valor *1/u* de su capacidad total. Los valores de *l*, T_u, T_l y *u* son parámetros de la política de reemplazo.

2.3.2.28 *Least Frequently Used* (LFU)

La política de reemplazo LFU mantiene un contador para cada documento almacenado en la caché que indica el número de veces que ha sido referenciado. Por tanto, cuando se solicita un documento almacenado en caché, este contador se incrementa en una unidad. La caché se encuentra ordenada en función de este contador, de forma que en las primeras posiciones de la caché se encuentran aquellos documentos con un menor contador de referencias, por lo que serán eliminados primero. Por otro lado, en las últimas posiciones de la caché se situarán los documentos con un mayor número de referencias.

2.3.2.29 LFU*

LFU* [Arlitt,1996] funciona como LFU, con la salvedad de que, al producirse un reemplazo, elimina de la caché únicamente los documentos que tienen un número de referencias de valor uno. En el caso de que no haya suficientes documentos con un número de referencias igual a uno para eliminar, el nuevo documento no se inserta en la caché.

2.3.2.30 *LFU-Aging*

LFU-Aging [Arlitt,1996] funciona como LFU con la diferencia de que, ocasionalmente, se reduce el número de referencias de los documentos para que aquellos que han conseguido muchas referencias durante cierto tiempo, y ya no vuelven a ser referenciados, puedan ser eliminados alguna vez de la caché y no se mantengan en ella indefinidamente, prevaleciendo sobre documentos de más reciente aparición.

2.3.2.31 *LFU* Aging*

LFU* Aging [Arlitt,1996b] es una mezcla entre las políticas de reemplazo LFU* y LFU-Aging. Al igual que LFU*, esta política reemplaza sólo los documentos con una sola referencia y, además, decrementa el valor de las referencias cada cierto tiempo para evitar la situación de que no pueda entrar ningún documento en la caché al tener todos los documentos existentes una referencia superior a uno, tal y como hace LFU-Aging.

2.3.2.32 *LFU Dynamic-Aging* (LFU-DA)

La política de reemplazo LFU-DA [Arlitt,1999] funciona como LFU, pero lleva un contador que almacena el valor de la frecuencia (o número de referencias previas) del documento almacenado en caché que menor frecuencia tiene. De esta forma, cuando se almacena un documento nuevo que no se encontraba en la caché, se le asigna este valor de frecuencia en lugar de asignarle una frecuencia unidad. Cuando se elimina un documento de la caché, al contador se le asigna el valor de la frecuencia del documento eliminado. De este modo se palia el castigo a documentos “jóvenes”.

2.3.2.33 *Size-adjusted Sliding Window LFU* (SSW-LFU)

SSW-LFU [Hou,2001] propone una modificación para LFU manteniendo una ventana de *n* referencias a documentos, siendo *n* un parámetro del algoritmo. Cuando un documento se elimina de la caché se introduce una referencia a dicho documento en la ventana incluyendo su número de referencias. Si el documento vuelve a ser referenciado una vez que está en la ventana, se inserta en la caché con el número de referencias que había acumulado cuando entró en la ventana más la referencia actual. Se eliminan de la caché los documentos con mayor valor de la función de valoración:

$$ssw_i = \frac{s_i}{n} \tag{2.16}$$

donde s_i es el tamaño del documento *i*.

2.3.2.34 *α-Aging*

α-Aging [Zhang,1999b] emplea una función lineal para reducir la valoración de los documentos almacenados en la caché. Dicha función es:

$$f(v) = \alpha \cdot v \quad \text{con } 0 \le \alpha \le 1 \tag{2.17}$$

De acuerdo con esta función, en cada intervalo de tiempo predefinido, todo documento almacenado en la caché decrementa su valoración. El algoritmo propone usar el número de accesos como valoración.

2.3.2.35 *Frequency Based Replacement* (FBR)

FBR [Robinson,1990] divide la caché en tres secciones: *nueva*, *media* y *vieja*. Cuando llega un nuevo documento a la caché, se añade a la sección *nueva*, que se gestiona mediante LRU. Cuando el documento se elimine de la sección *nueva* se traslada a la sección *media*, que también se gestiona mediante LRU. Si el documento sigue sin ser referenciado, pasará a la sección *vieja*. Solo los documentos de la sección vieja son eliminados definitivamente de la caché. Para ello se usa una política LFU y, en caso de igualdad entre documentos, se emplea LRU. Cuando se referencia un documento que está en la caché, se pasa al final de la sección *nueva* si ya estaba en la *nueva* y no se incrementa su contador de referencias. Si estaba en la sección *media* o en la *vieja*, se incrementa su contador de referencias.

2.3.2.36 *Least Recently/Frequently Used* (LRFU)

LRFU [Lee,2001] asigna un valor $CRF_{last}(x)=1$ a cada documento *x* cuando se almacena en la caché. Este valor es actualizado en cada acceso al documento *x* según la siguiente ecuación:

$$CRF_{last}(x) = 1 + F(t_c - LAST(x)) \cdot CRF_{last}(x) \tag{2.18}$$

con

$$F(y) = \left(\frac{1}{2}\right)^{\lambda y} \tag{2.19}$$

siendo t_c el instante de tiempo actual, *LAST(x)* el instante del último acceso a x y λ un parámetro configurable. Conforme λ se aproxima a cero, CRF_{last} *(x)* tiende a ser el número de ocurrencias de x y LRFU se convierte en LFU. Conforme λ se aproxima a uno, LRFU se convierte en LRU. Se eliminan de la caché los documentos con menor valor de CRF_{last} *(x)*.

Se propone también una versión de LRFU en el que el parámetro λ se calcula de forma adaptativa, denominada ALRFU (*Adaptive LRFU*).

2.3.2.37 *Least Unified Value* (LUV)

LUV [Bahn,2002] se basa en la política de reemplazo LRFU añadiendo pesos. Se reemplaza el documento i con menor valor de valoración *V(i)*, siendo:

$$V(i) = p(i) \cdot w(i) \tag{2.20}$$

donde

$$w(i) = \frac{c_i}{s_i} \tag{2.21}$$

mientras que

$$p(i) = \sum_{j=1}^{n} F(\delta_j) \text{ con } \delta_j = t_c - t_j \tag{2.22}$$

siendo

$$F(x) = \left(\frac{1}{2}\right)^{\lambda x} \text{ con } 0 \le \lambda \le 1 \tag{2.23}$$

donde, para el documento i, c_i es el coste de obtener el documento, s_i su tamaño, n el número de referencias, t_c el tiempo actual y t_j el tiempo en que se referenció en el momento j (el algoritmo calcula cuánto tiempo hace que se referenció). λ es un parámetro que le da más importancia a la frecuencia (λ=0) o a la "recencia" (cercanía de su referencia en el tiempo) (λ=1).

2.3.2.38 *server-weighted LFU* (swLFU)

swLFU [Kelly,1999] mantiene el número de referencias de cada documento *i* almacenado en la caché (N_i). Cuando se inserta un documento nuevo en la caché desde un servidor Web, éste viene acompañado por un peso W_i asignado por el servidor que indica cuánto valora el servidor Web un acierto sobre dicho documento. Se eliminan de la caché los documentos con menor valor de $N_i x W_i$.

A-swLFU (*Aged swLFU*) [Kelly,1999b] es una extensión de swLFU que elimina, cada *k* reemplazos, el documento LRU en lugar del asignado por swLFU. Si *k* vale cero se obtiene swLFU; con *k* igual a uno se obtiene LRU y con un valor mayor que uno se obtiene un algoritmo híbrido que combina, como criterios, la frecuencia y la cercanía en el tiempo de las referencias.

2.3.2.39 *One-Timers*

La política de reemplazo *One-Timers* [Belloum,1998] divide la caché en dos colas denominadas *A* y *B*. La cola *B* se puede gestionar mediante una política de reemplazo como LRU, LFU o SIZE. Los documentos que se eliminan de la cola *B* pasan a la cola *A*, que se gestiona mediante LRU. Si un documento que se encuentra en la cola *A* vuelve a ser referenciado, pasa a la cola *B* y se inserta según el algoritmo de ordenación que haya implementado en *B*. Si los documentos no vuelven a ser referenciados cuando se encuentran en la cola *A*, se eliminan de la caché. El número de documentos que pueden almacenarse en la cola *A* es una proporción fija del número total de documentos en la caché.

2.3.2.40 *Generational Replacement*

En la política de reemplazo *Generational Replacement* [Osawa,1997] Los documentos se almacenan en *n* ($n>1$) listas. Cada lista $i<n$ contiene los documentos que han sido solicitados *i* veces. La lista *n* contiene todos los documentos que han sido solicitados *n* o más veces. Una solicitud a un documento causa el paso de una lista al final de la siguiente. Los documentos de la lista *n* se pasan al final de la misma lista al ser accedidos. Los documentos se sacan del principio de la lista *1* para hacer sitio a los nuevos.

2.3.2.41 *Exponential smoothing* (Exp1)

Exp1 [Reddy,1998] tiene en cuenta el instante en que se predice que va a ser referenciado de nuevo cada documento. Para ello, se le da un peso $W(t_i)$ a cada documento *i* cuyo valor es:

$$W(t_i) = \frac{1}{\mu_i} \tag{2.24}$$

donde

$$\mu_i = \alpha \cdot t_i + (1-\alpha) \cdot \mu_{i-1} \tag{2.25}$$

siendo t_i el instante en que se referencia el documento i. α determina la importancia relativa de la frecuencia y la recencia para predecir comportamientos futuros. Se propone fijar α en el intervalo [0.1, 0.3]. Finalmente, se eliminan de la caché los documentos con menor valor de W.

2.3.2.42 *Greedy-Dual Size* (GD-Size)

GD-Size [Cao,1997] emplea una función de coste para valorar los documentos, siendo la función de valoración:

$$V(i) = \frac{C(i)}{S(i)} \tag{2.26}$$

donde *C(i)* es el coste asignado al documento *i* y *S(i)* su tamaño en bytes. Se proponen varias funciones de coste:

- Constante (*C(i)*=1) – Se considera que todos los documentos tienen el mismo coste.
- Latencia – Se considera el tiempo que tarda el documento en ser servido desde el servidor.
- Saltos – El coste es el número nodos intermedios (saltos) que el documento ha tenido que atravesar para llegar desde el servidor al *proxy*.
- Número de paquetes – El coste es el número de paquetes que han sido necesarios para transmitir el documento.

Se eliminan de la caché los documentos con una menor valoración *V(i)*.

2.3.2.43 *Greedy-Dual Size with Frequency* o *Greedy-Dual Frequency Size* (GDSF o GSFS)

GDSF [Cherkasova,1998] es una variante de GD-Size que tiene en cuenta también la frecuencia de acceso a los documentos, calculando la función de valoración como:

$$V(i) = F(i)\frac{C(i)}{S(i)} \tag{2.27}$$

donde *F(i)* define en número de accesos al documento *i*.

2.3.2.44 *N-gram*

N-gram [Yang, 2001b] amplía la política de reemplazo GDSF prediciendo la probabilidad de que un documento sea accedido en el futuro. *N-gram* emplea la siguiente función de valoración *K(i)* para cada documento *i*:

$$K(i) = L + (W(i) \cdot F(i)) \cdot \frac{C(i)}{S(i)} \tag{2.28}$$

siendo

$$W(i) = \sum_{j} P_{i,j} \tag{2.29}$$

donde *L* es el parámetro de envejecimiento, cuyo valor es el de la valoración *K* más pequeña en la caché, $P_{i,j}$ es la probabilidad predicha de que exista una sesión TCP S_j entre el *proxy y el* servidor Web para obtener el documento *i*, *F(i)* es la frecuencia, *C(i)* es el coste y *S(i)* es el tamaño del documento *i*. El valor de $P_{i,j}$ se calcula mediante un algoritmo de "minería" de datos. Se seleccionan para eliminar primero los documentos con menor valoración.

2.3.2.45 *generalized-GDFS* (g-GDFS)

g-GDFS [Cherkasova,2001] es una variante de GDSF que permite dar más o menos peso al tamaño y la frecuencia. Para cada documento *i* asigna la función de valoración *V(i)*:

$$V(i) = Clock + F(i)^{\alpha} \cdot \left(\frac{C(i)}{S(i)} \right)^{\beta} \tag{2.30}$$

donde *Clock* es una variable de envejecimiento (*aging*). α y β son valores racionales para dar mayor énfasis (si son mayores que 1) o rebajar el énfasis (si son menores que 1) de cada parámetro.

2.3.2.46 GreedyDual Size Popularity (GDSP)

GDSP [Jin,1999] funciona como GD-Size modificando la función de valoración para tener en cuenta la popularidad:

$$V(i) = F(i) \frac{C(i)}{S(i)} \tag{2.31}$$

$$F_{n+1}(i) = F_n(i) \cdot 2^{-t/T} + 1 \tag{2.32}$$

donde *t* es el tiempo transcurrido desde la última referencia al documento *i*, *T* es una constante que vale dos días y *n* es la enésima referencia al documento. Para la primera referencia (*n=1*) de un documento *i* se considera $F_1(i)=1/3$.

2.3.2.47 *GreedyDual**

*GreedyDual** [Jin,2000] es una modificación del GD-Size para tener en cuenta la popularidad a largo plazo y la correlación temporal a corto plazo. Para ello, la función de valoración de los documentos se calcula como:

$$V(i) = L + \left(F(i)\frac{C(i)}{S(i)} \right)^{1/\beta} \tag{2.33}$$

siendo $L=min\{V(q)|q \text{ en la caché}\}$ la valoración más pequeña de los documentos de la caché y β es un parámetro de la relación exponencial $T=t^{-\beta}$ que caracteriza la correlación de las referencias. El valor de β suele fijarse a 0.5.

2.3.2.48 *Lowest Relative Value* (LRV)

LRV [Lorenzetti,2000] asigna, para cada documento en la caché una valoración dada por la ecuación:

$$V(i) = P(i)\frac{C(i)}{S(i)} \tag{2.34}$$

siendo *S(i)* el tamaño del documento *i*, *C(i)* el coste de obtener el documento del servidor original y *P(i)* la probabilidad de que el fichero sea accedido en el futuro. Esta probabilidad se define como:

$$P(i) = 1 - D(t) \tag{2.35}$$

$$D(t) = 0.35\log(t+1) + 0.45(1 - e^{\frac{t}{2*10^6}}) \tag{2.36}$$

donde *t* es el instante de tiempo actual. Se eliminan de la caché los documentos con menor valor de *V*.

2.3.2.49 Landlord

Landlord [Young,1998] propone asignar un crédito a cada documento *i* (*credit(i)*) de la caché correspondiente a su coste *c(i)*. Cuando hay que eliminar documentos para hacer sitio para uno nuevo, se decrementa el crédito de todos los documentos de la caché siguiendo la ecuación:

$$credit(i) = credit(i) - \Delta \cdot size(i) \tag{2.37}$$

donde

$$\Delta = \min_{j \in cache} \frac{credit(j)}{size(j)} \tag{2.38}$$

mientras que *size(i)* indica el tamaño.

Una vez decrementado el crédito de todos los documentos, se eliminan aquellos que tienen un crédito igual a cero. Este proceso se repite hasta que haya suficiente espacio para el nuevo documento.

En el caso de un acierto en caché, al crédito del documento acertado se le asigna un valor aleatorio entre su crédito actual y su coste *c(i)*.

2.3.2.50 *Latency Estimation Algorithm* (LAT)

La política de reemplazo LAT [Wooster,1997] elimina de la caché aquellos documentos con el menor valor de la función de valoración:

$$d_i = clat_{ser(i)} + \frac{s_i}{cbw_{ser(i)}} \tag{2.39}$$

siendo

$$clat_j = (1-\alpha) \cdot clat_j + \alpha \cdot s_{clat} \tag{2.40}$$

$$cbw_j = (1-\alpha) \cdot cbw_j + \alpha \cdot s_{cbw} \tag{2.41}$$

donde *s(i)* es el tamaño del documento *i*, s_{clat} es la latencia del tiempo de conexión, s_{cbw} es el ancho de banda y α es una constante de suavizado que vale 1/8.

2.3.2.51 MIX

La política de reemplazo MIX [Niclausse,1998] elimina los documentos con el valor más pequeño de la función de coste:

$$V = \frac{lat^{r1} \cdot nref^{r2}}{tref^{r3} \cdot size^{r4}} \tag{2.42}$$

donde *lat* es la latencia del documento, *nref* el número de veces que ha sido referenciado, *tref* el tiempo que hace que no se referencia el documento y *size* es su tamaño. Por su parte, *r1* tiene un valor constante e igual a 0.1, así como las constantes *r2*, *r3*, y *r4*, que tienen un valor de uno. Estos valores han sido obtenidos experimentalmente.

2.3.2.52 *Media Characteristic Weighted* (MCW-n)

MCW-n [Yu,2001] propone dividir la caché en zonas de tamaño variable y en cada una de ellas almacenar un tipo de documento diferente (audio, video, texto, imágenes). Cada zona tiene un tamaño de bloque fijo. Así, se eliminan los documentos que tienen menor peso:

$$W = Priority \times (b \times Tendency + (1 - b) \times Frequency) \tag{2.43}$$

donde β es una constante de configuración, *Priority* representa la prioridad que el servidor le da a cada documento, *Tendency* muestra la probabilidad de que haya un acierto en las próximas peticiones (se puede modelar usando una distribución Gaussiana) y *Frequency* se calcula como:

$$Frequency = \frac{1}{MTTR} \tag{2.44}$$

donde

$$MTTR = \sum_{i \geq 0} (t_i - t_{i-1}) \cdot w(i) \tag{2.45}$$

$$w(i) = \alpha \cdot w(i-1), \ \alpha \leq 1 \text{ y } w(0) = 1 - \alpha \tag{2.46}$$

siendo MTTR (*Mean Time-To-Reaccess*) el tiempo medio estimado para que el documento vuelva a ser referenciado. Éste se calcula como la suma ponderada (mediante el parámetro de ponderación *w(i)*) de las anteriores referencias al documento. α es una constante de ponderación entre la frecuencia y la recencia.

2.3.2.53 Algoritmo de Bolot/Hoschka

La política de reemplazo Bolot/Hoschka [Bolot,1996] elimina los documentos con menor valor de la función de coste *W*, definiéndose ésta como:

$$W = \frac{w1 \cdot rtt_i + w2 \cdot s_i}{ttl_i} + \frac{w3 + w4 \cdot s_i}{t_i} \tag{2.47}$$

donde t_i es el tiempo transcurrido desde que el documento *i* fue referenciado la última vez, s_i es su tamaño, rtt_i es el tiempo que tarda en descargarse el documento y ttl_i es el tiempo de vida del documento. *w1*, *w2*, *w3* y *w4* son constantes de ponderación. Como el algoritmo asume que ttl_i resulta difícil de calcular, la función se simplifica y queda:

$$W = w1 \cdot rtt_i + w2 \cdot s_i + \frac{w3 + w4 \cdot s_i}{t_i} \tag{2.48}$$

2.3.2.54 *Virtual Cache*

Virtual Cache [Arlitt,1999b] propone dividir la caché en varias colas, cada una de ellas con una política de reemplazo diferente. Los documentos entran en la cola *0* y, cuando es eliminado de dicha cola, se inserta en la cola *1*. Cuando un documento es eliminado de la última cola es cuando se elimina definitivamente de la caché.

Se propone implementarlo usando dos colas. La cola *0* se gestiona con la política GDSF(packets) y la cola *1* mediante LFU-DA.

2.3.2.55 *Logistic Regression* (LR)

LR [Foong,1999] emplea regresión logística para aprender sobre el comportamiento de los documentos Web y predecir la distancia al siguiente acceso para cualquier documento de la caché. Dicha predicción se hace en función del tamaño del documento, de su tipo, del número de accesos anteriores dentro de una ventana y del tiempo desde el último acceso.

$$P_{LR} = P(\text{evento Y} \mid \text{predictores}=\{X_1,X_2,...,X_n\} = \frac{1}{1+e^{-z}} \tag{2.49}$$

siendo

$$z = \sum_{i=0}^{k} \beta_i X_i \quad -\infty<z<\infty \tag{2.50}$$

donde β_i representa los coeficientes de los n predictores X_i del evento Y, siendo:

$$Y = \begin{cases} 1 & \text{el documento es accedido en los próximos } N \text{ accesos} \\ 0 & \text{en otro caso} \end{cases} \tag{2.51}$$

El algoritmo funciona en una primera fase de aprendizaje donde se calculan los coeficientes β_i. Para el resto de la simulación se almacenan en caché los documentos que se estima que van a ser reaccedidos.

2.3.2.56 *M-Metric*

En *M-Metric* [Wessels,1995] cada documento i en la caché tiene asignado una valoración dada por:

$$R_i = F_i^f \cdot S_i^s \cdot \Delta T_i^t \tag{2.52}$$

donde F_i es la frecuencia de acceso al documento i, S_i es su tamaño y ΔT_i es el tiempo transcurrido desde la última petición al documento. Los valores de f, s y t son configurables para dar más importancia a la frecuencia, tamaño o recencia, respectivamente. Se eliminan primero de la caché los documentos con menor valor de R.

2.3.2.57 Taylor Series *Predictor* (TSP)

TSP [Yang,2001] emplea una serie de Taylor de segundo orden para ampliar la política de reemplazo GDSF. La función de valoración de cada documento es:

$$K_i = \frac{F_i \cdot C_i}{S_i \cdot \Delta(T)} \tag{2.53}$$

donde

$$\Delta(T) = T_p - T_c \tag{2.54}$$

siendo F_i el número de referencias al documento i, C_i su el coste, S_i el tamaño, T_p el momento en que se predice que va a ocurrir el próximo acceso y T_c el instante de tiempo actual. La predicción del momento del próximo acceso se realiza suponiendo que el comportamiento de las referencias a los documentos siguen una serie de Taylor. El valor estimado siguiendo dicha suposición resulta ser:

$$T_p \approx (5T_c - 4T_{c-1} + T_{c-2}) / 2 \tag{2.55}$$

donde T_{c-1} y T_{c-2} son los instantes de tiempo de las dos últimas referencias al documento.

Se eliminan de la caché los documentos con menor valoración K_i.

2.3.2.58 *Simple and Efficient* (SE)

La política de reemplazo "modestamente" denominada SE [Radhika,2003] propone asignar a cada documento una valoración dada por:

$$bValue(i) = W(i) \cdot H(i) \tag{2.56}$$

donde

$$W(i) = \frac{c_i}{s_i} \tag{2.57}$$

$$H(i) = RecRef(i) \cdot FreqRef(i) \tag{2.58}$$

siendo c_i el coste de obtener el documento i (se considera como coste la latencia), s_i el tamaño, *RecRef(i)* indica lo reciente que fue la última referencia al documento i y *FreqRef(i)* es la frecuencia de acceso a i. Se eliminan los documentos con menor valor de *bValue*.

2.3.2.59 *Least-Popularity-per-Byte Replacement* (LPPB-R)

LPPB-R [Kim,2001] calcula la función de utilización de cada documento j como:

$$U(j) = \frac{P(j)}{S(j)} \tag{2.59}$$

siendo *S(j)* el tamaño del documento *j* y *P(j)* su popularidad, que se puede definir de dos formas:

1. $P(j) = \frac{R(j)}{T}$, donde *R(j)* es el número de referencias del documento *j* y *T* es el número total de peticiones que ha recibido el *proxy* en el momento en el que se inserta el nuevo documento.
2. $P(j) = \frac{1}{\beta^{R(j)}}$ $0 < \beta < 1$, donde *R(j)* es el número de referencias al documento *j* y β es una constante para el factor de impacto. Se usa un valor de β entre 0.3 y 0.5.

Para gestionar los documentos en la caché se usan *N* listas. Cada lista *i* almacena los documentos que tienen un tamaño entre 2^{i-1} y 2^{i}-1. A su vez, cada lista se gestiona mediante la política de reemplazo LFU. Se eliminan de la caché aquellos documentos con un menor valor de *U(j)* de entre los menos frecuentemente usados de cada lista.

Para evitar la polución de caché se mantiene una lista LRU que se va comprobando periódicamente de forma que, si la diferencia entre el instante de tiempo del último acceso del documento LRU y el tiempo actual es mayor que un valor umbral, el algoritmo asigna el contador de referencias a dos. La segunda vez que se dé este caso para el mismo documento, se asigna el contador de referencias a uno. El periodo de comprobación de la lista LRU es de 10.000 peticiones y el valor umbral es de 1.000.000 de peticiones.

2.3.2.60 *Heuristic-Cache Replacement Policy* (HCRP)

HCRP [Tabassum, 2010] propone emplear cuatro factores a tener en cuenta a la hora de eliminar documentos de la caché. En primer lugar se tiene en cuenta la recencia, después la frecuencia de acceso, el tamaño y, si siguiera habiendo igualdad en la aplicación del criterio, se emplearía la siguiente función heurística:

$$h - value = \frac{rand}{d} \tag{2.60}$$

donde *rand* es un número aleatorio entre cero y uno (generado en cada aplicación del algoritmo) y *d* es el retardo en obtener el documento.

2.3.2.61 *Largest File First* (LFF)

La política de reemplazo LFF elimina de la caché los documentos con mayor tamaño.

2.3.2.62 *Hyper-G*

Hyper-G [Williams,1996] clasifica los documentos de la caché usando la política de reemplazo LFU. En caso de empate a la hora de eliminar un documento, se emplea LRU y si volviera a haber empate, se comprueba el tamaño del documento para eliminar el más grande.

2.3.2.63 *Lowest Latency First* (LLF)

La política de reemplazo LLF [Wooster,1997] elimina primero de la caché los documentos con una menor latencia.

2.3.2.64 Hybrid

Hybrid [Wooster,1997] elimina de la caché aquellos documentos que tienen el menor valor de la función:

$$V(i) = \left(clat_{ser(i)} + \frac{Wb}{cbw_{ser(i)}} \right) \cdot \frac{nref(i)^{Wn}}{s(i)} \tag{2.61}$$

donde $clat_{ser(i)}$ es el tiempo de conexión con el servidor para obtener el documento i, $cbw_{ser(i)}$ es al ancho de banda con el servidor, $nref(i)$ el número de veces que ha sido referenciado el documento desde que está en la caché y $s(i)$ es el tamaño del documento. Wb y Wn son constantes.

2.3.2.65 *Least Normalized Cost Replacement WWW* (LNC-R-W3)

LNC-R-W3 [Scheuermann,1997] es una adaptación para la Web de la política de reemplazo LNC-R (*Least Normalized Cost Replacement*) [Scheuermann,1996], que fue originalmente diseñada para sistemas de almacenamiento de datos.

Se selecciona para eliminar de la caché el documento que tenga un menor valor de la función:

$$profit(D_i) = \frac{K \cdot d_i}{(t - t_k) \cdot s_i^{b+1}} \tag{2.62}$$

donde K es una constante que vale 2 o 3 y es ajustado experimentalmente, d_i es el retardo medio de descarga del documento i, t es el instante de tiempo actual, t_k es el instante de tiempo de la *k-ésima* referencia a i, s_i el tamaño del documento y b es un valor entre 1 y 2 obtenido experimentalmente.

En [Shim,1999] se propone una ampliación para LNC-R-W3 que tiene en cuenta la consistencia de los documentos denominada LNC-W3-U (*Least Normalized Cost Replacement WWW Unified*).

2.3.2.66 *Nearest Neighbor Classifier* (NNC)

En NNC [Belloum,1998b] cada documento se representa por una 4-*tupla* (o conjunto de 4 parámetros) dada por <*size*, *ETime*, *RTime*, *Nref*>, donde *size* es el tamaño del documento, *ETime* es el instante de tiempo en que entró en la caché, *RTime* es el instante de la última referencia y *NRef* es el número de referencias al documento desde que se encuentra en la caché.

Esta 4-*tupla* se representa en un espacio de cuatro dimensiones, en el que los documentos a eliminar son los más cercanos (usando la distancia euclídea) al WCD (*Worst Cacheable Document*), que es el que tiene la 4-*tupla* <*size*, 1, 1, 1>, siendo *size* un parámetro de la política de reemplazo.

2.3.2.67 *Cubic Selection Scheme* (CSS)

La política de reemplazo CSS [Tatarinov,1998] se basa en una estructura en forma de cubo cuyos elementos son grupos (listas) de referencias a los documentos. Cada grupo contiene documentos con el mismo valor de $\lfloor \log_2 Size \rfloor$ y $\lfloor \log_2 NRef \rfloor$, siendo *Size* el tamaño de los documentos y *NRef* el número de referencias. Se fija un límite (*MAX_NREF*) para restringir el número de referencias que puede acumular un documento.

Cuando hay un acierto en caché de un documento, se incrementa su contador de referencias y se sitúa al final de la lista correspondiente o al final de otra lista si cambiara de grupo. En el caso de un fallo en caché se selecciona un conjunto de víctimas para eliminar y hacer sitio al nuevo documento. Si no existen dichas víctimas adecuadas, el documento no entra en la caché. Para seleccionar las víctimas se busca en la diagonal del cubo formado por los grupos y se van eliminando hasta que haya sitio suficiente. Si el coste de eliminar los documentos es mayor que el del documento a introducir, no se eliminan ni se almacena el nuevo documento.

Periódicamente se divide el contador de referencias de los documentos por la mitad. Este proceso se realiza cuando se cumple la siguiente condición:

$$\frac{\sum_{i=1}^{\#Layers} i \cdot DocsInLayer_i}{\#docsInCache} > \frac{\lfloor \log_2 MAX\_NRef \rfloor}{2} \qquad (2.63)$$

donde *#Layers* es el número de grupos, $DocsInLayer_i$ es el número de documentos en el grupo *i*, mientras que *#docsInCache* representa el número total de documentos almacenados en la caché.

2.3.2.68 *Rand*

La política de reemplazo *Rand* elimina los documentos de la caché de forma aleatoria para hacer sitio a los nuevos documentos.

2.3.2.69 *Randomized Replacement with General Value Functions* (RRGVF)

Cuando hay que hacen un reemplazo en RRGVF [Psounis,2001] la primera vez se seleccionan *N* elementos aleatoriamente y se eligen los de menor valor usando otra política de reemplazo (LRU, GD-Size, ...). Los *M* documentos restantes se mantienen para la próxima iteración. Se continúa con las iteraciones hasta que hay suficiente espacio para el nuevo documento. El valor óptimo de *M* es:

$$M = N - \sqrt{(N+1)100/n} \tag{2.64}$$

Donde *N* es la longitud de la muestra y *n* es un valor que informa de que ha ocurrido un error si el elemento eliminado no pertenece al *n* por ciento de los documentos menos útiles para todos los documentos.

En las siguientes iteraciones se seleccionan *N*-*M* documentos (para llegar a *N*) y se vuelve a realizar la misma operación. El algoritmo propone usar *N*=8 y *M*=2.

2.3.2.70 *Randomized Climb-C y Randomized* LRU-C

Propuestos en [Starobinski,2001], los algoritmos definen $c_{\max} = \max(c_1, c_2, ..., c_N)$ como el coste máximo de todos los *N* documentos almacenados en la caché. Se define el coste normalizado de cada documento *i* como: $\tilde{c}_i = c_i / c_{\max}$. Cuando se accede al documento *i*, se cambia de posición en la cola (siguiendo la política de reemplazo *Climb* o LRU respectivamente) con una probabilidad $\tilde{c}_i$. En otro caso (con probabilidad $1-\tilde{c}_i$), no se mueve de posición.

2.3.2.71 *Randomized Climb*-S y *Randomized* LRU-S

Propuestos también en [Starobinski,2001], definen $s_{\min} = \min(s_1, s_2, ..., s_N)$ como el tamaño mínimo de los N documentos almacenados en la caché y $\tilde{s}_i = s_{\min} / s_i$ como la densidad normalizada. Cuando hay un acierto en caché, los documentos cambian de posición en la cola (siguiendo la política de reemplazo *Climb* o LRU respectivamente) con una probabilidad $\tilde{s}_i$. En otro caso no se mueven.

2.3.2.72 Harmonic

Harmonic [Hosseini-Khayat,1997] elimina de la caché los documentos de forma aleatoria con una probabilidad inversamente proporcional a su coste específico:

$$\cos t_i = \frac{c_i}{s_i} \tag{2.65}$$

siendo c_i el coste y s_i el tamaño del documento respectivamente.

2.3.2.73 *Neural Network Proxy Cache Replacement* (NNPCR)

NNPCR [Cobb,2008] es una política de reemplazo que emplea una red neuronal usando el modelo del Perceptrón Multicapa con dos capas ocultas y *feed-forward*. NNPCR tiene una primera fase de entrenamiento empleando muestras de tráfico anteriores y usando el algoritmo *back-propagation* para actualizar los pesos de los enlaces de la red. Una vez entrenada la red neuronal se emplea para tomar decisiones de reemplazo. Como entradas se tiene en cuenta los parámetros frecuencia, recencia y tamaño de los documentos. La red neuronal devuelve una valoración para cada documento y se realiza el reemplazo en función de dicha valoración.

En [Romano,2011] se propone una implementación de NNPCR, denominada NNPCR-2, para el *proxy* Squid [Squid,2011]. NNPCR-2 emplea el concepto de ventana deslizante de una petición definido como cierto tiempo transcurrido antes y después de que la petición se haya efectuado. La frecuencia y la recencia se estiman dentro de la ventana deslizante. Cuando se pide un documento, el valor de su recencia viene dado por la ecuación:

$$recency(x) = \begin{cases} \max(SWL, \Delta T)_i & \text{si el documento } i \text{ se pidió antes} \\ SWL & \text{en otro caso} \end{cases} \tag{2.66}$$

donde ΔT_i es el tiempo transcurrido desde la última petición al documento i, mientras que SWL (*Sliding Window Length*) es el tamaño de la ventana deslizante.

La frecuencia se calcula mediante la fórmula:

$$frequency(x) = \begin{cases} frequency(x) + 1 & \text{si } \Delta T_i \le SWL \\ MAX\left[\dfrac{frequency(x)}{\left\lceil \dfrac{\Delta T_i}{SWL} \right\rceil}, 1 \right] & \text{en otro caso} \end{cases} \tag{2.67}$$

Se considera un valor máximo para la frecuencia de 128.

2.3.2.74 *Artificial Web Caching Model*

Esta política de reemplazo propuesta en [Ali,2007] está especialmente diseñada para su uso en el navegador del usuario. El modelo consiste en dos cachés: una caché denominada *short-web* almacenada en la memoria y otra denominada *long-web* que se almacena en disco.

Cuando el usuario accede a una página Web, todos los documentos susceptibles de ser almacenados en caché se almacenan en la caché *short-Web*. Cuando esta caché se llena o se termina la sesión, los documentos referenciados más de cierto número *B* veces se transfieren a la caché *long-Web*. Por otro lado, cuando la caché *long-Web* se llena, se emplea un perceptrón multicapa con *back-propagation* para decidir cuál se reemplaza. Para ello se asignan valores de prioridad a cada documento almacenado en la caché *long-Web*, eliminándose los de menor prioridad.

La red neuronal empleada tiene tres capas: una de entrada, una oculta y una de salida. La capa de entrada tiene cuatro nodos, uno para cada parámetro de entrada: frecuencia, instante del último acceso al documento, tamaño y latencia del documento. En la capa de salida hay un nodo que presenta la prioridad calculada.

2.3.2.75 *Genetic Algorithm caching model* (GA)

GA [Vakali,2002] es un procedimiento iterativo basado en algoritmos genéticos que busca una posible solución para elegir los documentos a eliminar de la caché. Los documentos se codifican como una cadena de bits siguiendo los siguientes parámetros:

$$act_i = \begin{cases} 0 & \text{si el documento } i \text{ se elimina de la caché} \\ 1 & \text{en otro caso} \end{cases} \tag{2.68}$$

$$sr_i = \frac{t_i - lm_i}{now - t_i} \tag{2.69}$$

$$df_i = \frac{1}{a_i} \tag{2.70}$$

$$rr_i = lat_s \cdot band_s \tag{2.71}$$

donde act_i denota la denominada *action function* (función de acción), sr_i es el *staleness ratio* (tasa de antigüedad), df_i es la *dynamic frequency* (frecuencia dinámica) *y* rr_i es el *retrieval rate* (tasa de recuperación). t_i es el instante de tiempo en el que el documento *i* se almacenó en la caché, lm_i es el instante en el que se modificó por última vez, y *now* es el instante de tiempo actual; a_i es el número de accesos a otros documentos desde que *i* fue referenciado por última vez; lat_s es la latencia para abrir una conexión con el servidor *s* y $band_s$ el ancho de banda de la conexión.

Los documentos son evaluados siguiendo un criterio de calidad denominado función objetivo. Se proponen tres funciones objetivo diferentes:

$$F_1(x) = \sum_{i=1}^{N} (act_i \cdot sr_i \cdot df_i) \tag{2.72}$$

$$F_2(x) = \sum_{i=1}^{N} \left(act_i \cdot \frac{df_i}{rr_i} \right) \tag{2.73}$$

$$F_3(x) = \sum_{i=1}^{N} \left(act_i \cdot sr_i \cdot \frac{df_i}{rr_i} \right) \tag{2.74}$$

Se emplean las técnicas de *cruce* y *mutación* para seleccionar documentos almacenados en la caché para crear nuevas generaciones de documentos más fuertes (que cumplen mejor los objetivos). Aquellos documentos que no cumplen con la función objetivo (es decir, los documentos más débiles) son eliminados de la caché.

2.3.2.76 *Fuzzy*

Fuzzy [Calzarossa,2003] es una política de reemplazo basada en lógica difusa. Se consideran como variables de entrada el tamaño, la frecuencia y el tiempo de acceso de cada documento y como variable de salida la probabilidad de reemplazo de cada documento. Para cada una de estas variables se definen conjuntos difusos empleando las denominadas funciones de afiliación (*Membership Funcitions* - MF*)* que describen el grado de pertenencia de cada variable al conjunto difuso. Existen tres funciones MF asociadas con las variables tamaño y frecuencia denominadas *LOW*, *MEDIUM* y *HIGH*. Para describir la variable *tiempo de acceso* se han usado cinco funciones MF: *VERY LOW*, *LOW*, *MEDIUM*, *HIGH* y *VERY HIGH*. Para describir la variable *probabilidad de reemplazo* se emplean cuatro MF: *LOW*, *MEDIUM*, *HIGH*, *VERY HIGH*. A partir de las MF especificadas se generan una serie de reglas (se proponen dos conjuntos, uno de doce y otro de veinte reglas) para calcular el valor de la variable de probabilidad de reemplazo. Una vez que se obtiene el valor de la variable *probabilidad de reemplazo* se realiza un proceso de d*efuzzyfication* empleando el método del centroide para obtener su probabilidad de reemplazo. Finalmente, se eliminan aquellos documentos con el mayor valor de dicha probabilidad.

2.3.2.77 Política de reemplazo de Sabeghi

La política de reemplazo basada en lógica difusa propuesta en [Sabeghi,2006] considera las variables de entrada *tiempo de acceso* (con valores *HIGH*, *MEDIUM* o *LOW*), *frecuencia* (con valores *HIGH*, *MEDIUM* o *LOW*) y *distancia* desde la última petición de un documento (con valores *SHORT*, *MEDIUM* o *LONG*). Como variable de salida se considera la *prioridad de reemplazo* (con valores *LOW*, *NORMAL* o *HIGH*). Las

variables de entrada se hacen corresponder con conjuntos difusos de valores entre cero y uno. Se usan tres reglas basadas en la operación *AND* para obtener el valor de la variable de salida a partir de los valores de entrada. Se eliminan de la caché los documentos con mayor valor de la variable *prioridad de reemplazo*.

2.3.3 Clasificación de las políticas de reemplazo

En este apartado se estudian, en primer lugar, las diferentes clasificaciones que se han realizado de las políticas de reemplazo por varios autores para después realizar una clasificación de las políticas de reemplazo comentadas en el apartado anterior.

2.3.3.1 Clasificaciones propuestas

En [Jin,2001] las políticas de reemplazo se clasifican, en función de los parámetros que tienen en cuenta, en los siguientes grupos:

- Basadas en peticiones recientes – Incorporan el tiempo del último acceso a los documentos (y el tamaño o coste) en el proceso de reemplazo.
- Frecuencia de peticiones – Consideran la frecuencia de acceso a los documentos (y el tamaño o coste).
- Basadas en peticiones recientes y frecuencia – Consideran tanto la frecuencia de acceso como el tiempo del último acceso.

Esta clasificación tiene la desventaja de no tener en cuenta aquellas políticas que no están basadas ni en el tiempo de acceso ni en la frecuencia como puede ser LFF, que únicamente evalúa el tamaño de los documentos, o LAT, que considera la latencia.

En [Balamash,2004] se clasifican las políticas de reemplazo de dos formas. La primera de ellas en:

- Deterministas
- Aleatorias

Y la segunda según estén basadas en:

- El tiempo desde la última petición
- La frecuencia de las peticiones
- El tamaño de los documentos

Esta segunda clasificación también se realiza en [Khayari,2003] y [Khayari,2005] y no es una clasificación excluyente, es decir, una política de reemplazo puede pertenecer a más de una clase simultáneamente e incluso a las tres a la vez tal y como puede observarse en la Figura 2.2. En ella se observan algunas políticas de reemplazo situadas en círculos que representan si dichas políticas tienen en cuenta la frecuencia (*Frequency information*), el tamaño (*Size*) y la cercanía en el tiempo de las peticiones (*Recency information*). En las

intersecciones entre los círculos se sitúan aquellas políticas que tienen en cuenta más de un parámetro.

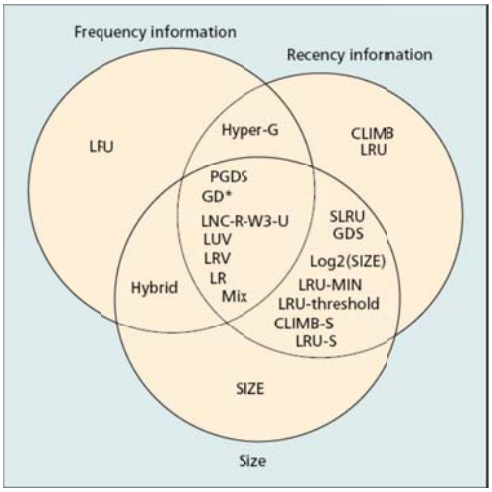


Figura 2.2. Clasificación de Balamash de las políticas de reemplazo – Fuente [Balamash,2004]

Esta clasificación también adolece de la misma desventaja que la anteriormente comentada de [Jin,2001].

En [Podlipnig, 2003] y [Wong,2006] se amplía la clasificación anterior de los algoritmos, distinguiendo entre políticas:

- Basadas en peticiones recientes – Usan el tiempo del último acceso a los documentos como factor principal para el reemplazo. Son variaciones de la política LRU.
- Frecuencia de peticiones – Usan la frecuencia de acceso a los documentos como factor principal para los reemplazos. Son variaciones de la política LFU.
- Basadas en peticiones recientes y frecuencia – Usan tanto el tiempo del último acceso como la frecuencia de acceso para realizar los reemplazos.
- Función de valoración – Asignan a cada documento una función de coste basada en parámetros como la frecuencia de acceso, el tamaño, la latencia, etc. Se reemplazan los documentos con menor función de coste.
- Aleatorias – La selección de los documentos a reemplazar se hace de forma aleatoria.

En [Nagaraj,2004] las políticas de reemplazo se clasifican en:

- Tradicionales – Incluye los algoritmos clásicos LRU, LFU y Pitkow/Recker y alguna de sus variantes.
- Basadas en clave – Incorpora los algoritmos que usan una clave primaria para eliminar los documentos de la caché. Esta clave primaria es alguna de las características del documento como el tamaño o la latencia.
- Basadas en coste – Aglutina los algoritmos que usan una función de coste basada en parámetros del documento como el tamaño, la latencia, etc.

2.3.3.2 Clasificación propuesta

En el presente trabajo se va a realizar una clasificación de las políticas de reemplazo en función de cada uno de los parámetros que se tienen en cuenta para realizar los reemplazos. Estos parámetros van a ser el tiempo transcurrido desde el último acceso (Rec.), el número de referencias (Frec.), el tamaño del documento (Tam.), el coste y la latencia (Lat.). De este modo, cada política de reemplazo puede incluirse en más de un grupo si el documento a reemplazar se calcula en función de más de un parámetro. También se va a considerar si la política de reemplazo emplea una única lista para administrar los documentos o utiliza más de una lista (multi-lista - ML), así como la aleatoriedad del algoritmo (Aleat.). La Tabla 2.1 muestra dicha clasificación. En dicha tabla se indica con el símbolo ☑ que la política de reemplazo tiene en cuenta dicho parámetro o es de dicho tipo, con '-' se indica que no lo cumple y con 'Opc.' que puede o no hacerlo de forma opcional.

	Parámetro					Tipo	
Política	Rec.	Frec.	Tam.	Coste	Lat.	ML	Aleat.
FIFO	-	-	-	-	-	-	-
LRU	☑	-	-	-	-	-	-
CLIMB	☑	-	-	-	-	-	-
PART	☑	-	-	-	-	☑	-
C-LRU	☑	-	-	-	-	☑	-
SLRU	☑	-	☑	-	-	-	-
SegLRU	☑	-	-	-	-	☑	-
PSS	☑	-	☑	-	-	☑	-
LRU-K	☑	-	-	-	-	-	-
LRU-MIN	☑	-	☑	-	-	-	-
ML-LRU	☑	-	-	-	-	☑	-
LRU-LSC	☑	-	☑	☑	-	☑	-
HLRU	☑	-	-	-	-	-	-
LRU*	☑	☑	-	-	-	-	-
LRU-SP	☑	☑	☑	-	-	☑	-
LRU-HOT	☑	-	-	☑	-	☑	-
LRU-QoS	☑	-	-	-	-	☑	-
LRU-HOT+QoS	☑	-	-	☑	-	☑	-
LRU-SLFR	☑	☑	-	-	☑	-	-
MQ	☑	☑	-	-	-	☑	-
2Q	☑	-	-	-	-	☑	-
Pitkow/Recker	☑	-	☑	-	-	-	-
LDF	☑	-	-	-	-	-	-
LLDF	☑	-	-	-	-	-	-
Value Aging	☑	-	-	-	-	-	-
Differential Aging	☑	-	-	-	-	-	-
LFU	-	☑	-	-	-	-	-
LFU*	-	☑	-	-	-	-	-
LFU-AGING	-	☑	-	-	-	-	-
LFU* AGING	-	☑	-	-	-	-	-
LFU-DA	-	☑	-	-	-	-	-
SSW-LFU	-	☑	-	-	-	-	-
α-Aging	-	☑	-	-	-	-	-

Tabla 2.1. Clasificación de políticas de reemplazo

	Parámetro					Tipo	
Política	Rec.	Frec.	Tam.	Coste	Lat.	ML	Aleat.
FBR	☑	☑	-	-	-	☑	-
LRFU	☑	☑	-	-	-	-	-
ALRFU	☑	☑	-	-	-	-	-
LUV	☑	☑	☑	☑	-	-	-
swLFU	-	☑	☑	-	-	-	-
One-Timers	☑	Opc.	Opc.	-	-	☑	-
Generational Replacement	-	☑	-	-	-	☑	-
Exp1	☑	☑	-	-	-	-	-
GD-SIZE	-	-	☑	☑	-	-	-
GDSF	-	☑	☑	☑	-	-	-
N-gram	-	☑	☑	☑	-	-	-
g-GDFS	-	☑	☑	☑	-	-	-
GDSP	-	-	☑	☑	-	-	-
GD*	-	☑	☑	☑	-	-	-
LRV	-	-	☑	☑	-	-	-
LANDLORD	-	-	☑	☑	-	-	-
LAT	-	-	-	-	☑	-	-
MIX	-	☑	☑	-	☑	-	-
MCW-n	-	-	-	☑	-	☑	-
Bolot/ Hoschka	☑	-	☑	-	-	-	-
Virtual Cache	-	☑	☑	☑	-	☑	-
LR	-	☑	☑	-	-	-	-
M-Metric	☑	☑	☑	-	-	-	-
TSP	-	☑	☑	☑	-	-	-
SE	☑	☑	☑	☑	☑	-	-
LPPB-R	☑	☑	☑	-	-	☑	-
HCRP	☑	☑	☑	-	-	-	-
LFF	-	-	☑	-	-	-	-
HYPER-G	☑	☑	☑	-	-	-	-
Log(Size)+LRU	☑	-	☑	-	-	-	-
LLF	-	-	-	-	☑	-	-
HYBRID	-	☑	☑	-	-	-	-
LNC-R-W3	☑	-	☑	-	☑	-	-
LNC-R-W3-U	☑	-	☑	-	☑	-	-
NNC	☑	☑	☑	-	-	-	-
CSS	-	☑	☑	-	-	☑	-
RAND	-	-	-	-	-	-	☑
RRGVF	Opc.	Opc.	Opc.	Opc.	Opc.	-	☑
CLIMB-C	-	-	-	☑	-	-	☑
CLIMB-S	-	-	☑	-	-	-	☑
LRU-C	☑	-	-	☑	-	-	☑
LRU-S	☑	-	☑	-	-	-	☑
HARMONIC	-	-	☑	☑	-	-	☑
NNPCR	☑	☑	☑	-	-	-	-
Artificial Web Caching Model	☑	☑	☑	-	☑	☑	-
GA	☑	☑	-	-	☑	-	-
Fuzzy	☑	☑	☑	-	-	-	-
Sabeghi	☑	☑	-	-	-	-	-

Tabla 2.1 (continuación). Clasificación de políticas de reemplazo

2.4 Políticas de control de admisión

En el presente apartado se va a realizar una descripción de las políticas de control de admisión (también denominadas políticas de control de acceso) propuestas en la literatura. Estas políticas suelen estar asociadas a políticas de reemplazo concretas y son las encargadas de decidir, para cada documento que llega a la caché, si éste debe almacenarse o no en la misma.

2.4.1 *LRU-Threshold*

Impone un límite máximo al tamaño de los documentos que pueden insertarse en la caché.

2.4.2 *LRU-Adaptive*

LRU-Adaptive [Markatos,1996] funciona como *LRU-Threshold* con la excepción de que el valor umbral del tamaño no es fijo, sino que se calcula dinámicamente. Se comienza con un umbral inicial y se incrementa o decrementa el su valor periódicamente. Si se observa que el rendimiento mejora se continúa realizando la misma operación (incremento o decremento), en otro caso se realiza la operación inversa. El valor inicial del umbral es de 16 kB y el paso de incremento o decremento se fija a 2 kB.

2.4.3 Control de admisión para política de reemplazo GDSF

Esta política de control de admisión asociada a GDSF [Cherkasova,2001] únicamente almacena un nuevo documento en la caché cuando la valoración aplicada por la política de reemplazo GDSF es mayor que la de los documentos que habría que eliminar de la caché para almacenarlo.

2.4.4 Política de Aggarwal

Para esta política de control de acceso [Aggarwal,1999] se emplea una caché auxiliar LRU que contiene identificadores de documentos, frecuencia, tiempo entre accesos y cantidad de accesos entre accesos. El tamaño de esta caché debería ser tal que admitiera dos veces la media de documentos que caben en la caché principal. La idea es evaluar la popularidad de un documento antes de introducirlo en la caché.

Si hay suficiente espacio en la caché, los documentos se almacenan en ella. Por el contrario, si no hay espacio y el documento no está en la caché auxiliar, no se guarda en la caché. Si ya estaba en la caché auxiliar, se compara el valor

$$c_{i_k}(1-r_{i_k})/\Delta T_{i_k k} \tag{2.75}$$

del documento a insertar con el valor

$$\sum_i c_i(1-r_i)/\Delta T_{ik} \tag{2.76}$$

$$r_i = \min\left\{1, \delta_{t_{i1}}/\delta_{t_{i2}}\right\} \tag{2.77}$$

del conjunto de documentos candidatos a ser eliminados usando la política de reemplazo. En estas ecuaciones, c_i representa el coste de obtener el documento, siendo δt_{i1} la diferencia entre t (instante de tiempo actual) y el momento en que fue accedido el documento i por última vez y δt_{i2} la diferencia entre el tiempo de expiración del documento y t. Finalmente, ΔT_i representa el número de accesos desde la última vez que se referenció i. El documento se inserta en la caché si el valor del documento es mayor que el de los documentos a eliminar de la caché.

2.4.5 *Aggarwal Dynamic Frequency* (Aggarwal DF)

Aggarwal DF [Aggarwal,1997] se aplica a las política de reemplazo LDF y LLDF. Se mantiene una caché auxiliar LRU con identificadores y tiempos de acceso a los documentos. Su tamaño debe ser el doble del número medio de documentos que caben en la caché principal. El algoritmo es el siguiente:

1. Sea i_t el documento que se desea insertar en la caché en la iteración t. Si hay suficiente espacio en la caché, se almacena.
2. Si no hay espacio suficiente, determinar el conjunto de posibles documentos a eliminar de la caché $o_1,...,o_n$, usando la política de reemplazo LDF (epígrafe 2.3.2.24).
3. Si el documento que llega no está en la caché auxiliar, no se inserta.
4. En otro caso, se compara el valor *DF* (epígrafe 2.3.2.24) del documento que quiere entrar en la caché con la suma de los valores *DF* del conjunto de posibles documentos a extraer.
5. Si el valor de *DF* del documento a insertar es mayor que la del conjunto de documentos a extraer, se inserta en la caché y sus datos en la caché auxiliar.

Cuando se accede a un documento que está en la caché auxiliar, esté o no en la caché principal, se actualiza su *time-stamp* y se coloca en orden LRU.

2.4.6 *Ignore First Hit* (IFH)

IFH [Wooster,1996] sólo almacena en caché aquellos documentos que son accedidos al menos dos veces, ya que la primera vez son desechados. Se mantiene una lista con los documentos que han sido desechados para, la segunda vez que sean referenciados,

insertarlos en la caché. El tamaño de esa lista es un parámetro de la política de control de acceso.

2.4.7 Política de control de admisión de Kaya

La política de control de admisión propuesta en [Kaya, 2009] calcula el tiempo de descarga de cada documento (z). Si dicho valor es menor que un determinado valor umbral ($\overline{z}$), el documento no se almacena en caché. El valor umbral se define como:

$$\overline{z} = b \cdot \left(\frac{n}{L^*} \right)^{1/a} \tag{2.78}$$

Siendo n el número de documentos que podrían ser potencialmente solicitados por los usuarios del *proxy*, a y b son los parámetros de forma y escala de una distribución de Pareto y L^* es el número óptimo de documentos marcados como *cacheables* en el conjunto de n documentos. El valor de L^* se calcula resolviendo la siguiente ecuación:

$$\frac{\partial}{\partial L}\left(L^{1-\frac{1}{\alpha}} \left(1 - \left(1 - \frac{R}{L} \right)^{\frac{1}{1-\alpha}} \right) \right) = 0 \tag{2.79}$$

siendo R el número medio de documentos almacenados en la caché y α un parámetro de sensibilidad de la edad de los documentos, modelado según un proceso de Poisson no homogéneo con tasa media instantánea dada por:

$$\theta(x) = \frac{1}{\alpha x + \beta} \text{ con } \alpha, \beta > 0 \text{ y } \alpha < 1 \tag{2.80}$$

2.5 Comparación de políticas de reemplazo

En los apartados anteriores se ha realizado un estudio pormenorizado tanto de las políticas de reemplazo en cachés Web como de las políticas de control de acceso, métricas de rendimiento y características del tráfico. En el presente epígrafe se va a realizar, en primer lugar, un estudio de una muestra de tráfico real para comprobar si se cumplen dichas características. Posteriormente se compara el rendimiento de las políticas de reemplazo aleatorias y la aplicación de varias de las políticas de reemplazo en función del tipo de los documentos. Además, se proponen una serie de métricas de rendimiento para cachés Web con políticas de control de admisión.

2.5.1 Caracterización de las muestras de tráfico utilizadas

Para evaluar el rendimiento de las políticas de reemplazo se ha utilizado una muestra de tráfico HTTP de un *proxy* del proyecto IRCache [IRCache,2011] para realizar las diferentes simulaciones. Dicho proyecto ofrece un marco para el estudio de las cachés al ofrecer, bajo suscripción, acceso a las muestras de tráfico que procesan los *proxies* del proyecto. De hecho, las muestras del proyecto IRCache han sido ampliamente utilizadas para dichos estudios, publicándose numerosos trabajos basándose en ellas [IRCache,2011]. Concretamente, el *proxy* del que se han empleado sus muestras se encuentra situado en el *Research Triangle Park* (Carolina del Norte, EEUU). Las muestras más actuales a las que se ha tenido acceso incluyen peticiones desde el 7 al 11 de Junio del 2004 generadas a un *proxy* Squid [Squid,2011]. Estas muestras de tráfico contienen información de cada petición HTTP procesada por el *proxy*, como el instante de tiempo en el que se inició la petición, el tamaño del documento, la URL, el tipo de documento, el tipo de petición (GET, POST,…) y el código de respuesta del servidor.

La muestra fue inicialmente preprocesada para eliminar las respuestas generadas dinámicamente mediante CGI (*Common Gateway Interface*) ya que los documentos devueltos mediante este tipo de peticiones no deben ser almacenadas en caché [Zhang,2000]. Por tanto, aquellas peticiones cuya URL contiene las cadenas ‘cgi’, ‘cgi-bin’ o ‘?’ han sido descartadas. Las peticiones que contienen la cadena ‘:3128’ también han sido eliminadas ya que este es el puerto que utiliza IRCache para intercambiar información entre *proxys* colaborativos. Se consideran códigos de respuesta que permiten almacenar el documento en caché los siguientes: 200 (*OK*), 203 (*Partial*), 206 (*Partial Content*), 300 (*Multiple Choices*), 301 (*Moved*) y 302 (*Redirects*) [Gourley,2002]. Para el código de respuesta 304 (*Not Modified*) el tamaño presente en las muestras es el del mensaje de respuesta y no el tamaño real del documento (simplemente informa de que el documento no ha sido modificado y sigue siendo válida la copia almacenada en caché). Consecuentemente dichos documentos han sido solicitados de nuevo al servidor Web original con el fin de obtener su tamaño real. Por otro lado, los documentos cuyo tipo de contenido era desconocido en las muestras han sido también solicitados de nuevo. La Tabla 2.2 resume las características de la muestra de tráfico después del proceso mencionado anteriormente.

Número de peticiones	4.040.036
Tamaño (GBytes)	40,4
Documentos diferentes	1.713.903
One-timers	1.299.217

Tabla 2.2. Características de la muestra de tráfico

Si se dividen las peticiones de documentos en función de su tipo de contenido (*content-type*) siguiendo los tipos definidos por IANA (*Internet Assisted Numbers Authority*) [IANA,2011], las principales estadísticas de cada uno de los tipos (*Application*, *Audio*, *Images*, *Text* y *Video*) se muestran en la Tabla 2.3, mientras que en la Figura 2.3 se representa el porcentaje de peticiones y tráfico en forma de histograma.

Estadístico	Application (Aplicación)	Audio	Images (Imágenes)	Text (Texto)	Video (Vídeo)
Media (kB)	41	123	4	14	260
Mediana (kB)	3	7	1	3	573
Desviación típica (kB)	761	948	21	104	974
Peticiones (%)	8,08	0,28	75,54	15,77	0,12
Bytes (%)	33,51	3,49	36,33	23,15	3,19
Documentos diferentes (%)	26,90	38,00	45,98	33,92	75,02

Tabla 2.3. Características de cada tipo de documentos

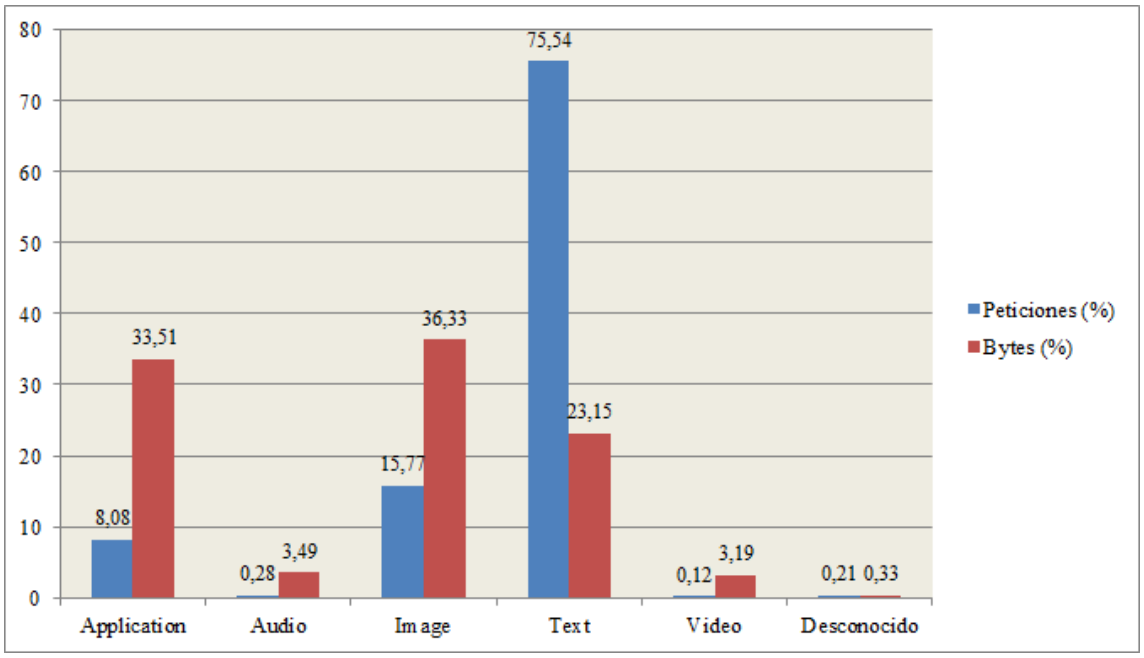


Figura 2.3. Histograma del porcentaje de peticiones y tráfico en función del tipo de los documentos

Los tipos *Image* y *Text* suman el 91% de las peticiones y el 59% del tráfico transferido, de modo que son los tipos más influyentes. Por otro lado, el tipo *Application* genera únicamente el 8% de las peticiones aunque es responsable del 33% del tráfico generado. Hay grandes diferencias entre el tamaño de los documentos de los diferentes tipos tal y como se puede comprobar si se compara la media y la mediana obtenida por cada uno de ellos. Estas diferencias se aprecian en la Figura 2.4, donde se representa la función de distribución acumulada del tamaño de los documentos por su tipo.

Debido a que la desviación típica de los tamaños es mayor que la media, se podría presentar un comportamiento de cola pesada, de modo que se ha representado la función de distribución acumulada complementaria de los tamaños en la Figura 2.5.

De acuerdo con los estudios de [Crovella,1997] se puede asumir el comportamiento de cola pesada si aparecen líneas rectas en esta representación durante al menos tres órdenes de magnitud. Tal y como se aprecia en la Figura 2.5, este comportamiento se cumple para todos los tipos de documentos, siendo estos resultados coherentes con los obtenidos en [Arlitt,1996c].

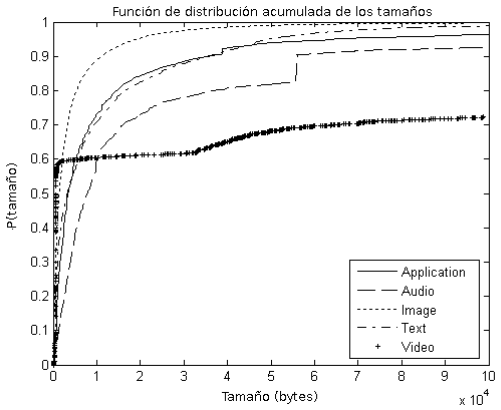


Figura 2.4. Función de distribución acumulada del tamaño de los documentos en función de su tipo

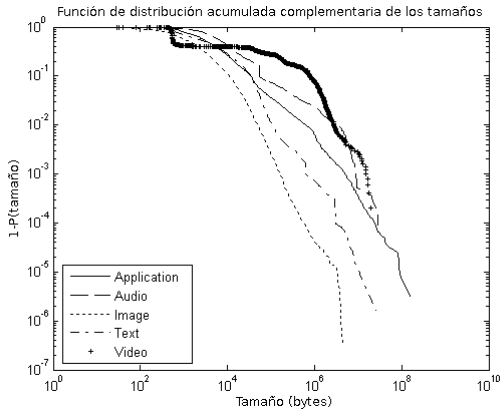


Figura 2.5. Función de distribución acumulada complementaria del tamaño de los documentos en función de su tipo

Para modelar la popularidad se ha empleado la ley de Zipf, obteniéndose como valores del parámetro α, tanto para la muestra total como para cada uno de los tipos de documento, los presentados en la Tabla 2.4. La Figura 2.6 muestra la representación de la popularidad para toda la muestra.

Tipo de documento	**α**
Muestra completa	0.64
Application	0.78
Audio	0.45
Image	0.62
Text	0.73
Video	0.50

Tabla 2.4. Valores de α para el modelado de la ley Zipf

Para el cálculo del parámetro β de la política de reemplazo GreedyDual* se necesita modelar la correlación temporal [Jin,2000]. Para ello se representa en escala logarítmica doble el porcentaje de ocurrencias de las distancias entre referencias de los documentos con la misma frecuencia de acceso k en la muestra completa, siendo β la pendiente de la distribución. Esta pendiente se calcula usando una regresión de mínimos cuadrados aproximando la ecuación:

$$T \propto \frac{1}{t^{\beta}} \text{ para documentos con frecuencia } k \tag{2.81}$$

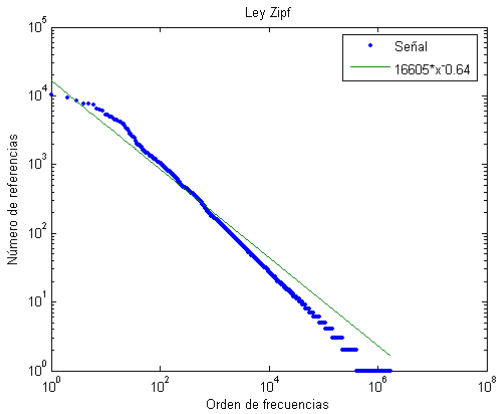


Figura 2.6. Representación de la popularidad para la muestra completa

Se ha elegido un valor de k igual a ocho para este análisis, aunque se obtienen valores similares para otros valores de k. La Figura 2.7 muestra el porcentaje de ocurrencias de las distancias entre referencias en la muestra completa para documentos que han sido referenciados en ocho ocasiones.

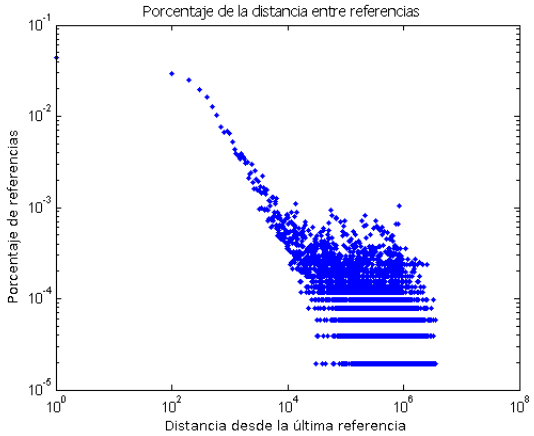


Figura 2.7. Porcentaje de ocurrencias de las distancias entre referencias para la muestra completa para k=8.

La Tabla 2.5 presenta los valores calculados para β tanto para la muestra completa como para cada uno de los tipos de documentos.

Tipo de documento	**β**
Muestra completa	0.46
Application	0.51
Audio	0.40
Image	0.46
Text	0.54
Video	0.95

Tabla 2.5. Valores de β para el modelado de la correlación temporal de la política de reemplazo GD*

Una característica importante a estudiar es la relación existente entre el tamaño de los documentos y su popularidad, ya que algunas políticas de reemplazo (como las de la familia GDSize) dan mayor preferencia a los documentos pequeños frente a los grandes puesto que suponen que los documentos pequeños son más solicitados que los grandes. Si se representa la popularidad en función del tamaño de los documentos se obtienen las gráficas mostradas en la Figura 2.8.

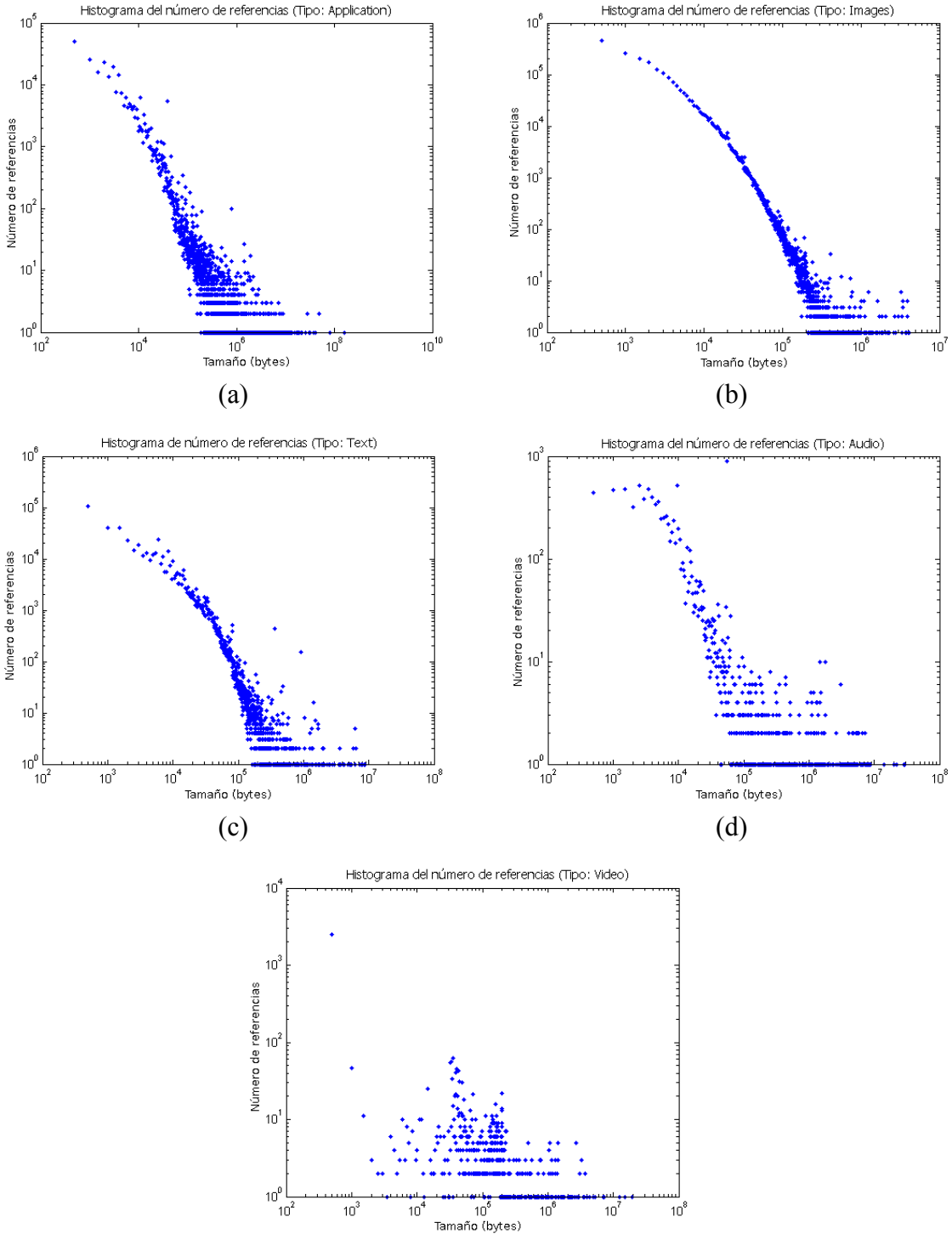


Figura 2.8. Número de referencias en función del tamaño de los documentos para los tipos *Application* (a), *Images* (b), *Text* (c), *Audio* (d) y *Video* (f).

Estas figuras demuestran que esta relación es válida para todos los tipos de documentos excepto para el tipo *Video*, donde la correlación no es tan evidente.

2.5.2 Evaluación de políticas de reemplazo

Las métricas utilizadas para evaluar y comparar el rendimiento de las políticas de reemplazo consideradas en el presente trabajo son las clásicas HR (*Hit Ratio*) y BHR (*Byte Hit Ratio*), ya que son, con diferencia, las más utilizadas a la hora de comparar el rendimiento de las políticas de reemplazo. El HR ofrece una idea de la reducción en la latencia que perciben los usuarios ya que las peticiones serán servidas más rápidamente desde la caché que desde el servidor Web original, dado que la caché se suele encontrar situada más cerca de los usuarios. Por otro lado, la métrica BHR da idea del ancho de banda ahorrado entre la caché y los servidores Web.

Se han tenido en cuenta una serie de consideraciones para realizar unas simulaciones de calidad. La primera consideración es el parámetro de "calentamiento" de la caché (*warm-up*), que es el tiempo que la caché va a estar funcionando antes de que se empiece a medir su rendimiento. Si se empezara a medir el rendimiento cuando la caché está vacía, el elevado número de fallos de caché (puesto que los documentos solicitados no están disponibles) ocasionaría un valor distorsionado del rendimiento, por lo que la medida debe empezar al menos cuando la caché está llena y ya se han realizado algunos reemplazos [Dykes,2000].

Otro asunto a tener en cuenta es cómo distinguir, en la muestra real analizada, la modificación de los documentos. De acuerdo con [Arlitt,1999b] si la diferencia entre los tamaños de peticiones sucesivas al mismo documento es menor del 5% del tamaño del documento, se considera que éste ha sido modificado; en otro caso se considera que la transferencia del documento fue cancelada.

Para realizar la evaluación del rendimiento primero se simuló una caché de tamaño infinito para determinar el tamaño ocupado en la caché en el caso de no tener que realizar ningún reemplazo. El resto de las simulaciones fueron realizadas usando diferentes porcentajes de este tamaño máximo ideal en el que ningún documento es eliminado de la caché.

2.5.2.1 Comparación de las políticas de reemplazo aleatorias

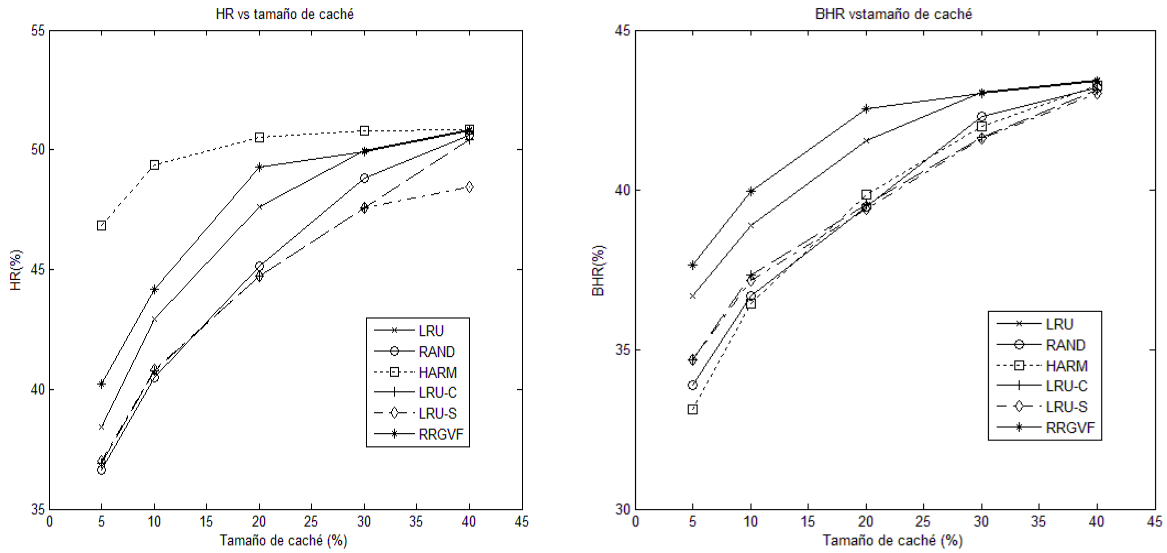
Se ha evaluado el rendimiento de todas las políticas de reemplazo aleatorias comentadas en el epígrafe 2.3.2 (Rand, Harmonic, LRU-C, LRU-S y RRGVF), excepto Climb-C y Climb-S ya que los propios autores [Starobinski,2001] demuestran que consiguen menor rendimiento que LRU-C y LRU-S respectivamente. Estas políticas de reemplazo aleatorias son comparadas también con la política de reemplazo LRU.

Debido a la naturaleza aleatoria de las políticas de reemplazo estudiadas, las simulaciones se han realizado cinco veces por cada política de reemplazo para poder

obtener una estimación del rendimiento haciendo la media de todas las simulaciones. Los tamaños de caché considerados son 40%, 30%, 20%, 10% y 5% del tamaño máximo. Además, se ha utilizado el 50% de la muestra para "calentar" la caché.

Algunas políticas de reemplazo incluyen parámetros que deben ser definidos. De esta forma, para la política de reemplazo Harmonic se ha considerado una función de coste constante, para LRU-C se ha considerado como función de coste el número de paquetes necesarios para transferir el documento y, finalmente, para el algoritmo RRGVF se han usado los valores N=30 y M=12 [Psounis,2001].

La Figura 2.9 muestra la evolución de los valores HR y BHR de las seis políticas de reemplazo evaluadas en función del tamaño de la caché. En estas figuras se muestra únicamente el rendimiento medio (sin mostrar el margen de confianza al 95% de las medidas) de las políticas de reemplazo aleatorias para que sean más fácilmente interpretadas. De todas formas, la variación del rendimiento entre las cinco simulaciones realizadas de cada política de reemplazo aleatoria está en un rango del 1%.



(a) (b)

Figura 2.9. HR (a) y BHR (b) en función del tamaño de la caché para políticas de reemplazo aleatorias

Con respecto al HR, Harmonic claramente resulta más eficiente que el resto de políticas. La diferencia de rendimiento con respecto a las demás se incrementa conforme el tamaño de la caché se decrementa. La segunda política en rendimiento es RRGVF, que mejora ligeramente a LRU. Finalmente, Random, LRU-C y LRU-S obtienen un HR similar, aunque menor que Harmonic, LRU y RRGVF.

Con respecto al BHR, la política de reemplazo RRGVF es mejor que todas las demás políticas, aunque LRU obtiene un rendimiento similar aunque ligeramente inferior. El resto de políticas de reemplazo presenta un comportamiento análogo aunque inferior

que RRGVF y LRU. Incluso Harmonic obtiene el peor resultados para tamaños de caché pequeños.

Como puede observarse, no existe una política de reemplazo que sea mejor que las demás para ambas métricas. Si se quisiera maximizar el HR, la política de reemplazo HARMONIC sería la mejor opción, ya que conforme disminuye el tamaño de la caché, la mejora en el rendimiento con respecto al resto de las políticas de reemplazo es considerable, aunque se obtendría un pobre BHR. Si se quisiera maximizar el BHR, podría elegirse el algoritmo RRGVF, con lo que también se obtendría un buen HR.

2.5.2.2 Comparación de políticas de reemplazo en función del tipo del documento

En este apartado se realiza una evaluación del rendimiento de las políticas de reemplazo LRU, LFU, LFU-DA, GD-Size, GDSF y GD* para cada uno de los tipos de documentos. El objetivo es encontrar cuál de dichas políticas de reemplazo se adapta mejor a cada una de las características del tipo de tráfico que genera cada tipo de documento.

Para las políticas de reemplazo GD-Size, GDSF y GD* se han empleado dos funciones de coste: coste de documento constante ($C(i)$=1) y coste proporcional al número de paquetes (denominado *packets*), considerándose que el coste en número de paquetes para un documento i, viene determinado por la ecuación:

$$C(i) = 2 + \frac{S(i)}{1460} \tag{2.82}$$

siendo $S(i)$ el tamaño del documento i y 1460 el tamaño de carga útil de datos de un paquete TCP (*Transmission Control Protocol*) sobre IP en Ethernet.

La Figura 2.10 muestra la evaluación del rendimiento de las políticas de reemplazo comentadas para el tipo de documento *Application*.

Usando el modelo de coste constante, las políticas de reemplazo basadas en coste resultan claramente mejores que las clásicas con respecto a la métrica HR, alcanzando un rendimiento del 60% para cachés pequeñas. Esta diferencia de rendimiento se hace más evidente conforme el tamaño de la caché se decrementa. Este comportamiento es debido a que LRU es la peor opción para maximizar el HR. Considerando la métrica BHR, el comportamiento es similar, aunque la diferencia de rendimiento entre las políticas es sólo del 4%. Usando el modelo de coste *packets*, GD* y GDSF resultan las mejores opciones para maximizar el HR, aunque son menos eficientes que usando el modelo constante. GD* y GDSF consiguen mayores BHR que las demás políticas de reemplazo, obteniendo mejores resultados que con el modelo de coste constante. Como resumen, para el tipo de documento *Application*, resultan ser las mejores opciones, obteniendo un alto HR aunque un bajo BHR, tal y como era de esperar debido a la relación entre el tamaño y el número de referencias mostrado en la sección 2.5.1. Los documentos pequeños son más solicitados que los grandes y la pendiente de la ley Zipf para el tipo *Application* es grande. Por lo tanto, se consigue un alto HR incluso para cachés pequeñas. Por otro lado, el

comportamiento de cola pesada del tamaño de este tipo de documentos es el más importante, por lo que los documentos de la cola son responsables de un gran porcentaje del tráfico, aunque dichos documentos grandes no son referenciados muchas veces. Esta es la razón por la que se obtienen bajas tasas de BHR.

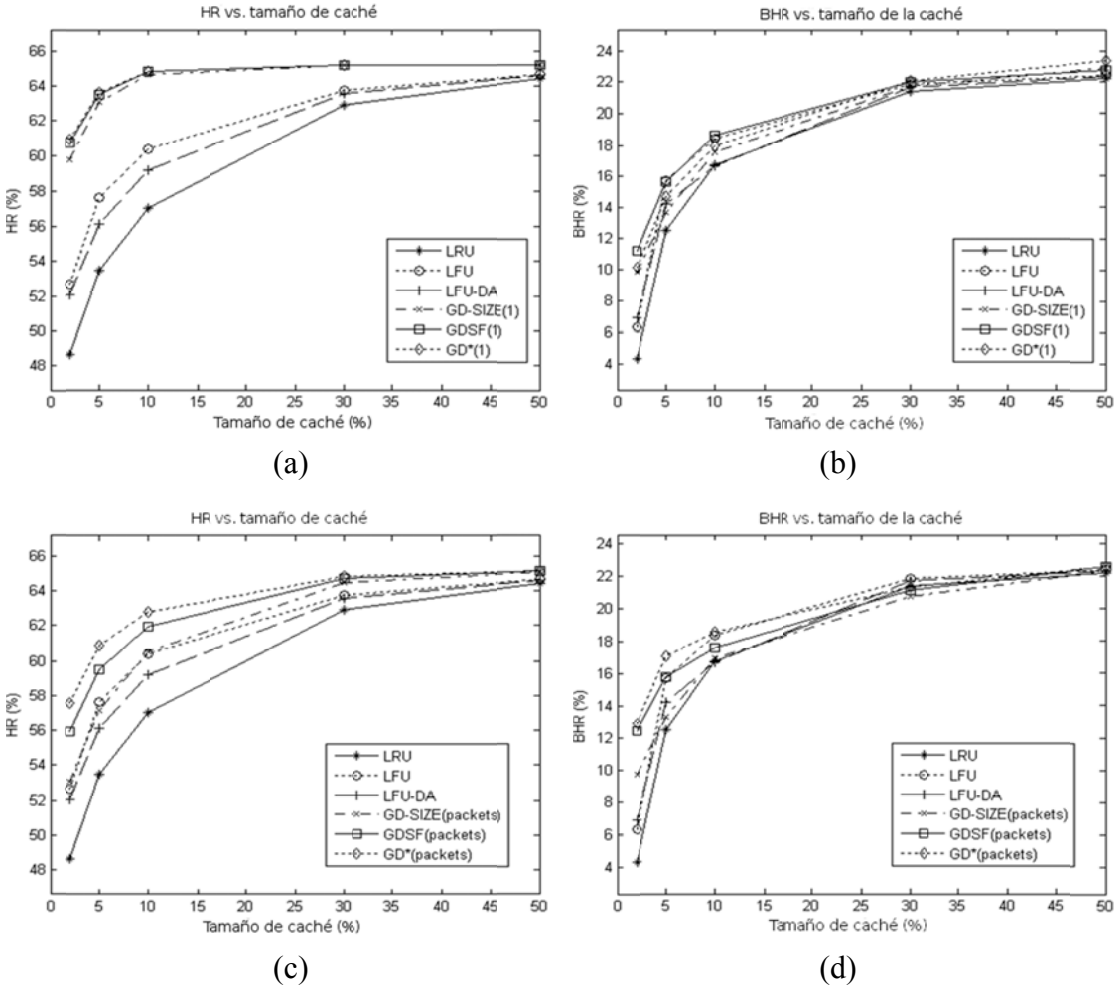


Figura 2.10. Evaluación de políticas de reemplazo para el tipo de documento *Application* empleando una función de coste constante (a) y (b) y coste *packets* (c) y (d)

Los resultados de las simulaciones para el tipo de documento *Audio* se muestran en la Figura 2.11.

GD*, GDSF y GD-Size obtienen un rendimiento similar para todos los tamaños de caché en el modelo de coste constante, resultando mejores que el resto de los algoritmos en más de un 20% si se considera el HR. Para el BHR, LFU y LFU-DA son las mejores opciones para tamaños de caché pequeños, aunque el rendimiento de todas las políticas de reemplazo evaluadas tiende a igualarse conforme el tamaño de la caché se incrementa. Bajo el modelo de coste *packets*, GD* es la política de reemplazo que consigue un mayor HR, mientras que el resto de las políticas, excepto LRU y GD-Size, consiguen el mismo buen rendimiento para el BHR. Para el tipo de documento *Audio*, ambas métricas no pueden se optimizadas simultáneamente para tamaños de caché inferiores al 10%. Para

tamaños de caché mayores, las políticas de reemplazo basadas en coste superan en rendimiento al resto tanto en el HR como en el BHR.

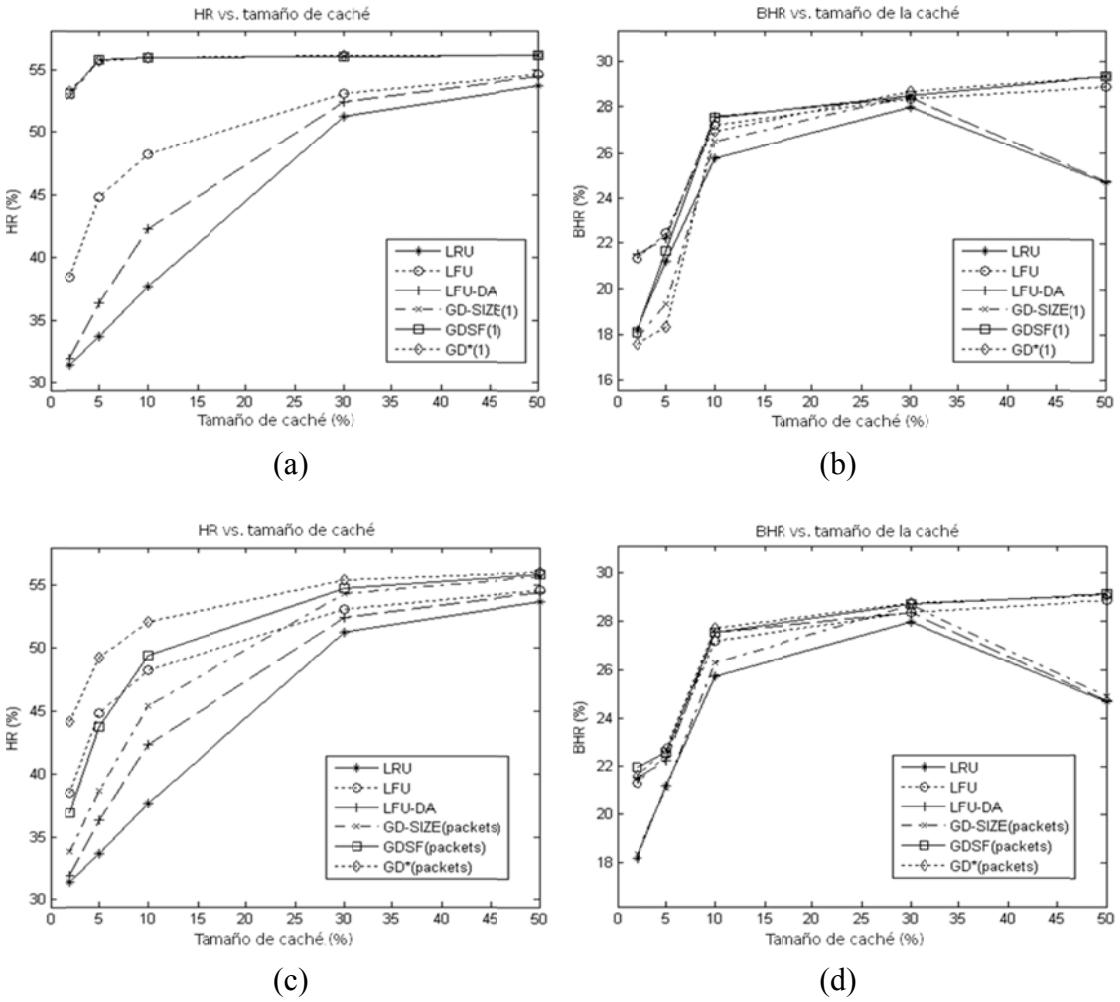


Figura 2.11. Evaluación de políticas de reemplazo para el tipo de documento *Audio* empleando una función de coste constante (a) y (b) y coste *packets* (c) y (d)

El estudio del rendimiento de las políticas de caché para el tipo de documentos *Image* se muestra en la Figura 2.12.

La figura muestra que las políticas basadas en coste maximizan el HR en los dos modelos de coste considerados, aunque el rendimiento es ligeramente superior para el modelo constante. Considerando el BHR, existen pocas diferencias entre las políticas de reemplazo, aunque LFU es la mejor opción para cachés con un tamaño superior al 10%, mientras que GDSF es la que mejor rendimiento obtiene para tamaños de caché menores.

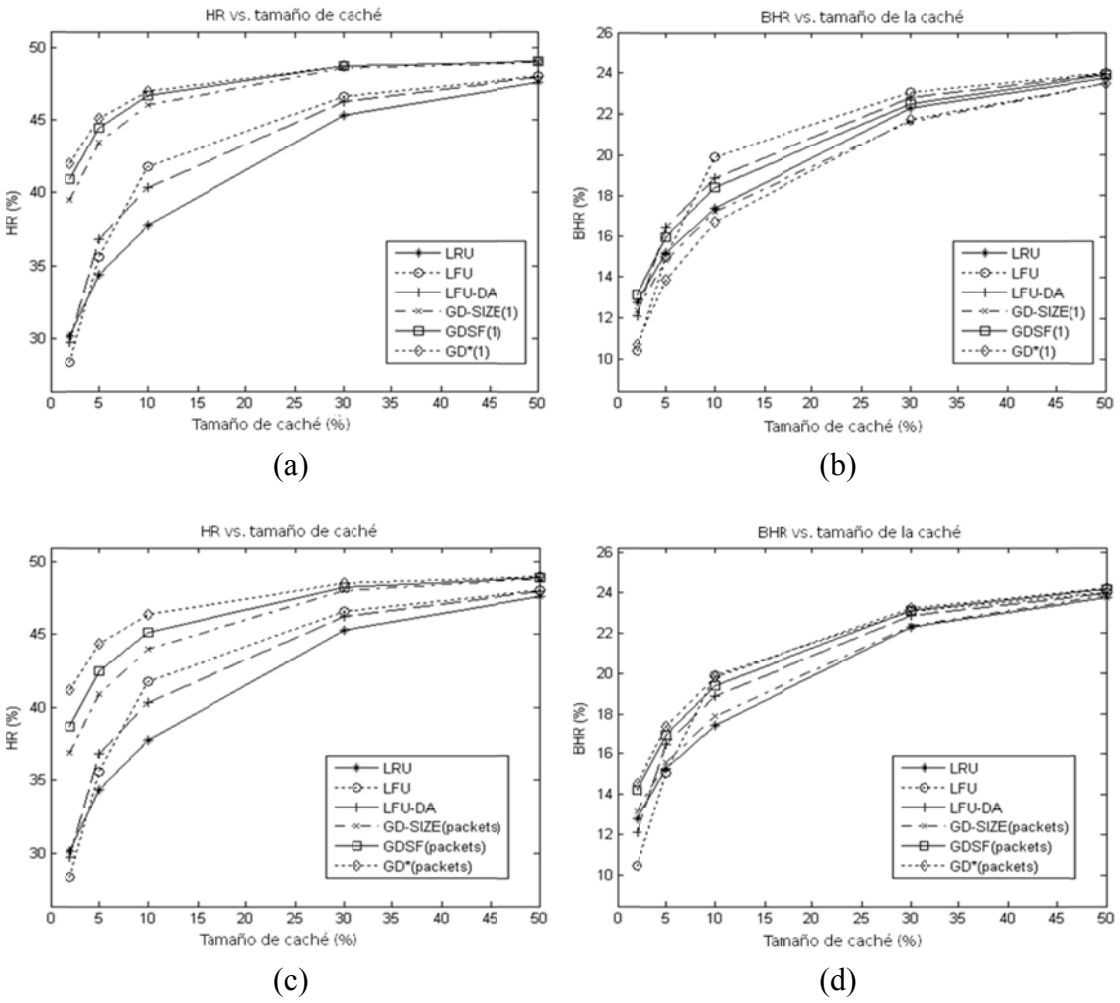


Figura 2.12. Evaluación de políticas de reemplazo para el tipo de documento *Image* empleando una función de coste constante (a) y (b) y coste *packets* (c) y (d)

La Figura 2.13 representa la evaluación del rendimiento de las políticas de reemplazo consideradas para el tipo de documento *Text*, mostrando características similares a las obtenidas para el tipo de documento *Image* para el HR. Las políticas de reemplazo GD* y GDSF obtienen los mejores rendimientos para todos los tamaños de caché y modelos de coste considerando el HR. El algoritmo LFU-DA resulta ser la mejor opción para maximizar el BHR, aunque GDSF con el modelo *packets* obtiene resultados similares. Para el tipo de documento *Text*, GD-Size es la peor de las opciones para ambas métricas. El comportamiento anómalo que presenta el BHR para algunas de las políticas de reemplazo en cachés pequeñas se puede justificar en la presencia de un documento de tipo *Text* muy grande. Este documento eliminará una gran cantidad de documentos en la caché, por lo que el porcentaje de BHR disminuirá, mientras que, para una caché pequeña (del 2% del total de la muestra), el documento no llega a entrar en la caché al ser más grande que la misma caché.

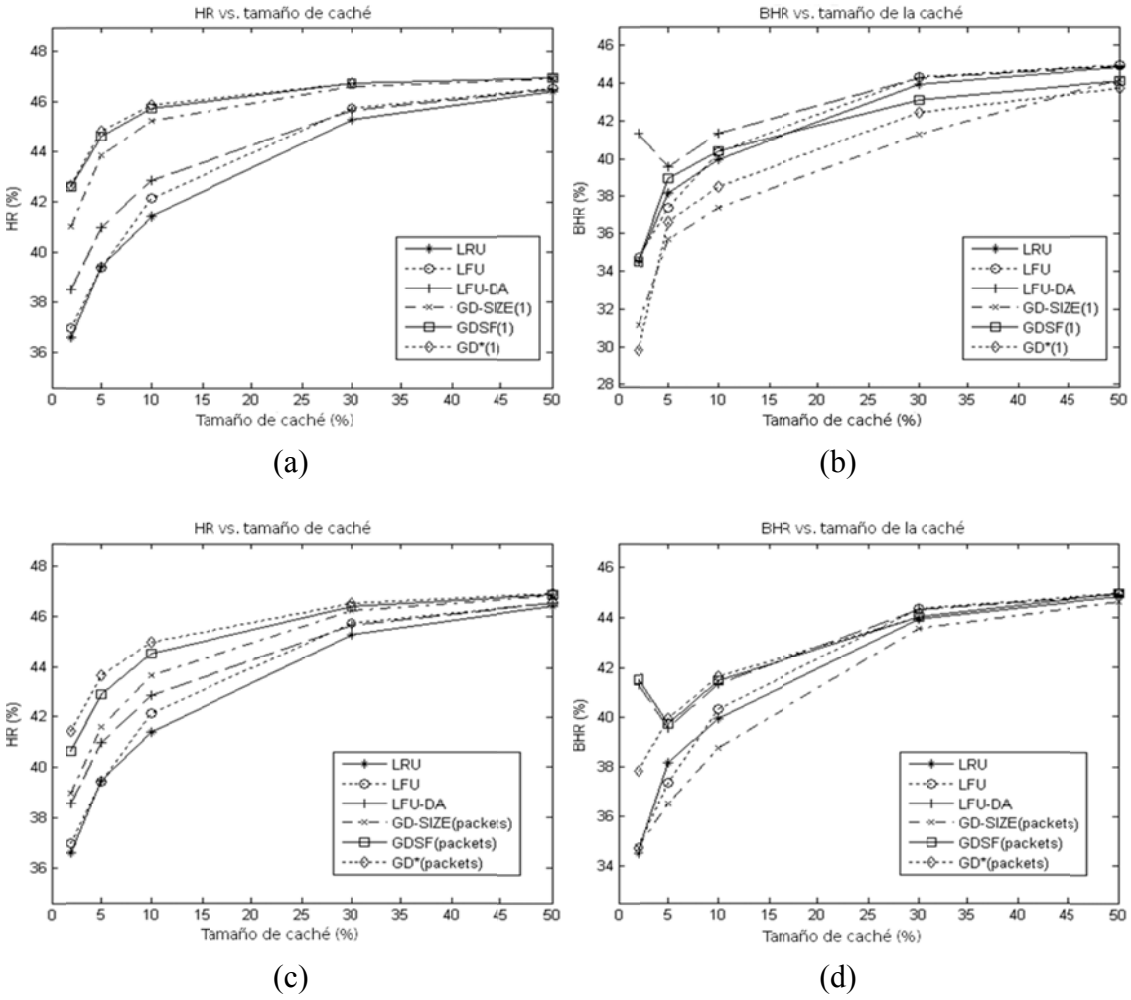


Figura 2.13. Evaluación de políticas de reemplazo para el tipo de documento *Text* empleando una función de coste constante (a) y (b) y coste *packets* (c) y (d)

Finalmente, la Figura 2.14 ilustra la evaluación del rendimiento de las políticas de reemplazo para el tipo de documento *Video*.

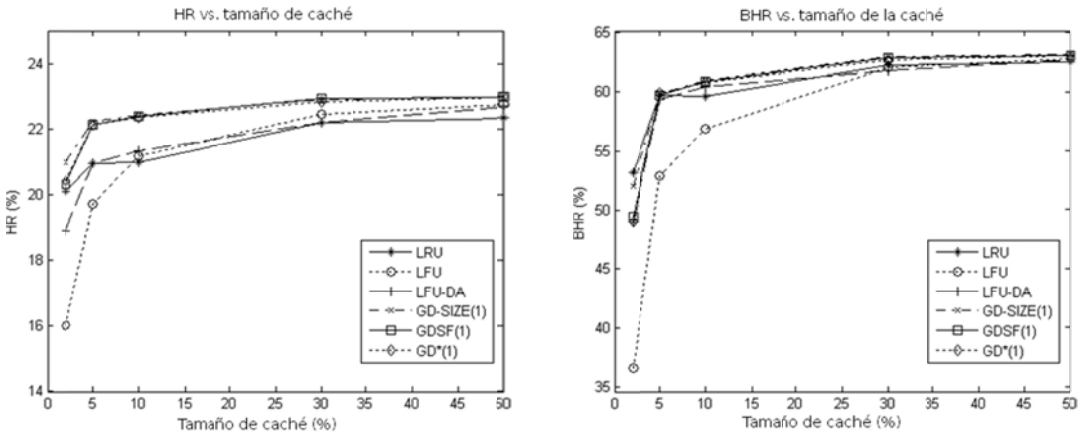


Figura 2.14. Evaluación de políticas de reemplazo para el tipo de documento *Video* empleando una función de coste constante. HR (a) y BHR (b)

Únicamente se muestran los resultados del modelo de coste constante ya que ambos modelos obtienen el mismo rendimiento. La figura muestra que, excepto LFU, el resto de políticas de reemplazo presenta un comportamiento similar para ambas métricas y modelos de coste, de modo que cualquiera de ellas puede ser una buena elección. Existen pequeñas diferencias entre políticas de reemplazo debido a que el porcentaje de referencias es sólo del 0.12% y la relación entre el tamaño de los documentos y el número de referencias es muy disperso, por los que las políticas basadas en coste no obtienen resultados óptimos.

2.5.2.3 Conclusiones

En el presente apartado se ha evaluado y comparado el rendimiento de varias políticas de reemplazo de cuyos resultados se puede concluir:

- Políticas de reemplazo aleatorias: Se ha evaluado el rendimiento de las políticas de reemplazo aleatorias Random, Harmonic, LRU-C, LRU-S y RRGVF y se han comparado entre ellas y con la política LRU. Las simulaciones basadas en la larga muestra de tráfico utilizada han mostrado que, para maximizar el HR, la política de reemplazo Harmonic es la mejor opción, aunque, por el contrario, obtiene un pobre BHR. Teniendo en cuenta el BHR, RRGVF consigue el mejor rendimiento para todos los tamaños de caché. Ya que esta política obtiene también un buen HR, sería adecuado para maximizar ambas métricas. Por otro lado, RRGVF obtiene un mejor rendimiento que LRU para ambas métricas, por lo que resultaría una buena opción para usar dicha política en lugar de LRU en los *proxy*. Por otro lado, si el objetivo es maximizar el HR, Harmonic obtiene un rendimiento mucho más óptimo que el resto conforme se va decrementando el tamaño de la caché.
- Políticas de reemplazo en función del tipo de documento: Se han evaluado seis políticas de reemplazo, tres políticas clásicas (LRU, LFU y LFU-DA) y tres políticas basadas en el tamaño y coste de los documentos especialmente diseñadas para la Web (GD-Size, GDSF y GD*), usando dos modelos de coste (coste constante y coste en función del número de paquetes necesarios para transmitir los documentos). Se ha realizado un estudio del rendimiento de dichas políticas de reemplazo para cada uno de los tipos de documentos presentes en la Web. De este estudio se puede concluir para cada tipo de documento:
    - Tipo *Application*: Las tres políticas de reemplazo basadas en coste usando el modelo de coste constante maximizan el HR. Por otro lado, GDSF(p) y GD*(p) maximizan el BHR.
    - Tipo *Audio*: Considerando el HR, GD-Size(1), GDSF(1) y GD*(1) obtienen los mejores rendimientos. Para maximizar el BHR, todas las políticas de reemplazo ofrecen el mismo rendimiento, excepto LFU que tienen un mayor rendimiento en cachés pequeñas.

- Tipo *Images*: GDSF(1) y GD*(1) superan al resto de políticas para el HR. Para maximizar el BHR, GDSF(p) y GD*(p) son la mejor opción para cachés pequeñas y LFU para las grandes.
- Tipo *Text*: GDSF(p) maximiza tanto el HR como el BHR.
- Tipo *Video*: Todas las políticas de reemplazo, excepto LFU, obtienen los mismos resultados tanto para el HR como para el BHR.

Como puede extraerse del anterior resumen, no existe una política de reemplazo que supere a las demás para todos los tipos de documentos. Por lo tanto, para desarrollar una caché que distinga el tipo de los documentos, se debería aplicar la mejor política de reemplazo para cada uno de ellos. Dicha elección estará supeditada a la métrica que se desee maximizar, ya que, excepto para los tipos de documentos *Text* y *Video*, la mejor política de reemplazo difiere en función de la métrica.

2.5.3 Métricas de rendimiento en cachés con control de admisión

Las clásicas métricas de rendimiento de políticas de reemplazo en cachés HR y BHR no están adaptadas para la evaluación del rendimiento de una caché con política de control de acceso, ya que tienen en cuenta todas las peticiones que llegan al *proxy*. Por lo tanto, aquellas peticiones cuyos documentos no han sido aceptados por la política de acceso también son contabilizadas, con lo que el rendimiento se reduce al no producirse ningún acierto. Sin embargo, el correcto rechazo de estas peticiones no debería reducir el rendimiento de la caché, sino todo lo contrario, ya que está siendo capaz de rechazar documentos que no van a causar aciertos.

2.5.3.1 Métricas propuestas

Para solucionar esta deficiencia, se proponen dos métricas para evaluar el rendimiento de una caché que tiene implementada una política de control de acceso. Se define el NUHR (*Not Unique Hit Ratio*) y NUBHR (*Not Unique Byte Hit Ratio*) tal y como se muestra en las siguientes ecuaciones:

$$NUHR = \frac{\sum_{i \in R} h_i}{\sum_{i \in R} f_i - \sum_{i \in R} R\_ok_i} \tag{2.83}$$

$$NUBHR = \frac{\sum_{i \in R} s_i \cdot h_i}{\sum_{i \in R} s_i \cdot f_i - \sum_{i \in R} s_i \cdot R\_ok_i} \tag{2.84}$$

Donde h_i, f_i, s_i y R se definen como se hizo en el apartado 2.3.1 de métricas de rendimiento y R_ok_i es la cantidad de veces que el documento i ha sido rechazado correctamente por la política de control de acceso. Para el caso de que no se utilice una política de control de acceso, el término R_ok_i es cero, por lo que NUHR y NUBHR se reducen a HR y BHR.

La dificultad de esta métrica radica en decidir cuándo se considera que un documento ha sido correctamente rechazado. Se propone considerar que un documento es correctamente rechazado cuando no vuelve a aparecer en la muestra de tráfico, es decir, cuando el documento rechazado es un *one-timer*. Por tanto, estas métricas miden el rendimiento de la caché con respecto a los documentos no únicos (aquéllos que aparecen más de una vez en la muestra).

Por otro lado, no existen métricas para medir el rendimiento de las políticas de control de acceso. Por lo tanto, se propone, por un lado, una métrica que mida la proporción de documentos correctamente rechazados y, por otra, la proporción de documentos correctamente aceptados. Como criterio para considerar los documentos correctamente rechazados se plantea seguir la propuesta realizada para las métricas NUHR y NUBHR, mientras que se consideran documentos correctamente aceptados aquellos para los que la política de control de acceso ha permitido el acceso en la caché de forma que, posteriormente, el documento ha sido referenciado al menos otra vez mientras se encontraba en la caché. Es decir, se consideran aquellos documentos que han ocasionado al menos un acierto en caché.

Se proponen, por tanto, las métricas ACHR (*Access Control Hit Ratio*) y ACBHR (*Access Control Byte Hit Ratio*) que se definen en las siguientes ecuaciones:

$$ACHR = \sqrt{\frac{\sum_{i \in RJ} R\_ok_i}{\sum_{i \in RJ} f_i} \cdot \frac{\sum_{i \in AC} Ac\_ok_i}{\sum_{i \in AC} f_i}} \tag{2.85}$$

$$ACBHR = \sqrt{\frac{\sum_{i \in RJ} s_i \cdot R\_ok_i}{\sum_{i \in RJ} s_i \cdot f_i} \cdot \frac{\sum_{i \in AC} s_i \cdot Ac\_ok_i}{\sum_{i \in AC} s_i \cdot f_i}} \tag{2.86}$$

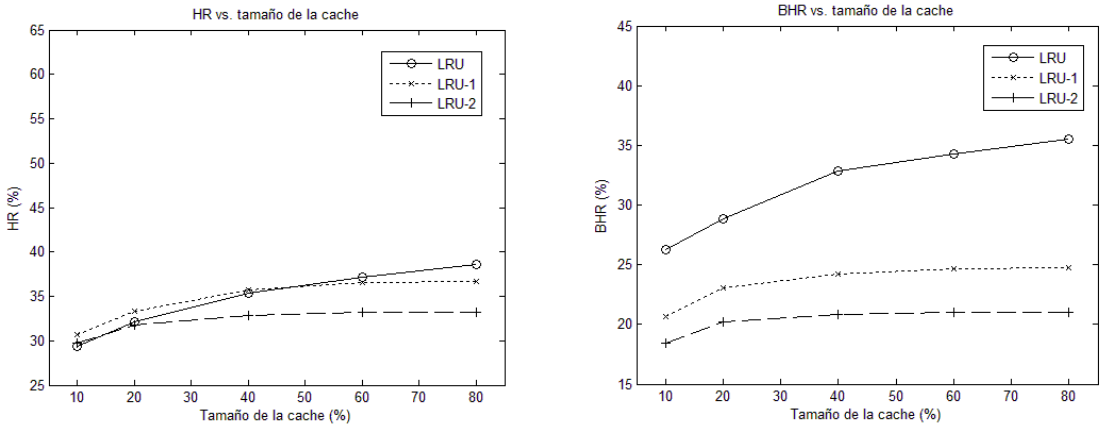
Donde $R = RJ \cup AC$, siendo RJ el conjunto de peticiones rechazadas por la política de control de acceso y AC el conjunto de peticiones aceptadas. Ac_ok_i indica el número de veces que el documento i ha sido aceptado de forma correcta, ya que ha causado al menos un acierto, en la caché. Las métricas ACHR y ACBHR realizan una ponderación entre los documentos correctamente rechazados (primer término de cada ecuación) y los correctamente aceptados (segundo término de cada ecuación).

2.5.3.1 Evaluación de políticas de control de admisión usando las métricas propuestas

Para evaluar la bondad de las métricas propuestas se han realizado una serie de simulaciones de una caché cuya política de control de acceso ha sido no permitir el acceso de los documentos a la caché hasta que éstos no han sido referenciados un número determinado de veces (en concreto, una y dos referencias, es decir, que el documento únicamente es almacenado en la caché a partir de que ha sido referenciado dos o tres veces respectivamente). La política de reemplazo utilizada ha sido LRU. Como tamaños de caché se han considerado el 10%, 20%, 40%, 60% y 80% del volumen máximo alcanzado de una hipotética caché infinita en el que se almacenaran todos los documentos de la muestra que

son referenciados al menos dos veces. Además, se ha usado el 10% de la muestra para "calentar" la caché.

La Figura 2.15 representa las medidas de HR y BHR de una caché que no implementa ninguna política de acceso (LRU), que implementa la política de no permitir el acceso a los documentos en la primera referencia (LRU-1) y en la segunda referencia (LRU-2).

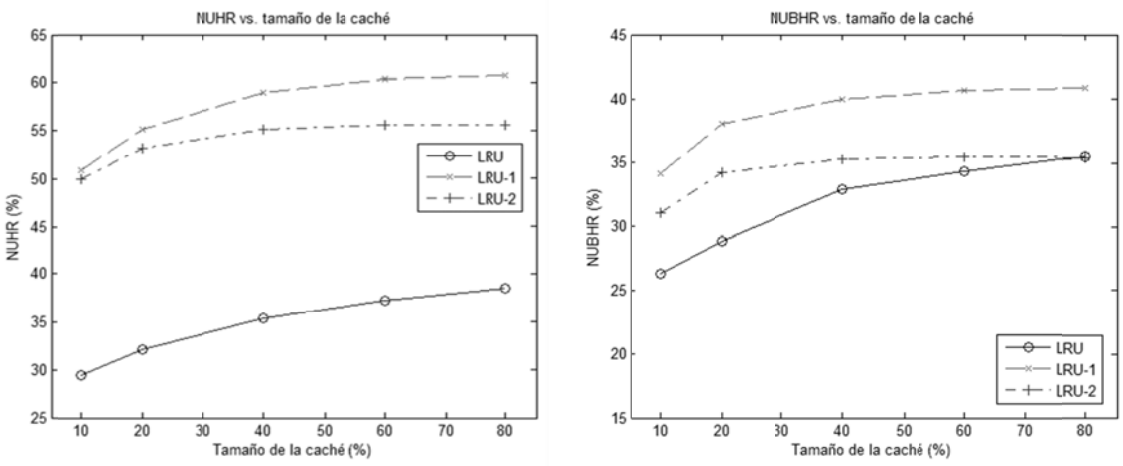


(a) (b)

Figura 2.15. HR (a) y BHR (b) para una caché con y sin usar política de admisión

Se observa que el HR se comporta mejor con tamaños de caché pequeñas cuando se usa una política de control de acceso, pero este rendimiento se va deteriorando conforme se incrementa el tamaño de la caché. Sin embargo, para el BHR el rendimiento es claramente superior si no se utiliza la política de control de acceso. Este comportamiento, que nos llevaría a pensar que es mejor no usar una política de control de acceso, no es realista debido a que estas métricas se ven influenciadas por el rechazo de documentos, incluso cuando el rechazo es adecuado.

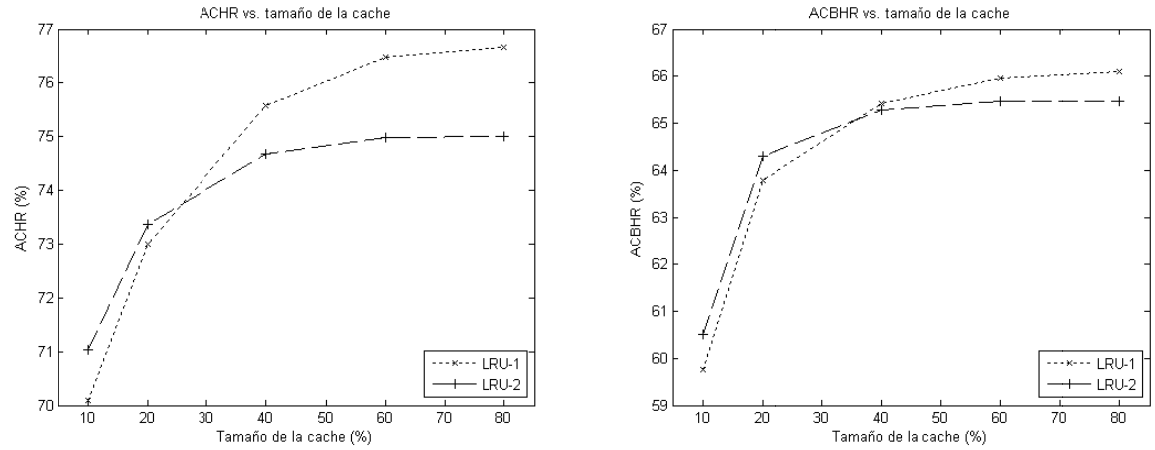
En la Figura 2.16 se ha representado el NUHR y el NUBHR, donde se observa que realmente el rendimiento es superior al obtenido si no se utiliza la política de control de acceso (hay que tener en cuenta que el NUHR y el NUBHR cuando no se utiliza política de control de acceso equivalen al HR y BHR). Esta diferencia es más notoria en el caso del NUHR, llegando a haber una diferencia de rendimiento de hasta un 20%, mientras que para el NUBHR la diferencia no es tan grande. De este hecho, se deduce que las métricas propuestas son más adecuadas para evaluar el rendimiento de una caché que implementa alguna política de control de acceso, ya que no penalizan el no introducir un documento en la caché a no ser que dicho documento realmente sea útil para la misma. También se observa, a partir de la figura, que el rendimiento obtenido es mejor cuando se rechazan los documentos la primera vez que son solicitados que si se hace las dos primeras veces.



(a) (b)

Figura 2.16. NUHR (a) y NUBHR (b) para una caché con y sin usar política de admisión

Por último, la Figura 2.17 representa la medida de las métricas ACHR y el ACBHR que se obtiene para cada una de las políticas de control de acceso consideradas. Teniendo en cuenta la métrica ACHR, LRU-2 es mejor para cachés pequeñas, mientras que para cachés grandes es mejor LRU-1, además de que la diferencia de rendimiento se hace cada vez más notoria conforme se aumenta el tamaño de la caché. Para el ACBHR el comportamiento es similar, aunque la diferencia de rendimiento es menos importante.

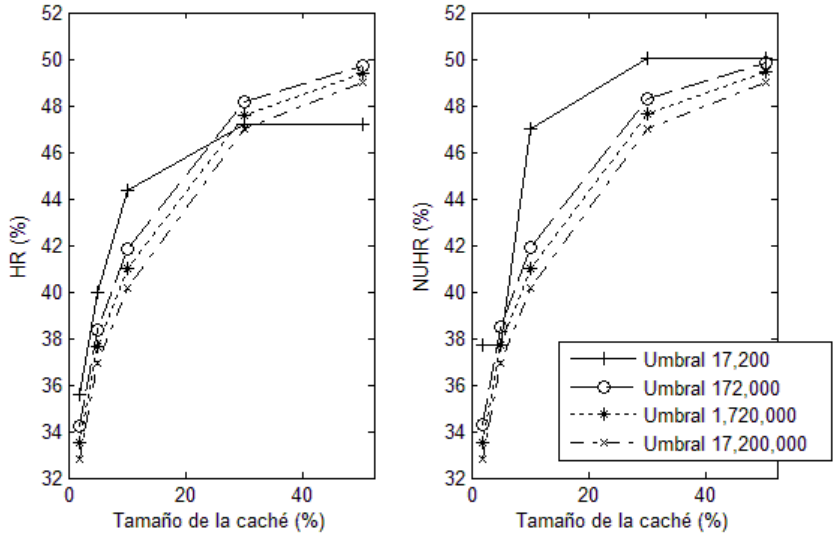


(a) (b)

Figura 2.17. ACHR (a) y ACBHR (b) para las políticas de admisión consideradas

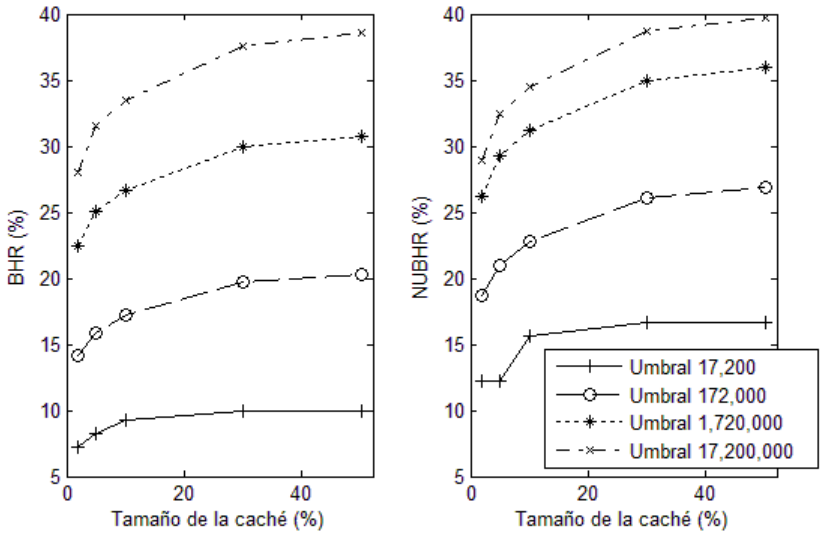
Como segundo experimento se implementó una política de control de acceso que no permite almacenar en la caché aquellos documentos cuyo tamaño es mayor que un determinado valor umbral (*Threshold)*. Los tamaños elegidos son los cuatro primeros múltiplos de diez de la mediana del tamaño de los documentos. Como en el experimento anterior, se ha seleccionado la política de reemplazo LRU y los tamaños de caché elegidos son el 2%, 5%, 10%, 30% y 50% del tamaño de caché infinita.

La Figuras 2.18 y 2.19 comparan el HR y BHR con el NUHR y NUBHR, respectivamente, para las cuatro políticas de acceso seleccionadas.



(a) (b)

Figura 2.18. HR (a) y NUHR (b) para las políticas de admisión consideradas

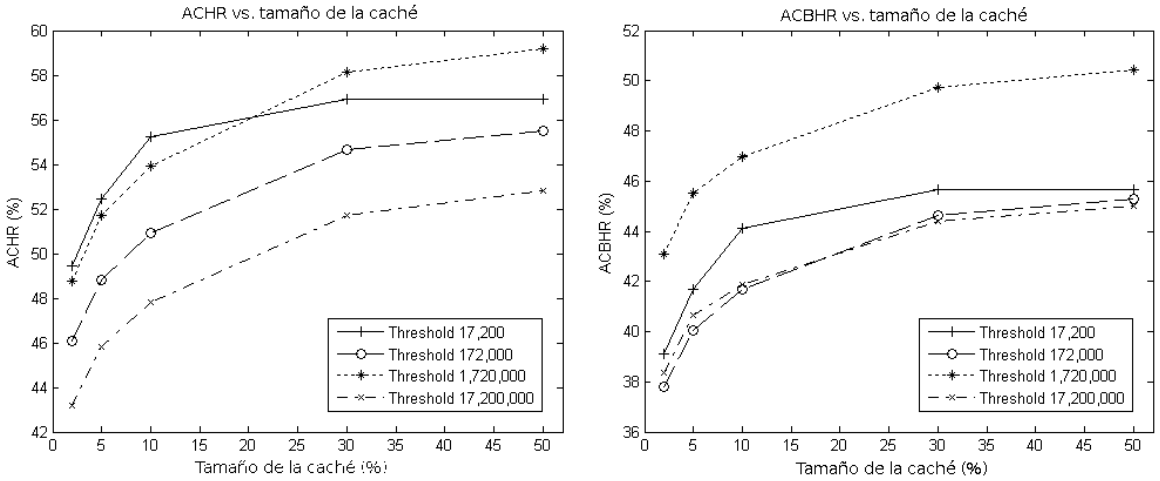


(a) (b)

Figura 2.19. BHR (a) y NUBHR (b) para las políticas de admisión consideradas

Como puede observarse, la diferencia entre el HR y el NUHR es insignificante excepto para los umbrales más restrictivos. De cualquier forma, las diferencias son más evidentes entre el BHR y el NUBHR y esta diferencia se incrementa conforme se reduce el umbral seleccionado. El principal inconveniente es que rechazar los documentos cuyo tamaño es mayor que un determinado valor umbral reduce drásticamente el BHR obtenido. Esto es debido a que el BHR no es una métrica adecuada para una caché con una política de control de acceso. Las métricas NUHR y NUBHR propuestas son, por lo tanto, más adecuadas para dicho tipo de cachés y revelan que rechazar algunos documentos no reduce el rendimiento de la misma tal y como había sido demostrado.

La Figura 2.20 ilustra el ACHR y el ACBHR obtenido por las políticas de control de acceso estudiadas.



(a) (b)

Figura 2.20. ACHR (a) y ACBHR (b) para las políticas de admisión consideradas

Estas figuras evidencian que la mejor opción es aplicar un umbral de 1720000 bytes (10000 veces al valor de la mediana) como tamaño máximo de los documentos a admitir en la caché, especialmente para la métrica ACBHR, aunque el ACHR obtenido es similar al umbral de 17200 bytes. Este efecto se debe a que, como se observa en la Figura 2.19, un tamaño umbral de 17200000 bytes sólo rechaza unos pocos documentos que son referenciados una vez, por lo que la tasa de rechazo correcto se decrementará. Alternativamente, el umbral de 172000 bytes o menos rechaza muchos documentos que son referenciados más de una vez y, por lo tanto, la tasa de aceptaciones correctas se decrementará. Como consecuencia, las métricas propuestas ofrecen una medida compensada de los dos principales propósitos de una política de control de acceso: maximizar la tasa de rechazos correctos y maximizar la tasa de admisiones correctas.

2.5.3.3 Conclusiones

En el presente apartado se han propuesto dos nuevas métricas (NUHR y NUBHR) para evaluar el rendimiento de una caché que implementa una política de control de acceso, ya que las métricas clásicas, HR y BHR, no tienen en cuenta estas políticas, con lo que se obtienen rendimientos que no se corresponden con la realidad. También se han propuesto las métricas ACHR y ACBHR para evaluar el rendimiento de las propias políticas de control de acceso.

Mediante simulaciones y usando una muestra de tráfico real, se ha medido el rendimiento de una caché que no implementa ninguna política de acceso y se ha comparado con una caché que sí las implementa. Por un lado se han evaluado las políticas de control de admisión que rechazan guardar en caché tras el primer acceso a un documento y también tras el segundo, y, por otro, la aplicación de un umbral del tamaño

máximo que pueden tener los documentos que se almacenan en la caché. Se ha comparado el rendimiento obtenido con las métricas clásicas y con las propuestas en este trabajo, demostrándose que las métricas propuestas ofrecen un rendimiento más realista que el que se obtiene con las métricas clásicas. También se han evaluado las políticas de control de acceso usando las métricas propuestas.

Capítulo 3

Mecanismos de caché para la optimización de redes ad hoc

El presente capítulo se estructura como sigue. En la sección 3.1 se estudia el funcionamiento de las redes ad hoc comentando los diferentes protocolos de encaminamiento y modelos de movilidad usualmente empleados. La sección 3.2 repasa los diferentes esquemas de caché propuestos en la literatura, realizando una clasificación de los mismos, además de enumerar las métricas de rendimiento comúnmente usadas para evaluarlas. La sección 3.3 repasa el estado de la técnica de los parámetros de simulación empleados para evaluar los esquemas de caché propuestos en la literatura. En la sección 3.4 se propone el esquema de caché CLIR, evaluando su rendimiento en redes móviles en la sección 3.5 y en redes estáticas en la sección 3.6.

3.1 Encaminamiento y modelos de movilidad en redes ad hoc

Las tecnologías móviles inalámbricas están cada vez más presentes en nuestra vida diaria en forma de teléfonos móviles, PDAs (*Personal Digital Assistants*), *tablets* PC y ordenadores portátiles que incorporan uno o más interfaces radio. En algunos casos, estos dispositivos funcionan de forma autónoma sin estar conectados a ningún operador de comunicaciones, aunque en ocasiones pueden llegar a construir su propia red de forma que puedan comunicarse entre ellos. En el caso particular de que no sea posible establecer una conexión directa entre los dispositivos, los nodos deben formar una Red Móvil Ad Hoc (MANET – *Mobile Ad Hoc Network*). Una característica fundamental de las redes MANET es la autonomía con la que se espera que funcionen. Sin embargo, esta autonomía se deriva de la cooperación de los nodos, tal y como ocurre en los protocolos de encaminamiento. En una MANET los paquetes deben ser retransmitidos por los nodos intermedios entre el nodo origen y el destino para asegurar que son recibidos por el dispositivo destinatario. Los paquetes, por tanto, saltan de un dispositivo a otro formando una red multisalto que es gestionada por el protocolo de encaminamiento ad hoc.

Se han propuesto numerosos protocolos de encaminamiento en redes MANET, que pueden clasificarse en tres grupos atendiendo al mecanismo que utilizan para actualizar la información de encaminamiento:

- **Proactivos o basados en tablas de encaminamiento**. Este tipo de protocolos de encaminamiento mantienen información renovada acerca de la topología que tiene la red. Para ello, los dispositivos de la red envían de forma periódica información a los demás para que mantengan actualizadas las rutas entre todos los nodos y, de esa forma, puedan encaminar los paquetes adecuadamente. En esta categoría se pueden mencionar los protocolos de encaminamiento DSDV (*Destination-Sequenced Distance-Vector Routing Protocol*) [Perkins,1994], OLSR (*Optimized Link State Routing Protocol*) [Clausen,2003], CGSR (*Cluster-head Gateway Switch Routing Protocol*) [Chiang,1997] y WRP (*Wireless Routing Protocol*) [Murthy,1996]. Este tipo de protocolos de encaminamiento resultan eficaces en entornos con una movilidad media de los nodos [Ariza,2009], ya que la topología de la red no cambia excesivamente. Sin embargo, en entornos de muy baja movilidad en la que las rutas se mantienen inalteradas durante largos periodos de tiempo, el envío periódico de mensajes de actualización de rutas repercute negativamente en el rendimiento de la red, ya que se genera un tráfico innecesario.
- **Reactivos o de encaminamiento bajo demanda**. En oposición a los protocolos de encaminamiento proactivos basados en tablas, en los protocolos de encaminamiento reactivos las rutas son creadas sólo cuando se desea enviar un paquete de datos. Cuando un dispositivo de la red requiere una ruta hacia el destino de su envío, se inicia un proceso de descubrimiento de ruta que termina cuando se encuentra dicha ruta o cuando se examinan todas las alternativas y ninguna ha conseguido su destino. Como protocolos más populares de encaminamiento pertenecientes a esta categoría se puede mencionar AODV (*Ad Hoc On-Demand Distance Vector Routing Protocol*) [Perkins,2003], TORA *(Temporary Ordered Routing Algorithm*) [Park,2001], ABR *(Associative-Based Routing Protocol*) [Toh,1996] y SSR (*Signal Stability Routing Protocol*) [Dube,1997].
- **Híbridos**. Combinan las mejores características de los protocolos de encaminamiento proactivos y reactivos. Para ello definen una zona de cobertura del dispositivo en la que se mantiene información de forma proactiva de los dispositivos que se encuentren a menos un número determinado de saltos. Para aquellos dispositivos que se encuentren fuera del área de cobertura se utiliza una aproximación reactiva que permite encontrar las rutas hacia ellos. Como ejemplo de protocolo de encaminamiento híbrido se encuentra ZRP (*Zone Routing Protocol*) [Haas,2002].

Un aspecto importante a tener en cuenta en una red móvil ad hoc es cómo se mueven los nodos en la red. Para imitar y reproducir en simulaciones el movimiento de los

dispositivos en las redes MANET se han propuesto numerosos modelos de movilidad en los últimos años. Estos modelos de movilidad se pueden categorizar en tres grupos:

- **Modelos aleatorios sin restricción**. En este tipo de modelos de movilidad, el punto de destino en el movimiento de los dispositivos se decide de forma aleatoria siguiendo algún tipo de métrica aleatoria o heurísticos que dependen del modelo de movilidad en concreto. Los modelos de movilidad de este tipo más usados son: RWP (*Random Way Point*) [Johnson,1996] que simplemente selecciona un destino aleatorio dentro del área de simulación para, una vez que se ha alcanzado y pasado un tiempo de espera, seleccionar otro; RD (*Random Direction*) [Royer,2001] selecciona la dirección que seguirá el dispositivo hasta que alcance los límites del área de simulación, momento en el que se selecciona otra dirección; los modelos de movilidad MWP (*Markovian Way Point*) [Hyytia,2006] y GM (Gauss-Markov) [Liang,1999] seleccionan el destino usando probabilidades Markovianas entre los puntos de destino.
- **Modelos geográficos**. El próximo destino de los dispositivos se selecciona siguiendo algún tipo de restricción geográfica. En esta categoría cabe destacar: OM (*Obstacle Model*) [Jardosh,2003] que define un conjunto de obstáculos en el área de simulación que deben evitarse; *FreeWay* y *ManhattanGrid* [Bai,2003] que limitan la movilidad de los dispositivos a una serie caminos, normalmente paralelos u ortogonales entre sí, definidos dentro del área de simulación.
- **Modelos grupales**. Estos modelos de movilidad intentan imitan el movimiento en grupos de humanos o vehículos tripulados. Como ejemplos, RPGM (*Reference Point Group Mobility*) [Hong,1999] incluye la posibilidad de formar grupos dinámicos de dispositivos con un líder que decide el próximo destino que debe alcanzar todo el grupo; el modelo DartMouth [Kim,2006] elige el destino dentro del área de simulación de acuerdo a una base de datos real de comportamientos humanos; CMM (*Clustered Mobility Model*) [Lim,2006b] divide el área de simulación en *clusters* a los que están asignados los dispositivos de la red, aunque estos pueden cambiar de *cluster* dependiendo del número de dispositivos en cada *cluster*; ORBIT [Ghosh,2007] define un conjunto aleatorio de *clusters* y los dispositivos se asignan a ellos moviéndose únicamente entre los *clusters* asignados; SLAW (*Self-similar Least Action Walk*) [Lee,2009] representa contextos sociales presentes entre la gente que comparte intereses comunes usando puntos de referencia fractales; TVCM (*Time-Variant Community Model*) [Hsu,2007] es un modelo realista obtenido de muestras reales de movimientos de personas en campus universitarios y edificios empresariales. TVCM considera dos suposiciones en su modelo: primero, el modelo supone que existen localizaciones "populares", denominadas comunidades, a las que los usuarios se mueven con mayor preferencia; además, TVCM considera la reaparición periódica de nodos. Teniendo en cuenta estas dos características, TVCM propone un modelo matemático que incorpora un comportamiento no homogéneo tanto en el

espacio como en el tiempo. La periodicidad en el tiempo se controla mediante un parámetro llamado *period duration*, mientras que el parámetro *average length* define el valor medio de una distribución exponencial que caracteriza la distancia que un dispositivo se mueve dentro de la comunidad.

Como puede observarse, cada modelo de movilidad intenta reproducir algunas características del movimiento de los dispositivos móviles, aunque ninguno de ellos sea lo suficientemente general para ser considerado el modelo de movilidad más adecuado para todas las circunstancias. Concretamente, modelos de movilidad como FreeWay y ManhattanGrid son adecuados para redes vehiculares, mientras que ORBIT, SLAW o TVCM están adaptados para movilidad de humanos.

3.2 Mecanismos de caché en redes ad hoc

Supongamos una red MANET con conectividad a redes externas como Internet tal y como se muestra en la Figura 3.1. Los dispositivos móviles (nodos móviles) cooperan para encaminar paquetes hacia los puntos de acceso que dan acceso a las redes exteriores. Los nodos móviles solicitan documentos a los servidores de documentos (por ejemplo, servidores Web) que se encuentran en las redes exteriores a la red móvil. Como todo el tráfico en la MANET es encaminado hacia los puntos de acceso, éstos pueden convertirse en un cuello de botella. Además, los puntos de acceso pueden estar temporalmente inaccesibles porque se encuentren fuera del área de cobertura de todos los nodos de la red debido a la movilidad de los mismos. Esta situación causa desconexiones de las redes externas. Los mecanismos de caché reducen el impacto de las desconexiones temporales de los puntos de acceso ya que los nodos cooperan para servir los documentos, que previamente han almacenado en su caché, a otros nodos. Por otro lado, dado que los nodos móviles también tienen la capacidad de servir documentos, el cuello de botella que se produce en los puntos de acceso se reduce ya que parte del tráfico de solicitud y servicio de documentos no les llega.

Los objetivos principales de un esquema de caché en redes inalámbricas pueden resumirse, por tanto, en tres puntos:

- Incrementar la accesibilidad a los documentos, es decir, facilitar que los documentos sean accesibles desde todos los nodos móviles de la red incluso, si fuera posible, en los casos en los que haya desconexiones temporales de los puntos de acceso.
- Reducir la sobrecarga de tráfico que exige el solicitar y servir los documentos. Esta reducción del tráfico puede reflejarse en un menor consumo de los nodos (si disponen de estados de bajo consumo cuando no transmiten ni reciben) y, en general, beneficia al resto de servicios y a las propias prestaciones de la red.
- Reducir el retardo que perciben los usuarios de la red inalámbrica.

Con el ánimo de conseguir dichos objetivos se han propuesto varios esquemas de caché que pasan a estudiarse a continuación.

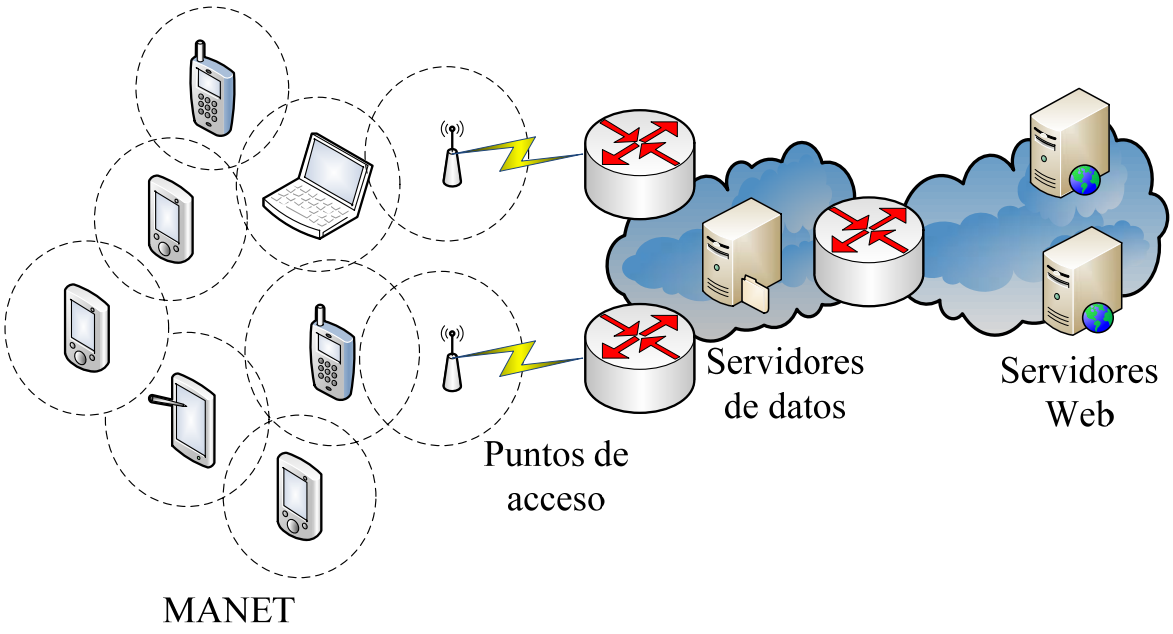


Figura 3.1. Red móvil inalámbrica con acceso a Internet

3.2.1 Estado de la técnica

En el presente apartado se va a realizar una descripción pormenorizada de los esquemas de caché para redes ad hoc que se han propuesto en la literatura. Dado que algunos de los esquemas propuestos no tienen un nombre propio asociado, se ha optado, en esos casos, por nombrarlos usando el apellido del primer autor del artículo en el que se presentaron.

3.2.1.1 Mobile Intercepting Proxy Cache (MobEye)

El esquema de caché MobEye (*Mobile intErcepting proxY cachE*) [Dodero,2006] propone implementar el algoritmo de reemplazo LRU en la caché local. Cuando un nodo solicita un documento usa un mensaje similar al del protocolo ICP (*Internet Caching Protocol*) en modo *broadcast*. Conforme el mensaje de petición se difunde por la red, los nodos que tienen una copia del documento solicitado en su caché local responden con un mensaje de confirmación (ACK). Cuando el nodo que pidió el documento recibe el primer mensaje ACK éste cursa la petición del documento en modo *unicast* hacia el nodo que generó el ACK.

3.2.1.2 Esquema de Sailhan

El esquema de caché propuesto por Sailhan en [Sailhan,2002] y [Sailhan,2003] cursa las peticiones de documentos directamente al servidor si éste se encuentra en el radio

de cobertura del nodo solicitante y el nodo no tiene una copia válida en su caché local. En otro caso el documento se solicita, mediante un mensaje en modo *broadcast* de un salto, a los nodos vecinos. Si ningún nodo en el vecindario responde a la solicitud del documento, el nodo pide el documento a aquellos nodos que están más cerca que el servidor. Para conocer los nodos a los que enviar esta solicitud, el nodo solicitante usa la función *F* definida como:

$$F_i = terminal\_profile_i / hops_i \tag{3.1}$$

donde $terminal\_profile_i$ es un número entero que indica la cantidad de aciertos en las peticiones que ha obtenido el nodo *i* y $hops_i$ es la distancia en saltos a la que se encuentra dicho nodo. Cada nodo tiene una función *F* definida para cada uno de los nodos que están en su área de cobertura. Los nodos que reciben esta solicitud y tienen una copia del documento en su caché local responden al nodo solicitante mediante un mensaje que incluye el TTL (*Time To Live* – Tiempo de vida) del documento y la capacidad de la batería del nodo (siendo la capacidad de la batería un valor real normalizado entre cero y uno). De entre los nodos que responden provocando un "acierto", el nodo cursa la petición hacia el nodo que maximiza la función *R*:

$$R = Capacity \cdot (\lambda \cdot TTL + \mu \cdot hops) \tag{3.2}$$

donde *Capacity* es la capacidad de la batería del nodo tal y como se indicó anteriormente, TTL es el tiempo de vida del documento, *hops* es el número de saltos a los que se encuentra el nodo que respondió mientras que los valores λ y μ son constantes predefinidas. Además de esta operación, el nodo incrementa el parámetro $terminal\_profile_i$ de cada uno de los nodos que respondieron a la petición indicando tener una copia del documento. Si no hay ningún nodo que responda mediante este método, el documento se pide directamente al servidor.

Como política de reemplazo para la caché local se propone asignar un peso *W* a cada documento almacenado según la ecuación:

$$W_i = \alpha \cdot Popularity_i + \beta \cdot AccessCost_i + \gamma \cdot Coherency_i + \delta \cdot size_i \tag{3.3}$$

donde $Popularity_i$ indica la popularidad del documento *i* y se define como el número de veces que se ha solicitado desde que se almacenó en la caché; $AccessCost_i$ es una estimación del coste de energía asociado a la obtención del documento *i* de forma remota si fuera eliminado de la caché; $Coherency_i$ es el TTL del documento *i* y $size_i$ es su tamaño. Finalmente, α, β, γ y δ son constantes predefinidas para dar más o menos peso a cada uno de los criterios. En el caso de tener llena la caché y requerir espacio para un nuevo documento se eliminan primero de la caché los documentos con un menor valor de *W*.

El esquema de caché propuesto no especifica el valor de ninguna de las constantes que define.

3.2.1.3 Zone Cooperative Caching (ZC)

El esquema de caché Zone Cooperative Caching (ZC) [Chand,2006] propone, cuando un nodo necesita un documento, mirar primero en la caché local. Si el nodo no tuviera una copia válida del mismo, se realiza una petición *broadcast* a sus nodos vecinos (es decir, a un salto) para ver si ellos lo tienen en sus cachés. Si alguno lo tiene, responde con un mensaje de confirmación *ACK*. Tras recibir este *ACK*, el nodo origen de la petición envía un mensaje *CONFIRM* al nodo que respondió que tiene el documento. Una vez recibido el mensaje *CONFIRM* el nodo envía el documento al nodo que lo solicitó. Si tras un periodo de tiempo definido el nodo que hace la petición no recibe ningún mensaje *ACK*, se manda la petición al servidor. Por el camino hacia el servidor, cada nodo que recibe la petición comprueba de nuevo si algún nodo que se encuentre en su vecindario tiene una copia del documento haciendo un *broadcast* a un salto.

Se propone una política de reemplazo para la caché local basada en una valoración de cada documento *i* de forma que se eliminan primero aquellos documentos con menor valor de cierto parámetro asociado VALUE:

$$VALUE_i = w_1 A_i^{new} + w_2 \delta_i + w_3 TTL_i + w_4 / s_i \tag{3.4}$$

siendo:

$$A_i^{new} = \alpha \frac{1}{t_c - t_1} + (1-\alpha) A_i^{old} \tag{3.5}$$

donde t_c es el instante de tiempo actual, t_1 es el instante de tiempo del último acceso a *i*, TTL_i es el tiempo de vida del documento *i*, δ_i la distancia en saltos al nodo que lo sirvió y $size_i$ su tamaño. Los parámetros α, *w1*, *w2*, *w3* y *w4* son constantes de peso con valores en el intervalo [0,1], cumpliéndose que $\sum_{j=1}^{4} w_j = 1$. Los autores proponen asignar un valor de 0.25 a cada factor de peso w_i. No se especifica el valor de α.

Para evitar que haya copias del mismo documento en nodos contiguos, no se almacenan en caché local los documentos que han sido servidos desde otro nodo que está a un salto.

Posteriormente en [Chand,2007] se propone sustituir la política de reemplazo propuesta por la política LUV (*Least Utility Value*). En esta política de reemplazo se eliminan primero de la caché los documentos con menor valor de utilidad. La utilidad se define para un documento *i* como:

$$utility_i = \frac{A_i \cdot \delta_i \cdot TTL_i}{s_i} \tag{3.6}$$

siendo:

$$A_i = \frac{a_i}{\sum_{k=1}^{N} a_k} \tag{3.7}$$

con:

$$a_i = \frac{K}{t^c - t_i^K} \tag{3.8}$$

donde A_i representa la popularidad del documento i, K es el número de últimos accesos al documento almacenado (para una ventana deslizante), t^c es el instante de tiempo actual, t_i^K es el tiempo de acceso almacenado más antiguo al documento i, δ_i es la distancia en saltos al nodo que respondió a la petición, TTL_i es el TTL del documento i y s_i es su tamaño.

3.2.1.4 Cluster Cooperative (CC)

El esquema de caché *Cluster Cooperative* (CC) [Chand,2007b] organiza los nodos en la red inalámbrica en *clusters*. Para ello divide el área de trabajo en una rejilla predefinida de forma que los nodos que se encuentran en cada una de las celdas de la rejilla forman un *cluster*. Los nodos conocen dónde se encuentran ubicados dentro del área de simulación gracias a un dispositivo GPS (*Global Positioning System*). Igualmente saben a qué *cluster* pertenecen ya que también conocen la distribución de celdas.

En cada *cluster* se selecciona un nodo CSN (*Cache State Node*) que es el encargado de mantener información sobre los nodos que hay en el *cluster*. Esta información, denominada CCS (*Cluster Cache State*), contiene datos de los documentos que están almacenados en cada uno de los nodos del *cluster*, así como el tamaño disponible en cada una de las cachés.

Cuando un nodo se une a un nuevo *cluster* comprueba la presencia del CSN enviando una petición *broadcast* de un salto con información de su localización. Los nodos vecinos que pertenecen al mismo *cluster* responden con el identificador del *cluster* así como el del nodo CSN. Finalmente, el nuevo nodo envía información sobre el estado de su caché al CSN. Si el nuevo nodo no recibe ninguna respuesta pasado un tiempo predefinido, el nodo se convierte en el CSN, ya que es el primero en entrar en el *cluster*. Para el caso de un nodo que va a salir de un *cluster*, éste informa al CSN enviándole un mensaje para que el CSN actualice la lista de nodos del *cluster* y elimine la información del nodo que va a salir del *cluster*.

Los nodos dentro de un *cluster* envían mensajes de forma periódica al CSN con información del tiempo de renovación de la actividad. Si, transcurrido dicho tiempo de renovación, el nodo no se ha puesto en contacto con el CSN para renovar su estado, el CSN supone que el nodo ha fallado y borra las entradas de información de la caché asociadas al nodo que ha fallado.

Si un nodo que no ejerce el rol del CSN detecta que el CSN no está disponible, inicia una elección de CSN enviando un mensaje de elección por *broadcast* a todos los nodos del *cluster*. Una vez que se elige un nuevo CSN (aunque el autor no especifica qué criterio se usa para dicha elección), éste envía un mensaje *broadcast* al resto de nodos del *cluster* informando de su identificador. Todos los demás nodos envían al nuevo CSN la información sobre sus cachés.

Cuando un nodo necesita un documento, primero comprueba en su caché local si existe una copia válida del mismo. Si no es así, el nodo envía una petición al CSN del *cluster* en el que se encuentra. Si el CSN conoce la ubicación del documento solicitado dentro del mismo *cluster* envía un mensaje *ACK* al nodo solicitante con el identificador del nodo que tiene la copia del documento. El nodo que recibe el mensaje *ACK* remite un mensaje *CONFIRM* al nodo con el identificador recibido solicitándole el documento. Finalmente, el nodo que recibe el mensaje *CONFIRM* responde con el documento solicitado usando un mensaje *REPLY*. Si el CSN no tuviera información acerca de la localización del documento dentro del *cluster*, éste envía el mensaje *ACK* al nodo solicitante adjuntando un identificador nulo. En ese caso, el nodo enviaría la petición hacia el servidor usando la ruta creada para ello. Cuando un nodo localizado en un *cluster* diferente del nodo solicitante recibe la petición, éste comprueba si tiene una copia válida en su caché. Si es así responde con un mensaje *REPLY* adjuntando el documento. Si no tuviera una copia, el nodo envía la petición al CSN de su *cluster* para comprobar si hay una copia almacenada en algún nodo del *cluster*. Si hubiera una copia del documento en el *cluster* se repite el paso de mensajes comentado anteriormente para conseguir el documento. Si no hubiera una copia en el *cluster* actual se envía la petición al siguiente nodo en la ruta hacia el servidor. Este proceso se repite *cluster* a *cluster* hasta que se encuentra una copia del documento o la petición alcanza el servidor.

El esquema propone usar la política de reemplazo LUV-Mi (*Least Utility Value with Migration*). LUV-Mi tiene un funcionamiento similar a LUV (ver apartado 3.2.1.3) salvo en que, cuando hay que eliminar un documento de la caché, el nodo comprueba si hay espacio suficiente en las cachés de los nodos vecinos. Esta comprobación se realiza contactando con el CSN, que es el nodo que mantiene información del estado de las cachés del *cluster*. Si el CSN informa de una caché con espacio suficiente para almacenar el documento, el nodo lo envía al nodo disponible para que lo almacene.

Como política de control de admisión en la caché se propone no almacenar en caché local aquellos documentos que han sido servidos por otro nodo del *cluster*. De esta forma se asegura que sólo habrá una copia de cada documento dentro del *cluster*, aprovechando de esta forma el espacio disponible de almacenamiento.

3.2.1.5 Esquema de Gianuzzi

El esquema de caché propuesto en [Gianuzzi,2003] sugiere dividir las réplicas de los documentos en copias primarias y copias secundarias. Las copias primarias son aquellas que no se pueden borrar de las cachés por ser consideradas importantes. La autora

no especifica cómo se etiquetan cada una de las copias de los documentos en copias primarias o secundarias.

Cuando un nodo necesita un documento primero comprueba en su caché local si existe una copia válida. Si no es así, el nodo cursa la petición directamente al servidor de documentos. Los nodos intermedios en la ruta hacia el servidor pueden responder informando de que tienen una copia del documento solicitado. El nodo solicitante espera a que le llegue la última respuesta (aunque no se especifica cómo se define este tiempo de espera) y cursa la petición al nodo que se encuentre más cerca en número de saltos.

Como política de reemplazo propone usar LFU-Aging. Además, propone que las copias primarias de los documentos nunca sean eliminadas de las cachés.

3.2.1.6 Distributed Greedy Algorithm (DGA)

El esquema de caché DGA (*Distributed Greedy Algorithm*) [Tang,2008] propone que cada nodo de la red mantenga una tabla en la que, para cada documento existente en la red, se almacene la dirección del nodo más cercano que tiene una copia del documento en su caché. También se almacena la dirección del nodo que posee la segunda copia más cercana de cada documento.

Cuando un documento se almacena en la caché local de un nodo, la información del segundo nodo más cercano pasa a ser la del nodo más cercano y el identificador del nodo más cercano pasa a ser la del propio nodo que ha almacenado el documento. Además, el nodo que ha almacenado el documento envía un mensaje *AddCache* en modo *broadcast* a sus vecinos informando de que ha almacenado el documento. Los nodos que reciben el mensaje *AddCache* actualizan la información de la localización de los documentos en sus tablas. Si éstas se modifican, cada nodo vuelve a mandar un mensaje *AddCache* a sus vecinos para que la información de las tablas sea coherente. El mensaje *AddCache* también se envía al servidor de documentos ya que éstos mantienen una lista con los documentos y los nodos donde están almacenadas cada una de las copias.

Cuando un nodo elimina un documento de su caché local, primero actualiza su tabla de nodos más cercanos para dicho documento con el valor del segundo nodo más cercano y elimina el segundo nodo más cercano. El nodo solicita al servidor la lista de los nodos que tienen una copia del documento que ha borrado y envía un mensaje *DeleteCache* a sus vecinos para que actualicen sus tablas. Los nodos que reciben el mensaje *DeleteCache* reenvían el mensaje para que se mantenga la coherencia de las tablas en la red.

Dado lo costoso de mantener la lista de localización de cada una de las copias de los documentos en el servidor, los autores proponen obviar el envío de mensajes *DeleteCache* de forma que, si se recibe una petición de un documento que ya no está almacenada en un nodo, ésta se reenvía al servidor de documentos.

Para redes móviles, en lugar de usar mensajes *AddCache* y *DeleteCache* para cada documento que se almacena o borra de las cachés, se propone que sea el servidor de

documentos el que informe periódicamente de la localización de los documentos a toda la red mediante un mensaje de *broadcast*.

Cuando un nodo necesita un documento se comprueba si existe una entrada para la copia más cercana del documento. Si es así, se envía la petición al nodo más cercano que tiene el documento, que puede ser el mismo nodo solicitante porque tiene una copia en su caché local. En el caso de que no haya información sobre la localización del documento, se envía la petición directamente hacia el servidor.

Como política de reemplazo se propone usar LRU. Como política de control de admisión se propone calcular una estimación del beneficio producido al añadir o eliminar documentos de la caché. El beneficio de almacenar o eliminar el documento *Dj* del nodo *i* se define como:

$$B_{ij} = t_{ij}\delta_j \tag{3.9}$$

donde t_{ij} es la frecuencia de acceso al documento *Dj* observada por el nodo *i* en su tráfico local y δ_j es la distancia en número de saltos entre el nodo *i* y el nodo más cercano o segundo más cercano que tiene una copia del documento. Este cálculo se realiza para el documento *i* a insertar en la caché y para los documentos a los que va a reemplazar en la caché. Si el beneficio de insertar el documento *i* en la caché es superior al de eliminar los ya almacenados, entonces se almacena el documento en caché y se eliminan los documentos necesarios. El beneficio del documento *i* debe, además, superar cierto umbral para que el reemplazo tenga lugar.

3.2.1.7 Index Push (IXP)

El esquema de caché IndeX Push (IXP) [Chiu,2005] [Chiu,2009] propone que cada nodo mantenga una tabla denominada *IV* (*Index Vector*) con una entrada para cada documento que existe en el la red. Cada entrada de la tabla *IV* tiene la siguiente forma:

(x, cached, cachednode, count)

Para el documento con identificador *x* en cada nodo, *cached* indica si el documento *x* se encuentra almacenado en caché local. El campo *cachednode* almacena el identificador del último nodo vecino que se sabe que tiene una copia del documento *x*. Por su parte, *count* indica el número de nodos vecinos que tienen una copia del documento *x*.

Cuando un nodo solicita un documento, primero comprueba si tiene una copia en su caché local. Si no dispone de esa copia, en nodo comprueba, consultando la tabla *IV*, si conoce algún vecino que lo tenga. Si no es así, se envía la petición del documento al servidor. Si algún nodo intermedio entre el nodo origen y el servidor tiene una copia del documento en su caché local devuelve la copia al nodo solicitante. Alternativamente, si un nodo intermedio conoce de algún vecino que tenga el documento, redirige la petición hacia ese nodo. Para evitar bucles de los mensajes debido a las redirecciones, los nodos

comprueban si una petición ha pasado por ellos dos veces. Si es así, la segunda la envían al servidor de documento en lugar de redireccionarla de nuevo.

Cuando se almacena un documento en la caché local y se borran otros para hacerle sitio, el nodo que ha almacenado el documento envía un mensaje a todos sus vecinos informando del documento que ha almacenado y de los que ha eliminado de su caché local. Los nodos que reciben este mensaje actualizan sus tablas *IV* para mantener la coherencia.

Como política de reemplazo propone el algoritmo denominado CV (los autores no especifican lo que significan estas siglas). CV propone eliminar de la caché para reemplazo los documentos con un mayor valor de la variable *count* en la tabla *IV*. De esta forma se eliminan los documentos que más se repiten entre los vecinos.

En [Kumar,2010] se propone una mejora a IXP al añadir un mensaje *LEAVE* que envían los nodos cuando van a salir del grupo al que pertenecen. El mensaje *LEAVE* incluye información sobre los documentos que tiene almacenados el nodo en su caché local para que los demás nodos actualicen sus tablas en consecuencia.

3.2.1.8 Data Pull/Index Push (DPIP)

El esquema de caché DPIP (*Data Pull/Index Push*) [Chiu,2005] [Chiu,2009], propuesto por los mismos autores que el anterior, funciona de forma similar a IXP, siendo una evolución de éste. Al igual que IXP, cada nodo mantiene una tabla *IV*. Cuando un nodo necesita un documento que no tiene en su caché local comprueba, consultando la tabla *IV*, si sabe de algún otro nodo vecino que lo tenga. Si así fuera, envía al nodo correspondiente un mensaje de solicitud del documento. Si no hubiera constancia de que otro nodo del vecindario tuviera el documento a solicitar, el nodo envía una petición *data_pull* en modo *broadcast* a sus nodos vecinos (a un salto). Esta petición incluye el identificador del documento que se solicita y la lista de los documentos que serán eliminados de la caché local cuando se reciba el documento. Esta información se utiliza para actualizar las tablas *IV* de los nodos vecinos. Cuando un nodo recibe un mensaje *data_pull* responde al nodo solicitante si cumple alguna de las siguientes condiciones:

a) El nodo posee una copia del documento solicitado en su caché local.
b) En nodo conoce algún vecino que tiene el documento pedido.

En el caso de que se cumpla la condición a), el nodo que recibe el mensaje *data_pull* responde con el documento. En el caso b), el nodo responde con un mensaje *location_reply* con la dirección del nodo que tiene el documento.

Cuando se envía un mensaje *data_pull* el nodo espera un tiempo (*DPIP_Timer*) antes de enviar la solicitud del documento al servidor.

3.2.1.9 CacheData

De acuerdo con en el esquema de caché *CacheData* [Guohong, 2004] [Yin,2006] los nodos envían las peticiones directamente al servidor. Los nodos almacenan en su caché local los documentos que reenvían si consideran que éstos son populares, es decir, ha habido muchas peticiones de los mismos, o si tienen espacio suficiente en su caché local. De esta forma, futuras peticiones a los documentos pueden ser servidas directamente por los nodos intermedios en la ruta desde el nodo solicitante hacia el servidor realizando una intercepción de la petición. Para evitar un excesivo almacenamiento de documentos en tránsito se propone que un nodo no almacene un documento si todas las peticiones vienen del mismo nodo origen.

En [Zhao,2008] y [Zhao,2010] se propone una mejora al esquema *CacheData* que consiste en que las respuestas que van desde el servidor hacia el nodo solicitante sólo suban a la capa de aplicación si el documento tiene que ser almacenado en caché. Esta decisión la toma el servidor en lugar de cada nodo de forma individual. De esta forma se ahorra el procesamiento en los nodos intermedios y se reduce el retardo extremo a extremo. Para realizar esta mejora, la petición va almacenando la lista de los nodos por los que va pasando en su camino hacia el servidor. La respuesta lleva la lista de los nodos por los que tiene que pasar (*tunneling*) para que los nodos intermedios que no estén en esa lista simplemente reenvíen el mensaje y no lo suban al nivel de aplicación. El servidor usa la distancia en saltos y el retardo agregado para determinar en qué nodos tiene que ser almacenado el documento.

3.2.1.10 CachePath

En el esquema de caché *CachePath* [Guohong, 2004] [Yin,2006] los nodos almacenan información acerca de dónde se encuentran los documentos almacenados basándose en las peticiones que pasan a través de ellos. Sólo se almacena información sobre los destinos que están más cerca, en número de saltos, que el servidor y siempre y cuando la distancia al servidor menos la distancia al nodo destino supera un cierto umbral (T_H). De esta forma, cuando los nodos realizan peticiones al servidor, ésta puede ser redireccionada por un nodo intermedio hacia otro nodo más cercano que tiene una copia del documento.

3.2.1.11 HybridCache

HybridCache [Guohong, 2004] [Yin,2006] es un esquema de caché que mezcla los esquemas CacheData y CachePath.

Los nodos realizan las peticiones directamente hacia el servidor. Cuando una petición llega a un nodo en el camino hacia el servidor, éste comprueba si tiene una copia válida en su caché local. Si es así, el nodo devuelve al nodo solicitante una copia del documento. No cumpliéndose esta condición, si el nodo conoce la existencia de otro nodo

que tenga una copia del documento reenvía la petición hacia ese nodo. En otro caso reenvía la petición al siguiente nodo en la ruta hacia el servidor.

Cuando un documento es servido, desde el servidor o desde otro nodo de la red, la respuesta que pasa por cada uno de los nodos hacia el nodo solicitante es inspeccionada por los nodos intermedios. De esta forma, cuando a un nodo le llega una respuesta para reenviar, si tiene una copia del documento que ha recibido en su caché local y esta copia es más antigua que la que ha recibido actualiza la copia usando el documento recibido. Si el nodo no tuviera una copia válida en su caché local y el tamaño del documento es menor que un valor umbral T_S o hubiera una copia no válida en la caché o se supiera de otro nodo que tiene el documento, entonces se almacena el documento recibido en la caché local. En otro caso, si la distancia en saltos hacia el servidor menos la distancia en saltos hacia el nodo origen de la petición es menor que un umbral T_H y el tiempo de vida del documento es mayor que un umbral T_{TTL}, entonces se almacena la dirección del nodo que va a recibir el documento.

HybridCache usa la política de reemplazo SXO (Size*Order), que elimina de la caché aquellos documentos que tienen un mayor valor de valoración. La valoración de un documento d_i viene determinada por:

$$value(d_i) = s_i \times Order(d_i) \tag{3.10}$$

donde s_i es el tamaño del documento d_i y $k=Order(d_i)$ es el k-ésimo documento más accedido. Debido a que $Order(d_i)$ no es un valor directamente computable en cada nodo, éste se deriva de la tasa de acceso a_i al documento d_i:

$$a_i = \frac{K}{T - T_{a_i}(K)} \tag{3.11}$$

siendo T el instante de tiempo actual, $T_{ai}(K)$ el instante de tiempo en el que se produjo la *K-ésima* petición anterior al documento d_i, lo que exige tener una memoria de K posiciones con información de los últimos instantes de petición de cada documento. Por tanto, los documentos se ordenan en la caché según su valor de tasa de acceso a_i, estimándose $Order(d_i)$ como la posición que ocupa el documento d_i en la lista ordenada.

3.2.1.12 Cooperative and Adaptive Caching System (COACS)

En el esquema de caché COACS (*COoperative and Adaptive Caching System*) [Artail,2005] [Artail,2008] los nodos pueden tomar uno de entre dos posible roles: nodos *QD* (*Query Devices*) y nodos *CN* (*Caching Nodes*). Los nodos *QD* se encargan de almacenar las peticiones realizadas por los demás nodos, mientras que los nodos *CN* se encargan de almacenar los documentos en sus cachés locales.

Los nodos *QD* mantienen una tabla distribuida con información acerca de los nodos en los que están almacenados los documentos. De esta forma, si un nodo *QD* recibe una

petición y éste no sabe dónde encontrar el documento, la reenvía al nodo *QD* más cercano usando el algoritmo MDPF (*Minimum Distance Packet Forwarding*). El mensaje de petición almacena una lista de los nodos por los que va pasando para evitar la formación de bucles.

Cuando los nodos se unen a la red, éstos mandan mensajes específicos de tipo *broadcast* (*HELLO*) para que todos los demás nodos sepan de su presencia. Tras el intercambio de mensajes *HELLO*, el primer nodo en la red que tenga que almacenar un documento en su caché, después de haber enviado un mensaje de petición *DRP* (*Data Request Packet*) y haber recibido el documento mediante un mensaje *DREP* (*Data Reply Packet*), envía un mensaje *CSP* (*COACS Score Packet*) a uno de sus nodos vecinos. El mensaje *CSP* contiene la puntuación que recibe el nodo, la dirección, la capacidad de la caché y una lista vacía que los autores denominan de *nodos exhaustos*. La puntuación de cada nodo depende del tiempo que se espera que esté activo en la red, la capacidad de su batería, su ancho de banda y la memoria disponible para cachear, aunque los autores no especifican cómo calcular el valor de esta puntuación. Cuando un nodo recibe un mensaje *CSP*, añade su propia puntuación, dirección y capacidad de caché al mensaje y lo reenvía al nodo más cercano de su tabla de encaminamiento que no esté ni en la lista de direcciones ni en la de *nodos exhaustos*. Si el nodo que recibe el mensaje *CSP* encuentra que todos los nodos que tiene en su tabla de encaminamiento están en la lista de direcciones del mensaje *CSP*, entonces añade su dirección a la lista de *nodos exhaustos* y reenvía el mensaje a alguno de los nodos que aparecen en la lista de direcciones y que no aparecen en la lista de *nodos exhaustos*. Esta estrategia asegura que el mensaje *CSP* pasará por todos los nodos de la red de forma secuencial. Cada nodo comprueba si la lista de direcciones del mensaje *CSP* incluye todos los nodos de la red excluyéndose a sí mismo. Si es así, el nodo pasará a tener el rol *QDA* (*QD Assigner*) y enviará un mensaje *QDAP* (*QD Assignment Packet*) al nodo con mayor puntuación en el que se incluye la tabla con todos los valores recopilados en el mensaje *CSP*. El nodo que recibe el mensaje *QDAP* pasa a ser el primer nodo *QD*. Este nodo calcula el número inicial (N_{QD}^{strt}) y máximo (N_{QD}^{max}) de nodos *QD* en la red, siendo:

$$N_{QD}^{strt} = \left\lfloor \sqrt{\frac{N \cdot R_{req} \cdot R_{hit}}{2 \cdot K_L}} \right\rfloor \tag{3.12}$$

donde N es el número de nodos en la red, R_{req} es la tasa de peticiones por nodo, R_{hit} es la tasa de aciertos que se desea conseguir y K_L es una constante umbral.

El valor de N_{QD}^{max} se calcula mediante la siguiente inecuación de forma recursiva:

$$E[H_{QD_{Data}}] + (1/R_{hit} - 1)E[H_{QD_{Last}}] - 0.4884a/r_0 - 2T_{out}/T_{in} < 0 \tag{3.13}$$

en la que:

$$E[H_{QD_{Data}}] = \sum_{i=1}^{N_{QD}} P_i \sum_{j=N_{QD}-i+1}^{N_{QD}} E\left[H \mid n = j\right] \tag{3.14}$$

$$E[H_{QD_{Last}}] = E[H_{N_{QD}} \mid n = N_{QD}] = \sum_{j=1}^{N_{QD}} E[H \mid n = j] \tag{3.15}$$

donde r_o es el radio de cobertura de los nodos, a es el lado del área de simulación (se supone un área cuadrada), T_{out} y T_{in} representan el tiempo de acceso a un nodo que está fuera de la red, como un servidor de documentos, y dentro de la red respectivamente. P_i es la probabilidad de que el nodo QD_i tenga el documento pedido y $E[H|n=j]$ es el número esperado de saltos dadas j opciones de nodos *QD* a los que mandar las peticiones y $E[H_{QDLast}]$ es el número esperado de saltos para acceder al servidor de documentos.

El nodo seleccionado como *QD* envía un mensaje *QDAP* a los N_{QD}^{strt}*-1* nodos con mayor puntuación de la lista, excluyéndose él mismo. Suponiendo que todos los nodos responden con un mensaje de confirmación, QD_1 envía un mensaje *CIP* (*COACS Information Packet*) en modo *broadcast* con las direcciones de todos los nodos *QD* asignados. El mensaje *CIP* se reenvía cuando la lista de nodos *QD* cambia. Además, cuando un nodo se une a la red, éste consigue la lista de nodos *QD* de sus nodos vecinos solicitando mediante un mensaje *CIP*.

Cuando una petición necesita ser almacenada en caché y ningún nodo *QD* acepta cachearla se añade un nuevo *QD* al sistema. El último *QD* en recibir la petición de cacheo inicia un *CSP*. Cuando un nodo *QD* deja la red, el primer nodo en descubrirlo inicia un *CSP* para encontrar un nuevo candidato para ese rol. En ambos casos, el primer nodo *QDA* calcula la mayor puntuación del mensaje *CSP* y envía un mensaje *QDAP* al candidato con mayor puntuación. Si el nodo acepta, éste envía un mensaje *CIP* en modo *broadcast* con él mismo añadido a la lista de *QDs*, en otro caso, responde con un mensaje de rechazo *NACK* (*Negative ACKnowledgement*) al nodo *QDA*. Para evitar situaciones en las que el candidato no hace nada, se define un temporizador en el nodo *QDA* que arranca tras haber enviado el mensaje *QDAP*. Si el *QDA* recibe un mensaje *NACK* o salta el temporizador, transmite un mensaje *QDAP* al segundo candidato con mayor puntuación, y así sucesivamente hasta que un candidato acepta la asignación.

Los nodos *CN* mantienen para cada entrada en la caché, no sólo los documentos en sí, sino también la petición y el nodo *QD* que la ha cacheado. Esta información se usa para el caso de que un nodo *QD* abandone la red y sea preciso reconstruirla en el nuevo nodo *QD*. Cuando se recibe el mensaje *CIP* del nuevo nodo *QD*, los nodos *CN* le envían las peticiones que referenciaban el nodo *QD* que ha abandonado la red. Esta información se le hace llegar usando un mensaje *QCRP* (*Query Caching Request Packet*). Si un nodo *CN* abandona la red, los nodos *QDs* lo detectarán y borran las entradas asociadas en su caché.

Un nodo únicamente se comporta como *CN* para la petición generada si la respuesta viene del servidor de documentos. En dicho caso, el nodo *CN* almacena el documento y la petición correspondiente y envía un mensaje *QCRP* al nodo *QD* más próximo. Si dicho *QD* no tiene memoria suficiente para almacenarla, la reenvía al siguiente *QD* más próximo, y así sucesivamente hasta que se almacena la petición o se crea un nuevo nodo *QD* para albergarla. Cuando un nodo *QD* almacena una petición, éste manda un mensaje *CACK*

(*Cache Acknowledgement Packet*) al nodo *CN*, que almacenará una referencia que enlaza el documento almacenado con dicho *QD*.

Suponiendo que todos los nodos de la red tienen conocimiento de los nodos *QD* que hay en la red, cuando un nodo necesita cierto documento, éste envía un mensaje *DRP* (*Data Request Packet*) al nodo *QD* más próximo. Si dicho *QD* no tiene la petición almacenada, añade su dirección al mensaje *DRP* y lo reenvía al siguiente nodo *QD* más cercano para indicarle que él no lo tiene. Este proceso continúa hasta que se encuentra la petición o hasta que se ha consultado a todos los nodos *QD*, en cuyo caso se contacta con el servidor de documentos para acceder a él. Si ocurre un acierto en un *QD*, la lista de nodos *QD* se elimina del mensaje *DRP* antes de ser enviado al nodo que tiene el documento almacenado en caché. Este nodo, al recibir el mensaje *RDP*, responderá al nodo solicitante con el documento usando el mensaje *DREP* (*Data Reply Packet*). Cuando un *CN* borra un documento de su caché, éste informa al nodo *QD* asociado de este hecho enviándole un mensaje *EDP* (*Entry Deletion Packet*).

3.2.1.13 Esquema de Denko

En el esquema de caché propuesto en [Denko,2007], los nodos móviles forman *clusters*. Para la formación de los *clusters* el esquema utiliza el algoritmo del menor identificador propuesto en [Gerla,1995].

Cada *cluster* tiene un nodo *CH* (*Cluster Head*) que se encarga de la comunicación con los otros *clusters*, un nodo *DS* (*Data Source*) que es el que tiene los documentos para ser servidos, nodos *CA* (*Caching Agents*) que se encargan de almacenar los documentos en caché y nodos *MH* (*Mobile Hosts*) que realizan las peticiones de los documentos. La conectividad de la red se mantiene usando mensajes periódicos *HELLO*.

Cuando un nodo necesita un documento que no tiene en su caché local, lo pide primero a sus vecinos, después al *CA*, después al *DS* y finalmente al *CH*. Si no lo encuentra en ninguno de ellos lo pide a otro *cluster* a través del *CH*.

En el caso de que un documento almacenado en la caché de un *CA* caduque y se detecte que dicho documento es solicitado frecuentemente, el nodo *CA* lo solicita de nuevo aunque no lo necesite en ese momento. Se realiza la misma acción si un *MH* necesita frecuentemente un documento que no está almacenado en sus vecinos, en cuyo caso lo solicita y se convierte en un *CN*. Para decidir si un documento es popular y debe ser solicitado de nuevo aunque haya caducado, cada nodo tiene un *PM* (*Prefetch Module*) que calcula, para cada documento D_i:

$$CPI(D_i) = \sum_{i=1}^{k} \frac{n_i}{N_j} \quad , i \geq 1 \tag{3.16}$$

donde N_j es la cantidad de peticiones distintas al nodo j, n_i es el número total de peticiones al documento D_i y k es el número de nodos. Si el valor de $CPI(D_i)$ supera un cierto umbral, el documento D_i se vuelve a solicitar.

Los nodos usan la política de reemplazo LRFU para manejar su caché local.

3.2.1.14 GroupCaching

El esquema de caché *GroupCaching* [Ting,2007] propone crear grupos para el almacenamiento de los documentos en caché. De esta forma, cada nodo en la red mantiene una tabla, denominada *self_table*, con la caché local. Cada entrada de esta tabla consiste en una *tupla* de la forma:

(*data_id*, *data*, *data_source_id*, *timestamp*)

donde *data_id* es el identificador del documento almacenado, *data* es el documento en sí, *source_id* es el identificador del nodo que envió dicho documento y *timestamp* es el instante de tiempo en el que se almacenó el documento. Los nodos también tienen otra tabla, denominada *group_table*, que mantiene información de los documentos que están almacenados en los demás nodos del grupo. Cada entrada de la tabla *group_table* está formada por una tupla de la forma (*data_id*, *data_source*, *group_member_id*, *timestamp*), donde *group_member_id* indica el identificador del nodo que tiene la copia del documento dentro del grupo y el resto de parámetros tienen el mismo significado que en la tabla *self_table*.

Los nodos vecinos de cada nodo, es decir, aquéllos que se encuentran a un salto, forman un grupo. Los nodos envían mensajes *HELLO* de forma periódica para saber si los nodos vecinos siguen estando en el grupo. Además, los nodos envían cada segundo un mensaje de control de caché con la tupla:

(*group_member_id*, *cached_data_id*, *timestamp*, *remaining_available_cache_space*)

donde *cached_data_id* es el identificador de un documento almacenado en el nodo *group_member_id* y *remaining_available_cache_space* informa del espacio que el nodo tiene disponible para almacenar documentos en su caché local. Estos mensajes mantienen actualizadas las tablas *group_table* de los nodos en el grupo.

Cuando un nodo necesita un documento primero mira en su caché local. Si no lo tiene almacenado, busca en la tabla *group_table* para comprobar si existe algún nodo en el grupo que tenga una copia. Si es así se envía la petición a dicho nodo. En otro caso la petición se envía al servidor. Los nodos intermedios en la ruta hacia el servidor miran en su caché local por si tienen una copia del documento, en cuyo caso responden directamente. Además, comprueban si existe algún nodo en su grupo que lo tenga para redirigir la petición hacia él.

Cuando el nodo recibe el documento lo almacena si tiene espacio suficiente en su caché local. En otro caso, el nodo comprueba si hay sitio libre en la caché de algún nodo

vecino, en cuyo caso lo reenvía para que almacene el documento. Si no hubiera sitio suficiente en ninguno de los nodos del vecindario, el nodo comprueba en su tabla *group_table* si hay algún otro nodo que ya tenga el documento almacenado. Si es así, el documento no se almacena en caché. En otro caso, el nodo selecciona aquel nodo que tenga una entrada en su tabla *self_table* con el campo *timestamp* más antiguo y le envía el documento. El nodo que recibe el documento lo almacena siguiendo una política de reemplazo LRU.

3.2.1.15 Esquema de Cho

Aunque no es un esquema de caché en sí, el trabajo de [Cho,2003] propone un funcionamiento similar al propuesto en *GroupCaching* al mantener un espacio de almacenamiento común para los nodos vecinos.

Los nodos de la red inalámbrica se dividen en nodos activos, que son los que están realizando peticiones, e inactivos, que son aquéllos que sólo se dedican, temporalmente, a reenviar las peticiones de los demás nodos. Los nodos inactivos ofrecen sus cachés para almacenar los documentos que no caben en las cachés de los nodos activos. Cada documento tiene asignado un valor absoluto y global de su *RUT* (*Recently Used Time* – Tiempo de uso reciente), de forma que los documentos que están almacenados en las cachés locales están ordenados según su valor *RUT*. Cada documento en la caché tiene asignado un valor de orden que va desde 1 hasta *n*, siendo *n* el número de documentos en la caché. El documento con orden 1 es el consultado más recientemente y el de orden *n* el que lleva más tiempo sin consultarse. Se define el *ranking threshold* para cada nodo como el valor *RUT* del documento localizado en la *k*-ésima posición en la caché del nodo, siendo *k* un valor programable para toda la red, de forma que todos los nodos tienen el mismo valor de *k* en el mismo instante. Todos los nodos conocen el *ranking threshold* de los nodos vecinos ya que se intercambian sus valores cuando tienen que comunicarse.

Cuando un nodo activo recibe un documento y tiene que sacar el documento menos referenciado de su caché local para hacer sitio para el nuevo documento, éste debe seleccionar un nodo vecino para almacenar el documento a eliminar. El nodo seleccionado debe cumplir las siguientes condiciones:

1. Debe ser el nodo menos activo. Para ello se supone que el nodo menos activo será el que tenga el menor valor del *ranking threshold*.
2. Cuando el documento se almacene en el nodo vecino, éste no debe ser eliminado inmediatamente. Para ello, el valor del *ranking threshold* del nodo seleccionado debe ser anterior que el *RUT* del documento.

Si no se encuentra ningún nodo que cumpla las condiciones anteriores el documento se elimina. En otro caso, el nodo negocia con el nodo vecino elegido enviándole un paquete de negociación que incluye el *RUT* del documento. Si el valor del *RUT* es posterior al *ranking threshold* del nodo elegido se devuelve un mensaje de aceptación. En otro caso, el nodo vecino responde con un mensaje de rechazo, con lo que

el nodo tendrá que enviar otro mensaje de solicitud al segundo nodo menos activo. Este proceso se repite hasta que se encuentre un nodo que acepte el documento o hasta que ya no haya más nodos en la lista correspondiente, en cuyo caso el documento se elimina.

En caso de aceptación por algún nodo vecino se envía el documento para que sea almacenado en su caché local. Además se anota en el nodo origen la dirección del nodo vecino que va a recibir el documento para realizar futuras redirecciones de las peticiones. En el nodo destino por su parte, se anota la dirección del nodo que le cedió el documento. Cuando el nodo que recibe el documento lo tiene que eliminar de la caché, éste le envía un mensaje al nodo que se lo cedió para informarle de dicho evento y obligarle a actualizar su información.

3.2.1.16 Esquema de Moriya

En el esquema de caché propuesto en [Moriya,2003] los nodos envían un mensaje *QUERY* a sus nodos vecinos cuando necesitan un documento que no tienen en su caché local. Si alguno de los nodos que reciben el mensaje *QUERY* tiene una copia del documento responde con un mensaje *REPLY* al nodo que lo pidió. En otro caso, se reenvía sucesivamente el mensaje *QUERY* a sus nodos vecinos. Si varios nodos responden al nodo origen de la petición, éste selecciona el primero de ellos y le envía un mensaje *REQUEST* al que el otro nodo responde enviando el documento solicitado.

Para evitar la inundación de mensajes *QUERY* se propone el modelo *NDLM* (*Neighbor-Dependent Link Model*) de estabilidad de las rutas para enviar el mensaje solamente a aquellos nodos que tienen una mayor probabilidad de que su enlace sea más estable. En *NDLM* la probabilidad de la estabilidad de una ruta formada por n enlaces ($l_1 \ldots l_n$) viene dada por:

$$P(l_1)P(l_2 \mid l_1)P(l_3 \mid l_2)...P(l_n \mid l_{n-1}) \tag{3.17}$$

donde las probabilidades condicionadas se calculan como:

$$P(l_i \mid l_{i-1}) = P(l_i \cdot l_{i-1}) / P(l_{i-1}) \tag{3.18}$$

Suponiendo que la probabilidad de existencia de un enlace *P(l)* sigue un proceso de Bernouilli, se estima el valor de esta probabilidad como:

$$P(l_i) \equiv p_i = k_i / n_i \tag{3.19}$$

donde n_i representa el número de segundos durante los que se está observando el estado del enlace l_i y k_i es la suma de los segundos que el enlace l_i está activo.

En [Moriya,2003] no se comenta en ningún momento cuál es la política de reemplazo utilizada en la caché local de los nodos.

3.2.1.17 Hamlet

En el esquema de caché *Hamlet* [Fiore,2009] los documentos se encuentran divididos en *C* trozos lo suficientemente pequeños como para caber en un paquete IP. Cuando un nodo necesita un documento que no tiene en su caché local envía una petición en modo *broadcast* solicitando los *C* trozos del documento. Si un nodo recibe esta petición y tiene alguno de los trozos solicitados responde al nodo solicitante informando de este hecho. En el caso de que existan trozos del documento almacenados en la caché del nodo que solicita el documento éste reenvía la petición con la información de los trozos restantes. Conforme el nodo solicitante va recibiendo respuestas va solicitando los trozos del documento a cada uno de los nodos que le han respondido. Una vez que el documento ha sido obtenido el nodo decidirá si lo almacena en su caché local y durante cuánto tiempo.

Hamlet define el parámetro *cache drop time* como el tiempo que tiene que estar un documento almacenado en la caché en función del conocimiento que se tiene sobre el almacenamiento de los documentos en el resto de las cachés. Para realizar este cálculo los nodos móviles funcionan en el llamado modo "promiscuo" que permite escuchar y analizar los paquetes que pasan por su radio de cobertura con independencia de a quién vayan dirigidos.

Para cada documento *i* en un nodo *n* y un instante *j*, el *cache drop time* se calcula como:

$$\chi_i(n,j) = M_C - p_i(n,j)(M_c - m_c) \tag{3.20}$$

siendo M_c y m_c los valores máximo y mínimo respectivamente que puede tener el *cache drop time*. Por su parte $p_i(n,j)$ denota el índice de presencia general del documento *i* vista por el nodo *n* en el instante *k*, y se define como:

$$p_i(n,j) = \min\left\{1, \sum_{k=j-\tau}^{j} \phi_i(n,k,j)\right\} \tag{3.21}$$

donde τ define el tamaño del filtro que se aplica. Para estimar $p_i(n,j)$, contribuciones anteriores a τ se ignoran. $\Phi_i(n,k,j)$ es un predictor de la completitud de información del documento *i* en el instante *j* considerando únicamente las aportaciones recogidas durante el intervalo de tiempo *k*:

$$\phi_i(n,k,j) = \min\left\{1, \omega_i(n,k,j)\frac{1}{C}\sum_{c=0}^{C} p_{ic}(n,k)\right\} \tag{3.22}$$

$$\omega_i(n,k,j) = \begin{cases} 1 & \text{si } j-k \le \Delta(n,k) \\ \alpha^{j-k-\Delta(n,k)} & \text{en otro caso} \end{cases} \tag{3.23}$$

$$\Delta(n,k) = \left\lfloor f_e \chi_i(n,k-1) - \log_\alpha W \right\rfloor \tag{3.24}$$

$$p_{ic}(n,j) = \min\left\{1, d_{ic}(n,j) + r_{ic}(n,j)\right\} \tag{3.25}$$

$$d_{ic}(n,j) = d_{ic}(n,j) + \frac{1}{h_Q} \tag{3.26}$$

$$r_{ic}(n,j) = r_{ic}(n,j) + \frac{1}{h_P + h_Q} \tag{3.27}$$

siendo α un factor de decrecimiento exponencial, f_e la frecuencia de estimación, $d_{ic}(n,j)$ un contador de las respuestas generadas por el nodo n a otro nodo Q que está situado a h_Q saltos y $r_{ic}(n,j)$ un contador de una respuesta escuchada por el nodo n que va del nodo Q al nodo P estando ambos nodos situados a h_Q y h_P saltos respectivamente. En el artículo no se comenta el significado del parámetro W.

3.2.1.18 SimpleSearch

En el esquema de caché *SimpleSearch* [Lim,2006], cuando un nodo necesita un documento que no tiene almacenado en su caché local se envía un mensaje de petición en modo *broadcast*. El nodo solicitante espera la respuesta durante un tiempo predefinido, pasado el cual la petición se considera que ha fallado. Cuando un nodo recibe un mensaje de petición, éste lo reenvía a sus nodos vecinos si no tiene una copia en su caché local. En otro caso, el nodo envía un mensaje de confirmación (*ACK*) al nodo que realizó la petición. En el mensaje *ACK* se almacena la dirección de cada uno de los nodos por los que va pasado el mensaje. Cuando el nodo solicitante recibe el primer mensaje *ACK*, envía un mensaje *CONFIRM* al nodo que respondió con el mensaje *ACK* siguiendo la ruta inversa. El nodo que recibe el mensaje *CONFIRM* responde enviando el documento siguiendo la ruta creada.

El algoritmo define un valor para el número máximo de saltos a los que puede hacerse la petición *broadcast* para reducir de esta forma el tráfico de petición, aunque no se especifica cuánto debe valer este parámetro. Además, *SimpleSearch* implementa una política de admisión en la caché local consistente en no almacenar en caché aquellos documentos que son servidos desde un nodo que se encuentre situado a una distancia menor que un determinado número de saltos. De esta forma se evita que documentos muy populares se encuentren almacenados en todos los nodos de la red.

SimpleSearch propone tres políticas de reemplazo en caché basadas en la distancia en número de saltos (δ) y la frecuencia (τ), definida como el inverso del tiempo transcurrido desde el último acceso al documento:

- TDS_D (*Time and Distance Sensitive – Distance*): Considera como primer criterio para eliminar documentos de la caché la distancia en número de saltos al nodo que sirvió el documento. Se eliminan primero los documentos que se sirvieron desde más cerca. Si hubiera un empate entre documentos se escoge aquél que tiene el menor valor de τ.

- TDS_T (*Time and Distance Sensitive – Time*): Se eliminan primero los documentos con menor valor de τ, es decir, aquéllos que fueron referenciados hace más tiempo.
- TDS_N (*Time and Distance Sensitive – Neutral*): Se eliminan primero de la caché los documentos con un menor valor del producto $\delta * \tau$.

3.2.1.19 Modified SimpleSearch

Modified SimpleSearch [Lin,2007] es una modificación del esquema de caché *SimpleSearch*. Se asume que cada nodo dispone de un dispositivo GPS que le permite conocer su localización y velocidad relativa hacia el servidor de documentos. De esta forma, cada nodo estima la conectividad con el servidor usando el algoritmo GPSCE (*GPS Conectivity Estimation*) propuesto junto con el esquema de caché. Este algoritmo permite discernir las peticiones que se alejan del servidor para no seguir propagándolas por la red y, de este modo, limitar el tráfico en la red.

En el esquema de caché *Modified SimpleSearch* las peticiones son siempre en modo *broadcast* en dirección hacia el servidor, de forma que los mensajes que se alejan del servidor no son reenviados. Cuando un nodo recibe una petición, comprueba si tiene una copia en su caché local. Si es así, envía la petición al servidor para averiguar si la copia está actualizada. Cuando el servidor recibe una petición puede responder directamente si nadie en la ruta hacia él tenía una copia o responder simplemente indicando que la copia que se había encontrado por el camino es válida. De esta forma el nodo intermedio puede mandar la copia del documento al nodo que lo solicitó usando el mismo mecanismo que en *SimpleSearch*.

3.2.1.20 COOP

El esquema de caché COOP [Du,2005] se divide en tres partes:

- *Adaptive flooding*: Las peticiones se hacen mediante *broadcast* pero con un límite de cuatro saltos para no inundar la red.
- *Profile-based resolution*: Cada nodo mantiene una tabla denominada RRT (*Recent Request Table*) gestionada de forma circular y con un tamaño predeterminado, en la que se va almacenando información sobre los documentos que se van pidiendo, tanto del nodo origen como del nodo destino, analizando las peticiones y respuestas que reenvía el nodo. Si el documento a solicitar no está en la caché local, primero se comprueba en esta tabla si se sabe de algún otro nodo que lo haya solicitado o servido con anterioridad. Si así fuera, se solicita el documento a ese nodo.
- *Roadside resolution*: Si usando los métodos anteriores no se ha encontrado una copia del documento a solicitar, se cursa la petición hacia el servidor. Si algún nodo intermedio en la ruta hacia el servidor tiene una copia del documento, éste responde directamente al nodo solicitante y no reenvía la

petición. Además, si algún nodo encuentra en su tabla *RTT* información sobre un nodo que tiene el documento pedido y se encuentra más cerca que el servidor, entonces la petición es redireccionada hacia ese nodo.

COOP divide los documentos en dos categorías: copias primarias y copias secundarias. Una copia de un documento en un nodo se considera primaria si no está en otro nodo vecino. De esta forma, cuando un nodo recibe un documento lo considera copia primaria si viene de fuera del "vecindario", es decir, proviene de un nodo que no está en su rango de cobertura. En otro caso, si la copia viene de dentro del vecindario y ésta estaba considerada como primaria, entonces la nueva copia se considera secundaria. Si la copia estaba considerada como secundaria, entonces se necesita saber dónde está situada la copia primaria. Si el nodo que tiene la copia primaria está dentro del vecindario, la nueva copia se considera secundaria. En otro caso la copia se considera primaria.

Como política de reemplazo COOP elimina primero de la caché local las copias secundarias y después las primarias, utilizando un política de reemplazo LRU dentro de cada una de las categorías.

3.2.1.21 Esquema de Wang

En el esquema de caché propuesto en [Wang,2006] los nodos almacenan información acerca de dónde se encuentran situados los documentos en la red. Esta información la obtienen analizando los mensajes de petición/respuesta que reenvían. Cada nodo tiene una caché que se administra mediante LRU con dicha información.

Cuando un documento se sirve desde un nodo *S* a otro *D*, el nodo *C* que se encuentra en la posición intermedia en número de saltos entre ambos también almacena la copia del documento. Los nodos que se encuentran situados entre el nodo *S* y el nodo situado a la mitad de la distancia entre *S* y *C* almacenan en su tabla que el nodo *S* tiene el documento. Los nodos que se encuentran situados entre la mitad del camino entre *C* y *D* almacenan que el nodo *D* tiene el documento. Finalmente, los nodos situados entre la mitad del camino entre *S* y *C* y la mitad del camino entre *C* y *D* almacenan que el nodo *C* tiene el documento.

Las peticiones son directamente enviadas al servidor cuando el nodo solicitante no tiene una copia en su caché local. Los nodos intermedios en la ruta hacia el servidor pueden responder directamente si tienen una copia del documento solicitado o también pueden redireccionar la petición hacia otro nodo que se conozca que tiene una copia del documento y se encuentra más cerca que el servidor. Esta comprobación se hace consultando la tabla correspondiente.

El artículo no especifica la política de reemplazo de la caché local.

3.2.1.22 Poware Aware Caching Heusistic (POACH)

POACH (POware Aware Caching Heusistic) [Nuggehalli,2003] es un esquema de caché que está dividido en dos partes: en la primera de ellas el servidor disemina los documentos en la red, mientras que en la segunda los nodos acceden a los documentos. El algoritmo de diseminación considera a la red como un grafo en el que tiene que maximizar la disponibilidad de los documento. POACH no especifica ninguna política de reemplazo en caché local.

3.2.1.23 ORION

El esquema de caché ORION (*Optimized Routing Independent Overlay Network*) [Klemm, 2003] se divide en dos partes: un algoritmo de búsqueda y otro de transferencia de documentos.

Cada nodo mantiene una caché local además de dos tablas de encaminamiento, una de respuestas y otra de documentos. La tabla de respuestas almacena los nodos desde los que se han recibido mensajes de respuesta, mientras que la tabla de documentos almacena, para cada identificador de documento, el próximo salto para la transferencia. Las dos tablas se manejan usando LRU.

En la fase de búsqueda, un nodo busca un conjunto de documentos enviando un mensaje *QUERY*, a lo que los nodos responden con un mensaje *RESPONSE*. Todos los mensajes llevan el identificador del nodo que origina el mensaje y un número único de secuencia para cada nodo que se va incrementando para evitar el reenvío de mensajes más de una vez.

Cuando un nodo pide una serie de documentos, el mensaje *QUERY* se distribuye en modo *broadcast* usando la capa de enlace, es decir, los datos de aplicación se insertan en modo *piggy-backing* en el mensaje de *broadcast* de la capa de enlace. Los nodos que reciben el mensaje responden si tienen alguno de los documentos solicitados y reenvían el mensaje extrayendo del mismo los identificadores de documento que ellos tienen. Gracias a la información de las respuestas se va sabiendo quién tiene cada documento en la red, actualizándose las tablas de documentos.

Cuando un nodo pide un documento que no tiene en su caché local, usando la tabla de documentos éste envía una serie de mensajes *DATA_REQUEST*, remitiendo un mensaje por cada trozo de documento, al nodo que tiene el documento. El nodo que recibe la petición responde empleando un mensaje *DATA_REPLY* con el trozo del documento pedido.

Para saber si las rutas se han roto se usa la información que ofrece la capa de enlace sobre ruptura de enlaces. Cuando se percibe que un enlace se ha roto se envía un mensaje *ROUTE_ERROR* informando de este hecho para actualizar las tablas.

3.2.1.24 Cooperative Caching (COCA)

El esquema de caché COCA (*COoperative CAching*) [Chow,2004] propone que cuando un nodo de la red inalámbrica necesita un documento primero mire en su caché local. Esta caché local se maneja mediante LRU. Si el documento no está en la caché local se envía un mensaje de petición en modo *broadcast* a los nodos vecinos, que responden con el documento si lo tuvieran almacenado en sus cachés locales. Si no se recibiera ninguna respuesta pasado un determinado tiempo de espera, el nodo enviaría la petición directamente al servidor. Si en el camino hacia el servidor algún nodo tuviera el documento, éste respondería directamente.

COCA divide los nodos en nodos LAM (*Low Activity Mobile host*) y nodos HAM (*High Activity Mobile host*) en función de si realizan muchas peticiones o no. Cada nodo detecta su propio estado calculando su C*ache Access Rate* (*CAR*) en cada periodo de tiempo φ. Este parámetro CAR se define como:

$$CAR(t_c, t_l) = \frac{N_a}{t_c - t_l} \tag{3.28}$$

donde $CAR(t_c,t_l)$ es el CAR calculado en el intervalo de tiempo $[t_c, t_l)$, N_a es el número de accesos a la caché durante φ, t_c es el instante de tiempo actual y t_l es el último instante de tiempo en el que el *CAR* fue calculado. Si el valor del S*haring Ability* (*SA*) de un nodo es mayor o igual que el umbral μ, el nodo es considerado *LAM*, siendo *SA* y μ:

$$SA_t = \omega \times \frac{1}{CAR_t + 1} + (1 - \omega) \times SA_{t-1} \tag{3.29}$$

$$\mu = SA_{t-T}(1 - \omega)^i + \frac{1}{\lambda + 1}(1 - (1 - \omega)^i) \tag{3.30}$$

donde ω un parámetro que pondera la importancia del valor más reciente de *SA* en el instante *t* con respecto al valor de *SA* en el instante *t-1* (considerando un tiempo discretizado), CAR_t es el valor del *CAR* calculado para el instante *t*, $T=i\cdot\varphi$ con $i=1,2,3\ldots$ y $T>0$, SA_{t-T} es el valor *SA* antes del instante *T* y λ es el *CAR* obtenido por un nodo durante el intervalo de tiempo *T*.

Para aprovechar el espacio de almacenamiento en caché de los nodos *LAM*, el servidor replica los documentos en ellos de forma que los nodos *HAM* puedan acceder a las réplicas. La elección de los documentos a replicar está basada en la probabilidad de acceso. El servidor almacena la probabilidad de acceso *p* de cada documento en el sistema. Para cada documento *i*, p_i se inicializa a cero y a t_l se le asigna el instante de tiempo actual. El valor de p_i se actualiza cada vez que se recibe una petición al documento *i* con la siguiente ecuación:

$$p_i^{new} = \omega \times \frac{1}{t_c - t_l} + (1-\omega) \times p_i^{old} \tag{3.31}$$

donde t_c es el instante de tiempo actual, t_l es el instante de tiempo del último acceso y ω es un parámetro que pondera la importancia del acceso más reciente.

Cuando un nodo cambia de estado *HAM* a *LAM*, éste envía un mensaje de petición de replicación al servidor. En dicho mensaje se incluye la información de los documentos almacenados en la caché local del nodo. El servidor ordena la lista de documentos de forma descendiente en función de su probabilidad de acceso *p*. El servidor obvia los δ primeros documentos con mayor probabilidad *p* y envía al nodo los documentos que se encuentran entre las posiciones $\delta+1$ y $\delta+CacheSize$, siendo *CacheSize* el tamaño de la caché del nodo que va a recibir los documentos. Para cada periodo de replicación μ, si el nodo sigue siendo *LAM*, éste envía otro mensaje de replicación al servidor, que procederá como se ha comentado anteriormente.

En [Chow,2004b] se proponen dos mejoras al esquema de caché *COCA*. La primera consiste en que, cuando un nodo solicita un documento a sus vecinos, los nodos que lo tienen responden con un mensaje *reply* en lugar de enviar el documento directamente. El nodo solicitante elige la respuesta que le llegó de más cerca para pedir el documento. La segunda mejora consiste en firmar los documentos mediante un mecanismo de doble *hash*, a través de las denominadas *signatures*. En primer lugar cada identificador de documento se hace pasar por una función *hash* que lo transforma en un número binario de longitud λ. Después, se hace pasar dicho número por una función módulo. Con todos los módulos calculados para cada documento de la caché se calcula la firma total superponiendo todas las firmas. Este método de firmas permite reducir la información asociada a cada documento así como facilitar y acelerar la búsqueda de los mismos.

Cuando un nodo no tiene el documento que necesita en su caché local genera una firma para el documento que necesita (*search signature*). Calcula también una firma denominada *peer signature* superponiendo las firmas de los demás nodos. Se comparan las dos firmas generadas aplicando una operación *AND* bit a bit. Si el resultado de esta operación es cero se debe a que ninguno de los demás nodos tiene el documento a solicitar. Si el resultado es igual al valor *search signature*, significa que alguno de los nodos vecinos tiene el documento, con lo que se genera una petición en modo *broadcast* para obtenerlo.

Los nodos pueden intercambiar sus firmas con los nodos vecinos de dos formas:

- Pasiva – Los nodos adjuntan las firmas en los mensajes de petición y respuesta usando la técnica de *piggy-backing*.
- Activa – Los nodos que no generan peticiones durante un periodo de tiempo μ envían su firma a los vecinos usando un mensaje en modo *broadcast*.

Los nodos borran la firma de los demás nodos si no reciben su firma durante un periodo de tiempo μ. De esta forma se detecta que los nodos salen del vecindario.

3.2.1.25 Group-based Cooperative Caching (GROCOCA)

GROCOCA (GROup-based COoperative CAching) [Chow,2004c] es un esquema de caché basado en COCA al que se le añade la formación de agrupaciones de nodos.

El servidor crea grupos de nodos basándose en el patrón de movilidad y en el patrón de acceso a datos que siguen los nodos de la red. Para conocer el patrón de movilidad los nodos tienen un dispositivo GPS que permite conocer la distancia y movilidad relativa entre ellos. Para conocer el patrón de acceso a los documentos, el servidor mantiene una matriz en la que se almacena, para cada documento y nodo del sistema, la cantidad de veces que un determinado documento ha sido solicitado por cada nodo.

Propone calcular la medida de similitud en el patrón de movimiento siguiendo la ecuación:

$$\| m_i m_j \|^{new} = \omega \times | m_i m_j | + (1-\omega) \times \| m_i m_j \|^{old} \tag{3.32}$$

donde $\|m_i m_j\|$ es la distancia media ponderada en el tiempo entre los nodos m_i y m_j y $|m_i m_j|$ es la distancia euclídea entre m_i y m_j. Finalmente, ω es un parámetro de ponderación, con valor entre cero y uno, para dar más o menos peso a la distancia más reciente. Un par de nodos se considera que tienen un patrón de movilidad común si la distancia media ponderada es menor o igual a un determinado umbral Δ. La distancia media ponderada para cada par de nodos se almacena en una matriz bidimensional denominada *WADM* (*Weighted Average Distance Matrix*).

Por otro lado, se propone también la medida de la similitud del patrón de acceso a datos siguiendo la ecuación:

$$sim(m_i, m_j) = \frac{\sum_{d=1}^{NData} A_i(d) \times A_j(d)}{\sqrt{\sum_{d=1}^{NData} A_i(d)^2} + \sqrt{\sum_{d=1}^{NData} A_j(d)^2}} \tag{3.33}$$

donde *NData* es el número de documentos del sistema, $A_i(d)$ es la frecuencia en que el nodo m_i accede al documento d. Si el valor de similitud del patrón de acceso de dos nodos determinados es mayor o igual a un determinado valor umbral δ, estos se consideran que tienen un patrón de acceso parecido. Los valores de similitud en el acceso a datos para cada par de nodos se almacenan en una matriz bidimensional denominada *ASM* (*Access Similarity Matrix*).

Cuando el servidor recibe una petición de un nodo m_i, éste extrae la información de la localización de la petición y calcula la distancia media ponderada entre dicho nodo y cada uno del resto de los nodos. Adicionalmente, el servidor también actualiza el valor de similitud del patrón de acceso del nodo y todos los demás nodos. Se considera que un nodo m_j pertenece al grupo del nodo m_i si $\|m_i m_j\| < \Delta$ y $sim(m_i,m_j) > \delta$. El servidor envía la información de cambio de grupos cuando tiene que responder a las peticiones.

Los nodos de cada grupo formado funcionan en conjunto para manejar sus cachés como una sola. Cuando un nodo no tiene el documento a solicitar en su caché local, primero envía una solicitud a sus compañeros de grupo. Si alguno de ellos tiene el documento y la caché del que lo pidió no está llena, lo almacena. Si estuviera llena no lo almacena en su caché local. Si el documento fuera servido por algún nodo que esté fuera del grupo del nodo solicitante, el documento se almacena en caché local, eliminando los documentos necesarios siguiendo la política de reemplazo LRU.

En [Chow,2007] se añade a GROCOCA el sistema de firmas propuesta en [Chow,2004b] así como tener en cuenta el TTL de los documentos.

3.2.2 Taxonomía de mecanismos de caché

Como puede observarse de la lectura del apartado anterior, existe una gran variedad de esquemas de cachés para redes MANET. Aunque todos ellos son diferentes y llevan a cabo su propósito de variadas formas, existen algunos aspectos que tienen en común y que permiten realizar una clasificación de las mismas. Se propone por lo tanto clasificar los esquemas de caché en las siguientes categorías no excluyentes:

- Esquemas basados en *broadcast* –Pertenecen a este grupo aquellos esquemas de caché que realizan las peticiones usando mensajes de *broadcast*. Los mensajes de *broadcast* pueden estar restringidos a un número determinado de saltos. Pertenecen a esta categoría los esquemas de caché: *MobEye*, *Moriya*, *SimpleSearch* y *Modified SimpleSearch*.
- Esquemas basados en información –Los nodos intercambian o almacenan información sobre la localización de los documentos en la red, como en *DGA*, *Wang*, *Cho* y *POACH*.
- Esquemas basados en roles – Cada nodo de la red tiene una función específica dentro de ella o de un subconjunto de la misma (*cluster*). *CC* y *Denko* son esquemas de caché que pertenecen a esta categoría.
- Esquemas con peticiones dirigidas – Los nodos realizan las peticiones directamente al servidor a la espera de que se pueda servir por el camino, como en *Gianuzzi*.

Al no ser excluyentes las categorías de clasificación nos encontramos con esquemas de caché que pertenecen a dos categorías simultáneamente. Tal es el caso de los esquemas de caché *COOP*, *ORION*, *IXP*, *DPIP* y *COCA* que usan tanto mecanismos de *broadcast* como de manejo de información. Además, *Sailhan* y *ZC* combinan las peticiones usando *broadcast* con las peticiones dirigidas. *COACS* y *GROCOCA* combinan los roles dentro de la red con el uso de información. Por último, *GroupCaching*, *CacheData*, *CachePath* e *HybridCache* emplean peticiones dirigidas e información.

La Figura 3.2 resume la clasificación anterior de los esquemas de caché estudiados en el estado de la técnica.

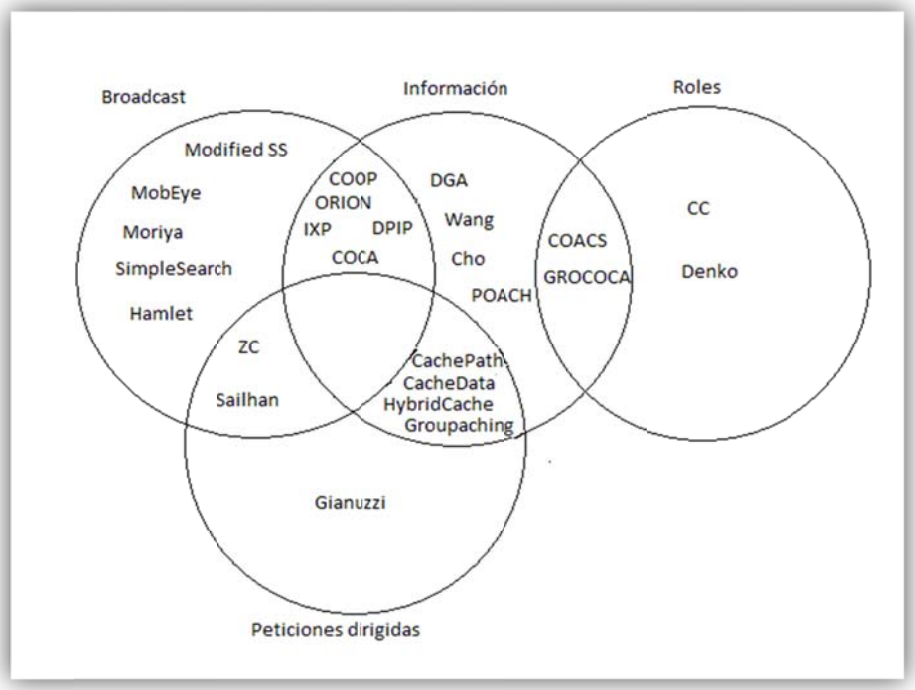


Figura 3.2. Clasificación de los esquemas de caché para redes MANET

3.2.3 Métricas de rendimiento empleadas en la literatura

Para evaluar la bondad de un esquema de caché es necesario el uso de unas métricas que permitan medir y comparar el rendimiento de cada una de ellas de forma adecuada. En el presente apartado se va a realizar una revisión de las métricas de rendimiento que se han usado en la literatura asociada. Éstas son las que siguen:

- **Número de documentos servidos** – Se define como la media de la cantidad neta de documentos que han sido servidos a los nodos de la red durante el tiempo de la simulación. Esta métrica permite comprobar qué esquemas de caché son capaces de servir más documentos a los nodos en un tiempo definido.
- **Tráfico generado** – Mide la media del tráfico que genera o reenvía cada nodo durante el tiempo de simulación. Debido a que el medio inalámbrico es un medio compartido escaso, cuanto mayor sea el tráfico generado mayor será la probabilidad de interferencias y colisiones.
- **Número de mensajes** – Mide la media del número de mensajes necesarios para obtener un documento, contando tanto los mensajes de petición como los de respuesta.
- **Retardo** – Se define como el tiempo medio que tarda en ser servida una petición de un documento, es decir, el tiempo que tendría que esperar el usuario del terminal móvil de la red MANET en recibir el documento.
- **Fallo de peticiones o *Timeouts*** – Porcentaje o *ratio* medio de peticiones que han fallado y han tenido que ser reenviadas debido a que ha saltado el

timeout de recepción tras un tiempo sin recibir el documento. Esta métrica indica la bondad del esquema de caché para localizar los documentos en la red.

- **Éxito de peticiones** – Es la métrica complementaria al porcentaje o ratio de fallo de peticiones o *timeouts*, ya que mide la proporción de peticiones que se han logrado servir con éxito.
- **Número de saltos** – Se define como el número medio de saltos que tiene que recorrer los mensajes de petición y respuesta para que un documento sea servido al nodo solicitante. Cuanto menor sea el número de saltos, menor será el número de retransmisiones y el medio radio estará menos saturado.
- **Aciertos en caché** – Generalmente se dividen en aciertos en caché local y aciertos en caché remota. Los aciertos en caché local indican el porcentaje de peticiones que son servidas desde la caché local de cada nodo. Un acierto en caché local implica un retardo nulo, evitan la generación del tráfico en la red y reducen la carga de los servidores. Los aciertos en caché remota indican el porcentaje de peticiones de documentos que son servidos por nodos de la red inalámbrica distintos del servidor. Los aciertos en caché remota implican la reducción de la carga en los servidores. Dependiendo del esquema de caché, los aciertos en caché remota pueden ser a su vez divididos en varias clases.
- **Accesos al servidor** – Esta métrica indica el porcentaje o ratio de peticiones que son servidas por el servidor de documentos.
- **Consumo de energía** – Indica el consumo energético de los nodos en la red. Para medir este parámetro de rendimiento es necesario definir un modelo de consumo de las baterías de los dispositivos inalámbricos de la red.
- **Ancho de banda** – Indica el consumo de ancho de banda de los nodos de la red. Para estimar este valor se multiplica el tamaño medio del mensaje por la tasa de mensajes que cada nodo tiene que procesar.

3.3 Estado de la técnica de la simulación

Dada la dificultad de crear modelos analíticos de todo el comportamiento de las redes MANET así como de realizar todo un abanico de pruebas con redes reales, lo habitual es realizar los estudios del rendimiento de este tipo de redes usando simuladores.

En el presente apartado se hará un repaso de los parámetros de simulación más habituales utilizados en las simulaciones en redes MANET con esquemas de caché. Este estudio servirá para elegir los parámetros más adecuados para la evaluación del rendimiento del esquema de caché CLIR y poder comparar los mismos en igualdad de condiciones con otros trabajos de la literatura.

En las tablas presentadas se indicará con el símbolo “N/E” (No Especificado) los casos en los que no se conozca el valor del parámetro utilizado ya que no se especifica en el artículo asociado. Se indicará con “NO” si el parámetro en cuestión no se tiene en cuenta en la evaluación. Por otra parte, en las tablas no se incluirán los parámetros de los esquemas de caché POACH, Wang y Gianuzzi ya que en los respectivos artículos no se realiza ningún tipo de evaluación de los mismos, los algoritmos simplemente se proponen sin efectuar validación alguna.

3.3.1 Simulador de redes

La primera elección a tener en cuenta a la hora de evaluar el rendimiento de un esquema de caché sin la necesidad de implementar físicamente los algoritmos en dispositivos reales y tener que crear una MANET real, es el simulador de redes usado. La Tabla 3.1 muestra el simulador usado para evaluar cada uno de los esquemas de caché analizados.

Esquema de caché	Simulador
MobEye	NS2
Sailhan	N/E
Zone Cooperative Caching	N/E
Cluster Cooperative	N/E
Distributed Greedy Algorithm	NS2
Index Push	NS2
Data Pull/Index Push	NS2
CacheData	NS2
CachePath	NS2
HybridCache	NS2
COACS	NS2
Denko	NS2
GroupCaching	NS2
Cho	NS2
Moriya	N/E
Hamlet	NS2
SimpleSearch	CSIM
Modified SimpleSearch	NS2
COOP	NS2
ORION	NS2
COCA	CSIM
GROCOCA	CSIM

Tabla 3.1. Simuladores empleados para evaluar el rendimiento de esquemas de caché

Como puede observarse, el simulador de redes NS2 [NS2,2011] es utilizado para evaluar 15 de los 22 esquemas de caché, mientras que CSIM [CSIM,2011] es usado sólo en 3 de ellos. Cabe destacar el hecho de que haya 5 esquemas de caché en los que no se especifique el simulador empleado para evaluarlos aunque sí se realice una evaluación de los mismos.

3.3.2 Área de simulación y servidores de documentos

Se define el área de simulación como el espacio en el que los nodos de la red inalámbrica van a estar situados y, en el caso de ser nodos móviles, el espacio en el que va a estar restringido su movimiento. Esta área es un espacio bidimensional definido por su largo y ancho. Por otro lado, el número y situación de los servidores de documentos van a definir la disponibilidad de los mismos en la red. La Tabla 3.2 recopila las medidas del área de simulación (en metros por metros), número de servidores y su situación, usados para la evaluación de los esquemas de caché estudiados. Además, se han incluido otros valores utilizados en las simulaciones.

Esquema de caché	Área de simulación	Otros	Número de servidores y posición (m,m)	Otros
MobEye	N/E (relación entre lados 0.7)		6 (posición aleatoria)	8 (posición aleatoria)
Sailhan	4000 x 4000		N/E	
ZC	1500 x 1500		1 (750, 750)	
CC	1500 x 1500		1 (750, 750)	
DGA	2000 x 500		2 (posición aleatoria)	
IXP	1500 x 500		1 (0, 0)	
DPIP	1500 x 500		1 (0, 0)	
CacheData	1500 x 320		2 (0,0) (1500,320)	
CachePath	1500 x 320		2 (0,0) (1500,320)	
HybridCache	1500 x 320		2 (0,0) (1500,320)	
COACS	1000 x 1000		1 (cerca de una de las esquinas)	
Denko	1500 x 1500		N/E	
GroupCaching	1500 x 500		Sin servidores, documentos repartidos entre todos los nodos	
Cho	N/E		Sin servidores, documentos repartidos entre todos los nodos	
Moriya	600 x 600		5 nodos tienen todos los documentos	
Hamlet	3000 x 3000	L de 100 m de ancho y alto	2 (3000,0) y (0,3000)	2 (0,50) y (400,400)
SimpleSearch	3000 x 3000		1 (1500, 1500)	
MSS	2000 x 750		1 (1000, 375)	
COOP	1500 x 300		N/E	
ORION	1000 x 1000		Sin servidores, documentos repartidos entre todos los nodos	
COCA	500x500		1 (250, 250)	
GROCOCA	1000x1000		1 (posición N/E)	

Tabla 3.2. Áreas de simulación y situación de los servidores empleados para evaluar el rendimiento de esquemas de caché

Las áreas de simulación pueden dividirse en cuadradas y rectangulares. Las áreas de simulación cuadradas se usan en la evaluación de 11 de los 22 esquemas de caché estudiados, mientras que en 10 se usa un área rectangular. Hay 2 trabajos sobre esquemas de caché en los que no se especifica el área de simulación. Cabe destacar que no existe una homogeneidad a la hora de elegir el tamaño del área de simulación.

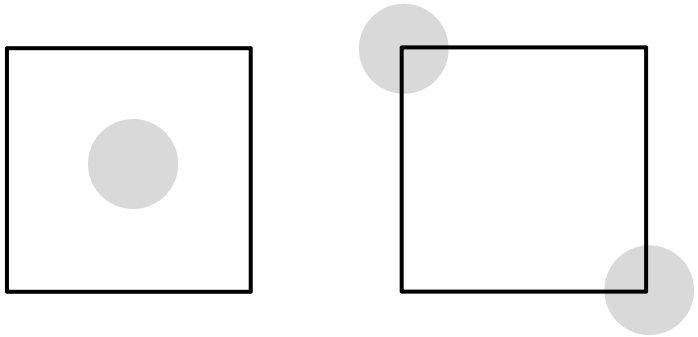
Como casos atípicos, MobEye no especifica explícitamente el área de simulación, sino que indica la relación que hay entre los lados de la misma, y Hamlet es evaluado usando, además de un área de 3000 metros por 3000 metros, un espacio en forma de 'L' en el que cada uno de los dos lados tiene una longitud de 100 metros.

Con respecto al número de servidores y su situación dentro del área de simulación, los esquemas de caché estudiados se pueden dividir en varias categorías en función del número de servidores, su situación y su movilidad. Si se tiene en cuenta el número de servidores, la mayoría de los esquemas de caché se evalúan con uno o dos servidores a excepción de Moriya y MobEye que se prueban con cinco y seis respectivamente. En los esquemas GroupCaching, Cho y ORION no existen servidores propiamente dichos, sino que los documentos están repartidos entre todos los nodos. Con respecto a la situación de los servidores, los esquemas de caché se pueden dividir en aquellos que tienen los servidores situados en las esquinas de área de simulación (DPIP, IXP, CacheData, CachePath, HybridCache, COACS y Hamlet) en el centro (ZC, CC, SimpleSearch, MSS y COCA) o situados de forma aleatoria (MobEye, DGA, GroupCaching, Cho y ORION). Finalmente, se ha de mencionar que en cuatro de los esquemas de caché no se especifica el número de servidores o su localización en la red.

3.3.3 Número de nodos en la red y radio de cobertura

El número de nodos en la red define, junto con el tamaño del área de simulación, la densidad de nodos en la red inalámbrica. Esta densidad, así como el radio de cobertura de los nodos, influye en la conectividad de la red ya que, cuantos más nodos por metro cuadrado y mayor radio de cobertura, mayor será probabilidad de conectividad entre nodos que estén situados a más de un salto de distancia.

La accesibilidad a los servidores, que también son nodos de la red, está fuertemente determinada por su situación en el área de simulación ya que, cuanto más cerca del centro del área de simulación se encuentren, mayor será el área de cobertura disponible para su conexión con otros nodos. La Figura 3.3 muestra un ejemplo de servidores situados en el centro y en las esquinas del área de simulación, indicándose el área de cobertura con un sombreado.



(a) (b)

Figura 3.3. Rango de cobertura para servidores situados en el centro (a) y en las esquinas (b) del área de simulación.

Suponiendo que la posición de los nodos en el área de simulación sigue una distribución uniforme, la probabilidad de que ninguno de los nodos de la red se encuentre dentro del área de cobertura de un servidor se puede calcular como:

$$P(aislado) = \left(1 - \frac{\pi r^2 \cdot prop}{a \cdot b}\right)^n \tag{3.34}$$

donde *r* es el radio de cobertura del servidor, *prop* es la proporción del área de cobertura que realmente está dentro del área de simulación (para el caso (a) de la Figura 3.3 *prop*=1 ya que toda la cobertura está dentro del área de simulación, mientras que para el caso (b) *prop*=1/4), *a* y *b* son el largo y el ancho del área de simulación y *n* es el número de nodos móviles (no servidores). Tanto *r* como *a* y *b* se expresan en la misma unidad de longitud. Para realizar estos cálculos se supone un modelo de propagación en el espacio libre de reflexiones.

La Tabla 3.3 muestra en número de nodos por defecto que se ha usado para evaluar cada uno de los esquemas de caché estudiados así como el radio de cobertura de las comunicaciones. Además, se indican otros valores de la cantidad de nodos usados así como la densidad de nodos por metro cuadrado y la probabilidad de que los servidores queden aislados para aquellos casos en que su posición sea fija y conocida.

Esquema de caché	**Número de nodos**	**Otros**	**Radio de cobertura (metros)**	**Otros**	**Densidad de nodos (nodos/m^2)**	**Prob. servidor aislado**
MobEye	40	80	180		N/E	
Sailhan	500	600	250		3.12×10^{-5}	
ZC	70	50-100	250		3.11×10^{-5}	0.0017
CC	70	50-100	250		3.11×10^{-5}	0.0017
DGA	100		250		10×10^{-5}	
IXP	70	60-100	250		9.33×10^{-5}	0.0088
DPIP	70	60-100	250		9.33×10^{-5}	0.0088
CacheData	100	50-100	250		2.08×10^{-5}	0.0000206
CachePath	100	50-100	250		2.08×10^{-5}	0.0000206
HybridCache	100	50-100	250		2.08×10^{-5}	0.0000206
COACS	100		100		10×10^{-5}	
Denko	200		250		8.88×10^{-5}	
GroupCaching	100		100		13.33×10^{-5}	
Cho	50	30-90	250		N/E	
Moriya	50	70	90		13.88×10^{-5}	
Hamlet	377	128	100	25	4.18×10^{-5}	0.7195
SimpleSearch	200		250		2.22×10^{-5}	0.0121
MSS	50		250		3.33×10^{-5}	0.000898
COOP	100		250		22.22×10^{-5}	
ORION	40	5-60	115		4×10^{-5}	
COCA	100		50	24 - 200	40×10^{-5}	0.0411
GROCOCA	100	100-500	100		10×10^{-5}	

Tabla 3.3. Número de nodos, radio de cobertura, densidad de nodos y probabilidad de servidores aislados usados para evaluar el rendimiento de esquemas de caché

Como puede comprobarse, existe una gran disparidad tanto en el número de nodos usados, oscilando entre 40 nodos para MobEye y ORION y 500 para Sailhan como valor por defecto, y la densidad de los mismos, que oscila entre los $2.08x10^{-5}$ nodos por metro cuadrado de CacheData, CachePath e HybridCache y los $40x10^{-5}$ nodos por metro cuadrado de COCA.

La probabilidad de que un servidor quede aislado (siempre asumiendo que la cobertura queda determinada por el alcance y que los nodos se distribuyen homogéneamente por el área de simulación) ronda entre prácticamente un valor nulo como ocurre en CacheData, CachePath e HybridCache y 0.0411 de COCA. El caso extremo se encuentra en Hamlet, donde esta probabilidad es de 0.7195. Cuanto mayor es la probabilidad de que los servidores se queden aislados, mayor será la probabilidad de que los nodos de la red no puedan acceder a ellos para obtener los documentos.

3.3.4 Estándar de conexión y modelo de propagación

El estándar de conexión define la capa física y de enlace de la capa OSI (*Open System Interconnection*), especificándose sus normas de funcionamiento en una WLAN (*Wireless Local Area Network*). Dependiendo del estándar de conexión se pueden alcanzan diferentes anchos de banda. Por su parte, el modelo de propagación caracteriza la propagación de las ondas de radio en función de la frecuencia, la distancia y otras condiciones. La Tabla 3.4 muestra los estándares de conexión así como los modelos de propagación empleados para evaluar los esquemas de caché estudiados.

Esquema de caché	Estándar conexión	Modelo de propagación
MobEye	N/E	*Two-Ray Ground*
Sailhan	802.11	N/E
ZC	N/E	N/E
CC	N/E	N/E
DGA	802.11	N/E
IXP	802.11	N/E
DPIP	802.11	N/E
CacheData	802.11	N/E
CachePath	802.11	N/E
HybridCache	802.11	N/E
COACS	802.11	N/E
Denko	N/E	N/E
GroupCaching	802.11	N/E
Cho	N/E	N/E
Moriya	N/E	N/E
Hamlet	802.11b	*Two-Ray Ground*
SimpleSearch	N/E	N/E
MSS	802.11	*Free space*
COOP	802.11	N/E
ORION	802.11	*Two-Ray Ground*
COCA	N/E	N/E
GROCOCA	N/E	N/E

Tabla 3.4. Estándares de conexión y modelos de propagación usados para evaluar el rendimiento de esquemas de caché

Se aprecia que en nueve de los esquemas de caché no se especifica el estándar de conexión, siendo 802.11 el más utilizado. Sólo Hamlet indica que emplea 802.11b. Por otro lado, sólo cuatro de los artículos especifican el modelo de propagación empleado, siendo el modelo *Two-Ray Ground* [Rappaport,1996] usado en tres de los esquemas de caché y *Free Space* [Rappaport,1996] en sólo uno.

3.3.5 Movilidad de los nodos

El modelo de movilidad de los nodos define cómo van a desplazarse éstos por el área de simulación, a qué velocidad y si van a realizar paradas. La Tabla 3.5 muestra estos datos para cada uno de los esquemas de caché estudiados.

Esquema de caché	Modelo de movilidad	Velocidad (m/s)	Otros	Tiempo de pausa (s)	Otros
MobEye	RWP	Aleatorio [0.5, 1.38]		Aleatorio [0, 60]	
Sailhan	N/E	N/E		N/E	
ZC	RWP	2	2-20	300	
CC	RWP	2	2-20	300	
DGA	RWP	Aleatorio [0, 20]		300	
IXP	RWP	Aleatorio [1, 20]		300	
DPIP	RWP	Aleatorio [1, 20]		300	
CacheData	RWP	2	Aleatorio [0, 20]	300	
CachePath	RWP	2	Aleatorio [0, 20]	300	
HybridCache	RWP	2	Aleatorio [0, 20]	300	
COACS	RWP	Aleatorio [0.01, 2]	Aleatorio [10, 20]	100	
Denko	RWP	Aleatorio [0, 20]			[200 – 2000]
GroupCaching	RWP	Aleatorio [1, 10]		N/E	
Cho	N/E	N/E		N/E	
Moriya	RWP	2	4, 5	60	120
Hamlet	IDM-IM	6,94	0,55	N/E	
SimpleSearch	RWP	Aleatorio [0, 1]			0, 100, 200, 400, 800, 1600, Inf
MSS	RWP	Aleatorio [0, 2]		100	
COOP	RWP	Aleatorio [1, 20]		120	
ORION	RWP	Aleatorio [0, 2]		50	
COCA	RWP	Aleatorio [0, 5]		1	
GROCOCA	RWP	Aleatorio [0, 5]		1	

Tabla 3.5. Modelo de movilidad, velocidad y tiempo de pausa usados para evaluar el rendimiento de esquemas de caché

Como puede observarse en la tabla, todos los esquemas de caché utilizan RWP (*Random Way Point*) excepto Hamlet, que usa IDM-IM (*Intelligent Driver Model with Intersection Management*) [Fiore,2007]. Tanto Sailhan como Cho no especifican el modelo de movilidad usado. Con respecto a la velocidad de los nodos se usan distribuciones aleatorias aunque el rango es variado, desde 0 m/s hasta 20 m/s, aunque también se usan velocidades constantes en esquemas de caché como ZC, CC, CacheData, CachePath, HybridCache, Moriya y Hamlet. En nueve de las políticas de reemplazo se ha usado una velocidad mínima o muy cercana a 0 m/s, lo que genera problemas de convergencia en la

simulación al poder quedar los nodos parados indefinidamente si se selecciona dicha velocidad. De hecho, si la simulación resultara lo suficientemente larga (idealmente infinita), los nodos convergerían hacia una situación en la que todos estarían parados [Yoon,2003]. El tiempo de pausa también es muy variado, con un rango entre 1 segundo y 300 segundos, dándose el caso de esquemas de caché que no lo especifican, como en GroupCaching, Cho o Hamlet.

3.3.6 Características de los documentos, política de reemplazo y tamaño de las cachés

En este apartado se analiza el número de documentos típicos usados en la evaluación de los esquemas de caché así como su tamaño. También se estudia la política de reemplazo empleada en las cachés locales y el espacio asignado para la misma.

La cantidad y tamaño de los documentos que haya en una MANET va a definir, junto al tamaño asignado para las cachés locales, la cantidad de documentos que van a poder ser almacenados en cada nodo de la red. Cuanto mayor sea esta cantidad, mayor será la probabilidad de que se produzcan aciertos en la caché local. La Tabla 3.6 muestra estos valores para cada uno de los esquemas de caché estudiados.

Esquema de caché	Num docs.	Otros	Tam. (kB)	Otros (kB)	Política reemp.	Tamaño caché (kB)	Otros
MobEye	60	80	10		LRU	5, 10, 15, 20% de 600 kB	5, 10, 15, 20% de 800 kB
Sailhan	N/E		1		Sailhan	N/E	
ZC	1000			[1, 10]	VALUE	800	200 - 1400
CC	1000			[1, 10]	LUV-Mi	800	200 - 1400
DGA	1000			[0.1, 1.5]	LRU	75	15 - 150
IXP	3000		1		CV		60 - 300
DPIP	3000		1		CV		60 - 300
CacheData	1000	80		[1, 10]	SXO	800	200 - 1200
CachePath	1000	80		[1, 10]	SXO	800	200 - 1200
HybridCache	1000	80		[1, 10]	SXO	800	200 - 1200
COACS	10000		10		N/E	200	
Denko	N/E		N/E		LRFU		100 - 1800
GroupCaching	1000		10		LRU		200 - 1400
Cho	50	30 - 90	2		LRU	5	
Moriya	N/E		N/E		N/E	1024	
Hamlet	10		1		N/E		10 – 100 %
SimpleSearch		1000, 10000		N/E	TDS_D, TDS_T y TDS_N	16	
MSS	1000		10		LRU		10 - 100
COOP	2000		N/E		LRU	50	
ORION	N/E		3096		N/E	N/E	
COCA	1000		1		LRU		5 - 30 %
GROCOCA	10000		1		LRU	100	1 - 20 %

Tabla 3.6. Cantidad y tamaño de los documentos, tamaño y política de reemplazo en caché local usados para evaluar el rendimiento de esquemas de caché

Como con otros valores, no existe un consenso al elegir los parámetros de simulación. El número de documentos en la red varía entre un mínimo de 10 y un máximo de 10000, aunque la cantidad de 1000 documentos es la más usada. El tamaño de los documentos utilizados varía entre 1 y 10 kB, siendo el valor más usado el de 1 kB. Cabe destacar que ZC, CC, DGA, CacheData, CachePath e HybridCache utilizan tamaños de documentos variables, mientras que el resto usa tamaños constantes. La política de reemplazo más habitualmente usada es LRU, excepto en aquellos esquemas de caché que proponen su propia política de reemplazo, como Sailhan, ZC, CC, IXP, DPIP, CacheData, CachePath, HybridCache y SimpleSearch. El único que usa una política de reemplazo no propuesta diferente de LRU es Denko, que emplea LRFU. Cabe destacar que COACS, Moriya, Hamlet y ORION no especifican la política de reemplazo. El tamaño de las cachés locales también es muy variable, encontrándose valores entre el 1% y el 30% del total del peso de los documentos de la red.

3.3.7 Otros parámetros de simulación

En el presente apartado se analiza el protocolo de encaminamiento, el tiempo de espera entre peticiones, el tipo de tráfico generado por los nodos, el TTL de los documentos y el tiempo de simulación empleado para evaluar los esquemas de caché. La Tabla 3.7 muestra todos estos parámetros.

AODV es, con diferencia, el protocolo de encaminamiento más usado para la evaluación de los esquemas de caché, seguido por DSDV y ZRP. Hamlet no emplea un protocolo de encaminamiento, sino que realiza un *broadcast* a nivel MAC para encontrar las rutas. Se ha de comentar además, que seis de los trabajos no indican qué protocolo de encaminamiento usan.

Los tiempos entre peticiones o tiempo de espera entre peticiones están definidos como la media de tiempo que los nodos deben aguardar desde que reciben un documento hasta que realizan la siguiente petición. La duración de esta espera se suele modelar con una distribución exponencial. Este tiempo medio de espera es muy variable en los esquemas de caché estudiados, estando entre 1 y 6000 segundos para los casos extremos, aunque los valores más comunes rondan entre 1 y 20 segundos de espera.

La popularidad de los documentos indica la distribución que se ha empleado para caracterizar la secuencia con que éstos se solicitan. En los esquemas de caché analizados la popularidad empleada es de tipo aleatorio uniforme, de acuerdo con la cual todos los documentos tienen la misma probabilidad de ser solicitados, o de tipo Zipf. La ley Zipf afirma, tal y como se comentó en el capítulo 2.2.1 que la probabilidad *P(i)* de que el i-ésimo documento más frecuentemente pedido sea solicitado es inversamente proporcional a su ranking de popularidad, tal y como se muestra en la ecuación (3.35):

$$P(i) = \frac{\beta}{r^{\alpha}} \tag{3.35}$$

donde α es la pendiente de la representación logaritmo/logaritmo del número de referencias al documento en función de su ranking de popularidad (r), mientras que β es una constante que determina el desplazamiento de la función.

Esquema de caché	Prot. encam.	T. pet. (s)	Otros	Popularid. doc.	Otros	TTL doc.	Otros	T. simul (s)
MobEye	AODV	N/E		Uniforme		NO		N/E
Sailhan	ZRP	N/E		N/E		N/E		N/E
ZC	N/E	5		Zipf (0.8)		5000	200 - 10000	N/E
CC	AODV	5		Zipf (0.8)		5000		N/E
DGA	DSDV	10	[3 – 40]	Zipf (0.8)			Uniforme [10.000 – 20.000]	N/E
IXP	N/E	20		Uniforme		∞		5000
DPIP	N/E	20		Uniforme		∞		5000
CacheData	AODV	5	[1 – 100]	Zipf (0.8)	Zipf (0 – 1)	5000	200 - 10000	N/E
CachePath	AODV	5	[1 – 100]	Zipf (0.8)	Zipf (0 – 1)	5000	200 - 10000	N/E
HybridCache	AODV	5	[1 – 100]	Zipf (0.8)	Zipf (0 – 1)	5000	200 - 10000	N/E
COACS	DSDV	10		Zipf		∞		2000
Denko	AODV	N/E		N/E		N/E		2000
GroupCaching	AODV	5		N/E		∞		6000
Cho	DSR	N/E		Zipf (1.0)		∞		N/E
Moriya	Dijkstra	200		N/E		∞		10000
Hamlet	NO		[1000 – 6000]	Uniforme		∞		N/E
SimpleSearch	ZRP	600		Uniforme	Zipf (0.95)	∞		N/E
MSS	N/E		[5 – 200]	Zipf (0.9)			10 - 1000	N/E
COOP	AODV	5		Zipf (0.8)		5000		N/E
ORION	AODV	N/E		N/E		∞		N/E
COCA	N/E	1		Zipf (0.5)		∞		20000 pet.
GROCOCA	N/E	N/E		Zipf (0.5)		∞		20000 pet.

Tabla 3.7. Protocolo de encaminamiento, tiempo entre peticiones, tipo de tráfico, TTL de los documentos y tiempo de simulación usados para evaluar el rendimiento de esquemas de caché

En la Tabla 3.7, en la columna sobre la popularidad, se ha especificado la pendiente usada para la ley Zipf entre paréntesis, siendo el valor más habitual el de 0.8. Por otro lado, Zipf es más usado para evaluar los esquemas de caché que la distribución aleatoria uniforme.

El TTL de los documentos define el tiempo que son considerados válidos dentro de la simulación. Sólo 8 de los esquemas de caché analizados tienen en cuenta la caducidad de los documentos, mientras que los demás suponen que los documentos siempre son válidos y nunca caducan. El TTL sigue en estos estudios una distribución exponencial de la que se especifica la media. La media del TTL más utilizada es de 5000 segundos. Esta media de tiempo de vida, junto con el tiempo de simulación, determina el número medio de veces

que los documentos caducan a lo largo de la simulación y, por tanto, deben ser solicitados de nuevo al servidor al quedar obsoleto. Curiosamente, ninguno de los esquemas de caché analizados que usan TTL especifica el tiempo de simulación.

Sólo 8 de los 22 esquemas de caché especifican el tiempo de simulación utilizado para evaluarlos, variando, para el caso de aquellos que los especifican, entre 2000 y 10000 segundos. Los esquemas COCA y GROCOCA no indican un tiempo de simulación absoluto, en su lugar especifican el tiempo necesario para realizar 20000 peticiones en la red.

3.4 Propuesta de esquema de caché para redes ad hoc: CLIR

CLIR es el acrónimo del esquema de caché *Cross Layer Interception and Redirection*. En este esquema de caché se propone aprovechar algunas de las ventajas de los esquemas de caché propuestos anteriormente pero evitando sus inconvenientes. Para ello se va a especificar el funcionamiento de CLIR dividiéndolo en cuatro partes: funcionamiento general, especificación de la caché local, intercepción de peticiones y redirección de peticiones. El funcionamiento de CLIR va a ser definido usando una versión modificada de la notación presentada en [Hosseini,1996] para la formulación del comportamiento de un esquema de cachés en redes MANET.

3.4.1 Funcionamiento general del algoritmo

En el esquema de caché CLIR se asume el empleo de un protocolo típico de petición y respuesta del estilo de HTTP (*HyperText Transfer Protocol*). Los nodos móviles realizan peticiones usando mensajes GET en los que se incluye el identificador del documento a solicitar y el tiempo de expiración de la petición tal y como se observa en la Figura 3.4.

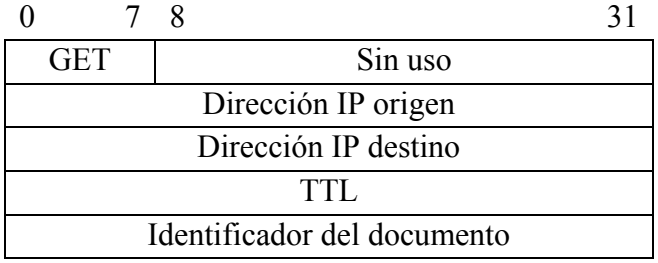


Figura 3.4. Formato del mensaje GET

Los campos “Dirección IP origen” y “Dirección IP destino” contienen las direcciones IP del nodo que realiza la petición y del nodo destino de la misma, respectivamente. El campo TTL define el instante de tiempo absoluto en el que la petición deja de ser válida. Si un nodo recibiera una petición que ha expirado o tuviera que reenviarla habiendo caducado mientras espera ser reenviada, ésta sería desechada. En la implementación realizada se ha definido, por simplicidad, que los documentos tengan un

identificador numérico de 32 bits, con lo que se podrían direccionar hasta un total de 2^{32}= 4294967296 documentos diferentes. Sin embargo, esta suposición no sería válida si se quisieran direccionar documentos de Internet, ya que no sería posible asignar un identificador único a cada uno de ellos dado que el número de documentos presentes en Internet resulta muy superior al posible direccionable. Para subsanar este problema, se propone modificar el formato del mensaje GET para que acepte cualquier URL (*Uniform Resource Locator*) necesaria para direccionar documentos en Internet, tal y como se muestra en la Figura 3.5.

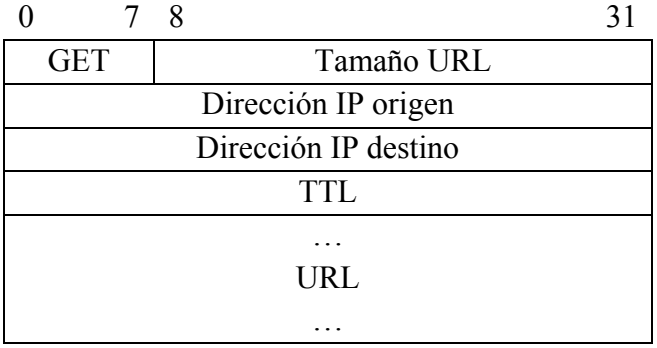


Figura 3.5. Formato modificado del mensaje GET

Donde el campo "Tamaño URL" indica el tamaño en bytes que tiene la URL del documento a solicitar. Como este campo tiene un tamaño de 24 bytes, se podrían construir URLs de 2^{24}= 16777216 bytes, tamaño más que suficiente para cualquier posible URL.

Una vez que el mensaje GET llega a su destino, el nodo correspondiente responde con un mensaje de tipo RESP (respuesta) incluyendo el documento solicitado. En la Figura 3.6 se muestra el formato del mensaje RESP.

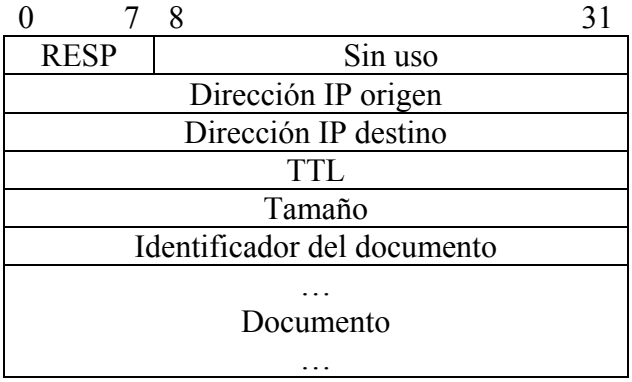


Figura 3.6. Formato del mensaje RESP

Los campos "Dirección IP origen" y "Dirección IP destino" especifican las direcciones IP de los nodos que generan la respuesta y destino de la misma respectivamente. Por otro lado, los campos "TTL" y "Tamaño" indican el tiempo de vida y el tamaño (en bytes) del documento solicitado. Con respecto al TTL, cabe reseñar que se considera como el instante de tiempo absoluto en el que el documento se considera obsoleto, siendo el servidor el que define su valor. De este modo, se sigue la misma filosofía de funcionamiento que el empleado por las respuestas de HTTP con el parámetro *Expires* [Gourley,2002]. Dicho parámetro especifica el tiempo absoluto de expiración de

los documentos Web tomando la hora del meridiano de Greenwich (GMT – *Greenwich Meridian Time*) como base.

Análogamente al caso de la petición GET, la respuesta RESP puede ser ampliada para manejar URLs tal y como se muestra en la Figura 3.7.

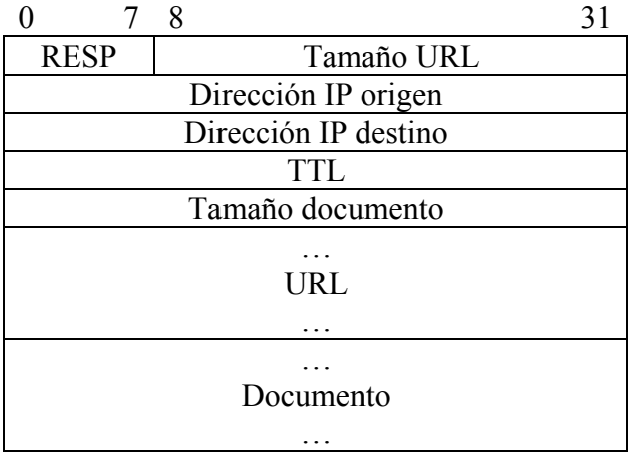


Figura 3.7. Formato modificado del mensaje RESP

Como ejemplo del funcionamiento de la propuesta, la Figura 3.8 muestra una instantánea de una MANET. En el ejemplo, el nodo *DS* es un servidor de documentos (*Document Server*), es decir, el nodo que almacena físicamente todos los documentos del sistema (o un *Gateway* o pasarela que da acceso a una red exterior), en tanto que los nodos *1*, *2*, *3*, *4*, *5* y *6* son nodos de usuario que pueden solicitar documentos al nodo *DS*. En la figura, las líneas que conectan los nodos representan las rutas creadas por el protocolo de encaminamiento.

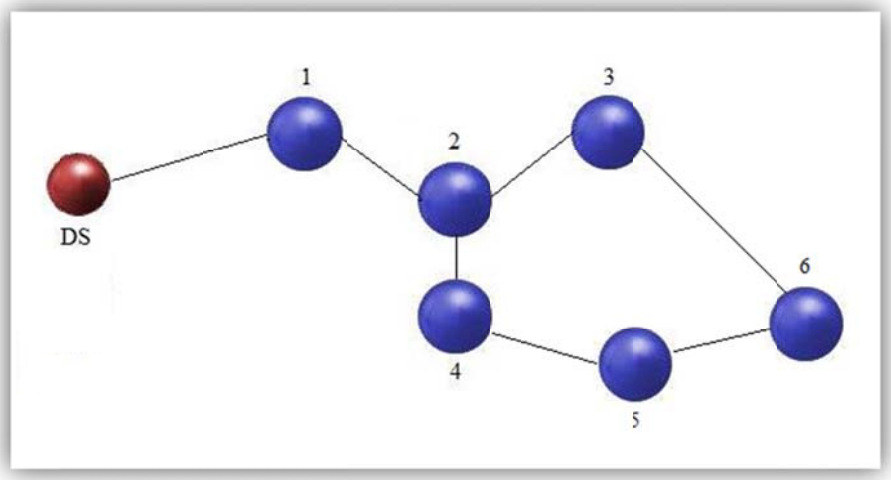


Figura 3.8. Ejemplo de MANET con conexión al servidor de documentos

La Figura 3.9 muestra el intercambio de mensajes que se produciría si el nodo *3* enviara una petición del documento *A* al servidor *DS* usando un mensaje GET y éste respondiera con el documento solicitado usando un mensaje RESP. Los nodos *2* y *1* se encargan de redirigir los mensajes de petición/respuesta hacia sus respectivos destinos.

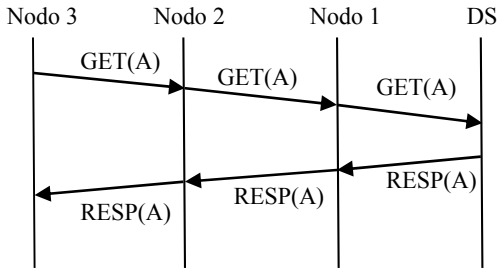


Figura 3.9. Diagrama de mensajes de petición y respuesta para obtener un documento de *DS*

3.4.2 Caché local

Para un nodo perteneciente a una MANET que solicita documentos al servidor de documentos (*DS*), la primera estrategia para reducir el tráfico en la red y el retardo sufrido para obtener los documentos es implementar una caché local que almacene los documentos solicitados.

Formalmente, sea $M = \{w_1, w_2, ..., w_w\}$ el conjunto de w nodos móviles en la red. Sea $U = \{1, 2, ..., n\}$ el universo de n documentos que pueden ser solicitados por los nodos móviles y $s(i)$ el tamaño del documento i con $i \in U$. Se define $TTL(i)$ como el instante de tiempo en el que el documento i caduca y, por lo tanto, su contenido deja de ser válido al quedar obsoleto. Sea $R_j = \{r_{j1}, r_{j2}, ..., r_{jm}\}$ la secuencia de peticiones realizadas por el nodo j, donde r_{jk} representa el documento solicitado por el nodo j en el instante k (considerándose el tiempo discretizado en intervalos constantes), considerando que $r_{jk} \in U$. El destino de las peticiones en R_j es un nodo fijo w_{DS} (*Document Server*) que almacena todos de los documentos pertenecientes a U. El número de servidores de documentos no tiene que ser uno, aunque las peticiones siempre van dirigidas al servidor que tiene almacenado el documento a solicitar.

Si se implementa una caché local en cada uno de los nodos móviles de la MANET, B_{jk} representa el conjunto de documentos almacenados en la caché local del nodo j en el instante k tras resolverse las peticiones de r_{jk}, donde $B_{jk} \subset U$. El conjunto de documentos almacenados en las cachés debe satisfacer las propiedades (3.36) y (3.37):

$$\sum_{\forall i \in B_{jk}} s(i) \leq S_j \tag{3.36}$$

$$\forall i \in B_{jk} \Rightarrow TTL(i) > k \tag{3.37}$$

donde S_j es el tamaño de la caché del nodo j. La condición (3.36) implica que el tamaño de los documentos almacenados en la caché local no puede superar el tamaño reservado para la misma. Por otro lado, la condición (3.37) afirma que los documentos almacenados en la caché no pueden haber caducado y, por lo tanto, ser obsoletos. La secuencia de estados $(B_{j0}, B_{j1}, ..., B_{jm})$ indica los estados de la caché del nodo j conforme las peticiones

contenidas en R_j se resuelven, siendo m el último instante de tiempo de análisis (k=1,2,…,m). B_{j0} es el estado inicial en el que se asume que la caché está vacía y B_{jm} es el estado de la caché cuando todas las peticiones han sido servidas.

Sea $p_{k,ij} = \{w_i, w_u, w_v, ..., w_j\}$ la ruta activa entre el nodo i y el nodo j en el instante de tiempo k, definida como la ruta existente para alcanzar al nodo j desde el nodo i en el instante k. Solamente dos nodos consecutivos en un camino están conectados directamente, es decir, existe una ruta creada entre ellos. Si $p_{k,ij}$=∅ entonces no hay una ruta creada entre el nodo i y el j en el instante k, en otro caso $card(p_{k,ij})\geq 2$, donde $card(p_{k,ij})$ es la cardinalidad del conjunto $p_{k,ij}$. Se define la distancia entre el nodo i y el nodo j en el instante k como:

$$dist(w_i, w_j)_k = card(p_{k,ij}) - 1 \tag{3.38}$$

Esta distancia define el número de saltos necesarios para alcanzar el nodo j desde el nodo i en el instante k. Cuando la distancia entre dos nodos es igual a uno se dice que son nodos vecinos. Por otro lado, se considera que $dist(w_i, w_j)_k = \infty$ si no hay una ruta creada entre los nodos i y j en el instante k ($p_{k,ij}$=∅).

Se define la secuencia de aciertos en caché local en el nodo j como $(h^l_{j1}, h^l_{j2}, ..., h^l_{jm})$, donde h^l_{jk} se define como se muestra en (3.39):

$$h^l_{jk} = \begin{cases} 1 & \text{si } r_{jk} \in B_{j(k-1)} \\ 0 & \text{si } r_{jk} \notin B_{j(k-1)} \end{cases} \tag{3.39}$$

Cuando h^l_{jk}=1 se considera que ha habido un acierto en caché local en el nodo j, en otro caso se considera un fallo en caché local ya que el documento a pedir no se encuentra en la caché local.

La política de reemplazo decide cuál o cuáles documentos deben ser eliminados de la caché local para hacer sitio a los nuevos documentos. Una buena política de reemplazo debería mantener en caché los documentos con una mayor probabilidad de ser solicitados de nuevo en un futuro próximo. Dada una secuencia de peticiones R_j en el nodo j, con una caché de tamaño S_j y un estado inicial B_{j0}, la política de reemplazo produce una secuencia de estados de caché $(B_{j1}, ..., B_{jm})$, con B_{jk} definida en (3.40):

$$B_{jk} = \begin{cases} (B_{j(k-1)} - \varepsilon_{jk}) \cup \{r_{jk}\} & \text{si } r_{jk} \notin B_{j(k-1)} \\ B_{j(k-1)} & \text{en otro caso} \end{cases} \tag{3.40}$$

En el primer caso de la ecuación (3.40), el documento solicitado no estaba almacenado anteriormente al instante k en la caché del nodo j (fallo de caché). El término $\varepsilon_{jk} \subset B_{j(k-1)}$ representa el conjunto de documentos que tendrían que ser eliminados de la caché local para hacer sitio al nuevo documento de la petición r_{jk}. Si hay espacio suficiente para el nuevo documento, entonces ε_{jk} será un conjunto vacío. En el segundo caso se da un

acierto en caché y, por tanto, ésta se mantiene inalterada. En consecuencia, B_{jk} representa el estado de la caché tras ser servida la petición r_{jk}.

Se han propuesto muchas políticas de reemplazo para cachés en *proxies* de Internet y cada una de ellas ha sido diseñada para ser óptima para un tipo de tráfico determinado [Wang,1999] [Podlipnig, 2003], tal y como se vio en el capítulo 2.3.2. Por lo tanto, si se conoce el tipo de tráfico que va a circular por la red se podrá seleccionar la política de reemplazo más adecuada. Desgraciadamente esto no siempre es posible y, por lo tanto, se propone usar la política de reemplazo LRU para el esquema de caché CLIR. Se ha optado por usar LRU ya que se trata de una política de reemplazo simple de implementar, es generalmente bastante efectiva [Hoof,2009], no necesita de ningún parámetro de configuración que deba ser calculado heurísticamente y la carga computacional y de manejo de estructuras de memoria que requiere es muy pequeña. Esta última consideración es muy importante ya que se pretende implementar en dispositivos que tienen una limitada capacidad computacional.

3.4.3 Mecanismo de intercepción de peticiones

El siguiente paso en el esquema de caché CLIR es que los nodos cooperen entre ellos para que respondan a las peticiones de los demás nodos usando sus cachés locales.

3.4.3.1 Intercepción de peticiones a nivel de aplicación

Supongamos que el nodo *2* en la Figura 3.8 solicita el documento *A*. La petición será reenviada usando el protocolo de encaminamiento hacia *DS*, que contestará con el documento *A* usando la ruta inversa. Cuando reciba el documento *A*, el nodo *2* lo almacenará en su caché local. En el caso de que el nodo *3* solicite posteriormente el documento *A* al nodo *DS*, la petición llegará al nodo *2*, que comprobará si tiene una copia válida del documento *A*, por lo que podrá responder directamente al nodo *3* sin necesidad de reenviar la petición al servidor. La Figura 3.10 muestra el intercambio de mensajes del caso comentado.

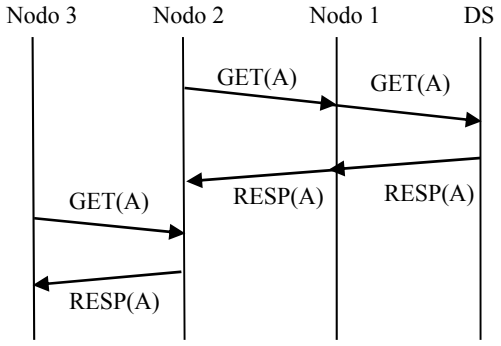


Figura 3.10. Diagrama de mensajes para un ejemplo de intercepción a nivel de aplicación

Esta técnica reduce el número de saltos necesarios para obtener los documentosasí como el retardo observado en obtenerlos ya que se sirven desde nodos que están más cerca que el servidor. Simultáneamente, se reduce la carga del servidor de documentos ya que ni las peticiones ni el tráfico que circula por la red llegan a él.

Se define la secuencia de aciertos en caché por intercepción en el nodo *j* como $(h^i_{j1}, h^i_{j2}, ..., h^i_{jm})$, donde h^i_{jk} se define en (3.41):

$$h^i_{jk} = \begin{cases} 1 & \text{si } r_{jk} \notin B_{j(k-1)} \wedge r_{jk} \in B_{i(k-1)} \mid i \neq j \wedge i \neq DS \wedge w_i \in p_{k,jDS} \\ 0 & \text{en otro caso} \end{cases} \quad (3.41)$$

donde w_i indica un nodo intermedio en la ruta desde el nodo *j* hacia *DS* del mensaje de solicitud del documento solicitado r_{jk}. Cuando h^i_{jk} =1 se considera un acierto de intercepción en caché en el nodo *j*.

3.4.3.2 Intercepción de peticiones usando técnicas intercapa (*cross-layer*)

El procedimiento de intercepción de peticiones a nivel de aplicación funciona exclusivamente cuando se conoce la ruta desde el origen de la petición hasta el nodo *DS*. Sin embargo, este requisito no siempre es posible satisfacerlo en una red móvil inalámbrica. Supongamos la situación en la que el nodo *1* de la Figura 3.8 se mueve ahora fuera del área de cobertura del nodo *DS* y, por lo tanto, el nodo *2* no puede acceder al nodo *DS*, tal y como se muestra en la Figura 3.11. Los nodos *3* y *4*, por su parte, continúan en el área de cobertura del nodo *2*.

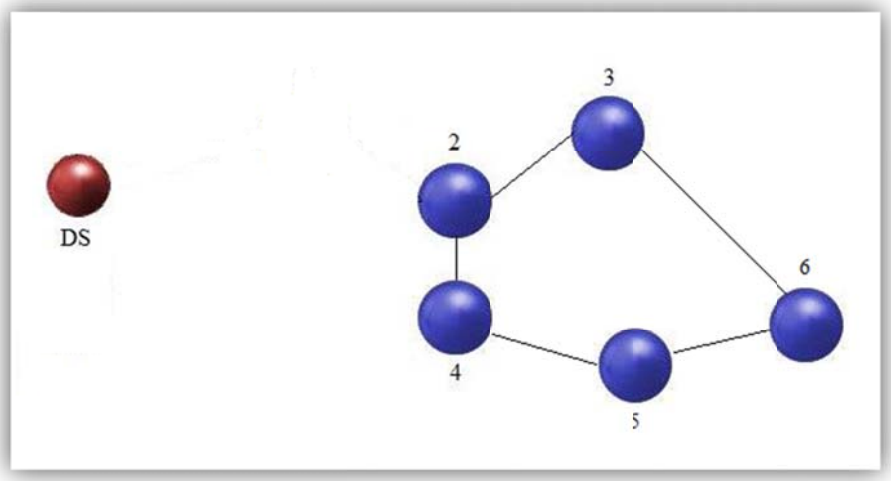


Figura 3.11. Red MANET sin conexión al servidor de documentos

Si el nodo *4* pidiera el documento *A* a *DS*, el nodo detectaría que no hay una ruta creada hacia *DS*. Bajo estas circunstancias, el nodo *4* iniciaría el procedimiento de creación de una nueva ruta hacia *DS*. Sin embargo, no hay forma de crear una ruta desde el nodo *4* hacia *DS*, por lo que el protocolo de encaminamiento terminará sin poder crear la ruta y,

por lo tanto, la petición del documento *A* no será cursada aunque el nodo *2* tenga una copia válida del mismo en su caché local.

Para evitar este problema se propone involucrar al protocolo de encaminamiento en el proceso de búsqueda de los documentos en la MANET en el denominado procedimiento de Intercepción *Cross-Layer* o Intercepción Intercapa. La técnica *cross-layer* permite compartir información entre capas del modelo de comunicaciones OSI saltándose sus restricciones. De esta forma se puede compartir información entre capas no adyacentes con la finalidad de optimizar el rendimiento de una red. Recientemente la técnica *cross-layer* está siendo adoptada en redes inalámbricas con el objetivo de optimizarlas dado las restricciones inherentes de las mismas [Shakkottai,2003] [Lixin,2009].

Supongamos que se emplea el protocolo de encaminamiento AODV (*Ad hoc On-Demand Distance Vector*) ya que es bastante empleado en los estudios sobre redes ad hoc y del que existen diversas implementaciones para diversos sistemas operativos. Debido a que AODV emplea mensajes RREQ (*Route REQuest*) en modo *broadcast* para crear las rutas hacia *DS*, se propone que dicho mensaje incluya información del documento a solicitar, es decir, el identificador del documento se transporta dentro del mensaje RREQ. En nuestro ejemplo, el nodo *4* enviará el mensaje RREQ para crear la ruta hacia *DS* con el identificador del documento *A* adjunto. Cuando el nodo *2* recibe el mensaje RREQ, éste comprueba si hay información del documento en el mensaje. Si es así, extrae el identificador del documento y comprueba si tiene una copia válida del mismo en su caché local. En caso afirmativo, el nodo *2* responde al nodo *4* usando el mensaje RREP (*Route REPly*) incluyendo también el identificador del documento. Cuando el nodo *2* recibe el mensaje RREP y, por lo tanto, se ha creado la ruta entre los nodos *4* y *2*, el nodo *4* envía la petición directamente hacia el nodo *2*. En el caso de que el nodo *2* no tuviera una copia del documento *A* reenviaría el mensaje RREQ para seguir buscando la ruta hacia *DS* y el documento *A*. La Figura 3.12 muestra el intercambio de mensajes para el ejemplo comentado anteriormente.

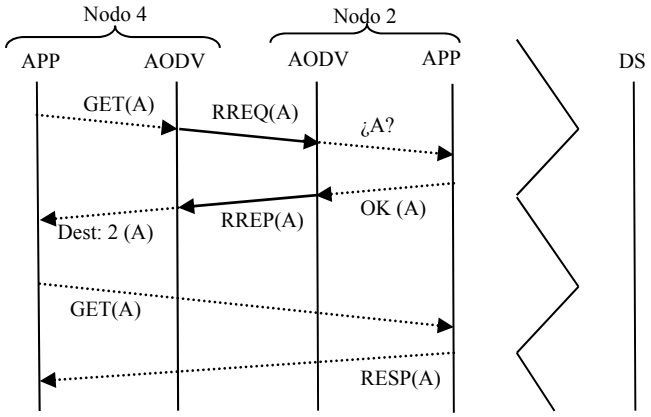


Figura 3.12. Diagrama de mensajes para un ejemplo de intercepción de peticiones usando *cross layer*

Usando este procedimiento, los nodos en la MANET pueden acceder a los documentos diseminados por la red incluso si el servidor de documentos está temporalmente inaccesible o encontrar los documentos en nodos más cercanos que el *DS* en el proceso de búsqueda de una ruta hacia el *DS*.

Aunque otros esquemas de caché como *MobEye* y *SimpleSearch* usan un método similar basado en mensajes de *broadcast* para realizar las peticiones, cuando un nodo recibe una petición éste tiene que crear la ruta inversa hacia el nodo solicitante para enviarle un mensaje *unicast* informándole de la localización del documento. Si hay varios nodos que tienen una copia del documento entonces el protocolo de encaminamiento también habrá de crear varias rutas, siempre mediante costosos mensajes RREQ, aunque sólo una de ellas sea necesaria. Esta operación incrementa innecesariamente la carga de tráfico en la red. Además, la creación de las rutas inversas toma su tiempo, mientras que nuestra propuesta aprovecha la creación de las propias rutas para localizar los documentos, por lo que la operación es mucho más rápida.

Para implementar la intercepción de peticiones usando esta técnica *cross layer* se requiere la modificación de los mensajes del protocolo de encaminamiento. Para el caso que nos ocupa, resulta necesario modificar los mensajes RREQ y RREP de AODV. La Figura 3.13 muestra el formato modificado del mensaje RREQ de AODV.

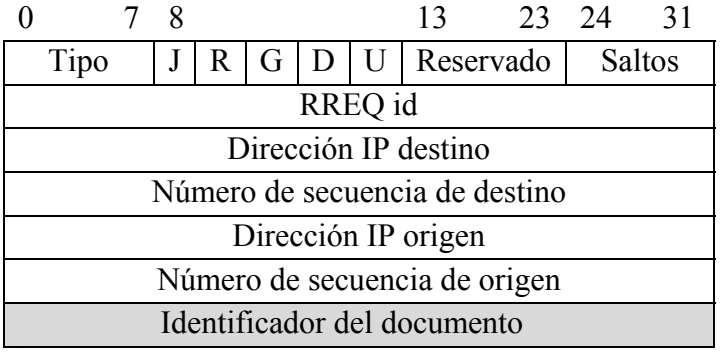


Figura 3.13**.** Mensaje RREQ de AODV modificado

El mensaje RREQ modificado incluye el identificador del documento solicitado, suponiendo que éste es un número de 32 bits. Para posibilitar la inclusión de direcciones URL en los mensajes RREQ se propone usar el espacio reservado del mensaje RREQ para incluir el tamaño de la URL a enviar. La Figura 3.14 muestra el mensaje RREQ modificado para admitir direcciones URL de tamaño variable. Al ser el campo donde se describe el tamaño de 11 bits, es posible direccionar URLs de 2^{11}=2048 bytes que resulta más que suficiente.

Por otro lado, se define la secuencia de aciertos en caché de intercepción durante la creación de la ruta en el nodo *j* por el nodo *i* como $(h^{rci}_{j1}, h^{rci}_{j2}, ..., h^{rci}_{jm})$, donde h^{rci}_{jk} se define como:

$$h_{jk}^{rci} = \begin{cases} 1 & \text{si } r_{jk} \notin B_{j(k-1)} \wedge r_{jk} \in B_{i(k-1)} \mid i \neq j \wedge i \neq DS \wedge p_{k,jDS} = \varnothing \\ 0 & \text{en otro caso} \end{cases} \quad (3.42)$$

donde *i* es el nodo que contesta a la petición de creación de ruta por parte del nodo *j* hacia el servidor *DS* porque tiene una copia válida del documento solicitado en su caché local cuando recibe el mensaje *RREQ*. Cuando h^{rci}_{jk}=1 se considera un acierto en caché por intercepción al crear la ruta en el nodo *j*. Como $dist(w_j, w_i)_k < dist(w_j, w_{DS})_k$, la cantidad de saltos necesaria para satisfacer la petición r_{jk} disminuye así como el retardo percibido al obtener el documento. Además, el documento solicitado puede ser obtenido incluso si no hubiera forma de crear una ruta hacia *DS* porque estuviera inaccesible.

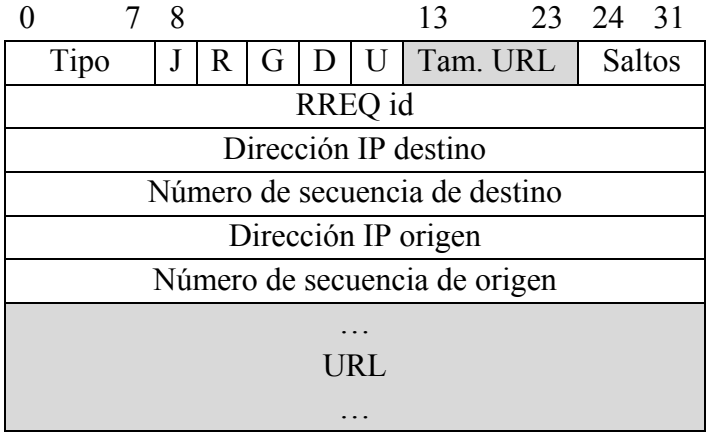


Figura 3.14**.** Mensaje RREQ de AODV modificado para aceptar URLs de tamaño variable

3.4.4 Redirección de peticiones

Se propone implementar en todos los nodos de la red una Caché de Redirecciones en la que se almacene información acerca de la localización de los documentos en la red. Esta Caché de Redirecciones se diferencia de otras propuestas similares de otros esquemas en la literatura en la manera cómo se administra dicha caché.

Cada entrada o registro de la Caché de Redirecciones está formada por una *tupla* de la forma:

(*id*, *IP_GET*, *hops_GET*, *TTL_GET*, *IP_RESP*, *hosp_RESP*, *TTL_RESP*)

donde *id* es el identificador del documento al que corresponde la información del registro, *IP_GET* es la dirección IP del nodo que realizó la petición del documento *id*, *hops_GET* es la distancia en saltos a la que se encuentra dicho nodo y *TTL_GET* es el instante en el que caduca el documento *id*. De forma análoga, los campos *IP_RESP*, *hops_RESP* y *TTL_RESP* hacen referencia a la dirección IP, la distancia en saltos y el TTL con respecto al nodo que respondió a la petición (si éste no es un servidor).

Supongamos la situación inicial de la red de la Figura 3.8, en la que el nodo *3* solicita el documento *B* a *DS*. La petición pasará a través de los nodos *2* y *1*. Estos crearán una entrada en la Caché de Redirecciones con el identificador *id B* y anotarán en el campo

IP_GET la dirección IP del nodo *3*. Análogamente, los nodos *2* y *1* anotarán en el campo *hops_GET* que el nodo *3* se encuentra a uno y dos saltos respectivamente. El campo *TTL_GET* permanecerá indefinido ya que no se conoce aún. Cuando *DS* responde con el documento usando la ruta inversa (*DS-1-2-3*), el valor del campo *TTL_GET* se actualizará en los nodos *1* y *2* con el tiempo de expiración del documento. Los campos *IP_RESP*, *hops_RESP* y *TTL_RESP* permanecen indefinidos ya que la respuesta vino de un servidor. Si tras esta operación el nodo *4* solicitara el documento *B* a *DS*, la petición alcanzaría el nodo *2*, que comprobará en su Caché de Redirecciones si hay una entrada para el documento *B*. Al existir una entrada para el documento *B* que indica que el nodo *3* está a un salto mientras que el nodo *DS* se encuentra a dos saltos y, por lo tanto, más lejos, el nodo *2* decidirá redireccionar la petición hacia el nodo *3*. El nodo *3*, al recibir la petición, comprobará si tiene el documento *B* almacenado en su caché local. Si es así responderá al nodo *4* con dicho documento. La Figura 3.15 muestra el diagrama de intercambio de mensajes para la situación comentada anteriormente.

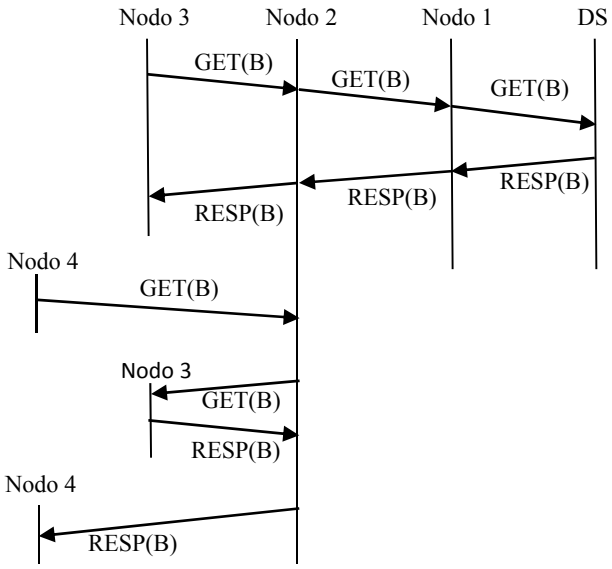


Figura 3.15. Diagrama de mensajes para un ejemplo de redirección de peticiones

En esta situación, si el nodo *2* solicitara el documento *B*, primero comprobaría que no lo tiene en su caché local y después comprobaría si tiene una entrada en la Caché de Redirecciones. En dicha caché tiene anotado que el nodo *4* pidió el documento y que se encuentra a un salto y que el nodo *3* respondió con el documento y también se encuentra a un salto. Ya que *DS* se encuentra a dos saltos, el nodo *2* redireccionará directamente su petición hacia el nodo que se encuentre más cerca. Como ambas opciones se encuentran a la misma distancia, se elegirá aquella opción que tenga un TTL más alto, ya que la probabilidad de que el documento haya sido eliminado de la caché es menor.

Los campos *IP_RESP*, *hops_RESP* y *TTL_RESP* sólo se rellenan cuando el nodo que responde no es un servidor, es decir, cuando hay una intercepción o redirección de peticiones. La información asociada a un determinado documento se considera obsoleta cuando el valor de los campos *TTL_GET* o *TTL_RESP* así lo indica. En estos casos se

elimina la información asociada en función de cuál esté obsoleta, la relacionada a la petición o a la respuesta. Si las dos estuvieran caducadas o sólo hubiera información de una de ellas y estuviera caducada, se elimina el registro completo de la Caché de Redirecciones. Cuando llega una petición o una respuesta a un nodo y hay que actualizar la Caché de Redirecciones prevalecerá la información del nodo que se encuentre más cercano. En caso de igualdad prevalecerá aquella información que tenga un mayor tiempo de vida.

Se definen varias restricciones para el procedimiento de redirección:

1. Una petición sólo puede ser redireccionada una vez.
2. Una petición es redireccionada únicamente si el tiempo de expiración asociado (*TTL_GET* o *TTL_RESP*) es conocido. Es decir, hay garantías de que se ha producido la respuesta.
3. Una petición puede ser redireccionada si se cumple alguna de las siguientes restricciones:
    a) El número de saltos hacia el nodo a redireccionar la petición es menor que el número de saltos al destino de la petición (normalmente *DS*).
    b) El número de saltos hasta el nodo destino es desconocido.
    c) El número de saltos hacia el nodo a redireccionar es desconocido (porque no existe ruta hacia él) pero el número de saltos almacenado en la Caché de Redirecciones para dicho nodo es menor que la distancia hacia el nodo destino y, además, el nodo destino está situado a más de dos saltos de distancia.

La restricción 1 previene la formación de bucles en el envío de las peticiones causados por múltiples redirecciones. La restricción 2 evita la redirección de peticiones hacia nodos que no está garantizado que hayan recibido el documento solicitado. Supongamos, en la Figura 3.8, que el nodo *6* solicita el documento *C* a *DS* usando la ruta *6-5-4-2-1-DS* y *DS* responde con el documento usando la ruta *DS-1-2-3-6*. En este estado, los nodos *4* y *5* han anotado en su Caché de Redirecciones que el nodo *6* solicitó el documento *C* aunque no tienen constancia de que lo haya recibido ya que la respuesta no ha pasado por ellos. Esta restricción evita también redirecciones incorrectas porque se hayan perdido mensajes de petición o respuesta por saturación de la red. La restricción 3 define las condiciones que se deben satisfacer para que un petición sea redireccionada: la condición a) asegura que si la distancia en saltos hacia el nodo al que se redirecciona y el destino de la petición son conocidos, la petición será enviada a aquél que se encuentre más cerca; si se cumple la condición b) la petición es redireccionada porque no se conoce la distancia hacia el nodo destino de la petición (usualmente *DS*) ya que no hay una ruta creada hacia él y puede que no sea accesible; para terminar, la condición c) redirecciona una petición si la distancia al nodo a redireccionar es desconocida (porque no hay una ruta creada hacia él) aunque, de acuerdo a la Caché de Redirecciones, está más cerca que el nodo destino de la petición (que se sabe que está como mínimo a dos saltos). La petición es

redireccionada incluso si el nodo al que se redirige no está en el vecindario, con la esperanza de encontrarlo más cerca que el nodo destino. La restricción 3 asegura que el tráfico inducido por el mecanismo de redirección es menor que cuando no se aplica.

La Caché de Redirecciones se administra usando dos listas LRU: la lista *KNOWN_TTL* contiene las entradas relacionadas con documentos cuyo TTL se conoce, mientras que la lista *UNKNOWN_TTL* contiene las entradas relativas a documentos con valor del TTL desconocido. El espacio asociado a cada una de las listas se asigna dinámicamente, mientras que la memoria total reservada para ambas listas es constante. Si alguna entrada de la lista *UNKNOWN_TTL* se actualiza con el TTL del documento, ésta será traspasada a la cabeza de la lista *KNOWN_TTL*. Cuando hay que almacenar una nueva entrada y no hay espacio suficiente porque la memoria reservada está llena, se elimina la entrada que hace más tiempo que no se referencia de la lista *UNKNOWN_TTL* o, si esta lista está vacía, se elimina la entrada más antigua de la lista *KNOWN_TTL*. Una entrada de la lista también se borra si toda la información asociada a los TTL ha expirado.

Otros trabajos anteriores en la literatura con políticas similares sólo tienen en cuenta el TTL asociado a los documentos como tiempo de expiración para la información acerca de las redirecciones. De esta forma, un nodo podría redireccionar una petición basándose en el TTL del documento aunque éste haya sido eliminado de la caché correspondiente antes de que quede obsoleto. Para tener en cuenta también la eliminación de los documentos de las cachés locales se propone usar como tiempo de validez de la información de la Caché de Redirecciones el valor mínimo entre el TTL del documento y la media de tiempo que los documentos permanecen en las cachés locales. Este valor puede ser calculado por cada nodo observando el comportamiento de su propia caché local. De esta forma, cada nodo calcula el tiempo transcurrido desde que cada documento es almacenado en la caché y el instante en el que es eliminado. Este tiempo se emplea para estimar el tiempo medio que los documentos permanecen almacenados en las cachés de los otros nodos de la MANET (suponiendo que todos muestran un comportamiento similar). Usando este mecanismo se disminuye la probabilidad de redireccionar peticiones hacia nodos que ya han eliminado el documento de su caché local. Ya que pueden darse casos en los que este procedimiento puede fallar, se propone enviar un mensaje de error cuando un nodo recibe un mensaje de petición redireccionado y ya no tiene el documento solicitado en su caché local. Concretamente, el nodo que recibe la petición responderá con un mensaje REDIRECTION_ERROR al nodo que redireccionó la petición, lo que permitirá actualizar la información asociada en la Caché de Redirecciones. Este mensaje obliga al nodo que realizó la redirección a enviar la petición al nodo destino original de la petición. El uso del mensaje de informe de error de redirección evita las redirecciones erróneas reiteradas a nodos que han eliminado el nodo solicitado de su caché local antes del tiempo estimado por la Caché de Redirecciones.

Como ejemplo, supongamos la situación de la red de la Figura 3.8, en la que el nodo *3* solicita el documento *D* al servidor *DS* siguiendo la ruta *3-2-1-DS* y éste responde con el documento usando la ruta inversa. El nodo *2* anotaría en su Caché de Redirecciones que el nodo *3* tiene una copia del documento *D*. Si, posteriormente, el nodo *4* solicitara

también el documento *D* al servidor, pero el nodo *3* lo hubiera eliminado de su caché local antes de que caducara el registro de la Caché de Redirecciones, la petición sería redireccionada desde el nodo *2* al *3*. El nodo *3*, al recibir la petición, comprobaría que no tiene una copia del documento *D* en su caché local, por lo que enviaría el mensaje REDIRECTION_ERROR al nodo *2*, que remitiría la petición a *DS*. Finalmente, el servidor respondería con el documento *D* al nodo *4*. El diagrama de la Figura 3.16 ilustra este ejemplo.

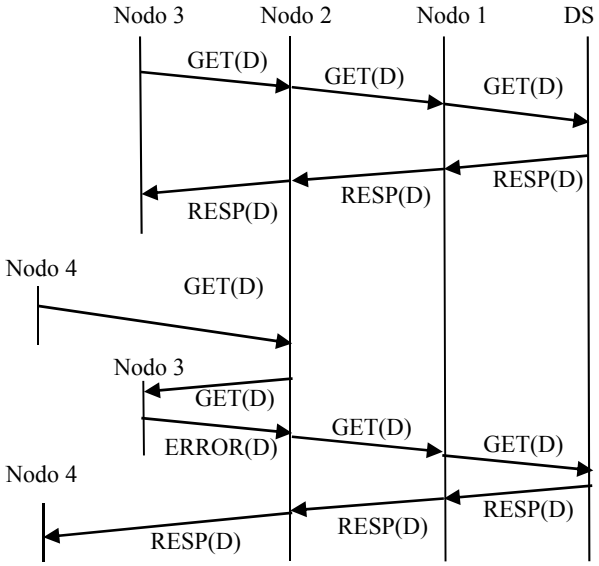


Figura 3.16. Diagrama de mensajes para un ejemplo de error de redirección

El esquema CLIR también implementa el mecanismo propuesto en [Wang,2006] que sugiere, como método de diseminación, que un nodo *C* situado en el centro de la ruta entre el nodo *S* origen de la respuesta que contiene el documento *d* y el nodo destino de la respuesta *D* también almacene el documento *d* en su caché local. Este método replica los documentos por la red para incrementar su disponibilidad aunque, en el caso particular de CLIR, esta acción sólo se lleva a cabo si la distancia desde el nodo que solicita el documento al nodo que responde es al menos de cuatro saltos. Esta restricción evita la replicación excesiva de documentos en los nodos vecinos. La Caché de Redirecciones también anota la información relativa al nodo intermedio donde se almacena el documento replicado. De esta forma, como se muestra en la Figura 3.17, se divide la ruta entre los nodos *S* y *D* en tres partes: los nodos situados en la primera mitad de la ruta entre los nodos *S* y *C* anotarán en su Caché de Redirecciones que el nodo *S* tiene el una copia del documento (si el nodo *S* no es un servidor); los nodos situados entre la segunda mitad de la ruta entre los nodos *S* y *C* y la primera mitad de la ruta ente *C* y *S* anotarán que el nodo *C* tiene una copia del documento *d*; finalmente, los nodos situados entre la segunda mitad de la ruta entre los nodos *C* y *D* y el nodo *D* anotarán que el nodo *D* tiene el documento *d*. Con esta información la próxima petición al documento *d* podrá ser enviada directamente o redireccionada a otros nodos que se sabe están más cerca que el nodo *DS*. La Figura 3.17 ilustra este ejemplo. Los nodos situados en la denominada "Zona 1" almacenarán en su Caché de Redirecciones que el nodo *S* (si no es un servidor) tiene almacenado el

documento solicitado. Por otra parte, los nodos de la "Zona 2" anotarán que el nodo *C* tiene almacenado el documento. Por último, los nodos de la "Zona 3" anotarán que el nodo *D* tiene el documento solicitado.

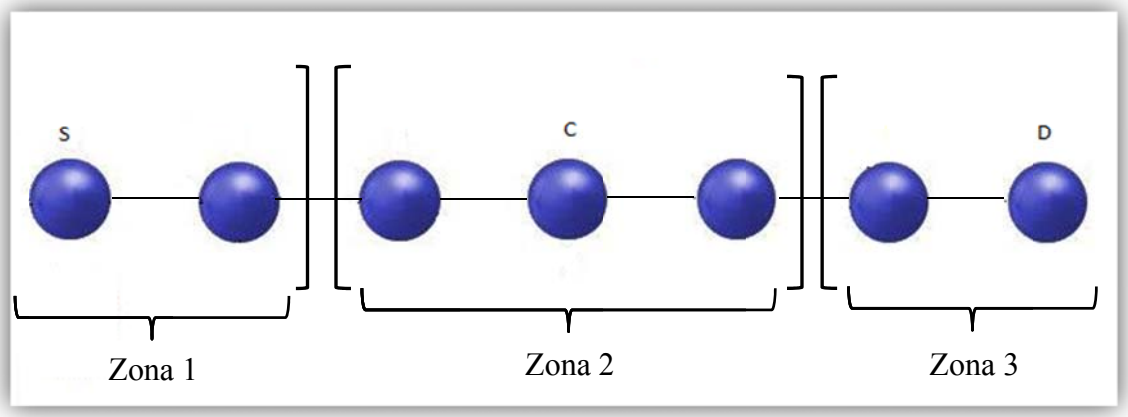


Figura 3.17. Distribución de documentos en la red

Si se implementa una Caché de Redirecciones en cada uno de los nodos móviles de la MANET, BR_{jk} representa la información almacenada en el nodo *j* en el instante *k* asociada a la situación y tiempo de vida de ciertos documentos del universo *U*. BR_{j0} será el estado inicial de la Caché de Redirecciones cuando ésta se encuentra vacía. Se define la secuencia de aciertos de redirección en el nodo *j* como $(h^{rd}_{j1}, h^{rd}_{j2}, ..., h^{rd}_{jm})$ donde h^{rd}_{jk} se define en (3.43):

$$h^{rd}_{jk} = \begin{cases} \text{si } r_{jk} \notin B_{j(k-1)} \wedge r_{jk} \notin B_{i(k-1)} \wedge r_{jk} \in BR_{i(k-1)} \wedge r_{jk} \in B_{r(k-1)} \mid \\ 1 \quad i \neq j \neq r \neq DS \wedge w_i \in p_{k,jDS} \\ \quad \wedge \exists p_{k,ir} \wedge \exists p_{k,rj} \wedge \ dist(w_i, w_r)_k < dist(w_i, w_{DS})_k \\ 0 \quad en\ otro\ caso \end{cases} \tag{3.43}$$

donde w_i representa un nodo intermedio en la ruta de la petición r_{jk} desde el nodo *j* a *DS* que además no tiene una copia válida del documento solicitado en su caché local, y que tiene información acerca de que el documento solicitado se encuentra en el nodo *r*, redireccionando la petición hacia él. La redirección se va a producir tanto si el nodo *r* se encuentra en la ruta del nodo *j* a *DS* como si no, siempre que se pueda establecer con éxito una ruta entre los nodos *i* y *r*, y además cuando el servidor de datos *DS* se encuentre a mayor distancia del nodo *i* que el nodo *r*, o lo que es lo mismo, cuando el valor de $dist(w_i, w_r)_k$ sea inferior a $dist(w_i, w_{DS})_k$. Además, se deberá poder crear una ruta (si no existía) desde el nodo *r* hasta el nodo *j* para que éste pueda enviar la respuesta. Cuando $h^{rd}_{jk}=1$, se considera que se ha producido un acierto de redirección en el nodo *j*.

3.5 Evaluación de CLIR en redes móviles ad hoc

En este apartado se evalúa el rendimiento de CLIR en redes MANET comparándolo con el rendimiento obtenido por los esquemas de caché MobEye, SimpleSearch (SS), COOP, DPIP e HybridCache (HC). Además, se compara con el caso de una MANET en la que no se usa ninguna técnica de caché.

Para la elección de estos esquemas de caché para comparar con CLIR se descartaron los esquemas MSS, GROCOCA y CC al necesitar un receptor GPS en cada nodo para poder operar, lo que supondría una comparación en desigualdad de condiciones. Por otro lado, el esquema de Moriya y de Sailhan fueron desechados ya que el primero no especifica la política de reemplazo en caché local y el segundo, aunque sí que la especifica, no asigna valores a ninguna de las constantes necesarias para valorar los documentos en caché local. El esquema ZC tampoco especifica el valor de una de sus variables para la ponderación de documentos en su caché local. Por otro lado, los esquemas de Gianuzzi y Wang no son evaluados en sus respectivos artículos, por los que no se consideró hacerlo nosotros. Los esquemas IXP, CacheData y CachePath son los predecesores de DPIP (para IXP) e HybridCache (para CacheData y CachePath), por lo que, según se demuestra en sus respectivos artículos resultan peores que los esquemas en los que evolucionan. El esquema de Cho, por su parte, se centra en diseminar los documentos que son extraídos de la caché local para almacenarlos en otra caché, de modo que no resulta interesante como esquema de caché completo. Por otro lado, POACH y Hamlet se centran en diseminar documentos por la red y en el cálculo del tiempo que los documentos tienen que estar en la caché respectivamente, por lo que tampoco fueron considerados. En COCA se requiere también que el servidor disemine documentos por la red, lo que implica una cierta inteligencia en el mismo. En el esquema de caché de Denko no se especifica cómo se asignan roles a los nodos, por lo que la elección de los mismos puede resultar crucial para su rendimiento. De los restantes esquemas de caché, se eligió a MobEye y SimpleSearch como representativas de los esquemas de caché basados en *broadcast*, a COOP y DPIP (descartando a ORION) como representativas de los esquemas de caché basados en *broadcast* y uso de información, a HybridCache como representativa de los esquemas basados en peticiones dirigidas y uso de información (descartando GroupCaching) ya que es el esquema de caché con el que se comparan otros esquemas de caché. Finalmente, se barajó la posibilidad de implementar COACS para comparar CLIR con algún esquema de caché basado en roles. Sin embargo se desechó esta posibilidad dado que los resultados que se presentan en el artículo asociado muestran un retardo obtenido por COACS, en igualdad de condiciones de simulación, en torno al doble del obtenido por CLIR.

3.5.1 Modelo del sistema

Para realizar el estudio del rendimiento de los esquemas de caché se ha usado el simulador de redes NS-2.33, en el que se han implementado, además del esquema de caché CLIR, los esquemas de caché MobEye, HybridCache, COOP, DPIP y SimpleSearch. En

las figuras que se muestran en este apartado, cada punto corresponde a la media de los resultados obtenidos en cinco simulaciones utilizando los mismos parámetros de simulación pero con diferentes semillas. Cada simulación varía los parámetros relativos a la movilidad, el punto inicial y final de los sucesivos movimientos de los nodos. Dependiendo de la simulación, se modifica el parámetro objeto de estudio mientras que el resto de parámetros se fijan a un valor por defecto. Las figuras incluyen el intervalo de confianza al 95% para cada uno de los parámetros de rendimiento medidos.

La Tabla 3.8 resume los principales parámetros utilizados en las simulaciones. Dichos parámetros han sido seleccionados usando los valores más comunes empleados en la evaluación de esquemas de caché, tal como se estudió en el apartado 3.3.

Parámetro	**Valor por defecto**		**Otros valores estudiados**
Área de simulación (metros x metros)	1000x1000		
Nodos	50		25-50-75-100
Servidores	2		
Número de documentos	1000		
Tamaño de los documentos (bytes)	1000		
Timeout de las peticiones (s)	3		
TTL (s)	2000		250-500-1000-2000-∞
Tiempo medio entre peticiones (s)	25		5-10-25-50
Modelado de la popularidad de los documentos (pendiente Zipf)	0.8		0.4-0.6-0.8-1.0
Política de reemplazo	LRU		
Tamaño de caché local	35		5-10-35-50
Tamaño de Caché de Redirecciones (registros)	35		
Tiempo de simulación (s)	20000		
Tiempo de calentamiento (s)	4000		
Protocolo MAC	802.11 b		
Modelo de propagación radio	*Two Ray Ground*		
Radio de cobertura (metros)	250		
Protocolo de encaminamiento	AODV	Reparación local RREP intermedios	
Patrón de movilidad	TVCM		RWP
	Velocidad constante: 1 m/s Tiempo de pausa: 0 s		Velocidad constante: 1-3-5 m/s

Tabla 3.8. Principales parámetros de simulación para redes móviles

Se supone un caso por defecto de 50 nodos móviles distribuidos en un área de simulación de 1000 metros por 1000 metros. Se estudia también el rendimiento en una red con 25, 75 y 100 nodos para estudiar la influencia de la densidad de nodos en la red. Los nodos móviles siguen el patrón de movilidad RWP modificado con una velocidad constante por defecto de 1 m/s sin pausa un vez llegan a su destino. También se evalúan los escenarios con velocidades constantes de 3 m/s y 5 m/s para estudiar la influencia de la velocidad de los nodos en el rendimiento. Para considerar un patrón de movimiento más realista también se ha usado el modelo de movilidad TVCM con los mismos parámetros.

Se consideran 1000 documentos diferentes (identificados mediante un número) distribuidos entre dos nodos Servidores de Documentos (*Document Server - DS*) sin movilidad en la red. Los servidores están situados en las posiciones (x,y)=(0,500) y (x,y)=(1000,500) dentro del área de simulación (siendo x e y coordenadas expresadas en metros). Cada *DS* está conectado a un *Gateway* o a una red externa. Para distribuir el tráfico entre los *DS*, los documentos con un identificador par se almacenan en un servidor mientras que los documentos con identificador impar se almacenan en el otro servidor. Además, cada documento tiene un TTL asociado que determina cuándo expira y es considerado obsoleto. Los documentos almacenados en caché que han expirado son eliminados de las mismas para no ocupar espacio innecesario. El TTL de los documentos sigue una distribución exponencial con medias entre 250 y 2000 segundos (dependiendo del experimento). De esta forma se modela una alta y baja variabilidad del tiempo de vida de los documentos. Además, se analiza en caso de un TTL de los documentos infinito, es decir, los documentos nunca expiran. El tamaño de todos los documentos es constante e igual a 1000 bytes.

Cada nodo que no es un servidor está programado para generar peticiones a los servidores durante el tiempo de simulación. Cuando una petición es servida, se genera otra petición una vez transcurrida una determinada cantidad de tiempo. El tiempo de espera entre la recepción de una respuesta y la próxima petición sigue una distribución exponencial con medias entre 5 y 50 segundos (el valor por defecto es de 25 segundos). Modificando el tiempo entre peticiones se puede evaluar un amplio rango de actividad de los nodos y, en consecuencia, de la carga de tráfico en la red. Los documentos son solicitados de nuevo a los servidores si la respuesta no es servida antes de un *timeout* constante cuyo valor ha sido fijado en 3 segundos. El patrón de elección de documentos para las peticiones sigue una distribución Zipf en la que, dependiendo del experimento, se han seleccionado valores entre 0.4 y 1.0 para la pendiente α, siendo el valor por defecto 0.8.

Cada nodo implementa una caché local que emplea la política de reemplazo LRU. Todos los nodos de la red tienen el mismo tamaño de caché, con un valor por defecto que les permite almacenar 35 documentos (35000 bytes). Para evaluar la influencia del tamaño de las cachés, también se han usado tamaños de caché de 5, 10 y 50 documentos. Para evitar la influencia de los fallos en caché al estar éstas vacías al inicio de la simulación [Dykes,2000], se ha definido un tiempo de "calentamiento" (*warm-up*) usando el primer 20% del tiempo de simulación. Dado que el tiempo de simulación se ha establecido en 20000 segundos, el tiempo de "calentamiento" de las cachés es de 4000 segundos. Consecuentemente, las estadísticas almacenadas en las simulaciones corresponden al tiempo transcurrido tras el tiempo de calentamiento. A la Caché de Redirecciones se le ha reservado una capacidad para almacenar 35 entradas.

Como protocolo MAC se ha usado 802.1b empleando el modelo de propagación *Two-Ray Ground* y un radio de cobertura de 250 metros. Como protocolo de encaminamiento se ha elegido AODV, con las características de "Reparación Local" y "RREP Intermedio" habilitadas.

Para el resto de esquemas de caché evaluados se han seleccionado los parámetros propuestos por sus autores, a excepción de aquéllos que no han sido definidos, a los que se le ha asignado un valor discrecional:

- HybridCache (HC): Los umbrales para la redirección y el parámetro TTL de los documentos se han fijado a dos saltos y 2000 segundos respectivamente. El tamaño de ventana *K* tiene un valor de 2 y cada nodo es capaz de almacenar información sobre la localización de todos los documentos en la red. Es decir, se supone un espacio de almacenamiento infinito para dicha información.
- COOP: La tabla RRT se ha dimensionado para almacenar 35 entradas, ya que es el mismo valor con el que se ha configurado la Caché de Redirecciones de CLIR y COOP no define ningún tamaño. Por la misma razón, el tiempo que hay que esperar una respuesta una vez que se ha hecho una petición en *broadcast* se ha definido como la mitad del *timeout* definido en CLIR, es decir, 1.5 segundos.
- DPIP: La tabla *Index Vector* es capaz de almacenar información sobre todos los documentos de la red y el *DPIP_Timer* es de 150 ms tal y como propone DPIP.
- SimpleSearch (SS): Se ha seleccionado la política de reemplazo TDS_T ya que es aquella que obtiene los mejores resultados. La política de admisión establece que no se almacene en caché local aquellos documentos que son servidos por un nodo que esté situado a menos de cinco saltos de distancia del nodo receptor.
- MobEye (ME): No necesita ningún parámetro.
- No Caché (NC): Se ha implementado usando los mimos mensajes de petición y respuesta que usa CLIR pero sin el uso de cachés.

Para comparar el rendimiento de los esquemas de caché se van a usar seis métricas: el número total de documentos servidos, el tráfico generado (en kB), el retardo, el porcentaje de *timeouts*, el porcentaje de aciertos en caché y el porcentaje de aciertos de redirección. Todas ellas han sido definidas en el apartado 3.2.3 excepto el porcentaje de aciertos de redirección, que se define como el porcentaje de redirecciones que han sido resueltas satisfactoriamente, porque el documento ha llegado a su destino tras una redirección, en relación al número total de redirecciones realizadas.

3.5.2 Evaluación en función de la velocidad de los nodos

En este conjunto de experimentos se varía la velocidad de los nodos para estudiar cómo se comportan los esquemas de caché cuando se modifica la topología de la red. Los cambios en la topología afectan a las cachés de redirección ya que la información que almacenan puede convertirse en obsoleta.

Las Figuras 3.18a y 3.19a muestran el número medio de documentos recibidos por nodo en el tiempo de simulación en función de la velocidad de los nodos para los patrones de movilidad TVCM y RWP respectivamente. Como puede observarse, DPIP es el esquema de caché que obtiene el menor número de documentos, incluso menos que la opción que no usa cachés. La causa es el alto porcentaje de *timeouts* (Figuras 3.18d y 3.19d) y el bajo porcentaje de aciertos de redirección (Figuras 3.18f y 3.19f). De hecho, conforme se incrementa la velocidad de los nodos, la cantidad de documentos obtenidos se incrementa ya que el porcentaje de aciertos de redirección también se incrementa. Esto es debido a la mala gestión que realiza DPIP de su tabla *IV*, ya que los nodos anotan que un nodo va a tener el documento que solicita una vez que les llega la petición, independientemente de que realmente lo reciba. Esto causa que, en el caso de desconexiones con el servidor, los nodos tengan información errónea de la localización de los documentos en su vecindario. Al incrementarse la velocidad de los nodos esta información se elimina de la tabla *IV* con más frecuencia al verse modificada la lista de nodos vecinos, con lo que la probabilidad de aciertos de redirección también se incrementa. Este comportamiento es un factor común en todos los estudios realizados y, por lo tanto, DPIP no se mostrará en las figuras relacionadas con la media de documentos obtenidos para, con ello, hacer el análisis más sencillo. Por otro lado, MobEye, CLIR y COOP obtienen una cantidad similar de documentos para todas las velocidades consideradas. SimpleSearch e HybridCache consiguen el menor número de documentos en comparación con los demás esquemas de caché, excepto para baja velocidad (1 m/s), donde la diferencia es menor. La cantidad de documentos obtenida es prácticamente igual tanto en TVCM como en RWP para todos los esquemas de caché.

Las Figuras 3.18b y 3.19b representan el tráfico medio procesado por nodo en función de la velocidad de los nodos para los patrones de movilidad TVCM y RWP respectivamente. Como se puede extraer de dichas figuras, todos los esquemas de caché excepto HybridCache y CLIR generan más tráfico que la opción que no usa técnicas de caché, debido al uso de peticiones en *broadcast*. MobEye realiza peticiones de *broadcast* indiscriminadas y, por tanto, es el esquema de caché que más tráfico genera. Conforme las peticiones de *broadcast* se restringen, el tráfico generado se reduce, tal y como muestran los resultados obtenidos por COOP, SimpleSearch y DPIP. La velocidad de los nodos no afectan significativamente a la carga de tráfico en los esquemas de caché evaluados excepto en DPIP, que genera más tráfico conforme se incrementa la velocidad. Al incrementarse la velocidad, la lista de vecinos usada en DPIP cambia frecuentemente y, por lo tanto, las peticiones no se pueden enviar a los nodos vecinos ya que han abandonado la vecindad del nodo, de modo que los nodos tienen que realizar la petición de *broadcast* más frecuentemente, incrementando de esta forma la carga de tráfico. El comportamiento de los esquemas de caché siguen las mismas tendencias con independencia del modelo de movilidad empleado, aunque el tráfico generado con RWP es superior.

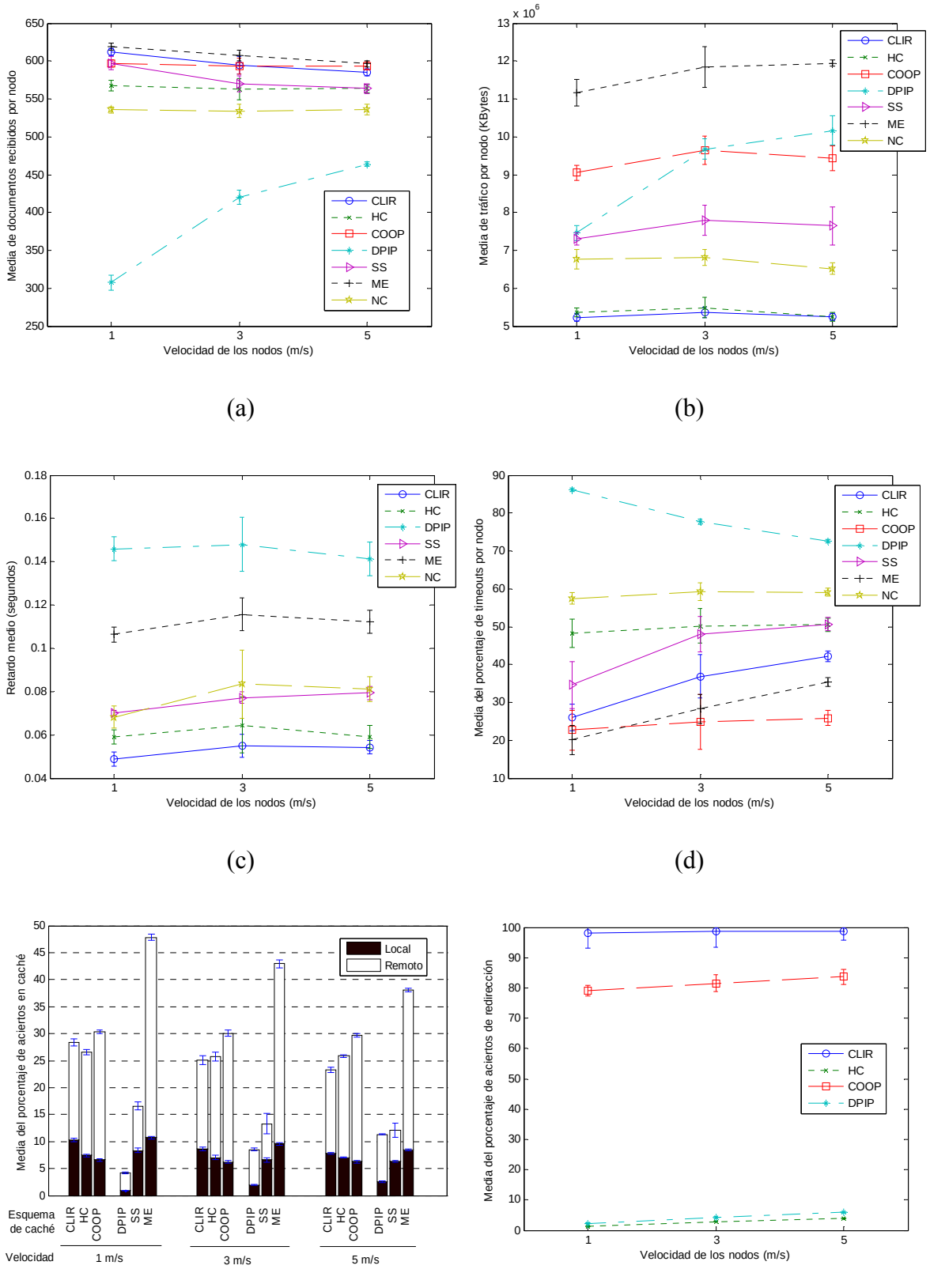


Figura 3.18. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función de la velocidad de los nodos para el modelo de movilidad TVCM

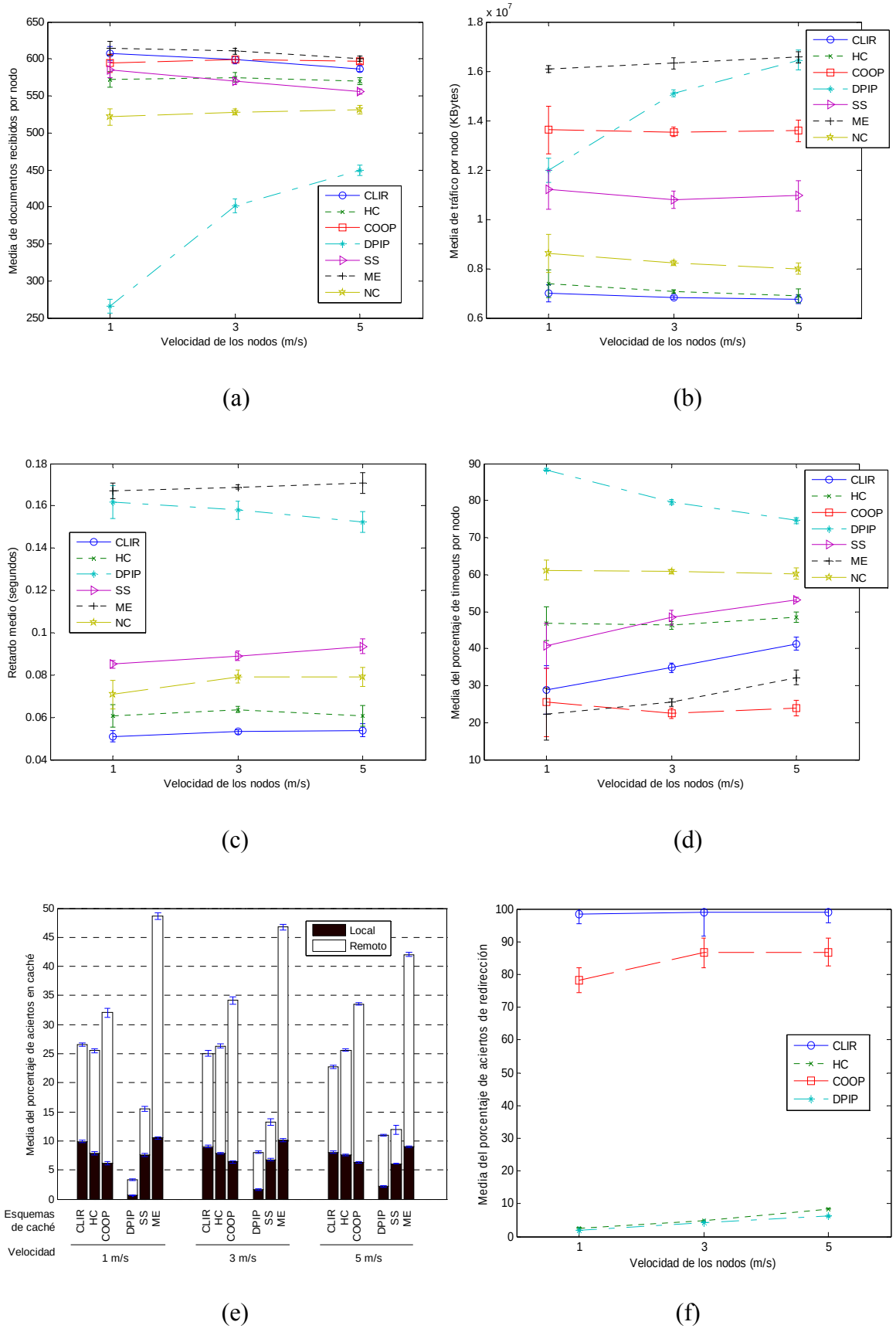


Figura 3.19. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función de la velocidad de los nodos para el modelo de movilidad RWP

Las Figuras 3.18c y 3.19c muestran el retardo medio necesario para obtener los documentos. El retardo obtenido por COOP no se muestra en las figuras ya que es cinco veces superior al obtenido por los restantes esquemas de caché. Esta circunstancia es debida al temporizador definido en COOP. Tras hacer una petición *broadcast*, COOP espera un cierto periodo de tiempo antes de suponer que la petición ha fallado. En ese caso, se envía la petición directamente al *DS*. Este tiempo de espera es un serio inconveniente de COOP ya que este temporizador debe ser correctamente ajustado para cada escenario si se desea obtener un buen rendimiento. De hecho, si el valor del temporizador se selecciona por debajo del valor óptimo, las peticiones se realizarán al *DS* incluso antes de que la petición *broadcast* pueda ser respondida, con lo que se estará generando tráfico innecesario. Por otro lado, CLIR e HybridCache son los esquemas de caché que obtienen el menor retardo, mientras que SimpleSearch consigue un retardo similar a la opción de no usar cachés en escenarios de movilidad TVCM y ligeramente superior usando RWP. SimpleSearch necesita cuatro mensajes para obtener los documentos que solicita (*get*, *ack*, *confirm* y *resp*) en lugar de dos mensajes (*get*, *resp*) que necesitan HybridCache y CLIR, con lo que el retardo obtenido también es mayor. Finalmente, MobEye y DPIP obtienen el mayor retardo, siendo éste dos y tres veces superior a CLIR respectivamente. El alto retardo de MobEye es debido al tiempo necesario para el intercambio de los cuatro mensajes usados para obtener un documento. MobEye, además, presenta un retardo más acusado usando RWP que en TVCM debido a que el número de aciertos en caché es inferior (Figuras 3.18e y 3.18e). Este comportamiento es debido a que en TVCM los nodos forman agrupaciones en torno a puntos de interés, con lo que la probabilidad de encontrar el documento a buscar mediante el uso de mensajes de *broadcast* será más alta. Por otro lado, el retardo en DPIP depende del parámetro *DPIP_Timer* que se convierte en un límite inferior para el retardo.

Las Figuras 3.18d y 3.19d representan el porcentaje medio de *timeouts* por nodo. Sólo DPIP obtiene un peor rendimiento que NC. DPIP supone que las peticiones van a ser servidas ya que el *DS* siempre va a estar disponible, aunque esta suposición no siempre es correcta. Cuando DPIP envía una petición *broadcast* a su vecindario, todos los nodos que reciben la petición anotan en su tabla *IV* que el nodo solicitante tendrá una copia del documento solicitado. Sin embargo, el documento puede que se obtenga mucho más tarde porque el *DS* esté temporalmente inaccesible. Por tanto, los nodos pueden almacenar información incorrecta de modo que posteriores peticiones al mismo documento pueden ser redirigidas a un nodo que no tiene el documento aún. De hecho, conforme la velocidad de los nodos se incrementa, el porcentaje de *timeouts* se decrementa ya que la información almacenada en las tablas *IV* se borra más frecuentemente porque los nodos cambian su localización con más asiduidad, con lo que cambia la configuración de los nodos vecinos. Por otro lado, MobEye y COOP obtienen un mejor rendimiento que CLIR ya que hacen uso de mensajes *broadcast*, por lo que tienen una mayor probabilidad de encontrar los documentos en la red, aunque a expensas de incrementar considerablemente la carga de tráfico generado. De cualquier modo, la diferencia no es significativa en aquellos escenarios en los que los nodos se mueven a poca velocidad.

Las figuras 3.18e y 3.19e muestran los aciertos en caché local y remota. DPIP es el esquema de caché que consigue una menor tasa de aciertos en la caché local, mientras que el resto obtiene un rendimiento similar. Como se espera, MobEye es el esquema de caché con el mayor porcentaje de aciertos remotos debido al uso indiscriminado de mensajes de *broadcast* para encontrar los documentos en la red. CLIR, COOP e HybridCache se comportan de forma similar en escenarios de baja movilidad, aunque el rendimiento de CLIR e HybridCache se deteriora conforme se incrementa la velocidad mientras que el rendimiento de COOP se mantiene. HybridCache consigue una buena tasa de aciertos en caché remota a expensas de tener una alta tasa de errores de redirección, tal y como se muestra en las Figuras 3.18f y 3.19f. Finalmente, DPIP y SimpleSearch obtienen el peor porcentaje de aciertos, aunque DPIP mejora conforme se incrementa la velocidad ya que la información almacenada en la tabla *IV* debe ser actualizada más frecuentemente.

Las Figuras 3.18f y 3.19f representan la media del porcentaje de aciertos de redirección. Los resultados obtenidos por DPIP e HybridCache son muy cercanos a cero. Este hecho demuestra que el mecanismo de redirección que implementan es inadecuado. Por otro lado, COOP y CLIR obtienen una tasa de aciertos de redirección aproximada de 80% y 100% respectivamente. Claramente, CLIR es mucho mejor que el resto de esquemas de caché ya que prácticamente no falla al redirigir peticiones.

Como conclusión, CLIR consigue una de las mejores tasas de obtención de documentos pero generando el menor tráfico en la red. La cantidad de *timeouts* es menor que el obtenido por el resto de esquemas de caché exceptuando a MobEye y COOP, que son ligeramente mejores aunque a expensas de doblar o triplicar la carga de tráfico generado. CLIR también obtiene el menor retardo a la hora de obtener los documentos. La Caché de Redirecciones prácticamente no falla a la hora de redirigir peticiones.

3.5.2 Evaluación en función del número de nodos de la red

En este conjunto de experimentos se analiza el comportamiento de los esquemas de caché cuando se modifica la densidad de los nodos en la red. De esta forma se evalúa su escalabilidad.

Las Figuras 3.20a y 3.21a presentan la cantidad media de documentos recibidos por nodo en función del número de nodos presentes en la red. Para redes con baja densidad (con sólo 25 nodos), MobEye y SimpleSearch obtienen una cantidad ligeramente superior de documentos que COOP y DPIP. Por otro lado, aunque HybridCache es mejor que la opción que no usa cachés, la cantidad de documentos es inferior a la obtenida por el resto de esquemas de caché. Conforme la densidad de nodos se incrementa, el comportamiento se mantiene para todos los esquemas. Sin embargo, para redes de más de 50 nodos el rendimiento de SimpleSearch y MobEye decae, mientras que CLIR, COOP e HybridCache mantienen el número de documentos obtenidos. Esta caída del rendimiento se debe al incremento del número de *timeouts* (Figuras 3.20d y 3.21d).

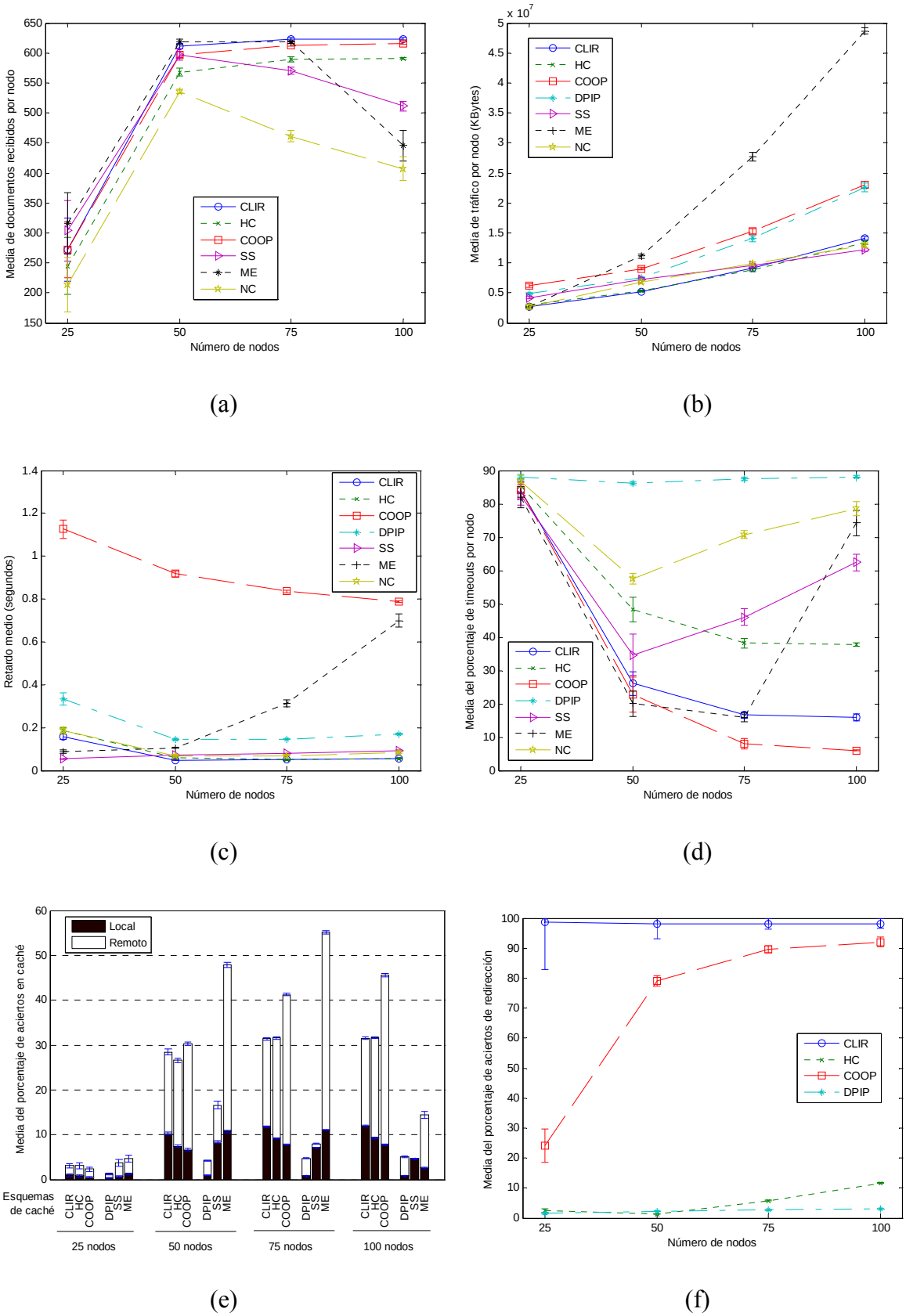


Figura 3.20. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función del número de nodos para el modelo de movilidad TVCM

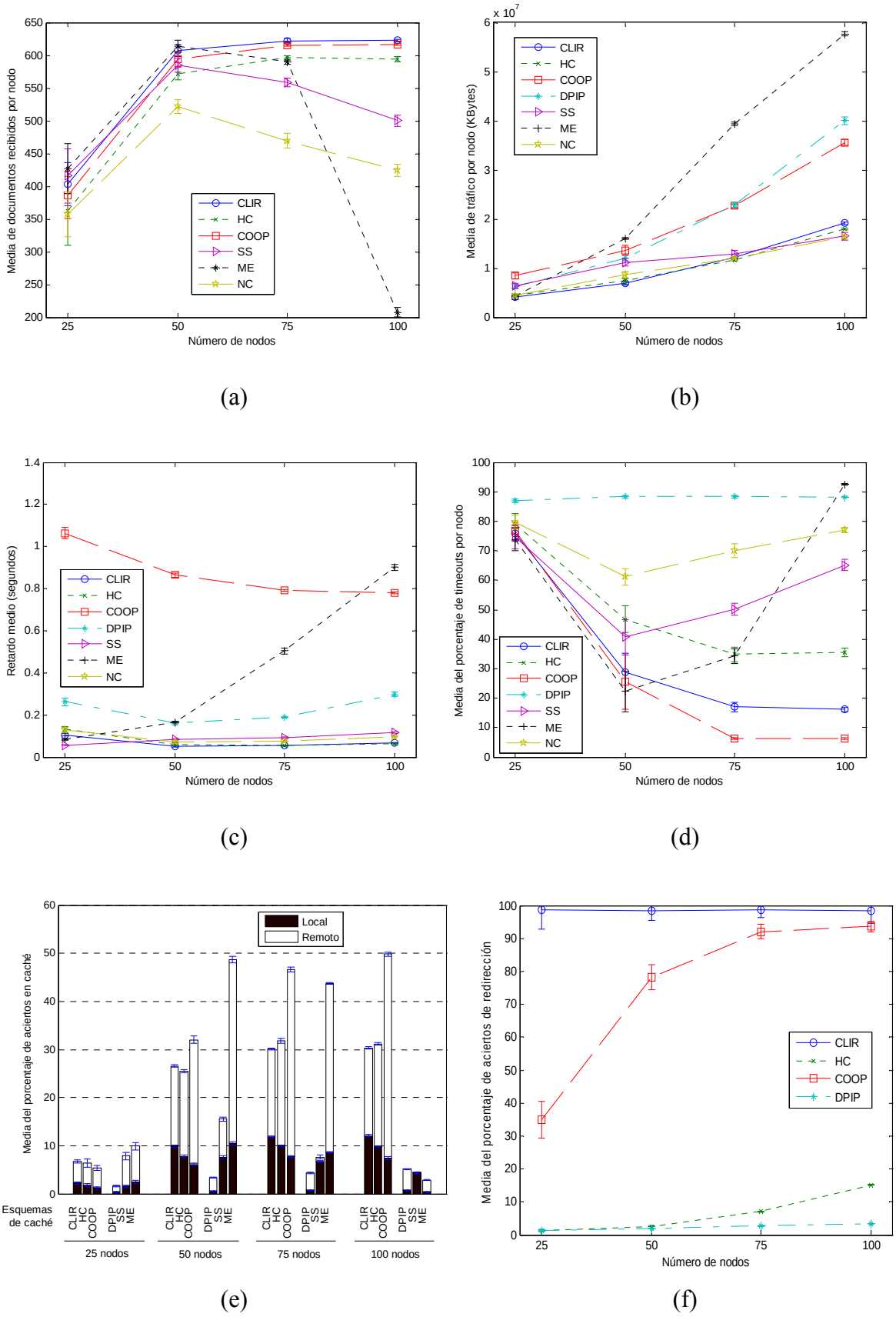


Figura 3.21. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función del número de nodos para el modelo de movilidad RWP

Las Figuras 3.20b y 3.21b muestran la media del tráfico procesado por los nodos. Los esquemas de caché se pueden dividir en tres tipos de comportamiento conforme se incrementa el número de nodos. Primero, aunque MobEye es uno de los esquemas de caché que genera menos tráfico en redes poco densas (25 nodos), la carga de tráfico se incrementa sustancialmente cuando se incrementa la cantidad de nodos en la red. Este incremento se debe al uso de mensajes de *broadcast* para realizar las peticiones de documentos. En segundo lugar, el tráfico generado por COOP y DPIP también se incrementa con el número de nodos en la red. Sin embargo, este crecimiento es más suave que en el caso de MobEye ya que realizan peticiones usando mensajes *broadcast* selectivos en lugar de realizar las peticiones a toda la red. En último lugar, el incremento de tráfico que presentan CLIR, HybridCache, SimpleSearch y la opción de no usar cachés es incluso más suave que en los casos mencionados anteriormente.

Las Figuras 3.20c y 3.21c representan el retardo medio de obtención de documentos en función de la cantidad de nodos en la red. MobEye y SimpleSearch consiguen el menor retardo en una red de 25 nodos pero, conforme se incrementa la densidad de la red, el retardo de MobEye se incrementa mientras que el de SimpleSearch se mantiene inalterado. Por otro lado, el retardo de que presentan CLIR e HybridCache se reduce para redes de más de 25 nodos. Finalmente, DPIP y COOP obtienen un rendimiento muy pobre comparado con el resto de esquemas de caché, debido a problemas anteriormente comentados referentes al temporizador de COOP y a la baja tasa de aciertos en caché (Figuras 3.20e y 3.21e), respectivamente.

Las Figuras 3.20d y 3.21d presentan la media del porcentaje de *timeouts*. El rendimiento para todos los esquemas de caché es similar para redes de 25 nodos. En concreto, se observa un alto porcentaje de *timeouts* en redes poco densas debido a que la probabilidad de poder crear una ruta al *DS* es muy baja ya que pueden estar inaccesibles durante un tiempo considerable de la simulación. Debido a esta falta de conectividad, los documentos no pueden ser servidos desde los *DS* y, por tanto, los nodos no pueden almacenar las copias en sus caché locales, con lo que la forma de administrar las cachés no tiene impacto en este parámetro para redes muy poco densas. Conforme el número de nodos se incrementa, el porcentaje de *timeouts* se reduce drásticamente excepto para DPIP, que mantiene dicho porcentaje. Además, los *timeouts* en MobEye y SimpleSearch se incrementan significativamente en una red de cien nodos debido a las peticiones de *broadcast* que realizan. Para grandes densidades de nodos, la cantidad de mensajes *broadcast* y el tráfico generado se incrementa, tal y como se muestra en las Figuras 3.20b y 3.21b, con lo que las interferencias y la saturación de la red degradan el rendimiento. Finalmente, cabe mencionar que únicamente COOP tiene un comportamiento mejor que CLIR para redes muy densas.

Las Figuras 3.20e y 3.21e muestran el porcentaje de aciertos en caché en función del número de nodos en la red. Como se ha mencionado anteriormente, una red con sólo 25 nodos el rendimiento es muy bajo. La tasa de acierto remoto de SimpleSearch se reduce conforme se incrementa la cantidad de nodos ya que la probabilidad de tener una ruta creada hacia el *DS* también se incrementa. Por lo tanto, las peticiones *broadcast* se llevarán

a cabo con menor probabilidad, lo que causará una reducción del tráfico generado (Figuras 3.19b y 3.20b). Por otro lado, los aciertos en caché de MobEye se reducen drásticamente para redes de 100 nodos, lo que implica una reducción de la cantidad de documentos recibidos (Figuras 3.20a y 3.21a) y el incremento del retardo (Figuras 3.20c y 3.21c) y del porcentaje de *timeouts* (Figuras 3.20d y 3.21d).

Por último, las Figuras 3.20f y 3.21f muestran el porcentaje de aciertos de redirección. Como en el estudio previo, CLIR obtiene una tasa de aciertos de redirección cercana al 100%. El rendimiento de COOP depende del número de nodos en la red y se decrementa cuando la densidad de nodos se incrementa. Además, HybridCache y DPIP tienen un rendimiento muy pobre aunque HybridCache mejora ligeramente cuando el número de nodos se incrementa.

Como conclusión, CLIR es el mejor esquema de caché a la hora de obtener documentos, generando muy poco tráfico, con el menor retardo y un porcentaje de *timeouts* sólo mejorado por COOP (que necesita generar el triple de tráfico para obtener dicho rendimiento). Estos resultados se mantienen independientemente del número de nodos en la red, por lo que se observa que CLIR es escalable.

3.5.3 Evaluación en función del tiempo entre peticiones

Este conjunto de experimentos se realiza para evaluar el rendimiento de los esquemas de caché cuando se modifica el número de peticiones y, por lo tanto, se modifica también la carga el tráfico en la red.

Las Figuras 3.22a y 3.23a representan el número medio de documentos recibidos por nodo en función del tiempo medio entre peticiones. Para redes muy cargadas (5 segundos entre peticiones), MobEye, COOP y CLIR obtienen un alto número de documentos comparados con el resto de soluciones. En cambio, HybridCache y SimpleSearch obtienen alrededor de un 45% menos de documentos. Conforme el tiempo medio entre peticiones se incrementa (y la carga de tráfico decrementa) el comportamiento de todos los esquemas de caché es similar, aunque las diferencias entre esquemas de caché se suavizan hasta llegar a desaparecer para redes muy poco cargadas.

Las Figuras 3.22b y 3.23b muestran el tráfico medio procesado por los nodos de la red. Para redes muy cargadas (5 segundos de tiempo de espera) DPIP y SimpleSearch tienen un rendimiento mejor que la opción de no usar caché, mientras que HybridCache tiene un comportamiento similar a la solución sin caché. CLIR introduce un poco más de sobrecarga, mientras que COOP y MobEye generan mucho más tráfico ya que usan peticiones en modo *broadcast*. Esta diferencia sólo se percibe para un tiempo entre peticiones de 5 segundos. Conforme el tiempo medio entre peticiones aumenta, la generación de tráfico obviamente se decrementa. Bajo estas circunstancias, CLIR e HybridCache generan un menor tráfico que el resto de esquemas de caché ya que requieren de menos mensajes para operar.

Las Figuras 3.22c y 3.23c representan el retardo obtenido por todas las estrategias de caché excluyendo a COOP. Para hacer el análisis más cómodo se ha decidido excluir los resultados de COOP ya que obtiene un retardo muy alto en comparación con el resto de esquemas de caché. CLIR consigue el retardo más bajo para todas las cargas de tráfico. SimpleSearch tiene un rendimiento similar a la opción de no usar cachés, mientras que HybridCache resulta peor que no cachear en redes muy cargadas mientras que va mejorando conforme la carga de tráfico se reduce. Finalmente, el retardo de MobEye y DPIP es tres y cuatro veces superior al de CLIR respectivamente.

Las Figuras 3.22d y 3.23d muestran la evolución del porcentaje medio de *timeouts* en función del tiempo medio entre peticiones. Todos los esquemas de caché, excepto DPIP, tienen un porcentaje de *timeouts* inferior a la opción de no usar cachés. CLIR resulta mejor que SimpleSearch e HybridCache para redes muy cargadas (5 y 10 segundos de tiempo de espera entre peticiones) y peor que los esquemas basados en peticiones *broadcast* COOP y MobEye. De todas formas, esta diferencia se atenúa conforme la carga de tráfico disminuye.

Las Figuras 3.22e y 3.23e ilustran el porcentaje de aciertos en caché en función del tiempo medio entre peticiones. En ellas puede observarse que todos los esquemas de caché excepto CLIR, HybridCache y SimpleSearch reducen su rendimiento conforme se decrementa la carga de tráfico en la red. CLIR y SimpleSearch usan peticiones *broadcast* (CLIR a nivel AODV y SimpleSearch a nivel de aplicación) cuando no existe una ruta creada hacia *DS*. En AODV las rutas se borran de la tabla de encaminamiento tras un periodo de tiempo sin uso. Este periodo de tiempo lo define el parámetro *Active Route Timeout* que, para el caso que nos ocupa, está configurado en 10 segundos. Por lo tanto, si el tiempo de espera entre peticiones es mayor que este parámetro, la ruta quedará obsoleta, con lo que CLIR enviará el mensaje de petición usando el protocolo de enrutamiento, con lo que el número de aciertos en caché remota mejora. Por otro lado, HybridCache obtiene el mejor porcentaje de aciertos de redirección en redes muy poco cargadas (Figuras 3.22f y 3.23f) y, como consecuencia, su tasa de aciertos en cachés remotas también se incrementa.

Las Figuras 3.22f y 3.23f muestran el porcentaje medio de aciertos de redirección. Como ocurría en los estudios previos, CLIR consigue una tasa de aciertos de redirección cercano al 100%, aunque este rendimiento cae al 95% para redes muy cargadas. COOP presenta un porcentaje de aciertos de redirección entre un 70% y un 80% excepto para un tiempo medio entre peticiones de 5 segundos, donde este parámetro cae al 50%. Finalmente, DPIP e HybridCache obtienen un porcentaje de aciertos de redirección muy pobre (cercano al 0%) aunque HybridCache eleva dicho rendimiento hasta el 30% para redes muy poco cargadas (50 segundos entre peticiones).

Como conclusión, CLIR es la mejor opción con respecto al retardo para todas las cargas de tráfico estudiadas. Al reducirse la carga de tráfico, el rendimiento de CLIR también mejora con respecto a la generación de tráfico y el porcentaje de *timeouts*, aunque obviamente la efectividad de la caché se reduce al distanciarse en el tiempo las peticiones.

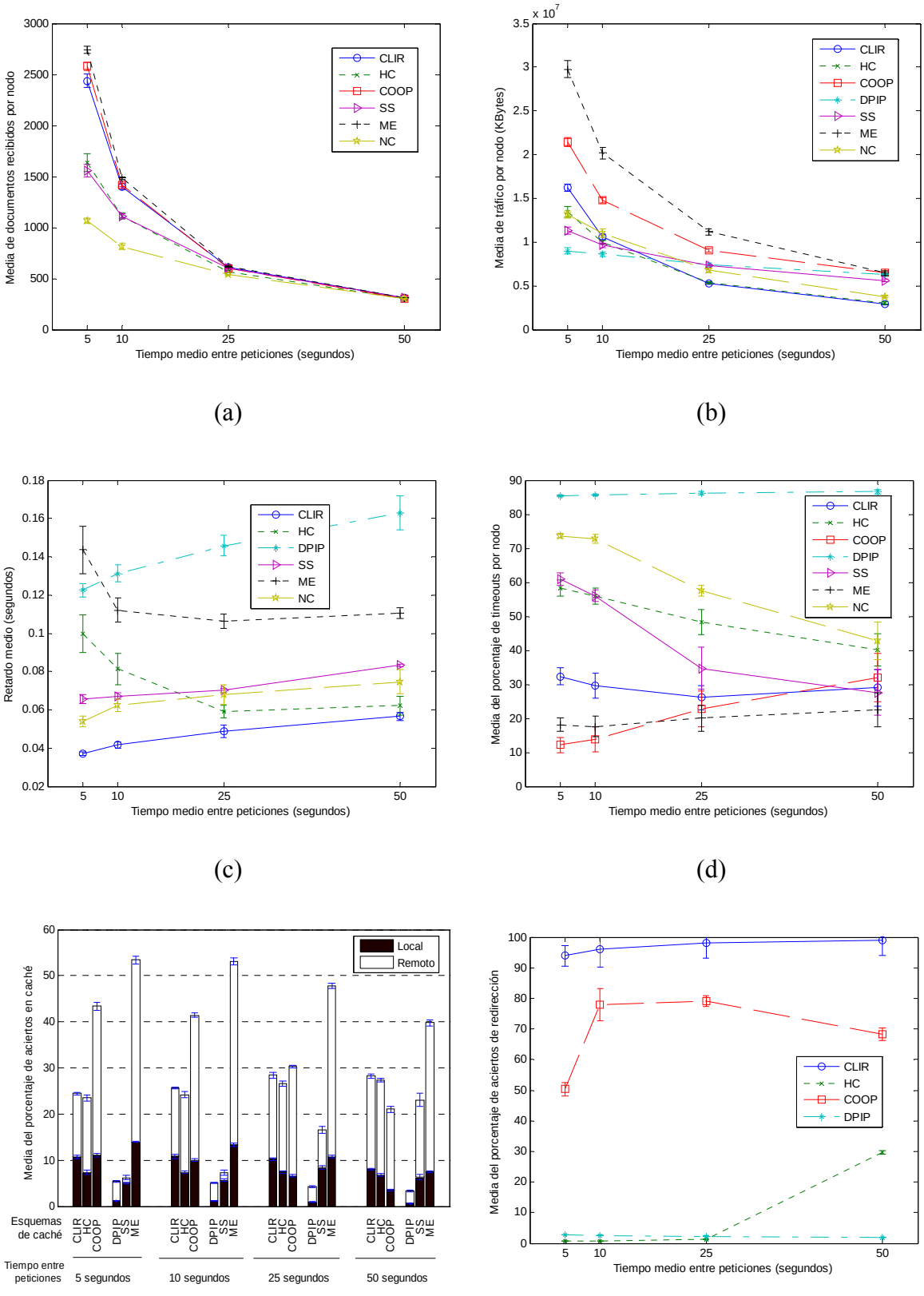


Figura 3.22. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función del tiempo medio entre peticiones para el modelo de movilidad TVCM

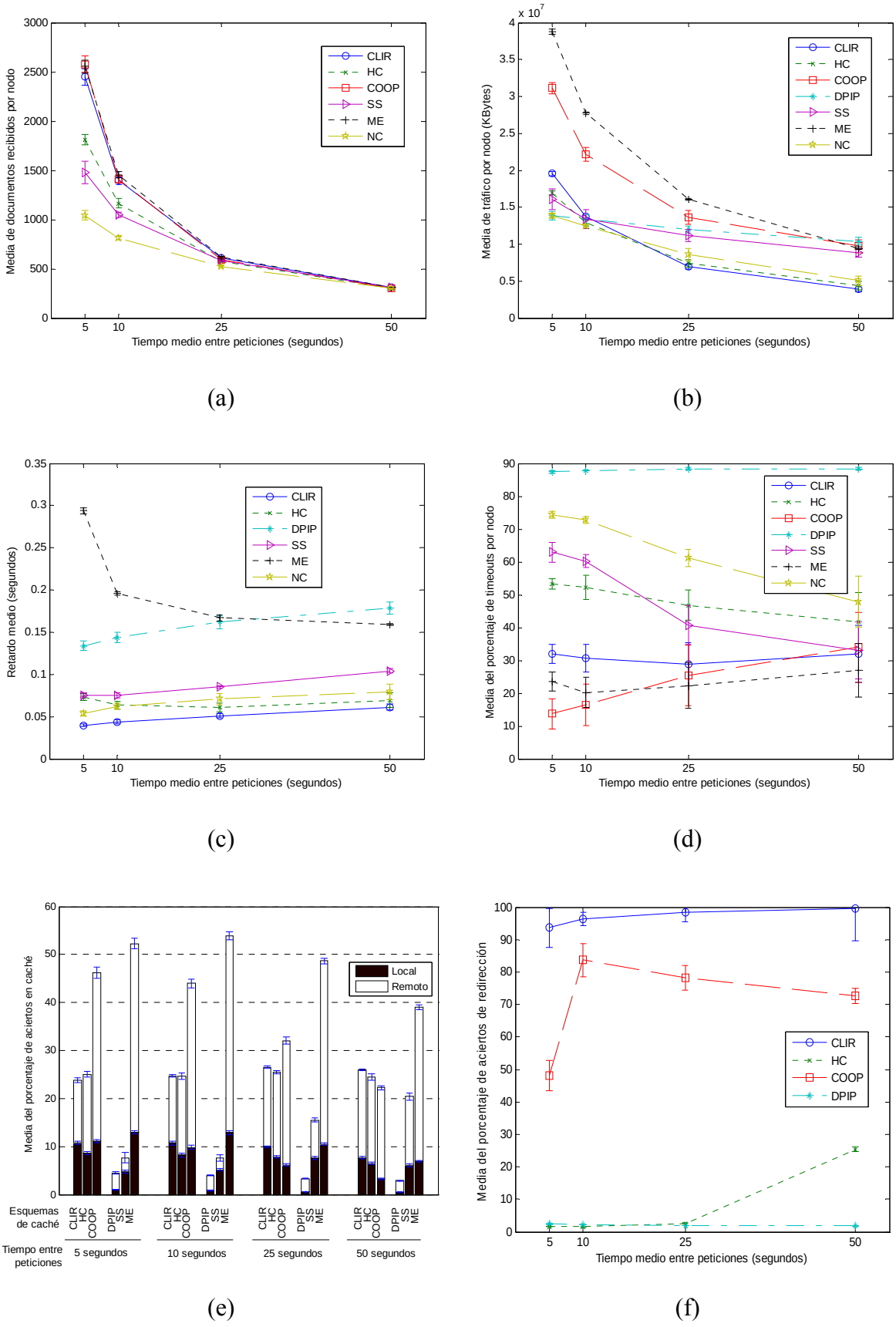


Figura 3.23. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función del tiempo medio entre peticiones para el modelo de movilidad RWP

3.5.4 Evaluación en función de la pendiente Zipf

En este conjunto de experimentos se evalúa el impacto que tiene la pendiente de la ley Zipf en el rendimiento de los esquemas de caché. La pendiente de la ley Zipf caracteriza la popularidad de los documentos, de forma que, conforme se incrementa su valor, la frecuencia de acceso a los documentos más populares también se incrementa, disminuyendo en cambio la probabilidad de acceso a los documentos menos populares.

Las Figuras 3.24a y 3.25a representan en número medio de documentos obtenidos por nodo en función de la pendiente Zipf. MobEye, CLIR, COOP y Simplesearch consiguen entre 620 y 580 documentos para todas las pendientes seleccionadas. El número de documentos obtenidos se incrementa cuando la pendiente Zipf se acerca a uno debido a que las cachés locales y remotas mejoran su rendimiento, tal y como se muestra en las Figuras 3.24e y 3.25e. El rendimiento de HybridCache es similar al obtenido sin usar métodos de caché, aunque se incrementa significativamente con la pendiente Zipf.

Por otro lado, las Figuras 3.24b y 3.25b muestran el tráfico medio procesado por los nodos en función de la pendiente Zipf. Como ocurría en los estudios anteriores, CLIR e HybridCache generan menos tráfico que el esquema sin cachés mientras que el resto de opciones requiere una mayor sobrecarga de tráfico para obtener un rendimiento equivalente. De hecho, las diferencias de rendimiento de CLIR e HybridCache con respecto al resto de esquemas de cachés se incrementan con la pendiente Zipf.

Las Figuras 3.24c y 3.25c comparan el tiempo medio necesario para acceder a los documentos. Sólo CLIR consigue, para α pequeña, un retardo inferior que el esquema que no emplea cachés, ya que el resto de esquemas de caché introducen un retardo en su procesamiento que los hacen menos eficientes para este caso. No obstante, SimpleSearch e HybridCache también reducen el retardo a valores inferiores a la opción de no usar cachés para pendientes de 0.8 y 1.0 respectivamente. Por otro lado, MobEye y DPIP obtienen retardos que duplican o triplican el obtenido por CLIR. Finalmente, el retardo de COOP no se representa en estas figuras al estar cercano a un segundo.

En las Figuras 3.24d y 3.25d se representa el porcentaje de *timeouts* en función de la pendiente de la función Zipf. Tal y como ocurría en estudios anteriores, los esquemas basados en peticiones *broadcast* MobEye y COOP consiguen un mejor rendimiento que CLIR, aunque dicha diferencia de rendimiento decrece cuando la pendiente Zipf está cercana a uno debido a los aciertos en caché local. HybridCache y SimpleSearch también obtienen un porcentaje de *timeouts* inferior a la opción de no usar cachés. Sin embargo, HybridCache se comporta de manera similar a no cachear para pendientes bajas. Finalmente, DPIP es el esquema de caché con el mayor número de *timeouts*.

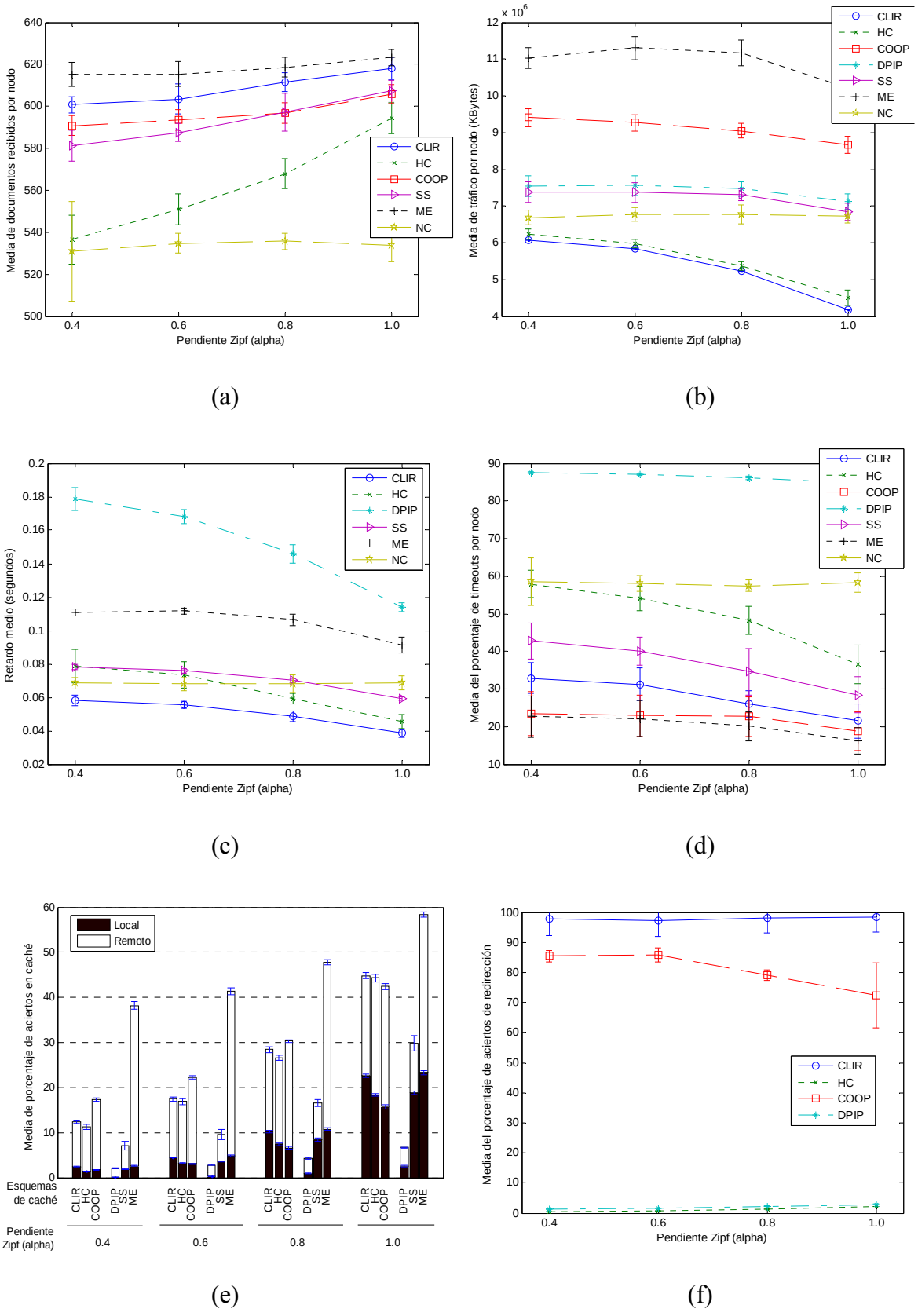


Figura 3.24. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función de la pendiente Zipf para el modelo de movilidad TVCM

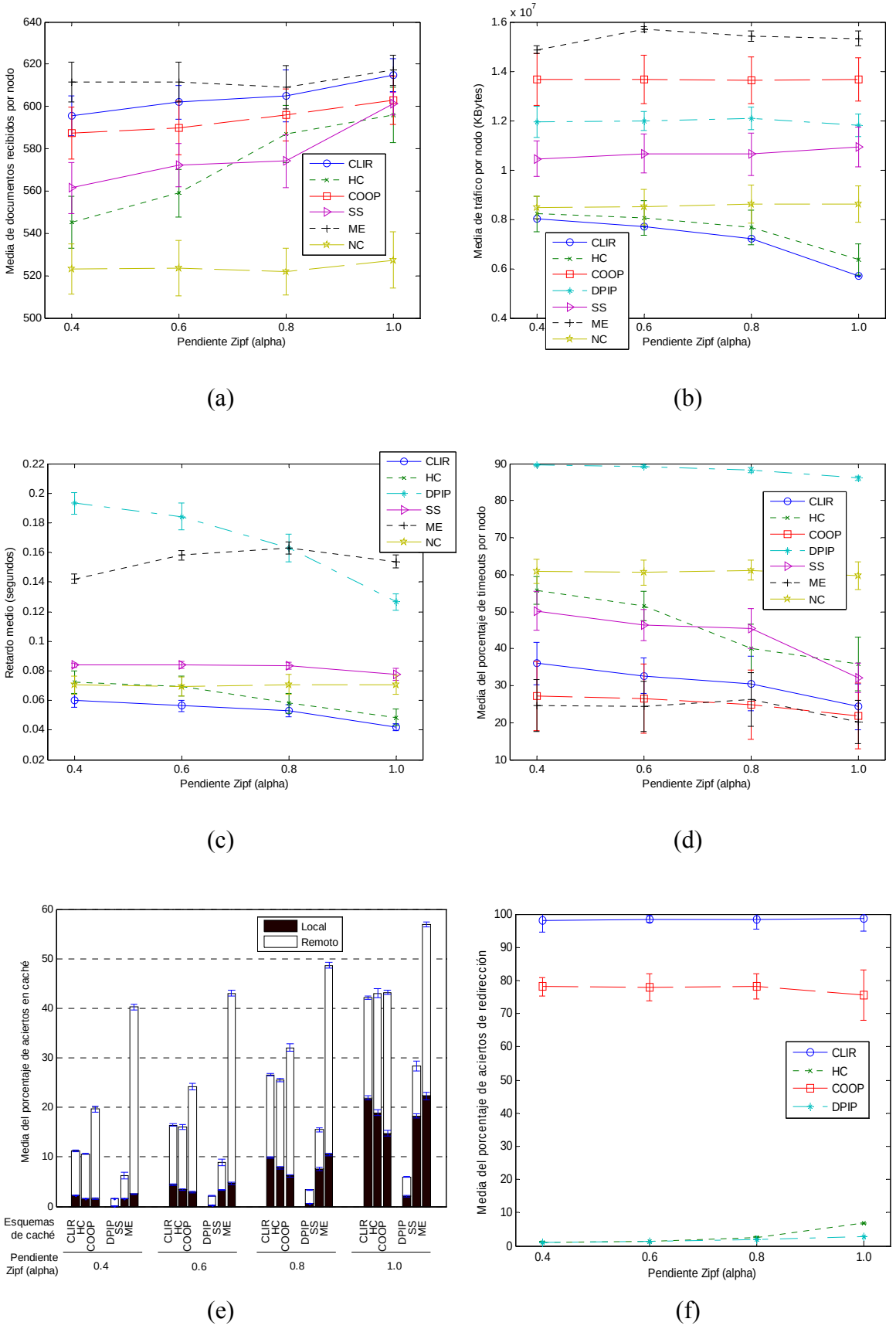


Figura 3.25. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función de la pendiente Zipf para el modelo de movilidad RWP

Conforme se incrementa la pendiente Zipf los documentos más populares se solicitan más veces y, por tanto, los aciertos en caché local aumentan. Un acierto en caché local no genera tráfico en la red y el retardo percibido es nulo. Las Figuras 3.24e y 3.25e confirman este comportamiento. Los aciertos en caché local de CLIR y MobEye son mayores que los aciertos en caché local de los demás esquemas de caché para todas las pendientes Zipf estudiadas.

Las Figuras 3.24f y 3.25f representan la media del porcentaje de aciertos de redirección en función de la pendiente Zipf. CLIR consigue un porcentaje de aciertos de redirección cercano al 100%. Por otro lado, COOP obtiene un rendimiento entre el 85% y el 70% conforme la pendiente Zipf se incrementa. Finalmente, como en estudios anteriores, HybridCache y DPIP muestran una tasa de aciertos de redirección cercana a cero.

Como conclusión, el esquema de caché CLIR aprovecha la popularidad de los documentos obteniendo unas altas tasas de aciertos en caché local y remota. Como consecuencia, CLIR muestra una tasa de documentos obtenidos en el mismo orden que los esquemas de caché que usan *broadcast* para realizar las peticiones pero reduciendo notablemente el tráfico generado así como el retardo.

3.5.5 Evaluación en función del periodo de vigencia de los documentos (TTL)

Este grupo de experimentos estudia la influencia del tiempo de vida de los documentos en el rendimiento de la red. El parámetro TTL define cuánto tiempo pueden ser almacenados los documentos en las cachés locales antes de ser considerado obsoletos.

Las Figuras 3.26a y 3.27a comparan la media de documentos obtenidos por nodo en función del TTL medio de los documentos. MobEye y CLIR son los esquemas de caché que más documentos consiguen para todos los valores del TTL considerados, seguidos por COOP y SimpleSearch. HybridCache obtiene un menor número de documentos para valores altos del TTL debido al incremento en los *timeouts* (Figuras 3.26d y 3.27d).

Las Figuras 3.26b y 3.27b representan el tráfico medio procesado por cada nodo en función del TTL de los documentos. CLIR e HybridCache generan una media de tráfico similar para todos los valores probados, aunque la carga de tráfico se reduce cuando los documentos tienen un tiempo de vida más alto. En cualquier caso, tienen un mejor rendimiento que la opción de no usar cachés. Por otro lado, el resto de esquemas de caché generan más tráfico que el esquema que no usa cachés, sobretodo COOP y MobEye.

Las Figuras 3.26c y 3.27c muestran el retardo medio en función del TTL de los documentos. CLIR e HybridCache obtienen un retardo similar para documentos con un corto tiempo de vida. En cambio, conforme se incrementa el valor del TTL, el retardo obtenido por HybridCache aumenta, particularmente si los documentos nunca expiran.

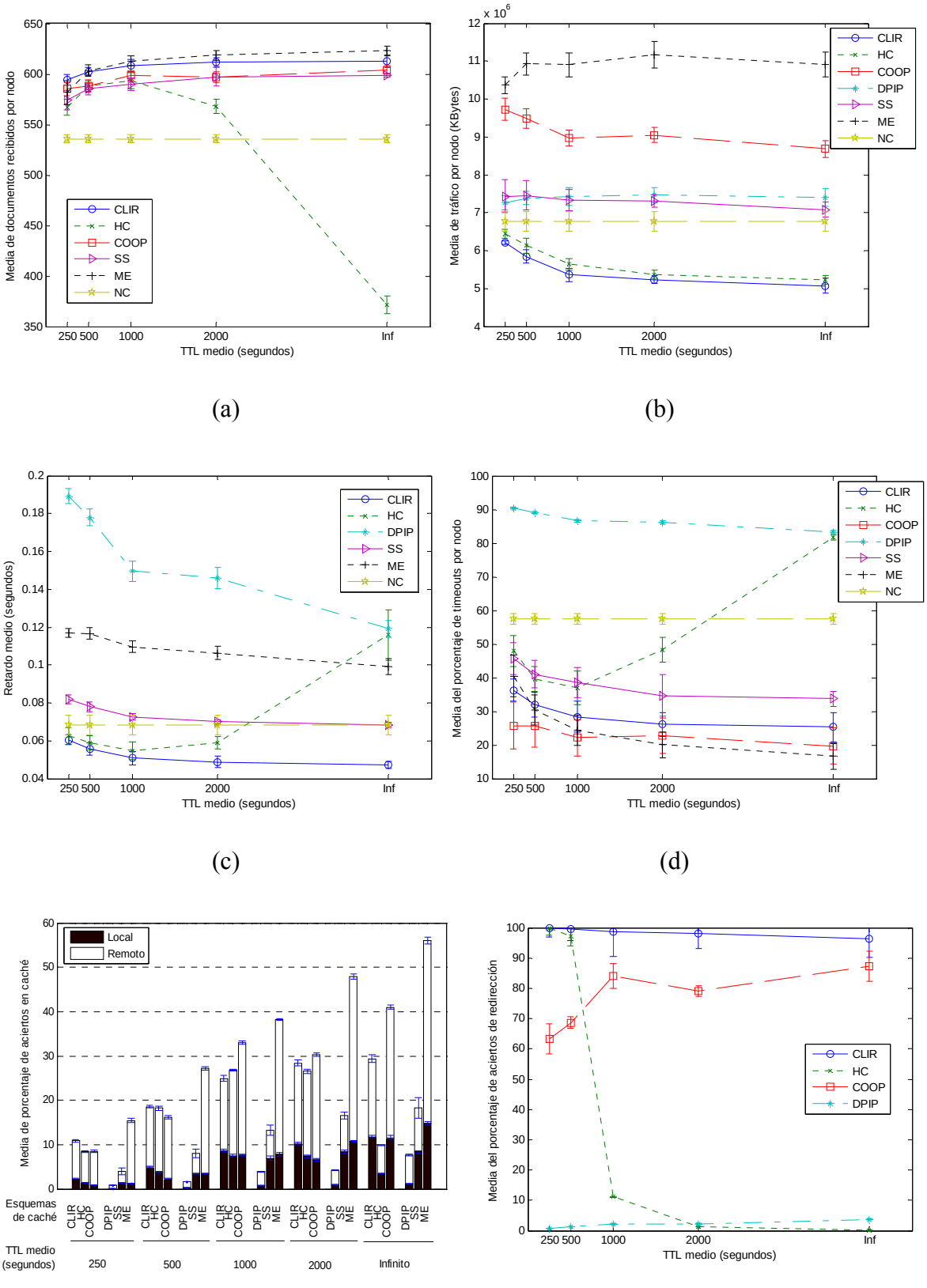


Figura 3.26. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función del TTL medio de los documentos para el modelo de movilidad TVCM

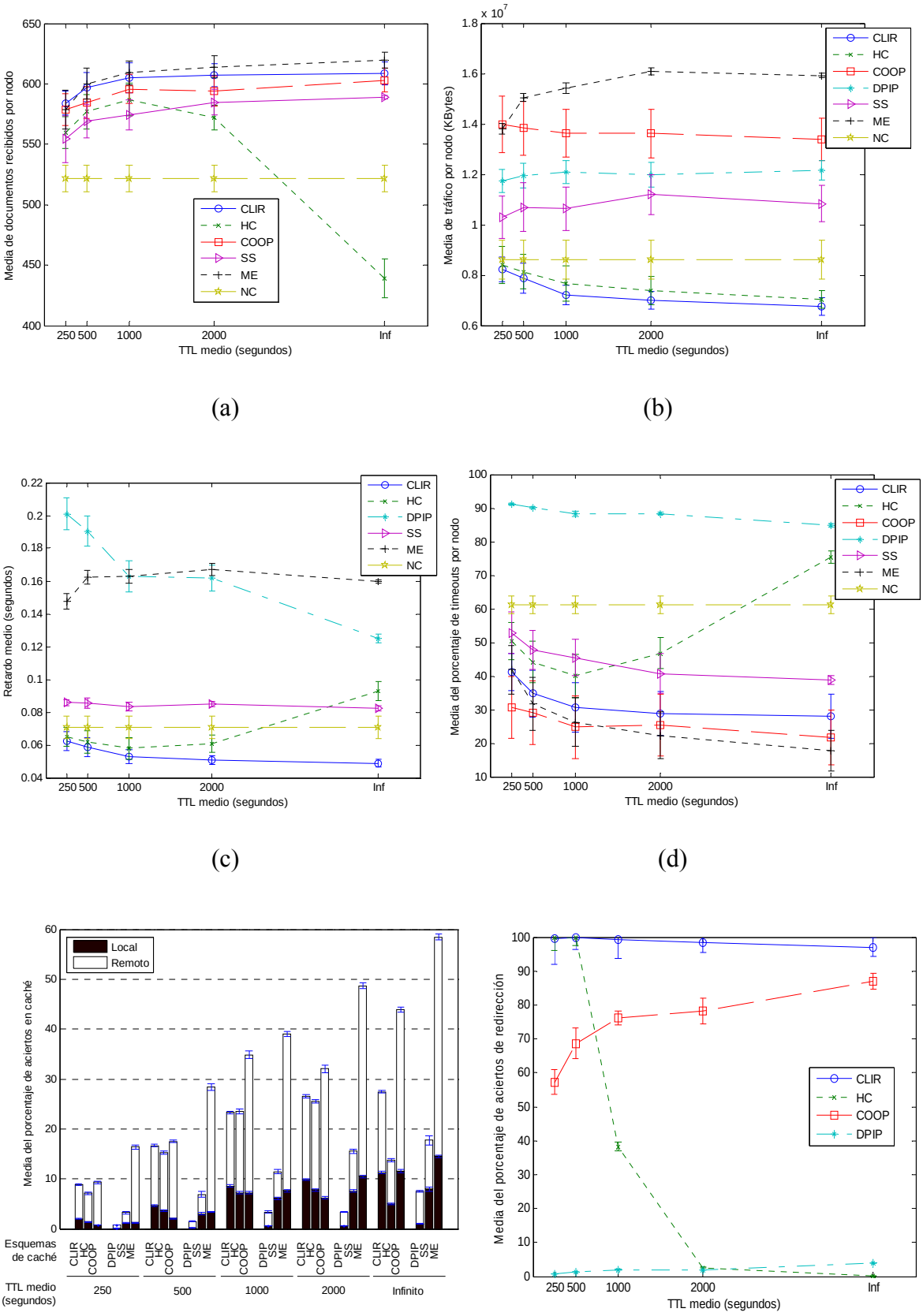


Figura 3.27. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función del TTL medio de los documentos para el modelo de movilidad RWP

Este comportamiento es debido al incorrecto manejo que se realiza en las redirecciones, tal y como se muestra en las Figuras 3.26f y 3.27f. Sin embargo, CLIR mantiene o reduce el retardo si la validez de los documentos es mayor ya que la técnica de redirección es más efectiva. Por otro lado, SimpleSearch tiene un rendimiento similar a la opción de no usar cachés. Finalmente, MobEye, DPIP y, especialmente, COOP obtienen un retardo muy elevado comparado con el resto de esquemas de caché.

Las Figuras 3.26d y 3.27d representan el porcentaje medio de *timeouts* en función del TTL de los documentos. El comportamiento observado es similar al encontrado en los estudios anteriores, excepto para el esquema de caché HybridCache. Tal y como se comentó anteriormente, HybridCache es muy sensible al tiempo de vida de los documentos y, por lo tanto, el número de *timeouts* se incrementa conforme el TTL aumenta.

Si la vigencia de los documentos es grande, su almacenamiento en las cachés locales puede ser más útil, ya que podrán permanecer más tiempo en ellas antes de caducar. En este sentido, las Figuras 3.26e y 3.27e muestran que los aciertos en caché local y remota se incrementa cuando el parámetro TTL crece. Los esquemas de caché con un mayor número de aciertos en caché son aquellos que usan peticiones en modo *broadcast*, ya que son capaces de encontrar los documentos en sus cachés locales o cachés remotas ya que los documentos están almacenados durante más tiempo.

Las Figuras 3.26f y 3.27f representan el porcentaje de aciertos de redirección en función del TTL medio de los documentos. El comportamiento de CLIR, COOP y DPIP es el mismo que se daba en los estudios previos. Sin embargo, HybridCache presenta una gran dependencia del TTL, ya que el mecanismo de redirección funciona mejor para documentos con un muy bajo TTL, mientras que el rendimiento cae conforme el TTL se incrementa.

Como conclusión, en los escenarios estudiados, CLIR obtiene el mínimo retardo para obtener los documentos, generando el menor tráfico mientras es capaz de obtener un gran número de documentos.

3.5.6 Evaluación en función del tamaño de las cachés

El tamaño de las cachés es un factor importante a tener en cuenta ya que, cuanto mayor sea la caché, más documentos podrán ser almacenados en ella y el porcentaje de aciertos en caché, tanto local como remota, también se incrementará.

Las Figuras 3.28a y 3.29a representan el número medio de documentos obtenidos en función del tamaño de las cachés (definido en función del número de documentos que puede albergar). MobEye, CLIR, COOP y SimpleSearch reciben más documentos que la opción de no usar cachés, mientras que HybridCache consigue menos para tamaños de caché de 10 y 20 documentos. En cualquier caso, el rendimiento de HybridCache es inferior al resto de esquemas de caché estudiados.

Teniendo en cuenta el tráfico medio procesado por los nodos en función del tamaño de las cachés, las Figuras 3.28b y 3.29b presentan el mismo comportamiento encontrado en los estudios anteriores. CLIR e HybridCache generan menos tráfico que el esquema que no usa cachés. Dicho tráfico se decrementa conforme se aumenta el tamaño de las cachés locales ya que los aciertos en caché local se incrementan (Figuras 3.28e y 3.29e). El tráfico generado por SimpleSearch y DPIP es similar al generado por el esquema que no usa cachés. Además, COOP y MobEye incrementan en tráfico generado en un 50% y 100% respectivamente comparado con CLIR. Finalmente, el tamaño de las cachés parece no influir en el tráfico generado, excepto para CLIR e HybridCache, que sí que lo decrementan al aumentar el tamaño de las mismas.

Las Figuras 3.28c y 3.29c muestran el retardo medio en función del tamaño de la caché. CLIR es, de nuevo, el mejor esquema de caché al obtener el menor retardo, aunque HybridCache consigue un retardo similar para grandes tamaños de caché. HybridCache obtiene un rendimiento peor que la opción de no usar cachés para tamaños de caché de 10 y 20 documentos debido a la tasa de aciertos de redirección, que es cero para esos tamaños de caché (Figuras 3.28f y 3.29f). Este comportamiento implica que las redirecciones siempre fallen y, como consecuencia, salten más *timeouts* (Figuras 3.28d y 3.29d) o las peticiones sean incorrectamente redireccionadas y finalmente enviadas de nuevo al *DS*. Por otro lado, el retardo de MobEye y DPIP es un 100% o 150% mayor que el obtenido por CLIR.

Las Figuras 3.28d y 3.29d presentan el porcentaje medio de *timeouts* en función del tamaño de las cachés. El rendimiento de COOP y MobEye es mejor que el de CLIR, aunque esta diferencia de rendimiento disminuye conforme el tamaño de la caché se incrementa y más documentos pueden ser almacenados en caché. SimpleSearch obtiene un rendimiento mejor que la opción de no usar cachés aunque es ligeramente inferior a CLIR. Como se comentó anteriormente, el porcentaje de *timeouts* de HybridCache es mayor que si no se emplean caché para tamaños pequeños de caché. Finalmente, DPIP obtiene el peor rendimiento.

Las Figuras 3.28e y 3.29e muestran los aciertos en caché en función del tamaño de la caché. Como se podía esperar, el número total de aciertos en caché local y remota se incrementa conforme más documentos pueden almacenarse en las cachés locales.

Las Figuras 3.28f y 3.29f comparan el porcentaje medio de aciertos de redirección en función del tamaño de las cachés. Conforme se decrementa el tamaño de las cachés, la cantidad de documentos que pueden almacenarse decrementa y el número de reemplazos en caché local se incrementa. Por lo tanto, la probabilidad de fallo al realizar una redirección se incrementa. Para una caché de 10 documentos, el porcentaje de aciertos de redirección ronda el 90% para CLIR y 65% para COOP. Para cachés mayores, los aciertos de redirección se incrementan alcanzando valores cercanos al 100% para CLIR y 80% para COOP. HybridCache y DPIP obtienen un rendimiento muy bajo que no sobrepasa el 5%.

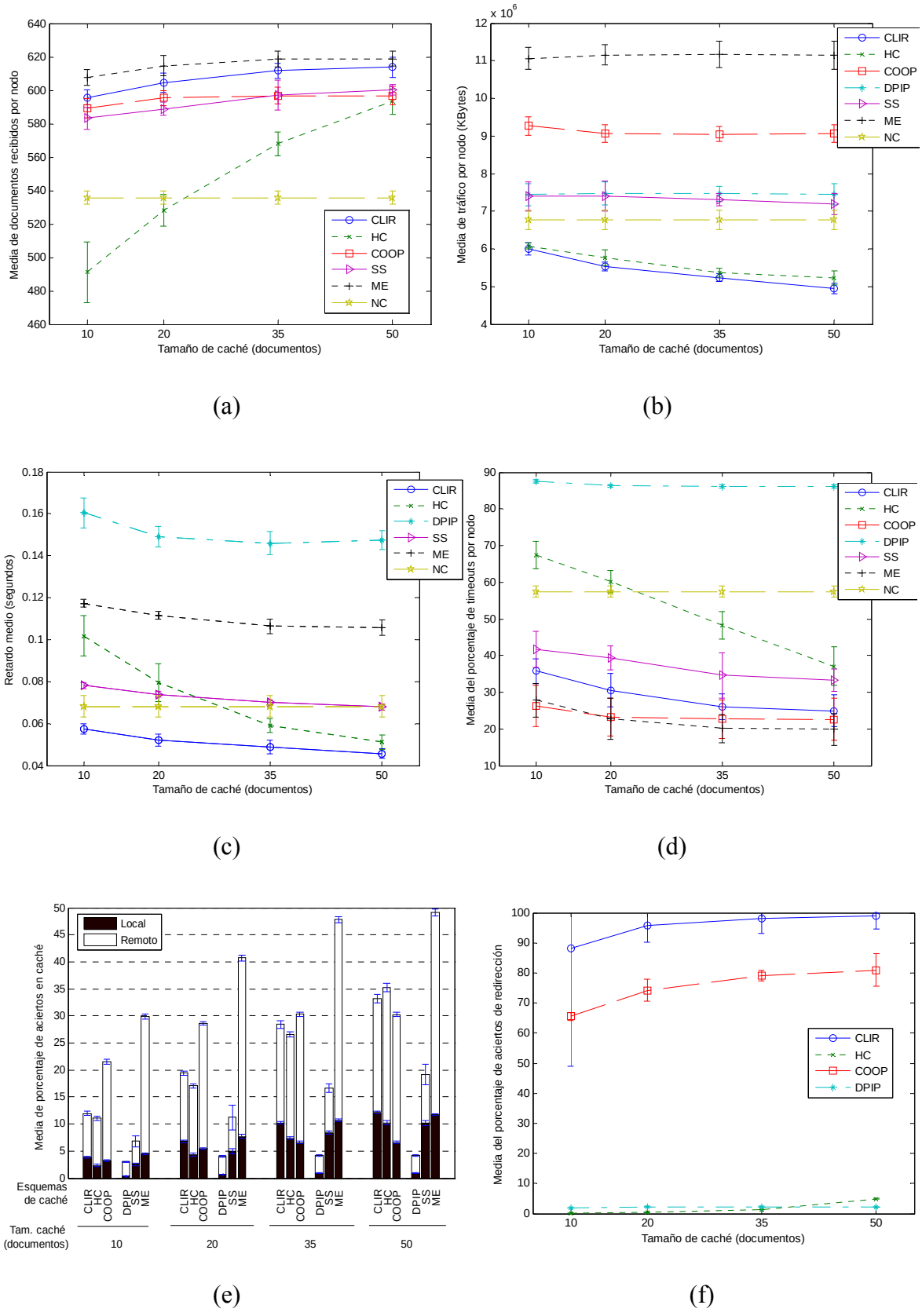
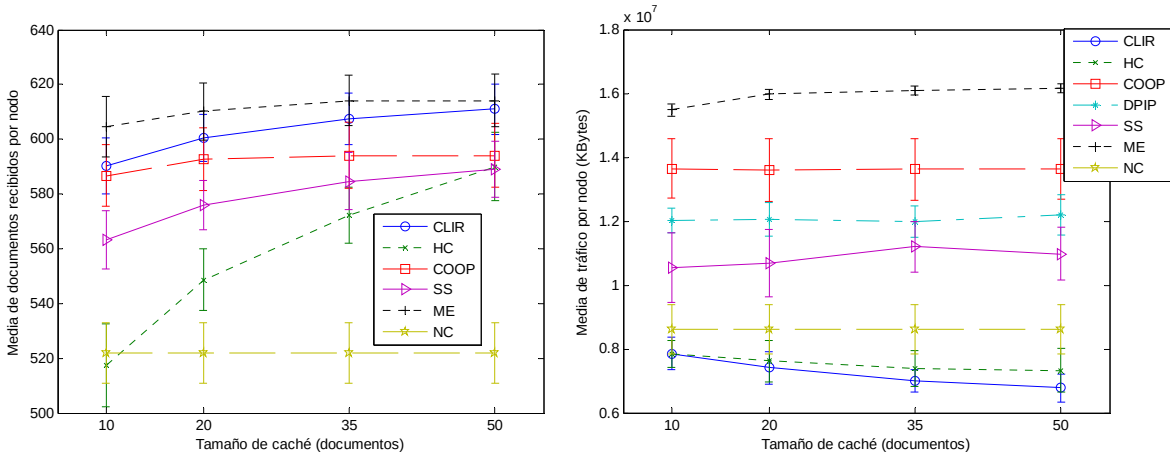
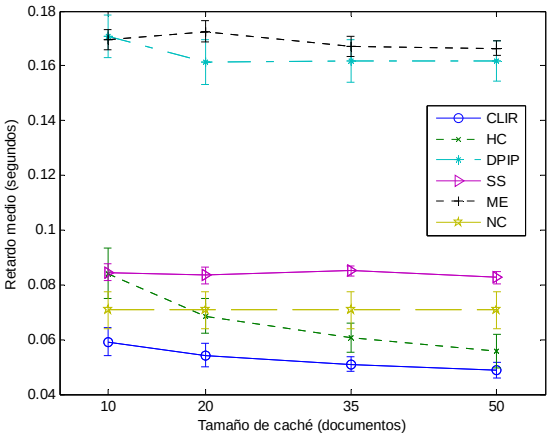


Figura 3.28. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función del tamaño de las cachés para el modelo de movilidad TVCM

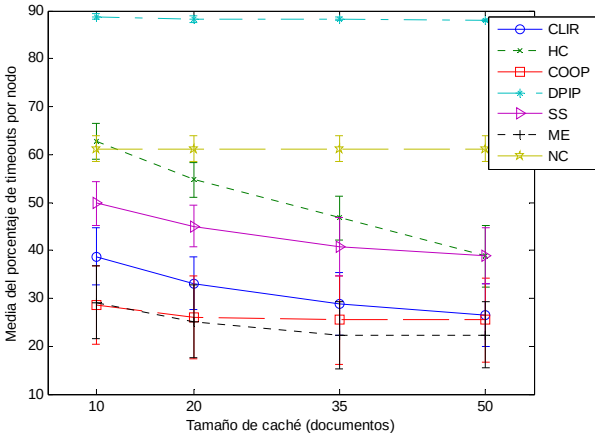


(a)

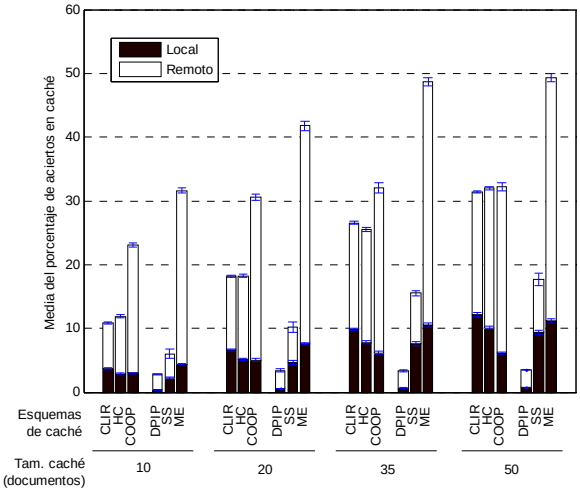
(b)



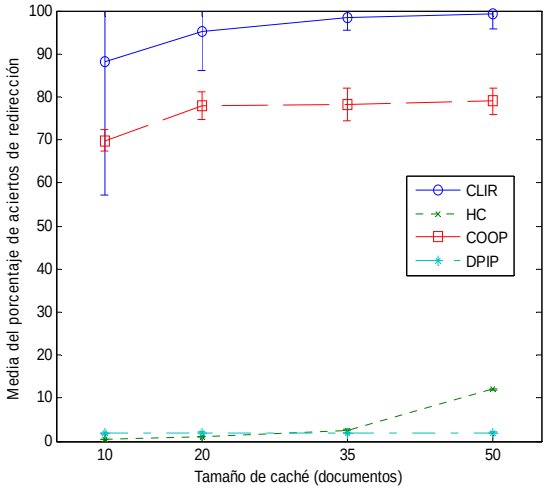
(c)



(d)



(e)



(f)

Figura 3.29. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función del tamaño de las cachés para el modelo de movilidad RWP

Bajo las condiciones analizadas, CLIR obtiene el menor retardo para todos los tamaños de caché. Además, el tráfico generado es también el más bajo pero obteniendo un número de documentos similar a los obtenidos por los esquemas de caché que usan peticiones de *broadcast*. Finalmente, el porcentaje de *timeouts* se reduce conforme se incrementa el tamaño de la caché hasta que resulta comparable con los esquemas de caché basados en *broadcast*.

3.5.7 Conclusiones

En este apartado se ha evaluado el rendimiento del esquema de caché CLIR en redes MANET empleando diferentes métricas: cantidad de documentos obtenidos durante el tiempo de simulación, tráfico procesado por cada nodo, retardo, *timeouts*, aciertos en caché y aciertos de redirección. Esta evaluación se ha realizado estudiando la influencia de la velocidad de los nodos, el número de nodos en la red, la carga de tráfico, la vigencia o TTL de los documentos, el patrón de popularidad de los documentos y el tamaño de las cachés. Además, se ha comparado el rendimiento de CLIR con los esquemas de caché HybridCache, COOP, DPIP, SimpleSearch y MobEye. También se ha comparado con el rendimiento de una red que no implemente ningún mecanismo de caché.

Del extenso conjunto de simulaciones realizadas, se puede concluir que el esquema de caché CLIR es capaz de obtener una cantidad similar, o incluso mayor, de documentos si lo comparamos con los esquemas de caché basados en *broadcast* COOP y MobEye. Sin embargo, CLIR es el esquema de caché que genera menos tráfico en la red, mientras que COOP y MobEye obtienen unos buenos resultados obteniendo documentos a costa de sobrecargar la red. Por otro lado, CLIR es también el esquema de caché que consigue el retardo más bajo para todas las variables estudiadas. Si se tiene en cuenta el porcentaje de *timeouts*, COOP y MobEye sólo obtienen un mejor rendimiento que CLIR para algunos de los parámetros estudiados a expensas de generar mucho más tráfico. Estas mejoras hacen que CLIR sea especialmente apropiado para redes MANET, donde el medio inalámbrico ofrece un ancho de banda reducido mientras que los usuarios demandan los mismos requisitos de calidad de servicio que en las redes cableadas. Finalmente, CLIR presenta un porcentaje de aciertos en caché local similar a los demás esquemas de caché reportando, a su vez, el mejor porcentaje de aciertos de redirección, ya que siempre es muy cercano al 100%. Este resultado demuestra que la administración de la información almacenada en la Caché de Redirecciones se realiza de forma eficiente.

De este estudio se han obtenido también las siguientes conclusiones:

- DPIP es el peor esquema de caché para todos los parámetros estudiados. Su bajo rendimiento es debido a la suposición de que los servidores siempre están accesibles, ya que esta condición no siempre es posible en las redes MANET. Además, el valor del parámetro *DPIP_Timer* no es suficiente en algunas circunstancias, ya que los mensajes de respuesta necesitan más tiempo para ser recibidos que el asignado a dicho temporizador.

- MobEye y SimpleSearch no son adecuados para redes muy densas debido al uso excesivo de peticiones *broadcast*.
- HybridCache es extremadamente sensible al tiempo medio de vida de los documentos y al tamaño de las cachés, ya que el algoritmo de redirección falla cuando los documentos tienen un gran tiempo de expiración o el tamaño de la caché es pequeño.
- El retardo obtenido por COOP es muy dependiente del valor temporizador seleccionado para enviar las peticiones a los servidores.

3.6 Evaluación de CLIR en redes ad hoc estáticas

Aunque tanto CLIR como el resto de esquemas de caché estudiados no estén inicialmente pensados para su uso en redes estáticas sino en redes móviles, se ha realizado también un estudio del rendimiento que ofrecen en redes estáticas. El presente apartado muestra los resultados de dicho estudio comparando el rendimiento obtenido por los esquemas de caché MobEye, SimpleSearch, COOP, DPIP e HybridCache. También se tiene en cuenta el rendimiento obtenido por la red al no implementar ningún esquema de caché.

3.6.1 Modelo del sistema

El modelo del sistema es el mismo que el utilizado en el apartado 3.5 para la evaluación del rendimiento en redes móviles con las siguientes excepciones:

- Los nodos de la red son estáticos y forman una malla regular de 5x5, 7x7 y 9x9 nodos equiespaciados respectivamente. Se evalúa, por tanto, el rendimiento de redes de 25, 49 y 81 nodos.
- Los servidores están situados en las esquinas del área de simulación, es decir, en las posiciones $(x,y)=(0,0)$ y $(x,y)=(1000,1000)$, siendo x e y coordenadas expresadas en metros.

Dado que el radio de cobertura de los nodos es de 250 metros y el área de simulación se fija en 1000x1000 m^2, la conectividad entre nodos es diferente para cada configuración de malla evaluada. La Figura 3.30 muestra la conectividad (nodos vecinos) para las distribuciones de 5x5, 7x7 y 9x9 nodos. Como puede observarse, cuando la densidad de nodos se incrementa, la cantidad de nodos vecinos disponibles también se incrementa.

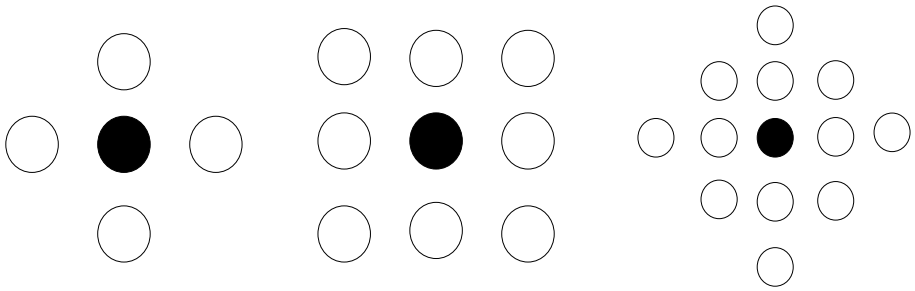


Figura 3.30. Conectividad de un nodo para vecinos a un salto en mallas de 5x5, 7x7 y 9x9 nodos respectivamente

La Tabla 3.9 resume los parámetros de simulación utilizados para la evaluación de redes estáticas.

Parámetro	Valor por defecto		Otros valores estudiados
Área de simulación (metros x metros)	1000x1000		
Nodos	49 (7x7)		25 (5x5) y 81 (9x9)
Servidores	2		
Número de documentos	1000		
Tamaño de los documentos (bytes)	1000		
Timeout de las peticiones (s)	3		
TTL (s)	2000		250-500-1000-2000-∞
Tiempo medio entre peticiones (s)	25		5-10-25-50
Patrón de tráfico (pendiente Zipf)	0.8		0.4-0.6-0.8-1.0
Política de reemplazo	LRU		
Tamaño caché local	35		5-10-35-50
Tamaño Caché de Redirecciones (registros)	35		
Tiempo de simulación (s)	20000		
Tiempo de calentamiento (s)	4000		
Protocolo MAC	802.11 b		
Modelo de propagación radio	Two Ray Ground		
Radio de cobertura (metros)	250		
Protocolo de encaminamiento	AODV	Local Repair Intermediate RREP	
Patrón de movilidad	Sin movilidad		

Tabla 3.9. Principales parámetros de simulación para las simulaciones con redes estáticas

3.6.2 Evaluación en función del tiempo entre peticiones

Las Figuras 3.31a, 3.32a y 3.33a representan la media de los documentos recibidos en el tiempo de simulación en función del tiempo medio entre peticiones para cada una de los esquemas de caché evaluados en redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. Todos los esquemas de caché obtienen un rendimiento similar para todos los tamaños de red excepto COOP y, sobretodo, HybridCache para tiempos entre

peticiones inferiores a 25 segundos en redes de 5x5 y 7x7 (Figuras 3.31a y 3.32a). Para redes con mucho tráfico (5 o 10 segundos entre peticiones) HybridCache obtiene un rendimiento muy pobre, llegando a obtener sólo la mitad de los documentos respecto a los demás esquemas de caché. Se observa también que, conforme el número de nodos en la red aumenta, el número de documentos obtenidos por MobEye se decrementa hasta el punto de resultar incluso inferior a HybridCache para redes muy cargadas de 9x9 nodos (Figura 3.33a). Esto es debido a que, cuanto mayor sea el número de nodos en la red, mayor será el tráfico generado (Figura 3.33b) al ser las peticiones generadas por MobEye en modo *broadcast*, con lo que se producirán más colisiones y pérdidas de paquetes, y se incrementará el número de *timeouts* (Figura 3.33d). El resto de esquemas de caché obtienen un número similar de documentos.

Las Figuras 3.31b, 3.32b y 3.33b muestran el tráfico medio procesado por nodo en función del tiempo entre peticiones para redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. Para redes pequeñas de 5x5 nodos (Figura 3.31b), CLIR, DPIP y COOP son los esquemas de caché que generan el menor tráfico, seguidos por SimpleSearch, MobEye y No Cache. HybridCache genera más tráfico que los demás esquemas de caché para un tiempo entre peticiones de 10 y 25 segundos y menor para redes muy cargadas (5 segundos entre peticiones). Al incrementarse el número de nodos en la red (Figuras 3.32b y 3.33b) el comportamiento es similar, aunque MobEye aumenta significativamente su tráfico generado ya que se utilizan más mensajes de *broadcast* para realizar las peticiones. SimpleSearch también ve incrementado el tráfico generado cuando el tiempo entre peticiones es muy alto (redes poco cargadas) debido también a que se realizan más peticiones *broadcast*.

Las Figuras 3.31c, 3.32c y 3.33c comparan el retardo medio que sufren las peticiones y respuestas en función del tiempo medio entre peticiones para redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. Para todos los tamaños de red, CLIR es el esquema de caché que menor retardo introduce y, para redes de 5x5 y 7x7 es el único que obtiene un retardo menor al obtenido cuando no se emplea caché. SimpleSearch y MobEye dependen de un sistema de cuatro mensajes petición-respuesta, por lo que obtienen un retardo mayor. Además, conforme la red se hace más grande, MobEye incrementa drásticamente su retardo dada la saturación de la red, que causa que los *buffers* en los nodos se llenen y tengan que esperar más tiempo para ser servidos. COOP no se ha representado en ninguna de las figuras dado que su retardo es muy alto, tal y como se ha comentado en apartados anteriores. Por otro lado, DPIP también obtiene un elevado retardo en comparación con el resto de esquemas de caché ya que depende del parámetro *DPIP_Timer*, que fija un límite inferior en el retardo de los mensajes. Finalmente, se aprecia que HybridCache tiene un rendimiento muy pobre en redes muy cargadas aunque el retardo obtenido se reduce significativamente conforme se decrementa la carga de tráfico en la red (tiempo entre peticiones de 50 segundos). Esto es debido a bucles en la redirección de peticiones causados por una mala gestión de la técnica de redirección. Cuando las peticiones se realizan cada más tiempo, las entradas de la tabla de redirecciones

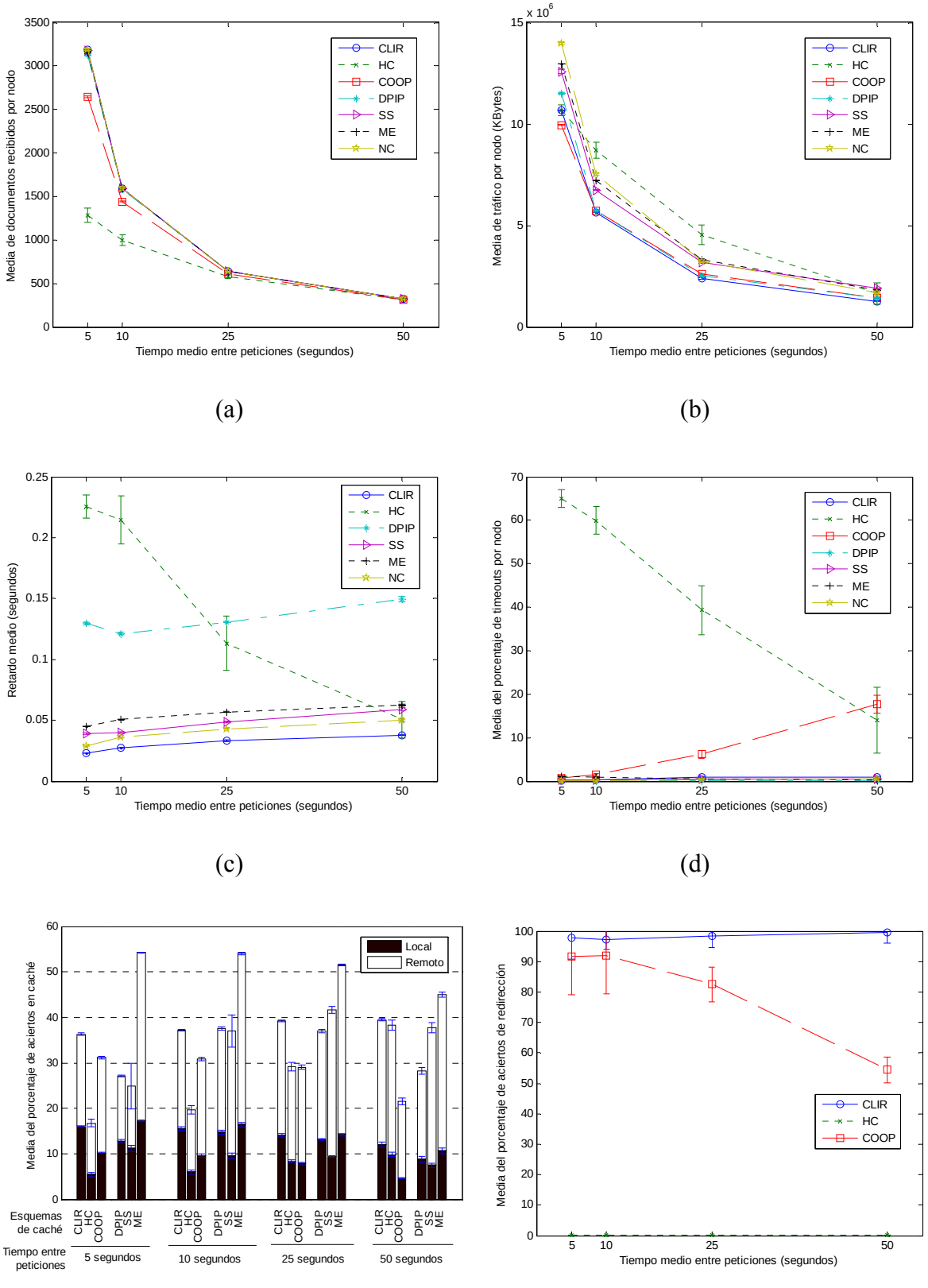


Figura 3.31. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función del tiempo medio entre peticiones para una malla de 5x5 nodos

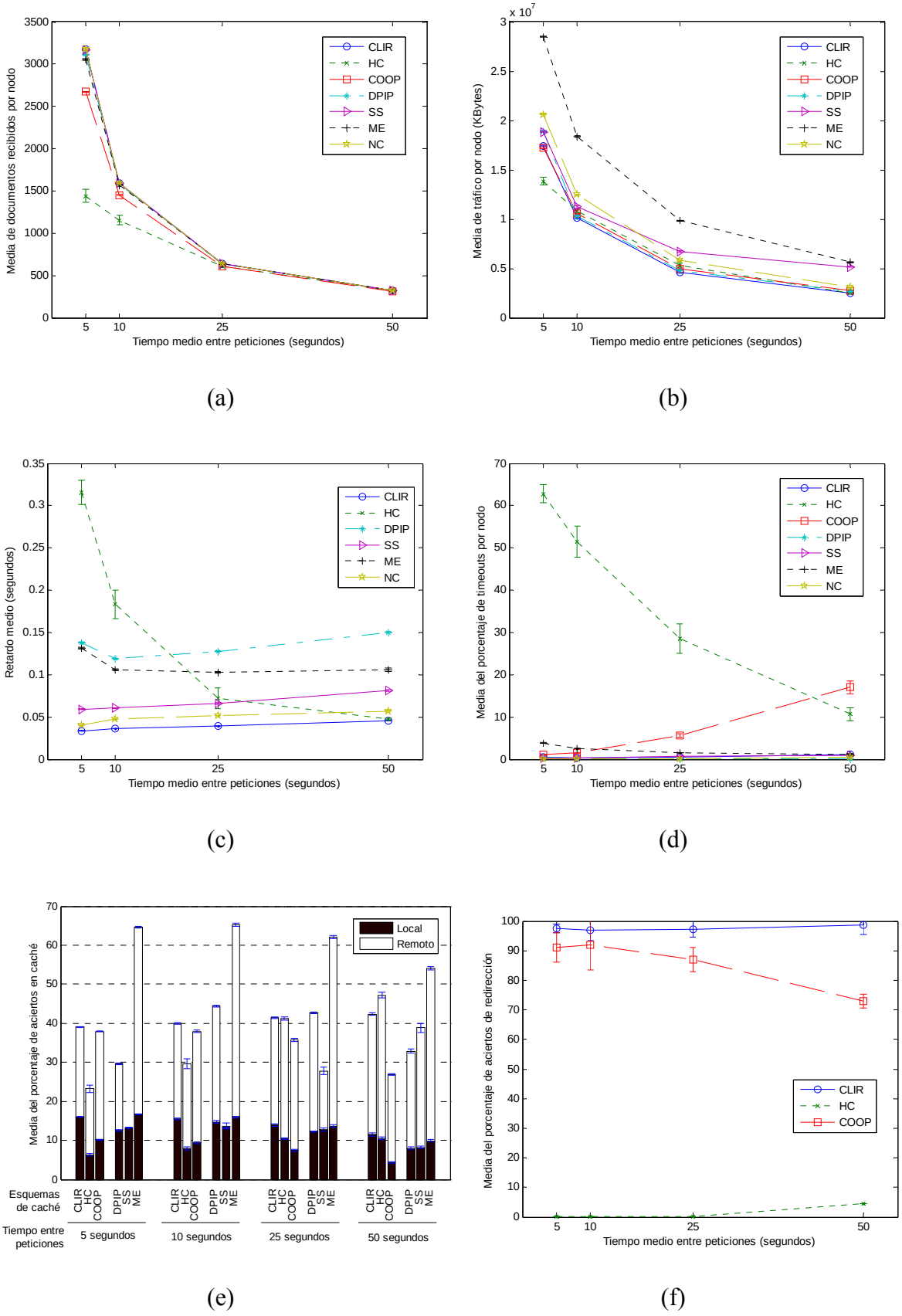


Figura 3.32. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función del tiempo medio entre peticiones para una malla de 7x7 nodos

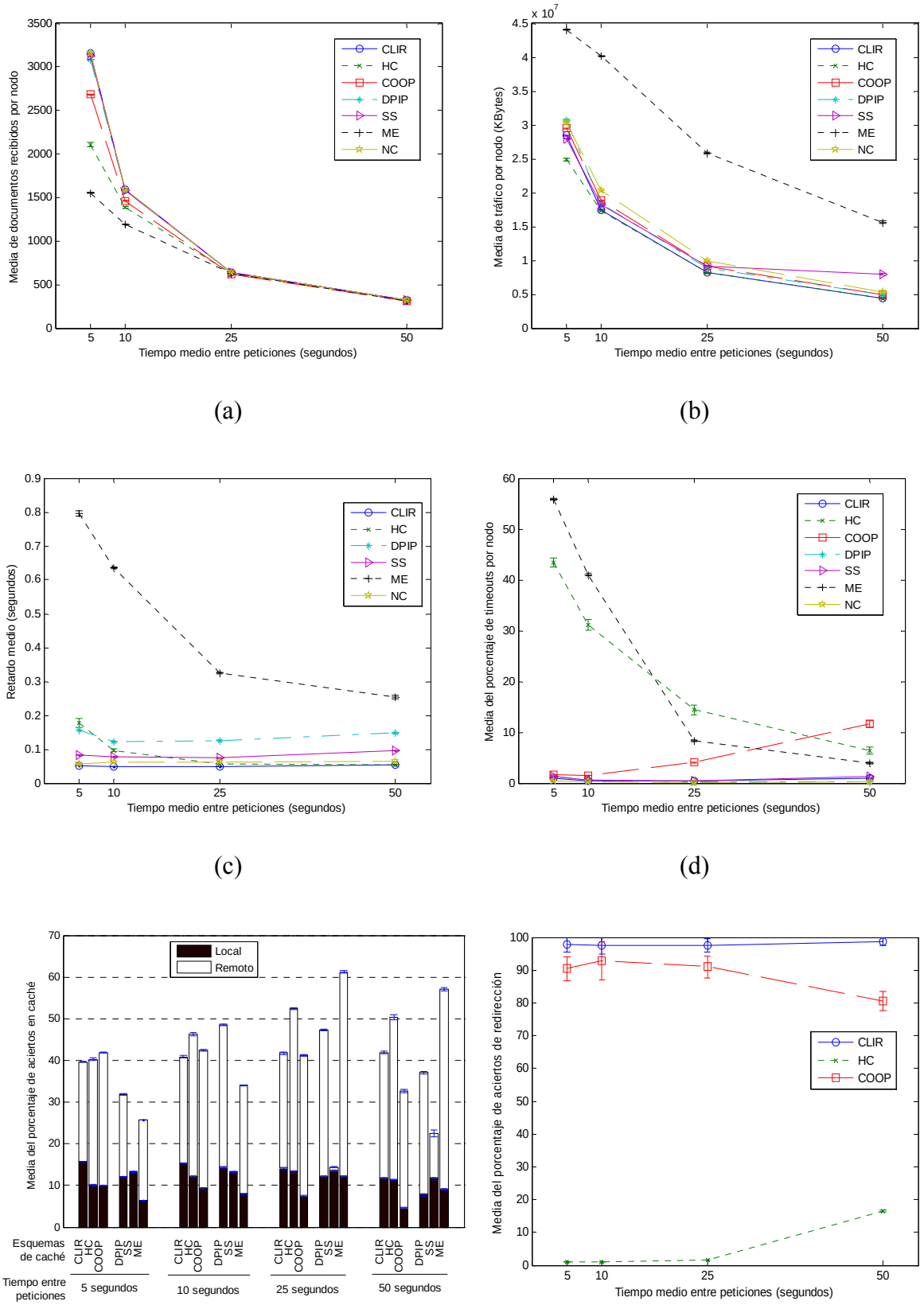


Figura 3.33. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función del tiempo medio entre peticiones para una malla de 9x9 nodos

están más actualizadas con respecto al estado de los documentos en las cachés, ya que hay menos documentos eliminados de las cachés locales, con lo que la caché de redirecciones puede acertar más al tener únicamente en cuenta el TTL de los documentos para eliminar una entrada de la tabla de redirecciones. Esta mala gestión de la tabla de redirecciones causa el incremento del retardo y también del número de *timeouts* (Figuras 3.31d, 3.32d y 3.33d).

Las Figuras 3.31d, 3.32d y 3.33d representan la media del porcentaje de *timeouts* en función del tiempo medio entre peticiones para redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. En dichas figuras se aprecia que HybridCache presenta un elevado porcentaje de *timeouts* para todos los tamaños de red por las razones explicadas anteriormente. COOP presenta también un comportamiento similar, aunque su porcentaje de *timeouts* se incrementa al decrementarse el tiempo entre peticiones. Esto es debido, al igual que ocurría con HybridCache, por un erróneo manejo de las redirecciones. En las Figuras 3.31f, 3.32f y 3.33f se aprecia cómo el porcentaje de aciertos de redirección de COOP se decrementa al incrementarse el tiempo entre peticiones, lo que causa el incremento del porcentaje de *timeouts*. MobEye, por otro lado, presenta un porcentaje de *timeouts* casi nulo para redes pequeñas (Figura 3.31d), aunque este porcentaje se va incrementando al hacerse más grande la red, sobretodo en redes muy cargadas, en las que el porcentaje llega al 55% (Figura 3.33d). Finalmente, el resto de esquemas de caché presentan un porcentaje de *timeouts* nulo o casi nulo.

Las Figuras 3.31e, 3.32e y 3.33e muestran el porcentaje de aciertos en caché en función del tiempo medio entre peticiones para redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. En redes pequeñas (Figura 3.31e) MobEye es el esquema de caché que más aciertos en caché consigue, aunque su rendimiento se va decrementando conforme decrece el volumen de peticiones en la red. Sin embargo, el resto de esquemas de caché presentan el comportamiento contrario. CLIR, DPIP y MobEye obtienen la mejor tasa de aciertos en caché local en redes poco cargadas, siendo COOP el que peor porcentaje obtiene para esta métrica. CLIR obtiene uno de los mejores rendimientos, sólo superado por MobEye y, ocasionalmente, por SimpleSearch. Conforme se incrementa el tamaño de la red, el porcentaje de aciertos en caché de HybridCache también se incrementa, lo que causa un ligero descenso en el retardo obtenido (Figura 3.33c). Para SimpleSearch, el porcentaje de aciertos remotos se hace prácticamente nulo para redes de 7x7 (Figura 3.32e) y 9x9 (Figura 3.33e) muy cargadas. Este comportamiento es debido a que, al realizarse las peticiones muy cercanas en el tiempo, no se emplean las peticiones en modo *broadcast*, ya que la ruta hacia los servidores sigue creada. Finalmente, la diferencia de rendimiento entre MobEye y el resto de esquemas de caché se ve reducido al aumentar el número de nodos en la red.

Las Figuras 3.31f, 3.32f y 3.33f comparan el porcentaje de aciertos de redirección en función del tiempo medio entre peticiones para redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. Como se ha comentado anteriormente, COOP ve reducida su tasa de aciertos en caché remota conforme el tiempo entre peticiones se va haciendo mayor. Esta reducción es más leve en redes de mayor número de nodos. CLIR obtiene un rendimiento

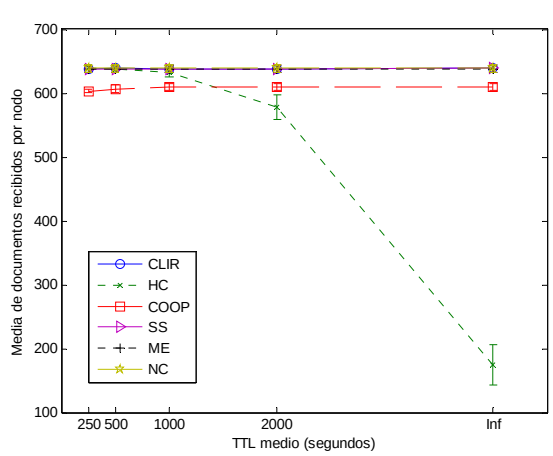
cercano al 100% para todos los valores estudiados, mientras que HybridCache muestra un porcentaje de aciertos de redirección prácticamente nulo a excepción de las redes con 7x7 nodos (Figura 3.32f) y 9x9 nodos (Figura 3.33f) con muy poco tráfico, en el que el rendimiento se incrementa hasta 5% y el 20% respectivamente.

Del presente estudio se puede concluir que, con respecto a la carga de tráfico en la red (tiempo medio entre peticiones), COOP e HybridCache no son adecuados para este tipo de redes ya que obtienen un retardo muy elevado y una tasa de *timeouts* también elevada. Además, son los esquemas de caché que menos documentos obtienen en redes cargadas, independientemente del número de nodos en la red. MobEye tampoco es adecuado ya que el excesivo tráfico que genera en la red produce que el retardo y el porcentaje de *timeouts* se incremente en redes de tamaño medio y grande. DPIP obtiene un gran retardo debido, en parte, al parámetro *DPIP_Timer*. Finalmente, CLIR resulta la mejor opción ya que obtiene un menor retardo que SimpleSearch para todos los tamaños de red y tiempo entre peticiones generando, además, menos tráfico.

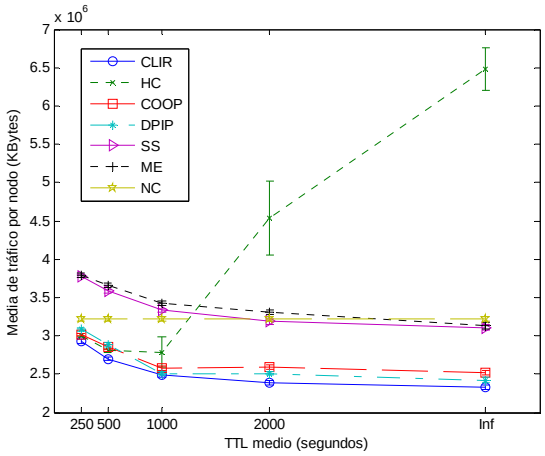
3.6.3 Evaluación en función del periodo de vigencia de los documentos (TTL)

Las Figuras 3.34a, 3.35a y 3.36a representan la media de documentos recibidos en función del TTL medio de los documentos para redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. Como puede observarse, todos los esquemas de caché obtienen un número similar de documentos excepto COOP, que presenta un rendimiento ligeramente inferior, e HybridCache cuyo rendimiento, para documentos con TTL mayor que 1000 segundos, cae drásticamente hasta llegar a obtener menos de un tercio de los documentos que obtienen el resto de esquemas de caché para el caso de que los documentos no caduquen. Este comportamiento ya se constató para redes móviles y demuestra que HybridCache es muy sensible al TTL de los documentos debido a la mala gestión de la tabla de redirecciones, ya que, para documentos con un TTL medio superior a 1000 segundos, el porcentaje de aciertos de redirección es prácticamente nulo (Figuras 3.34f, 3.35f y 3.36f). Esta mala gestión origina más tráfico en la red (Figuras 3.34b 3.35b y 3.36b) ocasionado por un mayor número de *timeouts* (Figuras 3.34d, 3.35d y 3.36d) causando un mayor retardo (Figuras 3.34c, 3.35c y 3.36c). Para redes de 9x9 nodos, MobEye también ve reducida la cantidad de documentos que es capaz de obtener debido a la sobrecarga en la red ocasionada por los mensajes de petición en modo *broadcast*.

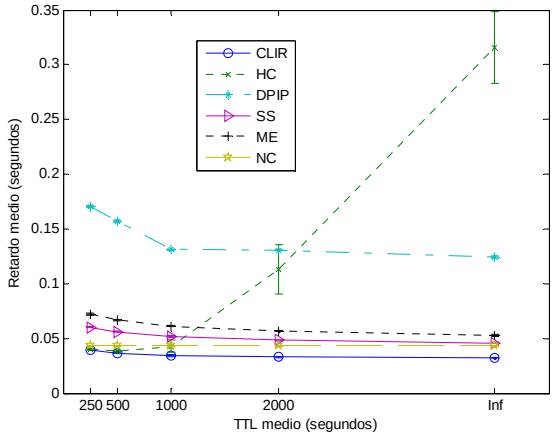
Las Figuras 3.34b, 3.35b y 3.36b muestran el trafico medio procesado por cada nodo en función del TTL medio de los documentos para redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. Para redes de 5x5 (Figura 3.34b), COOP, DPIP y CLIR, para todos los TTLs estudiados, generan menos tráfico que la opción de no cachear. Conforme el tamaño de la red se incrementa, el resto de esquemas de caché también consiguen generar menos tráfico que la opción de no cachear, excepto MobEye (Figuras 3.35b y 3.36b), que llega a generar más del doble de tráfico que el resto de esquemas de caché para redes de



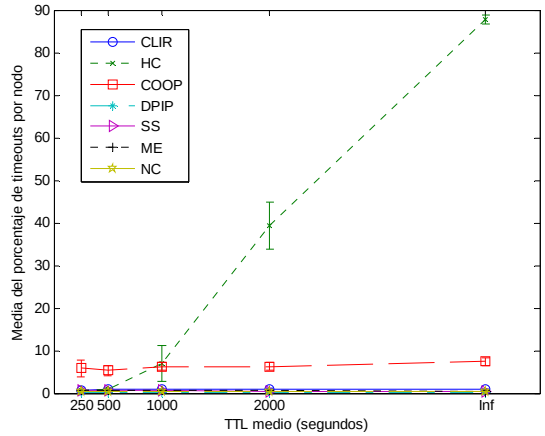
(a)



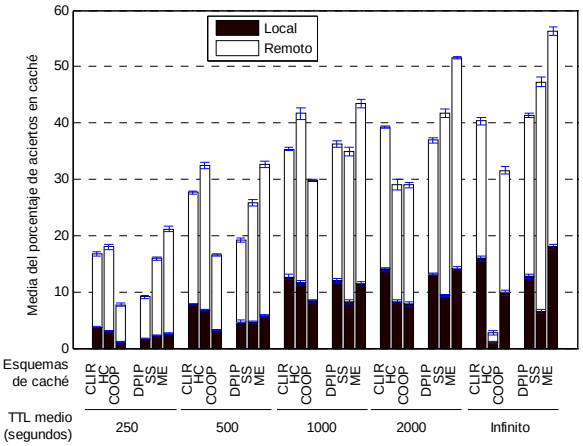
(b)



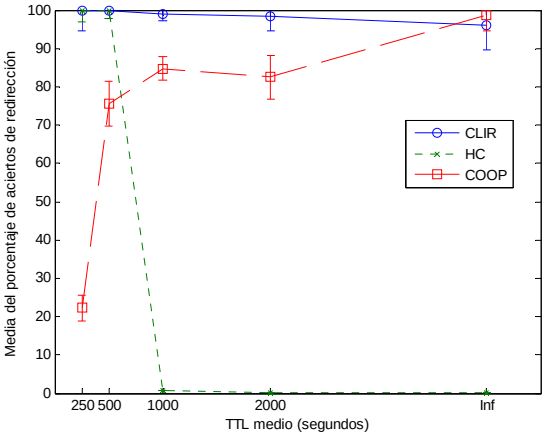
(c)



(d)



(e)



(f)

Figura 3.34. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función del TTL de los documentos para una malla de 5x5 nodos

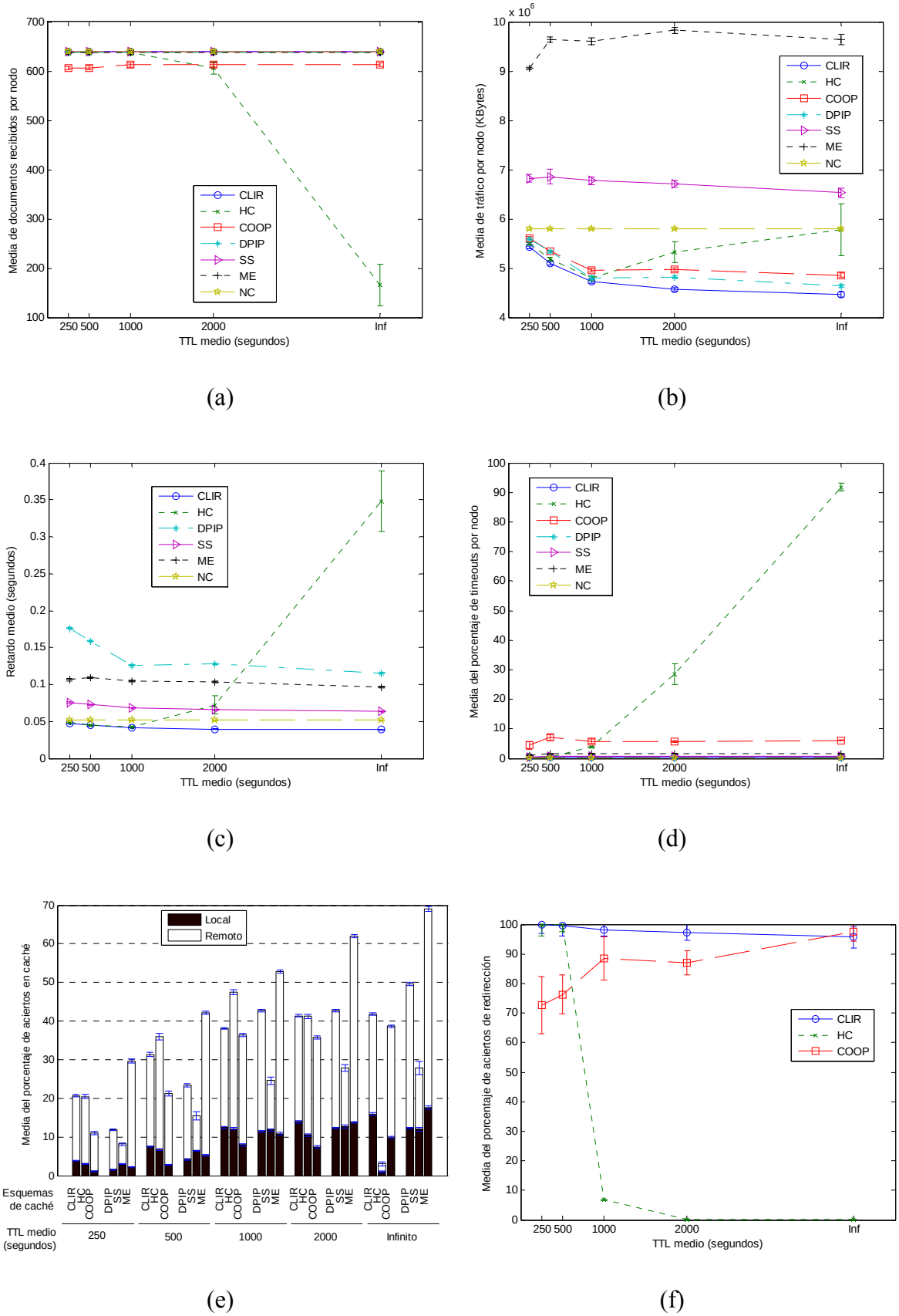


Figura 3.35. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función del TTL de los documentos para una malla de 7x7 nodos

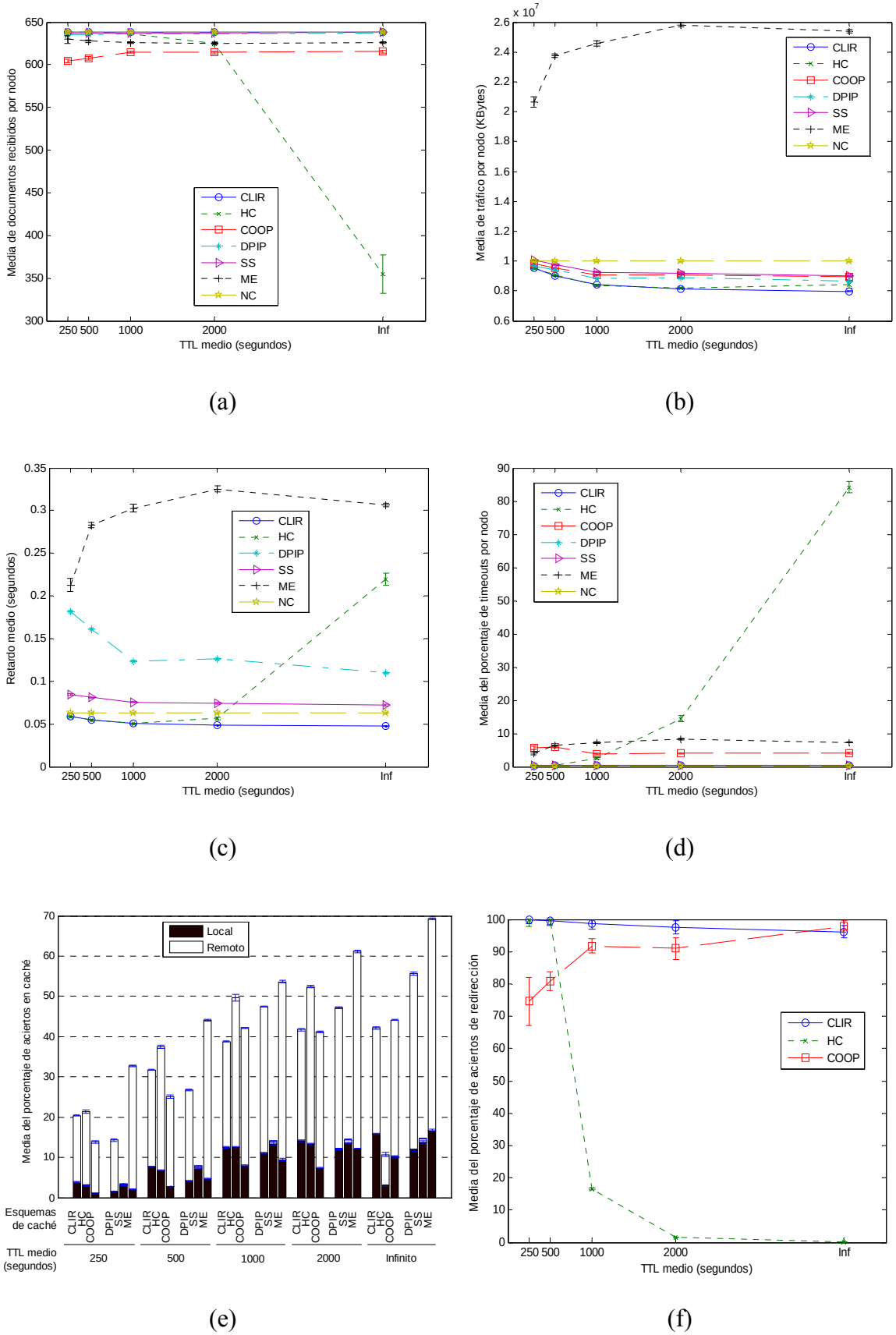


Figura 3.36. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función del TTL de los documentos para una malla de 9x9 nodos

9x9 nodos. Para todos los casos y, sobretodo, para redes pequeñas, CLIR es el esquema de caché que menos tráfico genera.

Las Figuras 3.34c, 3.35c y 3.36c comparan el retardo medio en la obtención de los documentos en función del TTL medio de los mismos para redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. Tal y como ocurría en estudios anteriores, CLIR es el esquema de caché que menor retardo presenta, sólo igualado por HybridCache para TTL medios inferiores a 2000 segundos. El resto de esquemas de caché obtiene retardos superiores a la opción de no usar cachés. Puede apreciarse también que, conforme se incrementa el número de nodos en la red, el retardo sufrido por MobEye se incrementa considerablemente debido al uso de peticiones de *broadcast* y la sobrecarga que introduce en la red. El retardo obtenido por COOP no ha sido representado por ser excesivamente elevado en comparación con los demás esquemas de caché.

Las Figuras 3.34d, 3.35d y 3.36d ilustran la evolución del porcentaje medio de *timeouts* en función del TTL medio de los documentos para redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. Para redes de 5x5 nodos (Figura 3.34d), sólo COOP e HybridCache presentan un porcentaje de *timeouts* distinto a cero debido a la mala gestión de las redirecciones tal y como se observa en la Figura 3.34f. Para redes mayores, COOP e HybridCache también presentan un porcentaje de *timeouts* mayor que cero por la misma razón comentada (Figuras 3.35f y 3.36f), al igual que MobEye, pero en este último caso el fenómeno es debido a la sobrecarga en la red por las peticiones de *broadcast*, causando pérdidas de peticiones y respuestas en la red.

Las Figuras 3.34e, 3.35e y 3.36e muestran el porcentaje de aciertos en caché en función del TTL de los documentos para redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. Conforme se aumenta el TTL medio de los documentos, éstos permanecen más tiempo en las cachés locales, con lo que la probabilidad de acierto, tanto en caché local como remota, también se espera que se vea incrementada. HybridCache obtiene un buen rendimiento tanto en aciertos en caché local como caché remota, aunque este rendimiento cae drásticamente cuando los documentos no caducan. CLIR obtiene un rendimiento del orden del resto de esquemas de caché y únicamente superado por MobEye y DPIP para TTLs medios muy grandes. SimpleSearch es el esquema de caché que obtiene peores resultados para todos los TTLs en redes medias (7x7 nodos) y grandes (9x9 nodos), donde el porcentaje de aciertos de redirección llega a ser casi cero. Este comportamiento se debe a que las peticiones en modo *broadcast* en SimpleSeach se realizan únicamente cuando no hay una ruta creada hacia el servidor y, conforme se incrementa el número de nodos, la mayor conectividad entre ellos causa que la probabilidad de que la ruta siga creada crezca, por lo que las peticiones *broadcast* se verán reducidas.

Las Figuras 3.34f, 3.35f y 3.36f comparan el porcentaje de aciertos de redirección en función del TTL de los documentos, para aquellos esquemas de caché que usan redirección, en redes malladas de 5x5, 7x7 y 9x9 respectivamente. Como se comentó anteriormente, HybridCache obtiene tasas de aciertos de redirección muy elevadas para valores de TTL medios pequeños aunque, conforme se incrementa la vigencia de los

documentos, esta tasa cae hasta hacerse prácticamente nula si el TTL es superior o igual a 2000 segundos. Por otro lado, se observa que COOP también es sensible al valor del TTL de los documentos, obteniendo un porcentaje menor de aciertos de redirección cuando los documentos son muy variables (TTL de 250 segundos). Este hecho se hace más patente en redes de 5x5 en las que la tasa de aciertos es cercana al 20 % (Figura 3.34f). Conforme se incrementa el TTL medio de los documentos, COOP incrementa su tasa de aciertos hasta hacerla cercana al 100 %. Finalmente, CLIR es capaz de obtener un porcentaje de aciertos de redirección cercano al 100 % independientemente del TTL medio de los documentos.

Del estudio realizado sobre el efecto de la vigencia (TTL) de los documentos se puede concluir que HybridCache es muy sensible al mismo, resultando sólo adecuado con un TTL es muy bajo, es decir, cuando los documentos caducan con mucha frecuencia. MobEye presenta una elevada generación de tráfico y un elevado retardo causados por las peticiones en modo *broadcast*. COOP presenta un elevado retardo y, además, produce *timeouts* debido a los errores de redirección de las peticiones. Del resto de esquemas de caché, CLIR es el que menor tráfico genera y menores retardos obtiene para todos los valores de TTL estudiados, por lo que es el más adecuado para este tipo de redes.

3.6.4 Evaluación en función de la pendiente Zipf

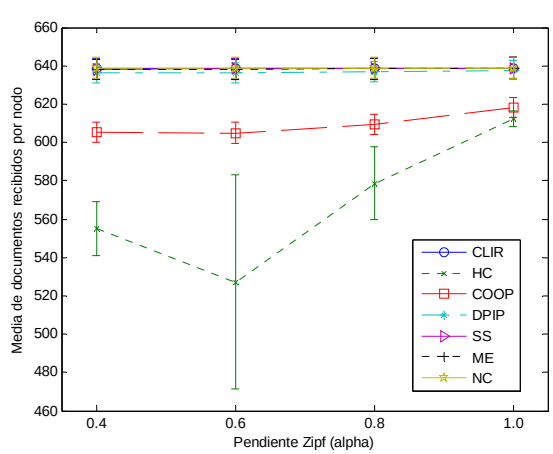
Las Figuras 3.37a, 3.38a y 3.39a representan la media de documentos recibidos en función de la pendiente de la distribución Zipf para redes malladas de 5x5, 7x7 y 9x9 respectivamente. Como en estudios anteriores HybridCache obtiene los peores resultados con respecto a la cantidad de documentos obtenidos, aunque esta métrica aumenta cuando la pendiente Zipf se incrementa, ya que se consiguen mayores aciertos en caché. Debido a los fallos de redirección de HybridCache (Figuras 3.37f 3.38f y 3.39f) se producen altas tasas de *timeouts* (Figuras 3.37d, 3.38d y 3.39d) lo que provoca una reducción del número de documentos obtenidos. Por la misma razón, COOP también consigue un rendimiento inferior al del resto de esquemas de caché. El número de documentos rescatados por MobEye con respecto al resto de esquemas de caché se reduce conforme el número de nodos en la red se incrementa. De esta forma, para redes de 5x5 nodos (Figura 3.37a), MobEye consigue una cantidad de documentos similar a CLIR, SimpleSearch, DPIP y la opción de no usar cachés. En redes de 7x7 nodos (Figura 3.38a) MobEye ya obtiene un rendimiento ligeramente inferior, mientras que en redes de 9x9 nodos (Figura 3.39a), el rendimiento de MobEye decae considerablemente debido al incremento del tráfico generado en la red (Figura 3.39b), lo que provoca un incremento del porcentaje de *timeouts* (Figura 3.39d). CLIR, DPIP, SimpleSearch y la opción de no usar cachés obtienen una cantidad de documentos similar para todos los tamaños de red estudiados y pendientes Zipf.

Las Figuras 3.37b, 3.38b y 3.39b muestran el tráfico medio procesado por nodo en función de la pendiente de la distribución Zipf para redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. En redes de 5x5 nodos (Figura 3.37b), CLIR, DPIP y COOP generan menos tráfico que la opción de no usar caché para todas las pendientes estudiadas,

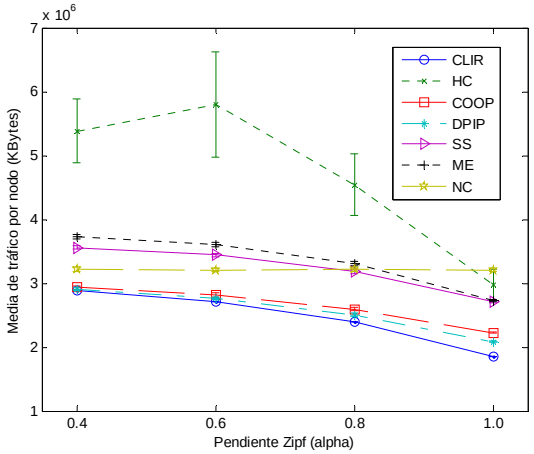
aunque MobEye y SimpleSearch también consiguen generar menos tráfico que no usar caché para una pendiente Zipf de 1.0. HybridCache es el esquema de caché que más tráfico genera en redes de 5x5 nodos. Conforme la cantidad de nodos de la red se incrementa (Figuras 3.38b y 3.39b) el tráfico generado por MobEye se incrementa considerablemente respecto al generado por el resto de esquemas de caché debido al uso de mensajes de petición en modo *broadcast*, mientras que el tráfico que genera HybridCache se hace del orden del resto de esquemas de caché. CLIR es el esquema de caché que menos tráfico genera para todos los tamaños de red y pendientes Zipf estudiadas.

Las Figuras 3.37c, 3.38c y 3.39c comparan el retardo medio en función de la pendiente de la distribución Zipf para redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. El retardo obtenido por COOP no se muestra al ser muy superior al del resto de esquemas de caché. Únicamente CLIR y MobEye para redes medias y grandes con pendientes Zipf altas obtienen retardos inferiores al obtenido por la opción de no usar cachés. DPIP obtiene un retardo unas tres veces superior al resultante de aplicar CLIR en redes de 5x5 nodos (Figura 3.37c), aunque este retardo se vea reducido conforme se incrementa la pendiente Zipf debido al aumento de los aciertos en caché (Figuras 3.37e, 3.38e y 3.39e). Esta diferencia en el retardo se mantiene para redes mayores (Figuras 3.38c y 3.39c). Por otro lado, el retardo de MobEye también se ve incrementado conforme se aumenta el número de nodos en la red debido a la sobrecarga de tráfico (Figuras 3.38b y 3.39b) hasta llegar a ser el más elevado (si exceptuamos a COOP) tal y como se observa en la Figura 3.39c). CLIR es el esquema de caché que consigue el menor retardo para todos los tamaños de red independientemente de la pendiente Zipf.

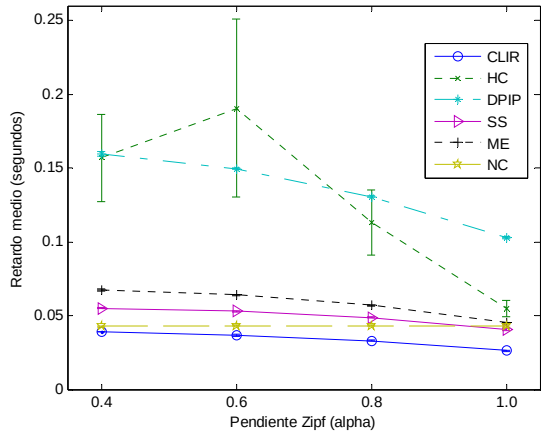
Las Figuras 3.37d, 3.38d y 3.39d representan el porcentaje medio de *timeouts* en función de la pendiente de la distribución Zipf para redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. Como en estudios anteriores, únicamente HybridCache, COOP y MobEye (en redes de 7x7 y 9x9) muestran un porcentaje de *timeouts* diferente a cero. HybridCache y COOP presentan dicho comportamiento debido a los errores de redirección (Figuras 3.37f, 3.38f y 3.39f). Por otro lado MobEye ve incrementado su porcentaje de *timeouts* debido al incremento del tráfico generado, lo que provoca un incremento de las interferencias y de la pérdida de paquetes. El porcentaje de *timeouts* que presentan HybridCache y COOP se ve decrementado conforme se aumenta la pendiente Zipf aunque el porcentaje de aciertos de redirección se vea reducido en COOP. Esto es debido a que, conforme se incrementa la pendiente Zipf, el porcentaje de aciertos en caché local y remota también se incrementa, con lo que es más probable que los documentos puedan ser servidos antes de que salte el *timeout.* Sin embargo, el comportamiento de MobEye es el contrario para redes grandes (Figura 3.39d), comprobándose que el porcentaje de *timeouts* crece con la pendiente Zipf. Esto se justifica en que el porcentaje de aciertos en caché, tanto local como remota, ronda entre el 58% y 68% (Figura 3.39e) y, aunque el porcentaje de aciertos en caché local se incrementa con la pendiente Zipf, el porcentaje de aciertos en caché remota se reduce. En consecuencia, aunque se disminuye el tráfico generado al aumentar el porcentaje de aciertos en caché local, aumenta el tráfico al reducirse el



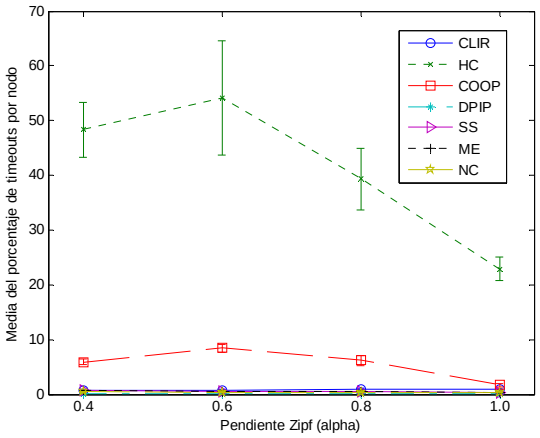
(a)



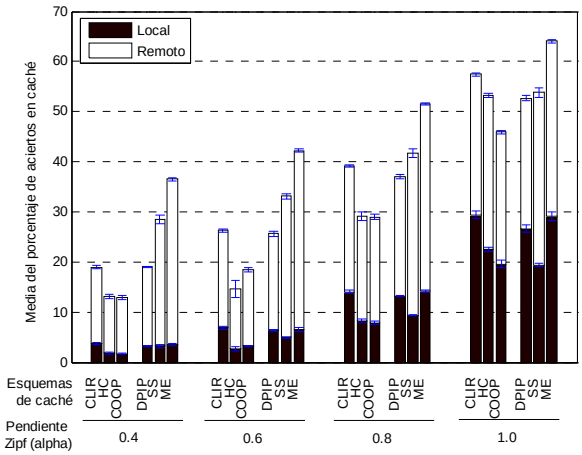
(b)



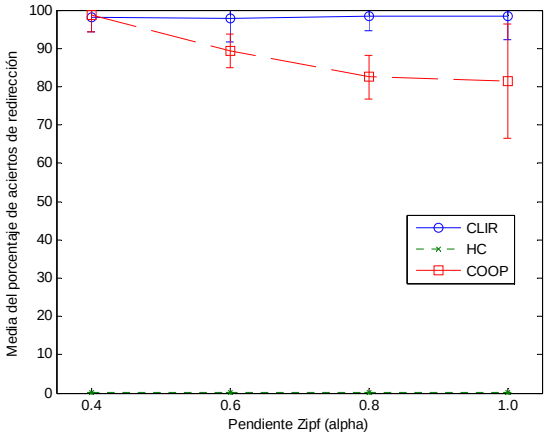
(c)



(d)



(e)



(f)

Figura 3.37. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función de la pendiente Zipf para una malla de 5x5 nodos

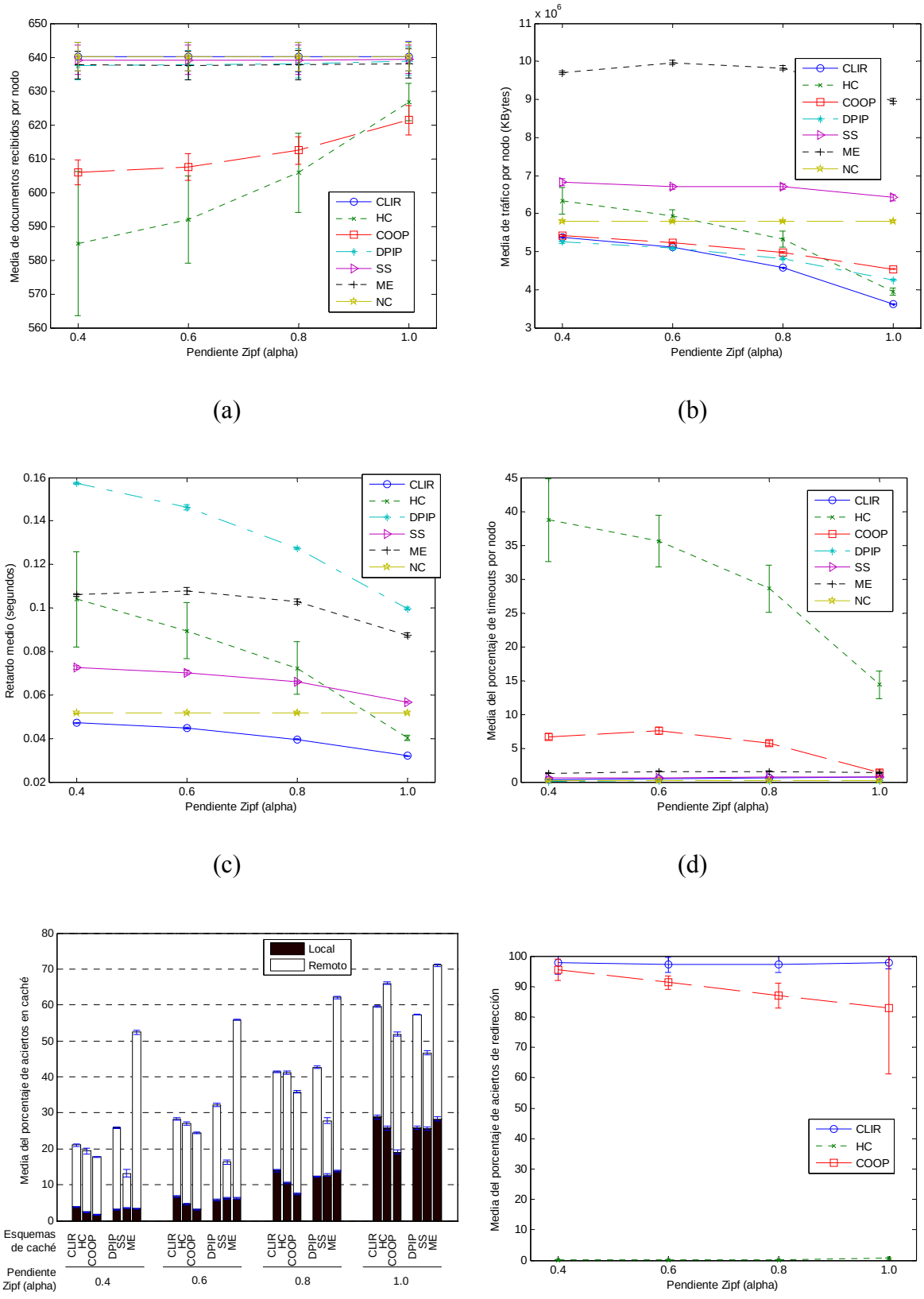
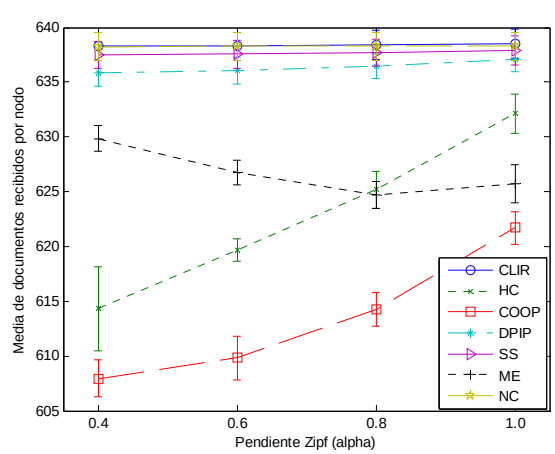
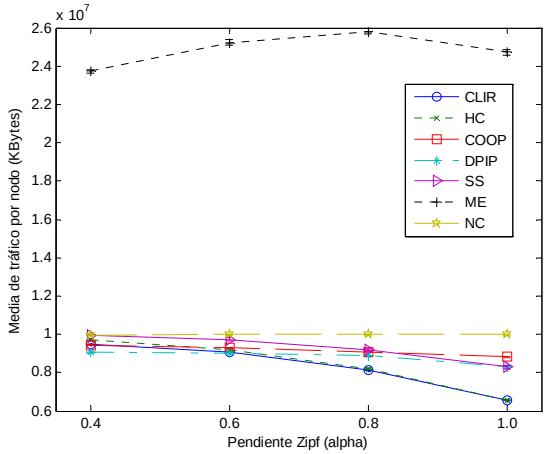


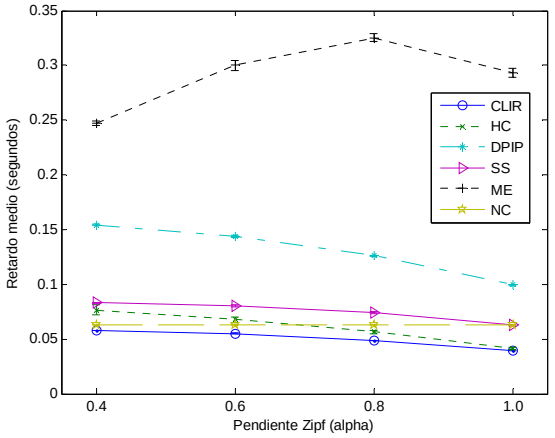
Figura 3.38. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función de la pendiente Zipf para una malla de 7x7 nodos



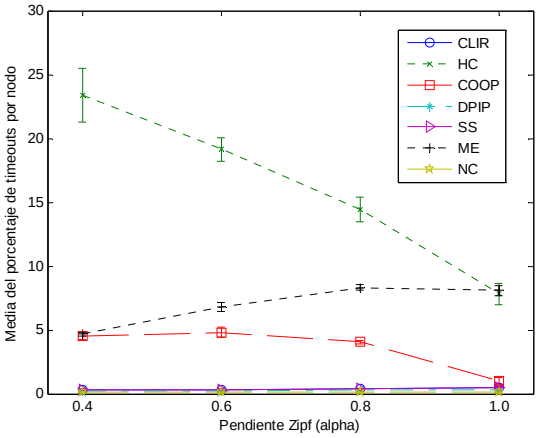
(a)



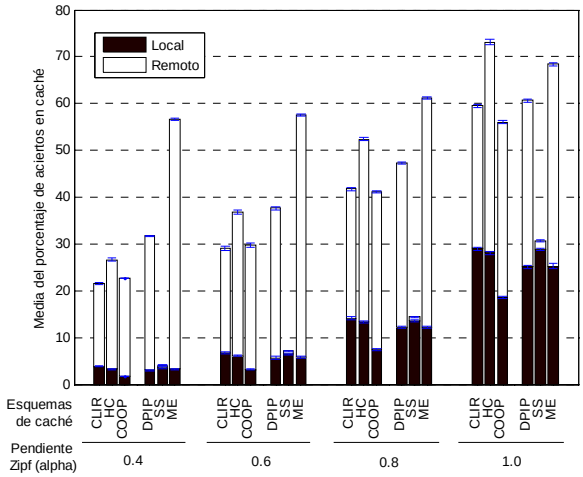
(b)



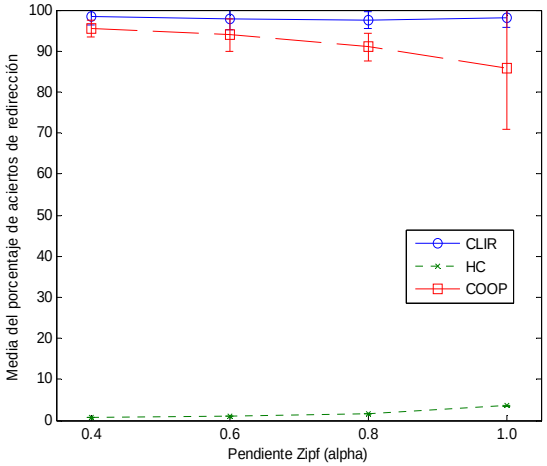
(c)



(d)



(e)



(f)

Figura 3.39. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función de la pendiente Zipf para una malla de 9x9 nodos

porcentaje de aciertos en caché remota. Como consecuencia, el tráfico global generado también se incrementa con la pendiente Zipf (Figura 3.39b), excepto para una pendiente 1.0, en la que el porcentaje de aciertos en caché es superior (Figura 3.39e), causando que el porcentaje de *timeouts* no crezca para dicha pendiente.

Las Figuras 3.37e, 3.38e y 3.39e muestran el porcentaje de aciertos en caché en función de la pendiente Zipf para redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. La pendiente Zipf indica la probabilidad de que los documentos más populares sean referenciados más veces, por lo que se espera que, al incrementarse la pendiente Zipf, los aciertos en caché local también se vean incrementados. Dicho comportamiento se puede constatar en todas las figuras. MobEye, como en otros estudios, es el esquema de caché que más aciertos en caché consigue debido al uso de peticiones en modo *broadcast*, aunque HybridCache le supere en redes de 9x9 nodos y pendiente 1.0 (Figura 3.39e). SimpleSearch, tal y como ocurría en estudios previos, reduce su tasa de aciertos en caché conforme aumenta el tamaño de la red llegando a ser prácticamente nula para pendientes Zipf pequeñas. CLIR es el esquema de caché que mayor porcentaje de aciertos en caché local presenta para todas las pendientes Zipf y tamaños de red.

Las Figuras 3.37f, 3.38f y 3.39f comparan el porcentaje de aciertos de redirección en función de la pendiente Zipf para aquellos esquemas de caché que usan redirección, en redes malladas de 5x5, 7x7 y 9x9 respectivamente. Como ocurría en estudios anteriores, HybridCache tiene una tasa de aciertos de redirección prácticamente nula, aunque en redes de 9x9 nodos (Figura 3.39f) llegue al 5% cuando la pendiente Zipf es 1.0. COOP ve reducido su porcentaje de aciertos de redirección conforme se incrementa la pendiente Zipf mientras que los resultados de CLIR se mantienen siempre cercanos al 100% independientemente de la pendiente Zipf utilizada.

Del estudio realizado sobre el efecto de la pendiente Zipf se puede concluir que el número de documentos que tanto HybridCache como COOP son capaces de obtener es muy sensible a este parámetro, siendo en todos los casos inferior al del resto de esquemas de caché. MobEye ve incrementado el tráfico que genera al aumentarse el tamaño de la red por lo que no es aplicable en grandes redes. Por otro lado, HybridCache es el esquema de caché que más tráfico genera y más retardo presenta en redes pequeñas, reduciéndose la diferencia de generación de tráfico y el retardo con el resto de esquemas de caché al incrementarse el tamaño de la red. Únicamente HybridCache, COOP y MobEye en redes grandes presentan *timeouts*. Por todas estas razones, HybridCache, COOP y MobEye no son adecuados para este tipo de redes, independientemente de la pendiente Zipf. Por otro lado, CLIR es el esquema de caché que menos tráfico genera y menor retardo introduce para todos los tamaños de red y pendientes Zipf analizadas.

3.6.5 Evaluación en función del tamaño de las cachés

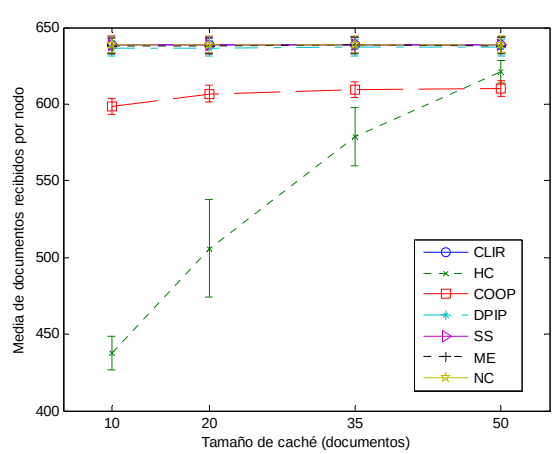
Las Figuras 3.40a, 3.41a y 3.42a representan la media de documentos recibidos en función del tamaño de las cachés implementadas en los nodos para redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. HybridCache es muy sensible al tamaño de las cachés

locales, consiguiendo muy pocos documentos cuando se emplean cachés muy pequeñas, aunque este número se ve incrementado conforme aumenta el tamaño de la caché. Este bajo rendimiento de la política de reemplazo SXO de HybridCache con pequeños tamaños de caché ocasiona altos porcentajes de *timeouts* (Figuras 3.40d, 3.41d y 3.42d), lo que incrementa también el retardo (Figuras 3.40c, 3.41c y 3.42c). Sin embargo, conforme el tamaño de la red se va haciendo mayor, la diferencia con respecto al resto de esquemas de caché que presenta HybridCache en términos de retardo y documentos obtenidos, se hace menor. COOP, por su parte, también obtiene menos documentos que CLIR, DPIP, SimpleSearch y la opción de no cachear, mientras que el rendimiento de MobEye también es menor para redes grandes (Figura 3.42a). El número de documentos obtenidos en el tiempo de simulación por CLIR, DPIP y SimpleSearch es independiente del tamaño de las cachés.

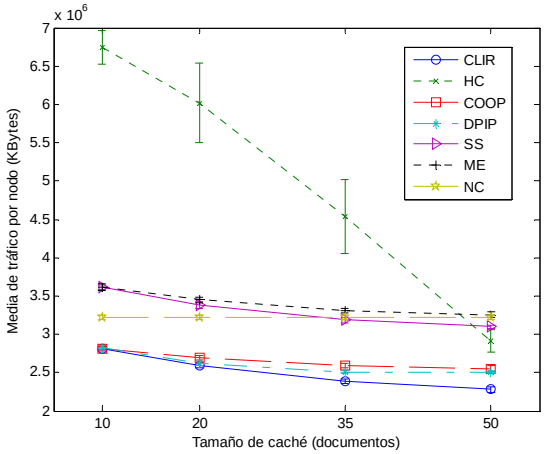
Las Figuras 3.40b 3.41b y 3.42b muestran el tráfico medio procesado por nodo en función del tamaño de las cachés para redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. Conforme el tamaño de las cachés se incrementa, el tráfico generado se decrementa ya que la cantidad de aciertos en caché también se ve incrementado (Figuras 3.40e 3.41e y 3.42e), siendo CLIR, DPIP y COOP los únicos esquemas de caché que generan menos tráfico que la opción de no cachear para todos los tamaños de caché y tamaño de la red. MobEye, al usar peticiones en modo *broadcast* genera más tráfico cuanto mayor es la red, siendo, con diferencia, el esquema de caché que más tráfico genera en redes de 7x7 y 9x9 nodos (Figuras 3.41b y 3.42b). Por otro lado, SimpleSearch únicamente es capaz de generar menos tráfico que la opción de no usar cachés en redes de 9x9 nodos, precisamente cuando obtiene un menor porcentaje de aciertos en caché remota, ya que no hace uso de las peticiones en modo *broadcast*.

Las Figuras 3.40c, 3.41c y 3.42c comparan el retardo medio en la obtención de los documentos en función del tamaño de las cachés para redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. Como en estudios anteriores, CLIR es el esquema de caché que menor retardo consigue a la hora de obtener los documentos y, en este caso, el único que es capaz de conseguir un retardo menor que la opción de no usar cachés para todos los tamaños de caché y de red. HybridCache muestra un retardo muy grande cuando el tamaño de la caché es muy pequeño, aunque este retardo se reduce drásticamente conforme se incrementa el tamaño de las cachés, hasta el punto de hacerse menor que la opción de no cachear en redes de 7x7 nodos y tamaño de caché de 50 documentos (Figura 3.41c) y redes de 9x9 nodos a partir de cachés de 35 documentos (Figura 3.42c). SimpleSearch siempre obtiene un retardo superior a no cachear en todos los casos. DPIP mantiene el retardo cercano a 150 ms independientemente del tamaño de la red y de las cachés debido al límite impuesto por el *DPIP_Timer*. Finalmente, MobEye ve incrementado el retardo conforme aumenta la densidad de la red debido a la saturación de los enlaces provocada por el uso masivo de peticiones en modo *broadcast*.

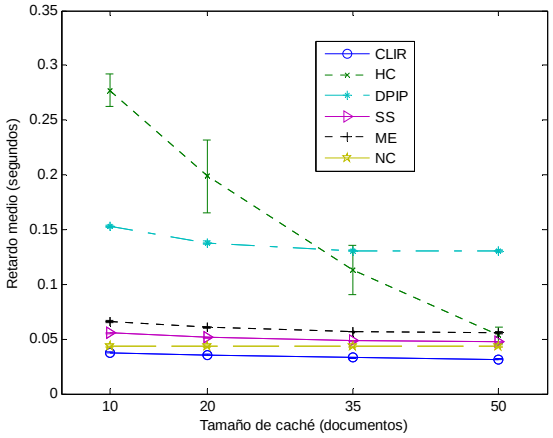
Las Figuras 3.40d, 3.41d y 3.42d representan el porcentaje medio de *timeouts* en función del tamaño de las cachés para redes de 5x5, 7x7 y 9x9 nodos respectivamente.



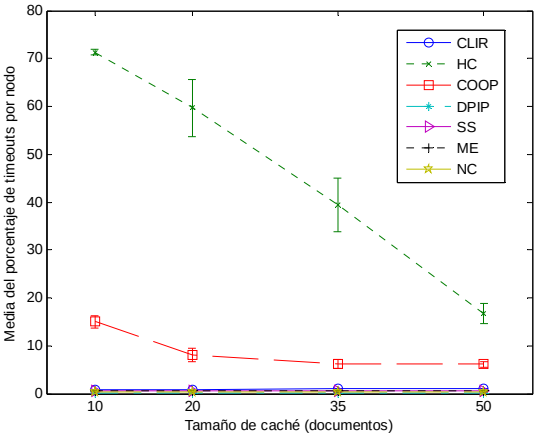
(a)



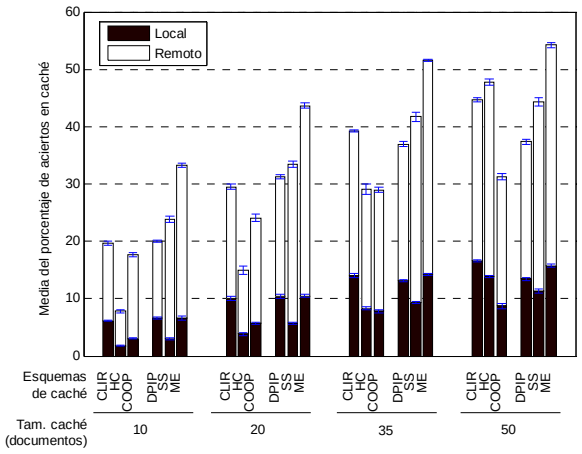
(b)



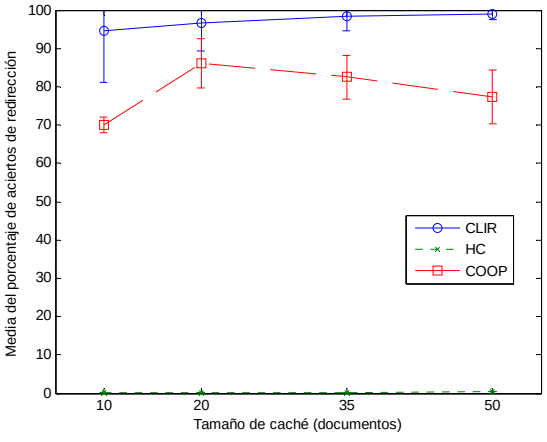
(c)



(d)



(e)



(f)

Figura 3.40. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función del tamaño de las cachés para una malla de 5x5 nodos

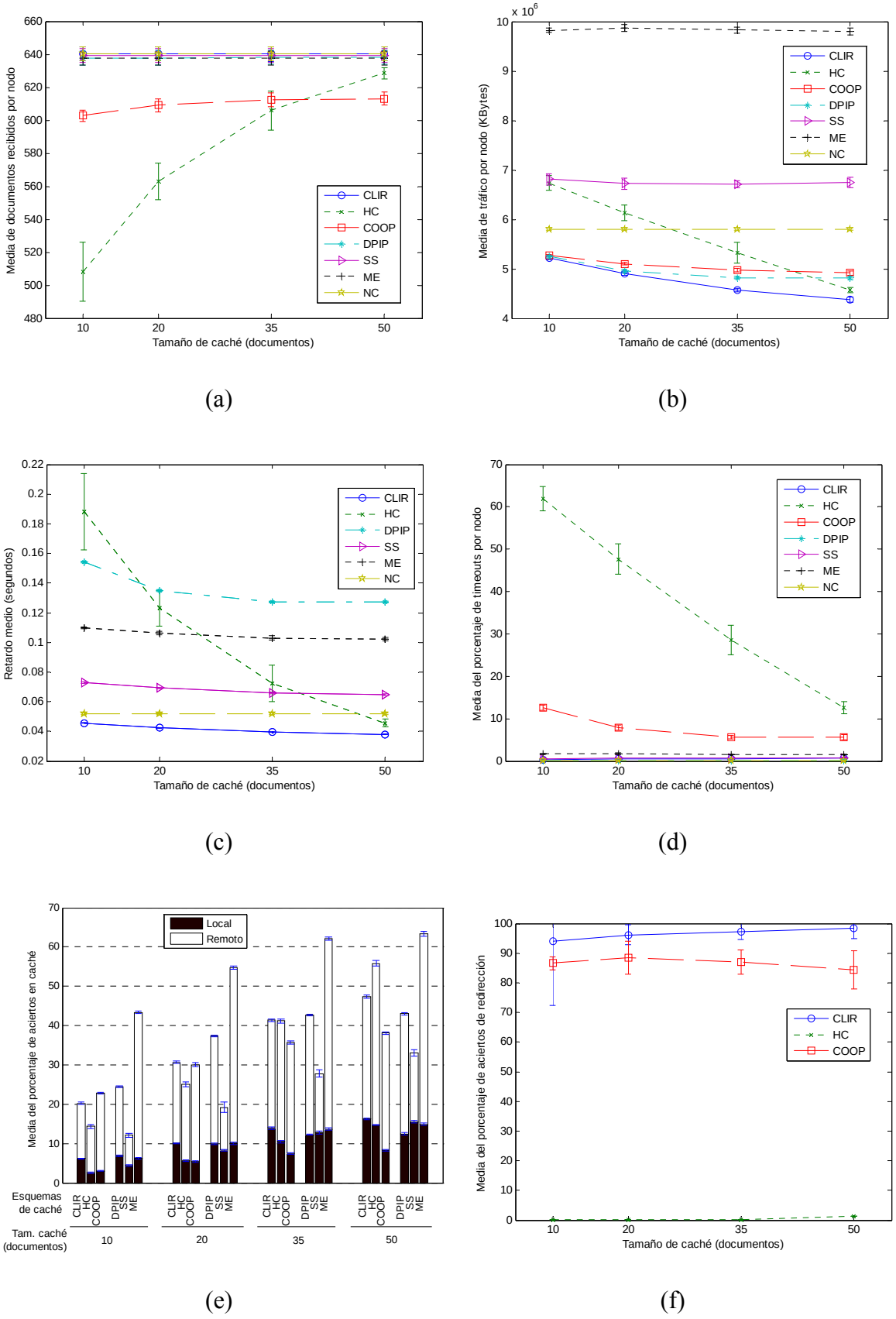


Figura 3.41. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función del tamaño de las cachés para una malla de 7x7 nodos

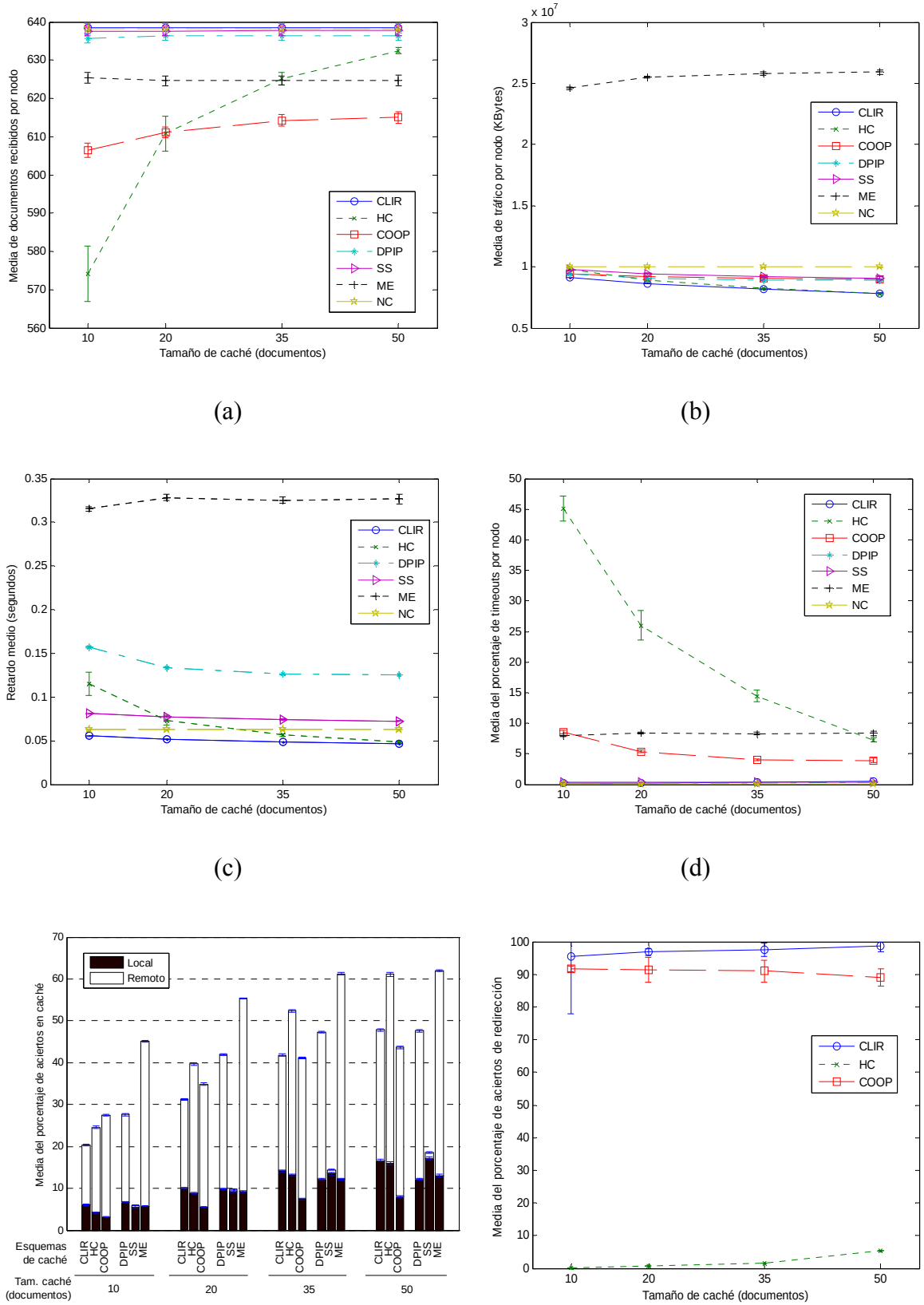


Figura 3.42. Media de documentos recibidos (a), tráfico (b), retardo (c), *timeouts* (d), aciertos en caché (e) y aciertos de redirección (f) en función del tamaño de las cachés para una malla de 9x9 nodos

Como en estudios anteriores, únicamente HybridCache, COOP y MobEye en redes grandes obtienen un porcentaje de *timeouts* superior a cero. Este porcentaje se ve reducido, sobretodo en HybridCache, conforme el tamaño de las cachés aumenta, ya que la probabilidad de aciertos tanto en caché local como remota también aumenta.

Las Figuras 3.40e, 3.41e y 3.42e muestran el porcentaje de aciertos en caché local y remota en función del tamaño de las cachés para redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. Cuanto mayor es la capacidad de almacenamiento de una caché se espera que el porcentaje de aciertos en la misma aumente tal y como se constata en las figuras. MobEye es el esquema de caché que mayor porcentaje de aciertos consigue, sobretodo en aciertos remotos debido al uso de peticiones en modo *broadcast*. CLIR consigue, junto a MobEye y SimpleSearch los mejores aciertos en caché local. El mejor rendimiento de HybridCache se consigue con tamaños de caché grandes, mientras que para cachés pequeñas su rendimiento es el peor de los esquemas de caché. Como en estudios anteriores, SimpleSearch prácticamente presenta un porcentaje de aciertos en caché remota nulo en redes de 9x9 nodos.

Las Figuras 3.40f, 3.41f y 3.42f comparan el porcentaje de aciertos de redirección en función del tamaño de las cachés, para aquellos esquemas de caché que usan redirección, en redes malladas de 5x5, 7x7 y 9x9 nodos respectivamente. El comportamiento es similar al de estudios anteriores, consiguiendo HybridCache un porcentaje de aciertos cercano al cero, aunque se ve ligeramente incrementado en redes de 9x9 nodos con tamaños de caché superiores a 35 documentos. COOP obtiene un porcentaje de aciertos que oscila entre el 80% y 90% conforme el tamaño de la red aumenta, si bien este rendimiento cae hasta el 70% en redes de 5x5 nodos con un tamaño de caché de 10 documentos. Finalmente, CLIR obtiene un rendimiento entre el 95% y el 100% para todos los tamaños de red, aunque cabe destacar la enorme variabilidad que presenta con tamaños de caché de 10 documentos dada la alta probabilidad de reemplazos.

Del estudio elaborado sobre el efecto del tamaño de las cachés se puede concluir que HybridCache es muy sensible al tamaño de las mismas y que su política de reemplazo SXO no maneja bien la sustitución de los documentos cuando la caché es muy pequeña. Como en estudios anteriores, MobEye sobrecarga la red con mensajes que causan grandes retardos y pérdidas. COOP es capaz de obtener menos documentos que el resto de esquemas de caché (exceptuando HybridCache) a costa además de un elevado retardo. CLIR, como en estudios anteriores, es el esquema de caché que menor tráfico genera en la red y menor retardo obtiene independientemente del tamaño de las cachés y del tamaño de la red.

3.6.6 Conclusiones

En este apartado se ha evaluado el rendimiento del esquema de caché CLIR en redes ad hoc estáticas con distribución mallada de los nodos usando diferentes métricas: cantidad de documentos obtenidos durante el tiempo de simulación, tráfico procesado por cada nodo, retardo, *timeouts*, aciertos en caché y aciertos de redirección. Esta evaluación se

ha realizado estudiando la influencia de la carga de tráfico, el TTL de los documentos, el patrón de peticiones y el tamaño de las cachés. Las pruebas se han efectuado en redes malladas de 5x5, 7x7 y 9x9 nodos con el fin de evaluar también la influencia de la densidad de nodos en la red. Además, se ha comparado el rendimiento de CLIR con los esquemas de caché HybridCache, COOP, DPIP, SimpleSearch y MobEye. También se ha comparado con el rendimiento de una red que no implemente ningún mecanismo de caché.

Del conjunto de simulaciones realizadas, se puede concluir que, tanto MobEye, como COOP e HybridCache no son esquemas de caché adecuados para redes estáticas. Esta conclusión se justifica en el hecho de que son los esquemas de caché que menos documentos son capaces de obtener en el tiempo de simulación y tienen un porcentaje de *timeouts* superior a cero. Este comportamiento con respecto a los *timeouts* es inaceptable en este tipo de redes en las que los servidores están siempre disponibles ya que los nodos no se mueven. Con respecto al resto de esquemas de caché (DPIP, SimpleSearch y CLIR), CLIR obtiene un número similar de documentos que DPIP y SimpleSearch. Sin embargo, CLIR es el esquema de caché que menor tráfico genera y menor retardo consigue en todas las situaciones estudiadas. Con respecto a estos parámetros, CLIR siempre consigue mejor rendimiento que no usar cachés, por lo que también es adecuado en redes estáticas.

Capítulo 4

Conclusiones

Esta tesis se ha centrado en el estudio del uso de cachés en redes inalámbricas ad hoc. En este capítulo se presentan las principales conclusiones del trabajo realizado, así como propuestas para el trabajo futuro a realizar. La sección 4.1 resume el contenido de la tesis. La sección 4.2 describe las principales conclusiones obtenidas del trabajo. Seguidamente, la sección 4.3 presenta las líneas futuras de trabajo. Finalmente, la sección 4.4 enumera las publicaciones emanadas directamente de esta tesis, así como otras publicaciones relacionadas en las que ha participado el autor.

4.1 Síntesis de la tesis

El aumento exponencial del número de usuarios y servicios que se ofrecen desde Internet ha hecho necesaria la utilización de mecanismos que optimicen el uso de las redes de comunicación. Estos mecanismos, en forma de cachés situadas en lugares estratégicos de las redes, permiten aumentar la calidad de los servicios ofrecidos a los usuarios, al ser capaces de acceder a los contenidos con un menor tiempo de espera y evitando, al mismo tiempo, sobrecargar las redes de comunicación. Por otro lado, el acceso a Internet, u otros servicios telemáticos, desde dispositivos inalámbricos también ha sufrido un incremento considerable. De este modo, las técnicas de caché aplicadas a las redes fijas también pueden adaptarse para optimizar los escasos recursos presentes en las redes inalámbricas ad hoc.

En esta tesis se realiza un estudio de las cachés para acceso Web presentes en *proxies*, centrando la atención en las políticas de control de admisión y, sobre todo, en las políticas de reemplazo. La misión de este tipo de políticas es decidir, por un lado, qué documentos deben almacenarse en la caché y cuáles deben eliminarse de la misma al almacenarse un nuevo documento, respectivamente. A continuación, y basándose en los estudios anteriores, se analizan los esquemas de caché propuestos en la literatura para redes ad hoc. De este análisis se propone una clasificación de los esquemas de caché para redes ad hoc que permite, además, conocer las ventajas e inconvenientes de cada una de las aproximaciones realizadas, lo que hace posible el desarrollo de una nueva propuesta de caché en redes inalámbricas ad hoc que aproveche dichas ventajas, evitando los inconvenientes.

4.2 Contribuciones

Esta tesis ha contribuido en los siguientes temas de investigación:

- Revisión del trabajo relacionado con las políticas de reemplazo en cachés Web. El estudio realizado, el más amplio que conocemos hasta la fecha, enumera y describe setenta y nueve políticas de reemplazo.

- Propuesta de una clasificación de las políticas de reemplazo en cachés Web. Hasta la fecha se habían realizado algunas clasificaciones de las políticas de remplazo, aunque éstas no eran completas al no tener en cuenta todos los parámetros considerados en ellas.

- Revisión del trabajo relacionado con las políticas de control de admisión en cachés Web. Este estudio, del que no se ha encontrado ninguno similar en la literatura técnica asociada, enumera y describe siete políticas de control de admisión.

- Comparación del rendimiento de políticas de reemplazo aleatorias. Se ha evaluado el rendimiento de este tipo de políticas de reemplazo con la clásica política LRU. Las simulaciones realizadas indican que, dependiendo de la métrica a optimizar, Harmonic y RRGVF consiguen mejores resultados que LRU.

- Comparación del rendimiento de políticas de reemplazo en función del tipo de los documentos. Se realiza un estudio del rendimiento de tres políticas de reemplazo clásicas (LRU, LFU y LFU-DA) con otras tres basadas en coste (GD-Size, GDSF y GD*) aplicadas únicamente a un tipo de documento concreto (*Application*, *Audio*, *Images*, *Text* y *Video*). Se concluye que no existe una política de reemplazo que sea la mejor para todos los tipos de documentos ni para todas las métricas de rendimiento.

- Propuesta de métricas de rendimiento adecuadas para una caché con política de control de admisión. Se detecta el problema de que las métricas clásicas en cachés Web (Hit Ratio y Byte Hit Ratio) no resultan adecuadas para evaluar cachés que implementan una política de control de acceso. Por lo tanto se proponen y evalúan las nuevas métricas propuestas para estudiar su idoneidad. Las simulaciones realizadas demuestran que estas nuevas métricas sí son adecuadas para este tipo de cachés.

- Revisión del trabajo relacionado con los esquemas de caché aplicados a redes inalámbricas ad hoc. El estudio realizado, también de gran amplitud, enumera y describe veinticinco esquemas de caché para redes MANET.

- Propuesta de una clasificación de los esquemas de caché para redes inalámbricas ad hoc. Basándose en las características y funcionamiento de los esquemas de caché propuestos en la literatura, se plantea una clasificación de los mismos que, hasta la fecha, no había sido realizada.

- Revisión del trabajo relacionado con la metodología y parámetros de simulación en esquemas de caché para redes ah hoc. Se realiza un exhaustivo análisis del estado de la técnica de simulación en este tipo de redes para, de este modo, realizar unas simulaciones acordes a lo propuesto en la literatura.

- Propuesta de un nuevo esquema de caché para redes inalámbricas ad hoc. Se define un nuevo algoritmo para cachés distribuidas en redes ad hoc basándose en cuatro pilares: la caché local, la intercepción de peticiones, la redirección de peticiones y la intercepción de peticiones empleando información intercapa.

- Evaluación del nuevo esquema de caché propuesto. Se evalúa el rendimiento del esquema de caché propuesto y se compara con otros cinco esquemas de caché, además de con la opción de no usar cachés en la red. Hasta la fecha, ningún nuevo esquema de caché había sido comparado con tal número de esquemas, ni estos esquemas de caché habían sido comparados entre sí. Esta comparación se realiza tanto en redes móviles como en redes estáticas inalámbricas, concluyéndose que la nueva propuesta mejora las prestaciones de las anteriores.

4.3 Trabajo futuro

El trabajo relacionado con las políticas de control de admisión y de reemplazo en cachés Web podría ampliarse en las siguientes líneas de desarrollo:

- Dada la ingente cantidad de políticas de reemplazo existentes en la literatura asociada y que, cada vez que se propone una nueva política de reemplazo, ésta se compara con el rendimiento de, a lo sumo, otras dos políticas, existe un amplio campo para comparar entre sí la mayor cantidad posible de políticas de reemplazo. Dicho estudio podría realizarse, no solo con muestras de tráfico real más actuales a las utilizadas en la presente tesis, sino también utilizando el generador sintético de muestras de tráfico desarrollado. De esta forma, sería posible evaluar el rendimiento de cada una de las políticas de reemplazo en función de cada uno de los parámetros del tráfico por separado.

- Las métricas propuestas para la evaluación de cachés con control de admisión pueden ser refinadas, sobre todo la estimación de los documentos correctamente

rechazados. Dependiendo del tipo de política de reemplazo podría estimarse el tiempo que permanecería el documento almacenado en la caché y así, si la siguiente referencia se encuentra más allá de ese tiempo, el rechazo sería correcto. Tal y como se realiza actualmente, un rechazo se considera correcto si no vuelve a solicitarse el documento en el resto de la muestra de tráfico.

El trabajo relacionado con los esquemas de caché en redes inalámbricas ad hoc puede ampliarse en las siguientes líneas de desarrollo:

- El esquema de caché propuesto utiliza el protocolo de enrutamiento AODV para realizar las intercepciones usando mecanismos intercapa. Sería adecuado realizar una evaluación del mecanismo utilizando otros protocolos de enrutamiento. Para ello habría que estudiar la idoneidad de modificar cada uno de los protocolos y, una vez realizadas dichas modificaciones, evaluar y comparar el rendimiento del esquema de caché propuesto con cada uno de los protocolos de enrutamiento.

- El esquema de caché propuesto utiliza la política de reemplazo LRU en las cachés locales de cada uno de los dispositivos de la red. Sería conveniente realizar una evaluación del comportamiento del esquema de caché utilizando otras políticas de reemplazo. Las políticas de reemplazo a evaluar deberían ser aquellas cuyo coste computacional y número de acceso y movimientos en la estructura de datos a gestionar sea mínima. Estos requisitos se deben a que la política de reemplazo debe implementarse en dispositivos cuyas capacidades, tanto de memoria como de velocidad de cómputo, son reducidas.

- El espacio reservado para la tabla de redirecciones en el esquema de caché propuesto podría optimizarse aún más si cada registro con información relativa a las redirecciones se dividiera en dos subregistros, uno relativo a la petición y otro a la respuesta. De esta forma, debido a que hay ocasiones en que ambos no existen de forma simultánea, se aprovecharía mejor el espacio asignado al mantener en memoria únicamente los datos que pueden ser útiles.

- Aunque se ha elegido un subconjunto representativo de cada uno de los tipos de esquemas de caché para redes ad hoc, sería conveniente comparar también el esquema de caché propuesto con otros de los propuestos en la literatura.

4.4 Publicaciones

A continuación se enumeran las publicaciones realizadas directamente relacionadas con la tesis:

Publicaciones en revistas:

- F.J. González-Cañete, E. Casilari, A. Triviño-Cabrera, *A Cross Layer Interception and Redirection Cooperative Caching Scheme for MANETs*, EURASIP Journal on Wireless Communications and Networking, Pendiente de publicación.
  Factor de impacto: 0.815.

Publicaciones en libros:

- F.J. González-Cañete, E. Casilari, *Impact of the Mobility Model on a Cooperative Caching Scheme for Mobile Ad Hoc Networks*, Mobile Ad-Hoc Networks: Applications, InTech Open Access Publisher, pp. 265-286, Enero 2011.

Publicaciones en congresos internacionales:

- F.J. González-Cañete, L.B. Ríos-Sepúlveda, E. Casilari, A. Triviño-Cabrera, *A MANET with caching Capabilities Visualization Tool*, Proceedings of the IADIS International Conference Applied Computing 2010, pp. 159-166, Timisoara (Rumanía), Octubre 2010.

- F.J. González-Cañete, E. Casilari, A. Triviño-Cabrera, *An Application Level Caching Scheme Implementation for MANETs Using NS2*, Proceedings of the IADIS International Conference Applied Computing 2009, pp. 343-346, Roma (Italia), Noviembre 2009.

- F.J. González-Cañete, E. Casilari, A. Triviño-Cabrera, *Proposal and evaluation of a Caching Scheme for Ad Hoc Networks*, Proceedings of the VIII International Conference on Ad Hoc Networks and Wireless, pp. 366-372, Murcia (España), Septiembre 2009.
  Core B.

- F.J. González-Cañete, E. Casilari, A. Triviño-Cabrera, *Proposal and Evaluation of an Application Level Caching Scheme for Ad Hoc Networks*, Proceedings of the 5th International Wireless Communications and Mobile Computing Conference (IWCMC 2009), pp. 952-957, Leipzig (Alemania), Junio 2009.
  Core B.

- F.J. González-Cañete, E. Casilari, A. Triviño-Cabrera, *A Windows Based Web Cache Simulator Tool*, Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (Simutools 2008), Marsella (Francia), Marzo 2008.

- F.J. González-Cañete, J. Sanz-Bustamante, A. Triviño-Cabrera, E. Casilari, *Evaluation of Randomized Replacement Policies for Web Caches*, Proceedings of the IADIS International Conference WWW/Internet 2007, pp. 227-234, Vila Real (Portugal), Octubre 2007.

- F.J. González-Cañete, E. Casilari, A. Triviño-Cabrera, *A Content-Type Based Evaluation of Web Cache Replacement Policies*, Proceedings of the IADIS Applied Computing 2007, pp. 90-96, Salamanca (España), Febrero 2007.

- F.J. González-Cañete, E. Casilari, A. Triviño-Cabrera, *Characterizing Document Types to Evaluate Web Cache Replacement Policies*, Proceedings of the IV European Conference on Universal Multiservice Networks (ECUMN'2007), pp. 3-11, Toulouse (Francia), Febrero 2007.
  Core C.

- F.J. González-Cañete, E. Casilari, A. Triviño-Cabrera, *Evaluation of a Multi-Queue Web Caching Scheme that Differentiates the Content-Type of Documents*, Proceedings of the International Conference on Internet Surveillance and Protection (ICISP 2006), Cap Esterel Côte d'Azur (Francia), Agosto 2006.

- F.J. González-Cañete, E. Casilari, A. Triviño-Cabrera, *Two New Metrics to Evaluate the Performance of a Web Cache with Admission Control*, Proceedings of the IEEE Mediterranean Electrotechnical Conference (MELECON 2006), pp. 696-699, Benalmádena (España), Mayo 2006.

Publicaciones en congresos nacionales:

- F.J. González-Cañete, R. Jiménez-Jiménez, E. Casilari, *Generador de tráfico sintético para la evaluación del rendimiento de cachés*, Actas de las X Jornadas de Ingeniería Telemática (JITEL 2011), pp. 340-347, Santander (España), Septiembre 2011.

- F.J. González-Cañete, E. Casilari, A. Triviño-Cabrera, *Propuesta y evaluación de un esquema de caché para redes ad hoc*, Actas de las VIII Jornadas de Ingeniería Telemática (JITEL 2009), pp. 471-474, Cartagena (España), Septiembre 2009.

- F.J. González-Cañete, A. Ruiz-Alcántara, E. Casilari, *Política de acceso a cachés Web basada en el tipo de los documentos*, Actas del XXIII Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2008), Madrid (España), Septiembre 2008.

- F.J. González-Cañete, J. Sanz-Bustamante, A. Triviño-Cabrera, E. Casilari, *Evaluación de políticas de reemplazo aleatorias en cachés Web*, Actas de las VI Jornadas de Ingeniería Telemática (JITEL 2007), pp. 83-88, Málaga (España), Septiembre 2007.

- F.J. González-Cañete, J. Pulido-Vázquez, E. Casilari, A. Triviño-Cabrera, *Evaluación de Políticas de Reemplazo Multinivel en Cachés Web*, Actas del XXI Simpusium Nacional de la Unión Científica Internacional de Radio (URSI 2006), pp.264-267, Oviedo (España), Septiembre 2006.

- F.J. González-Cañete, E. Casilari, A. Triviño-Cabrera, *Evaluación de políticas de Control de Acceso en Caché Web*, Actas de las XVI Jornadas Telecom I+D 2006, Madrid (España), Noviembre 2006.

- F.J. González-Cañete, E. Casilari, A. Triviño-Cabrera, *Evaluación de políticas de reemplazo para una caché con distinción de tipo de contenido*, Actas de las XV Jornadas Telecom I+D 2005, Madrid (España), Noviembre 2005.

Seguidamente se enumeran otras publicaciones no directamente relacionadas con la tesis en las que ha participado el autor:

Publicaciones en revistas:

- A. Triviño-Cabrera, J. García-de-la-Nava, E. Casilari, F.J. González-Cañete, *Application of path duration study in multihop ad hoc networks*, Telecommunication Systems, Springer Netherlands, Abril 2008.
  Factor de impacto: 0.396

- A. Triviño-Cabrera, E. Casilari, F.J. González-Cañete, *Active gateway switching in hybrid ad hoc networks*, IEEE Electronics Letters, vol. 42, 21:1252-1254, Octubre 2006.
  Factor de impacto: 1.063.

- E. Casilari, F.J. González-Cañete, F. Sandoval, *Modelling of HTTP Traffic*, IEEE Communications Letters, vol. 5, 6:272-274, Junio 2001.
  Factor de impacto: 0.689

- F.J. González, E. Casilari, G. Gómez, F. Sandoval, *Calidad de Servicio: Internet sin Atascos*, Sólo Programadores, vol. VI (2ª época), 71:16-21, Octubre 2000.

- F.J. González, E. Casilari, G. Gómez, F. Sandoval, *QoS: Internet sin Atascos (II)*, Sólo Programadores, vol. VI (2ª época), 72: 56-61, Noviembre 2000.

Publicaciones en congresos internacionales:

- A. Triviño-Cabrera, M.C. González-Linares, E. Casilari, F.J. González-Cañete, *Impact of Gateway Discovery on TCP-Connections in MANETs*, Proceedings of the IADIS International Conference Applied Computing 2010, pp. 215-219, Timisoara (Rumanía), Octubre 2010.

- A. Triviño-Cabrera, J. García-de-la-Nava, E. Casilari, F.J. González-Cañete, *Application of Path Duration Study in Multihop Ad Hoc Networks*, Proceedings of the 12th IFIP International Conference on Personal Wireless Communications (PWC 2007), pp.63-74, Praga (República Checa), Septiembre 2007.

- A. Triviño-Cabrera, G. Casado-Hernández, E. Casilari, F.J. González-Cañete, *Study of Gateway Selection Criteria in Hybrid MANETs*, Proceedings of the 6th Conference on Telecommunications (ConfTele 2007), Peniche (Portugal), Mayo 2007.

- A. Triviño-Cabrera, E. Casilari, F.J. González-Cañete, *Asymetrical Performance of Hybrid MANETs*, Proceedings of the IADIS Applied Computing 2007, pp. 231-237, Salamanca (España), Febrero 2007.

- A. Triviño-Cabrera, J. García-de-la-Nava, E. Casilari, F.J. González-Cañete, *An Analytical Model to Estimate Path Duration in MANETs*, Proceedings of the 9th ACM Sysmposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2006), Torremolinos (España), Octubre 2006.
  Core A.

- A. Triviño-Cabrera, I. Nieves-Pérez, E. Casilari, F.J. González-Cañete, *Ad Hoc Routing Based on Stability of Routes*, Proceedings of the ACM International Workshop on Mobility Management and Wireless Access (MobiWAC 2006), Torremolinos (España), Octubre 2006.
  Core B.

- A. Triviño-Cabrera, G. Casado-Hernandez, E. Casilari, F.J. González-Cañete, *Anticipated DAD for Global Connectivity in Hybrid MANETs*, Proceedings of the 3rd International Symposium on Wireless Communications Systems (ISWCS 2006), pp. 238-242,Valencia (España), Septiembre 2006.

- A. Triviño-Cabrera, S. Singh, E. Casilari, F.J. González-Cañete, *Integration of Mobile Ad Hoc Networks into the Internet without Dedicated Gateways*, Proceedings of the International Conference on Wireless and Mobile Communications (ICWMC 2006), Bucarest (Rumanía), Julio 2006.

- A. Triviño-Cabrera, E. Casilari, F.J. González-Cañete, *An Improved Scheme for the Integration of Mobile Ad Hoc Networks into the Internet without Dedicated Gateways*, Proceedings of the 11th International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networs (CAMAD 2006), pp. 16-21, Trento (Italia), Junio 2006.

- E. Casilari, J.M. Cano-Garcia, F.J. Gonzalez-Canete, F. Sandoval, *Modelling of Individual and Aggregate Web Traffic*, Proceedings of the 7th IEEE International Conference on High Speed Networks and Multimedia Communications HSNMC'04 , pp. 84-95, Toulouse (Francia), Junio 2004.

- E. Casilari, A. Reyes, F.J. González-Cañete, F. Sandoval, *Characterization of Web traffic*, Proceedings of the IEEE GLOBECOM 2001, San Antonio (EEUU), Noviembre 2001.
  Core B.

Publicaciones en congresos nacionales:

- A. Triviño-Cabrera, J. García-de-la-Nava, E. Casilari, F.J. González-Cañete, *Análisis de la duración de las rutas en redes móviles Ad Hoc*, Actas de las VI Jornadas de Ingeniería Telemática (JITEL 2007), pp.25-33, Málaga (España), Septiembre 2007.

- A. Triviño-Cabrera, E. Casilari, I. Nieves-Pérez, F.J. González-Cañete, *Routing Ad Hoc Basado en la Estabilidad de las Rutas*, Actas del XXI Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2006), Oviedo (España), Septiembre 2006.

- A. Triviño-Cabrera, G. Casado-Hernández, E. Casilari, F.J. González-Cañete, *Evaluación de Criterios de Selección de Pasarelas en Redes MANET Híbridas*, Actas de las XVI Jornadas Telecom I+D 2006, Madrid (España), Noviembre 2006.

- A. Triviño-Cabrera, E. Casilari, F.J. González-Cañete, *Conmutación activa en MANETs híbridas con soporte de múltiples gateways móviles*, Actas de las XV Jornadas Telecom I+D 2005, Madrid (España), Noviembre 2005.

- F.J. González-Cañete, E. Casilari, F. Sandoval, *Modelado de tráfico WAP en redes IP*, Actas de las IV Jornadas de Ingeniería Telemática (JITEL 2003), pp. 463-469, Gran Canaria (España), Septiembre, 2003.

- F.J. González-Cañete, O. López-Martín, E. Casilari, F. Sandoval, *Encaminador IP con calidad de servicio RSVP para Windows*, Actas del XVII Simposium Nacional de la Unión Científica Internacional de Radio (URSI 2002), pp. 291-292, Alcalá de Henares (Madrid), Septiembre, 2002.

Apéndice A

Descripción del software desarrollado

Para el desarrollo de esta tesis hubo que diseñar e implementar un conjunto de herramientas software que permitiera, por un lado, realizar simulaciones del comportamiento de varias políticas de reemplazo de caché y, por otro, generar muestras de tráfico sintético con diferentes características, así como ampliar la funcionalidad del simulador de redes NS2 para dar cabida a los esquemas de caché para redes ad hoc evaluados. Finalmente, para facilitar el desarrollo del esquema de caché CLIR, se desarrolló un visualizador de redes que permite observar el estado de la red y de las cachés en cada momento de la simulación. El presente anexo describe someramente cada una de estas herramientas desarrolladas.

A.1 Simulador de políticas de reemplazo en caché

Debido a la falta de un simulador de cachés en *proxies* de acceso público y globalmente aceptado, algunos investigadores han desarrollado sus propios simuladores con el objetivo de evaluar las políticas de reemplazo o políticas de control de admisión que proponen. Como ejemplo de algunos de estos simuladores pueden mencionarse los programas *Winsconsin Web Cache Simulator* [Cao,2011], *Multikey Web Cache Simulator* [Cardenas,2004], el simulador de Web caché y *prefetching* propuesto en [Márquez,2008], WebCASE (*Web Caching Algorithm Simulation Environment*) [Zhang,1999] y *WebTraff* [Markatchev,2002]. Todos estos simuladores están desarrollados para sistemas operativos UNIX/Linux. Sin embargo, ninguno de estos simuladores se ajustaba a las necesidades de la presente tesis ya que no implementaban todas las políticas de reemplazo evaluadas. De hecho, ninguno de ellos implementa la política de reemplazo que divide la caché según el tamaño de los documentos ni aquélla que divide la caché por el tipo de los documentos. Además, tampoco se calculan métricas relacionadas con el control de admisión.

A.1.1 Arquitectura del simulador

El simulador de caché se ha dividido en dos módulos. El primer módulo corresponde al módulo *Filtro* y el segundo módulo es el simulador en sí. La función del módulo *Filtro* es procesar los ficheros de registro del *proxy* Squid y adaptarlos a un

formato más manejable por el simulador. La Figura A.1 muestra la arquitectura del módulo de filtrado.

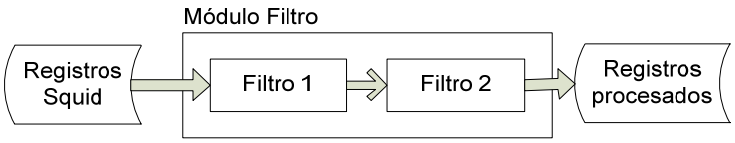


Figura A.1. Arquitectura del módulo Filtro

El módulo *Filtro* toma como entrada los ficheros de registro generados por el *proxy* caché Squid y realiza un primer proceso de filtrado (Filtro 1) eliminando aquellas peticiones generadas dinámicamente y las que van dirigidas al puerto 3128, por ser el puerto de intercambio de información entre *proxies* Squid. Además, únicamente se admiten aquellas peticiones con un código de respuesta *cacheable*, es decir, 200 (OK), 203 (*Partial*), 206 (*Partial Content*), 300 (*Multiple Choices*), 301 (*Moved*), 302 (*Redirect*) y 304 (*Not Modified*). El segundo proceso de filtrado (Filtro 2) descarta algunos de los parámetros de las peticiones incluidos en los registros de Squid y genera un fichero de salida en el que, para cada petición, se especifica el instante de petición, el retardo de la transferencia, el tamaño del documento, el identificador del mismo y su tipo de contenido (*content-type*). El identificador de los documentos y su *content-type* son codificados numéricamente para conseguir una mayor velocidad de operación del simulador.

La arquitectura del simulador se muestra en la Figura A.2.

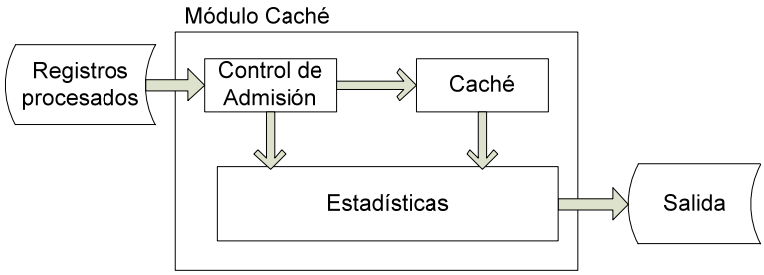


Figura A.2. Arquitectura del módulo Caché

Los ficheros de registro previamente filtrados pasan primero por el proceso *Control de Admisión*, que decide si el documento supera la política de admisión seleccionada (si la hubiera) y, por tanto, el documento pasa al proceso *Caché*. El proceso *Caché* ejecuta la política de reemplazo seleccionada decidiendo, en el caso de que fuera necesario, qué documentos se eliminan de la caché para hacer sitio a uno nuevo. Finalmente, el proceso *Estadísticas* recopila información de los procesos *Control de Admisión* y *Caché*, tales como el número de peticiones, los documentos aceptados y rechazados, tasas de acierto, etc. Una vez que la simulación termina, el proceso *Estadísticas* almacena los resultados en un fichero de texto que puede ser procesado automáticamente.

El proceso Control de Admisión implementa dos políticas de control de admisión: *Threshold size* y *Minimum number of references*. Por su parte, el proceso Caché implementa catorce políticas de reemplazo: FIFO, LRU, LFU, LFF, LFU-DA, GD-SIZE,

GDSF, GD*, Random, CLIMB, CLIMB-C, CLIMB-S, Harmonic, RRGVF, C-LRU y PART. C-LRU y PART son dos políticas de reemplazo que clasifican los documentos de acuerdo con su tamaño en varios grupos. Cada grupo es administrado mediante una cola LRU. El simulador permite definir cualquier combinación de grupos de tamaños, así como aplicar cualquiera de las políticas de reemplazo anteriores (no sólo LRU). Además, el simulador implementa una política de reemplazo que clasifica los documentos en función de su *content-type* y permite aplicar cualquier política de reemplazo a cada uno de ellos [Khayari,2005].

El proceso *Estadísticas* inspecciona continuamente la cantidad de almacenamiento en caché ocupado, el número de documentos almacenados, el número de documentos eliminados, modificados y descartados, así como el número de peticiones y tráfico procesado y servido por la caché (acierto en caché) y el número total de documentos rechazados por el proceso Control de Admisión. Basándose en los datos recopilados, este proceso calcula las métricas de rendimiento HR (*Hit Ratio*) y BHR (*Byte Hit Ratio*), así como las métricas NUHR (*Not Unique Hit Ratio*) y NUBHR (*Not Unique Byte Hit Ratio*). Para medir el rendimiento de la política de control de admisión, el simulador calcula el ACHR (*Access Control Hit Ratio*) y el ACBHR (*Access Control Byte Hit Ratio*). Una vez terminada la simulación, todos los datos y estadísticas comentadas anteriormente se almacenan en un fichero de texto para su posterior procesamiento, aunque esta información también puede ser almacenada para cada paso de simulación con el fin de estudiar la evolución de la caché en el tiempo.

A.1.2 El interfaz de usuario

El interfaz de usuario del simulador se encuentra organizado en cinco pestañas:

- *General Configuration* (Configuración General): Permite configurar los parámetros no específicos de ninguna política de reemplazo o control de admisión, tales como seleccionar los ficheros para filtrar o procesar, así como los ficheros por lotes que contienen los parámetros de simulación o el porcentaje de peticiones usado para "calentar" las cachés.
- *Standard Configuration* (Configuración Estándar): Diseñada para configurar los parámetros generales de simulación cuando se selecciona una política de reemplazo que emplea una sola cola. Se puede especificar el tamaño de la caché, la políticas de reemplazo y de control de admisión así como sus correspondientes parámetros.
- *Content-type Configuration* (Configuración por tipo de contenido): Se usa para configurar las políticas de reemplazo que tienen en cuenta el tipo de contenido de los documentos. Se puede asignar una política de reemplazo y control de admisión para cada tipo de documento.
- *Size-based Configuration* (Configuración basada en tamaño): Gestiona los rangos de tamaño asignados para cada cola, así como el tamaño de las mismas y la política de reemplazo que se les aplica.

- *Simulation Data* (Datos de Simulación): Muestra, en tiempo de simulación, toda la información acerca del proceso de simulación así como las estadísticas.

La Figura A.3 muestra una captura de pantalla de la aplicación.

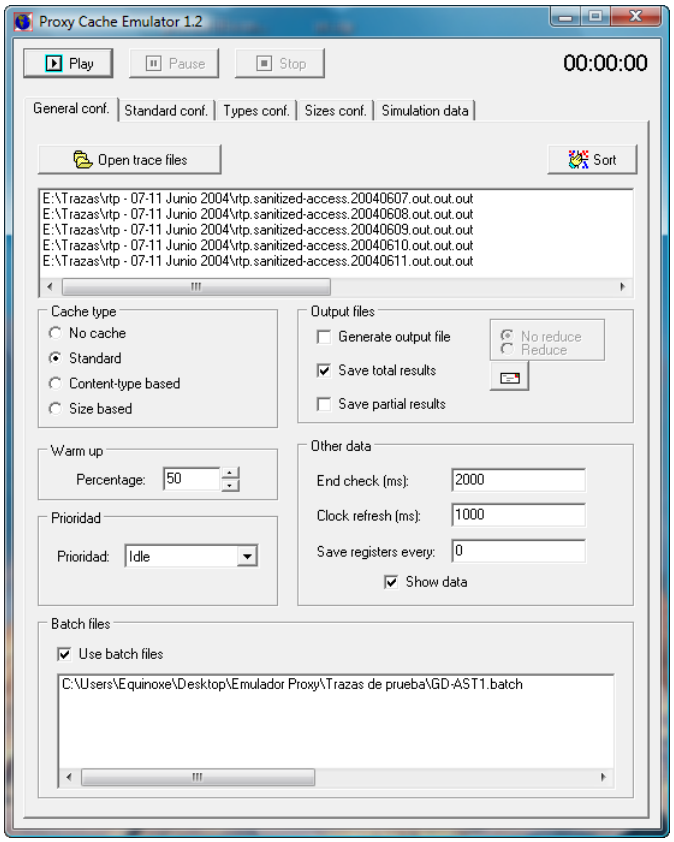


Figura A.3. Interfaz de usuario del simulador

A.1.3 Detalles de implementación

El simulador ha sido implementado usando el entorno de desarrollo integrado Borland C++ Builder 2006 y, por lo tanto, se ha utilizado la librería VCL (*Visual Common Library*) para desarrollar el interfaz y las clases para gestionar las colas. El diagrama de clases simplificado se muestra en la Figura A.4.

La clase *Cache* contiene un objeto que implementa uno de los tipos de políticas de reemplazo, es decir, *SimpleCache* (para las políticas de reemplazo que utilizan una única cola), *SizeCache* (para políticas de reemplazo multicola con diferenciación de tamaños) o *Conten-TypeCache* (para políticas de reemplazo multicola con diferenciación del tipo de los documentos). La clase *SimpleCache* utiliza un objeto *ReplacmentPolicy* para implementar la política de reemplazo seleccionada. Por el contrario, las clases *SizeCache* y *Content-TypeCache* utilizan más de un objeto *ReplacementPolicy* ya que cada cola puede operar con diferente política de reemplazo.

La clase *ReplacementPolicy* es una clase abstracta que implementa la mayoría de los métodos necesarios para gestionar las estructuras de datos de las políticas de

reemplazo. Ésta es una superclase para las clases que implementan las políticas de reemplazo y existe una clase por cada política de reemplazo implementada, aunque la clase *GDS* es una clase abstracta que implementa una generalización de la familia de políticas GDS (GD-Size GDSF y GD*).

La clase *Cache* contiene un objeto *AdmissionControl* que gestiona las políticas de admisión asignadas a la simulación en curso, siendo una clase abstracta y superclase para las clases que implementan cada una de las políticas de control de admisión (actualmente *Sizes* y *Occurrences*).

Como puede observarse de la descripción anterior, la arquitectura modular de la aplicación permite añadir nuevas funcionalidades, como nuevas políticas de reemplazo o de control de admisión, derivando por herencia de las clases base y sobrescribiendo los métodos necesarios.

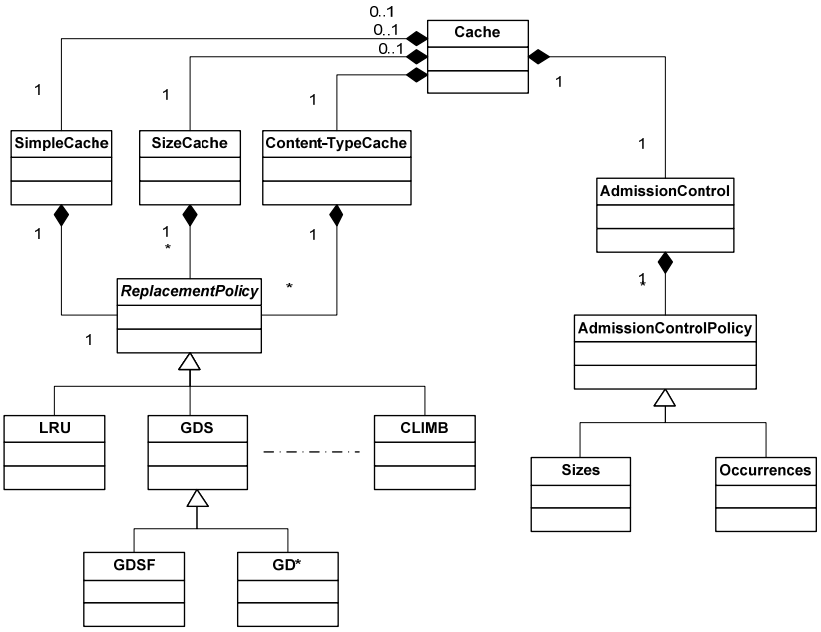


Figura A.4. Diagrama de clases simplificado del simulador

A.2 Generador de tráfico para servicios de caché

Actualmente, existen diferentes generadores de muestras de tráfico como ProWGen [Busari,2002], TrGen [Ridruejo,2005] y GenSyn [Heegaard,2000]. Estos generadores son capaces de crear muestras de tráfico sintético a partir de la introducción de una serie de parámetros o bien mediante la introducción de muestras de tráfico real, que son analizadas para posteriormente simular su comportamiento. Sin embargo, sólo ProWGen es capaz de generar muestras de tráfico para evaluar el rendimiento de cachés, ya que es el único que genera muestras de tráfico a nivel de peticiones de documentos Web. El resto, en cambio, genera muestras de tráfico a nivel de paquetes. Desafortunadamente, ProWGen no es capaz

de distinguir el tipo de los documentos que generan sus muestras, por lo que se consideró la creación de un nuevo generador que sí lo hiciera para poder ser usado en el simulador de cachés descrito en el apartado A.1.

A.2.1 Implementación

Se ha creado una herramienta sobre el entorno MATLAB que tiene como objetivo la generación de muestras de tráfico para su posterior procesado en simuladores de caché. El resultado de la aplicación es la creación de ficheros donde se especifican estas muestras de tráfico que serán las que utilicen posteriormente dichos simuladores.

El generador de tráfico incorpora cinco características de tráfico seleccionadas, las cuales han sido identificadas como importantes en estudios previos de muestras de tráfico en servidores [Busari,2002] [Abdulla,1997] [Mahanti,1999] [Badford,1998]: los documentos referenciados una sola vez, la popularidad de los documentos, la distribución del tamaño de los documentos, la correlación entre el tamaño de los documentos y su popularidad y, finalmente, la localidad temporal.

- Documentos referenciados una sola vez (*one-timers*): El enfoque utilizado para modelar las referencias a documentos *one-timers* es determinar cuántos de los documentos distintos en la muestra de tráfico a crear deberían ser *one-timers*. Utilizando el porcentaje de *one-timers* como un parámetro, se permite al usuario especificar el valor deseado. Una vez especificado, se puede hallar con facilidad el número de *one-timers*, con lo que las referencias a estos documentos se fijan a uno.
- Popularidad de los documentos: Se ha usado la ley Zipf para modelar la popularidad de los documentos.
- Distribución del tamaño de los documentos: Ésta se ha efectuado del siguiente modo:
    1. Se modela la cola de la distribución usando una distribución de Pareto.
    2. Se modela el cuerpo de la distribución usando una distribución logarítmica normal.
    3. Se unen ambas distribuciones.
- Correlación entre el tamaño de los documentos y su popularidad: Las muestras de tráfico generado pueden tener correlación positiva, negativa o cero. Una correlación positiva implica que los documentos de mayor tamaño tienen mayor popularidad y correlación negativa implica que los documentos de menor tamaño tengan más probabilidad de ser requeridos (mayor popularidad). Mientras que correlación cero no otorga ninguna correlación entre la popularidad de los documentos y su tamaño. El hecho de permitir la existencia o no de correlación radica en la posibilidad de explorar distintos algoritmos de caché según cada una de estas opciones. Modelar e incorporar estas características de correlación dentro de la generación de muestras de tráfico se realiza en tres etapas:

1. Generar un conjunto de popularidades de documentos usando la ley Zipf.
2. Generar un conjunto de tamaños de documentos tal y como se ha comentado anteriormente.
3. Usar una técnica de *mapeo* para introducir correlación positiva, negativa o cero entre la popularidad de los documentos y su tamaño.

- Localidad temporal: La aproximación utilizada para el modelado de la localidad temporal está basada en el modelo de pila finita LRU (*Least Recently Used*). Una pila LRU es una lista de todos los documentos ordenados según el momento de referencia [Almeida,1996], esto es, el último que haya sido referenciado estará en primer lugar de la pila y el que fue referenciado hace más tiempo estará en la última posición. La pila es actualizada dinámicamente cada vez que se procesa una referencia. En muchos casos, esta actualización implica el tener que añadir un nuevo elemento en la cima de la pila empujando el resto hacia abajo, en otros casos, implica extraer un elemento existente en el interior de la pila y trasladarla a la cima de la misma, desplazando al resto de los elementos hacia abajo. Una pila LRU de tamaño finito es una pila LRU que sólo puede almacenar un número *m* de documentos. El aspecto más importante en una pila LRU es que cada posición en la pila tiene asociada una probabilidad de referencia. Las probabilidades son asociadas a la posición de la pila y no a los documentos.
- Tipos de documentos: Los documentos existentes en la Web se clasifican en función de su tipo en: aplicaciones, audio, imágenes, mensajes, texto y vídeo. La aplicación desarrollada permite seleccionar el porcentaje de cada tipo de documentos que se desea generar. Además, permite la generación de muestras de tráfico para cada uno de los tipos de documentos especificando todos los parámetros anteriormente mencionados y posteriormente mezclar las muestras generadas para generar una muestra conjunta.

A.2.2 Interfaz de usuario

A la herramienta de generación de tráfico desarrollada se le ha provisto de un interfaz de usuario para facilitar la configuración de los parámetros de generación del tráfico. La ventana principal de la aplicación puede verse en la Figura A.5.

La aplicación se ha dividido en cuatro secciones a las que se accede mediante los botones situados en la parte superior de la ventana principal:

• General: En esta sección se introducen todos los parámetros necesarios para la generación de las muestras de tráfico sintético comentadas en apartados previos. Además se especifica el nombre del archivo en el que se generará la muestra.

• Por tipos de archivos: Esta sección es muy similar a la anterior, la principal diferencia es que antes de introducir los parámetros para la generación de las muestras de

tráfico, se deberá indicar de qué tipo de documentos serán las muestras que se van a generar. Una vez elegido el tipo de documento, se introducirán los parámetros necesarios y el nombre del archivo a generar.

• Analizar: Esta sección permite verificar el correcto funcionamiento de la aplicación, ya que muestra información relativa a las muestras que contiene el archivo seleccionado, como la Ley de Zipf, la función de probabilidad acumulada del cuerpo de la distribución de tamaño, la función de probabilidad acumulada de la cola de la distribución de tamaño, la correlación entre el tamaño de los documentos y su frecuencia de aparición, la función de probabilidad de los tamaños de los documentos y la localidad temporal de las repeticiones de las peticiones de los documentos comparadas con sus respectivas gráficas ideales.

• Mezclar: Esta sección permite mezclar muestras generadas en distintos archivos. A priori, se realiza una mezcla aleatoria uniforme, dejando abierta la posibilidad de incluir cualquier otro tipo de distribución para dicha mezcla.

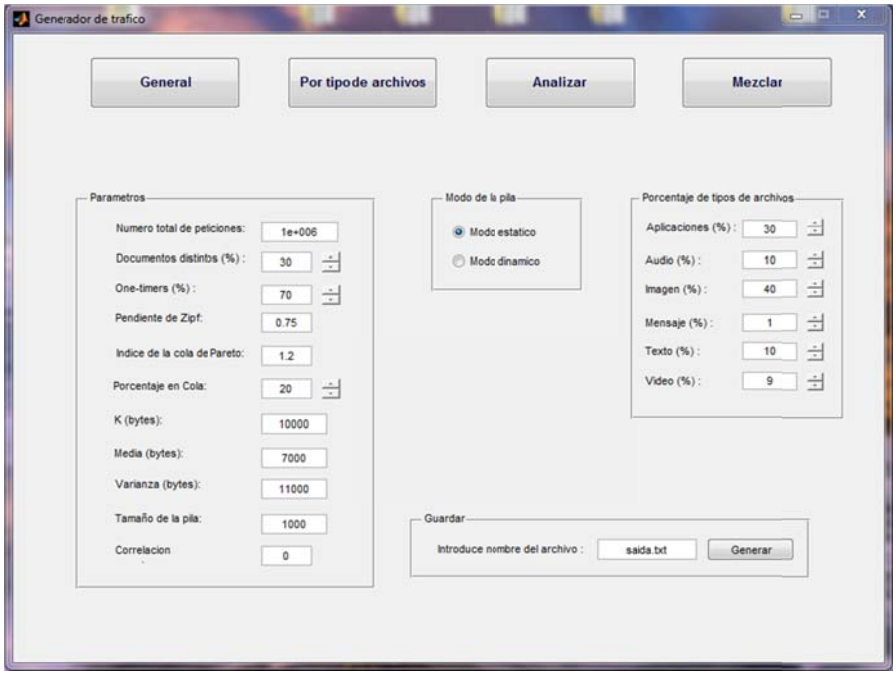


Figura A.5. Interfaz de usuario del generador de tráfico

Una vez que se han introducido todos los parámetros y se ejecuta la generación de muestras, la aplicación genera un archivo con el nombre indicado por el usuario. Este archivo consta de una lista en la que cada fila hace referencia a una solicitud de documento. Cada fila del fichero tiene tres campos: el primer campo es el identificador de archivo, especificado mediante un número para facilitar su procesamiento; el segundo campo indica el tamaño del documento en bytes; y el tercer campo indica el tipo de documento generado: aplicación, audio, imágenes, mensajes, texto y vídeo.

A.3 Ampliación del simulador de redes NS2

Para poder evaluar el rendimiento del esquema de caché CLIR y compararlo con los esquemas de caché DPIP, SimpleSearch, MobEye, HybridCache y COOP fue necesario implementar todos ellos en el simulador de redes NS2, ya que no se disponía de la implementación de ninguno de ellos ni en NS2 ni en ninguna otra herramienta de simulación de acceso libre.

A.3.1 Arquitectura

La Figura A.6 ilustra el diagrama de clases implementado para ampliar la funcionalidad de NS2.

Las clases *MobEye*, *DPIP*, *COOP*, *CLIR*, *HybridCache* y *SimpleSearch* heredan de la clase de NS2 *Agent*. Estas clases implementan las funcionalidades de cada uno de los esquemas de caché tales como el envío, recepción y reenvío de mensajes o los diferentes temporizadores. Todas ellas usan una instancia de la clase *ReplacementPolicy*, que es una clase abstracta que implementa las características comunes de una política de reemplazo. Esta clase, así como todas sus clases hijas, son una adaptación de las clases que se implementaron para realizar el simulador de políticas de reemplazo en caché del apartado 2.1. Para ello, se sustituyeron las clases de la librería VCL de Borland C++ Builder, incompatibles en sistemas operativos Linux, por otras que hubo que implementar para obtener la misma funcionalidad. De la clase *ReplacementPolicy* heredan todas las clases que implementan las diferentes políticas de reemplazo, tales como *CV*, *SXO*, *LRU*, *TDS_D* y otras políticas que, aunque se han implementado, no han sido usadas para ninguno de los esquemas de caché, tales como *LFU-DA*, *CLIMB* o la familia de políticas de reemplazo *GDS*. Por último, se ha de comentar que la política de reemplazo *COOP* emplea dos listas gestionadas mediante LRU. En principio, cada esquema de caché usa una política de reemplazo específica (por ejemplo HybridCache utiliza SXO), sin embargo, la implementación realizada permite asignar cualquier política de reemplazo a cualquier esquema de caché sin ningún tipo de limitación.

La clase *IV* implementa la tabla IV del esquema de caché DPIP, usando múltiples instancias de la clase *IVEntry*, que representan cada uno de los registros de la tabla IV. De forma similar, la clase *RRT* implementa la funcionalidad de la tabla RRT de COOP, siendo *RRTEntry* la clase que implementa cada una de las entradas de dicha tabla. Finalmente, las clases *RedirectionCache* e *HybridCacheRedir* representan las tablas de redirección de CLIR e HybridCache respectivamente, siendo *RedirectionEntry* e *HybridCacheRedirEntry* las clases que representan cada una de las entradas de dichas tablas.

Finalmente, cabe mencionar que la clase *AODV* se ha ampliado haciendo que herede, además de la clase *Agent*, de la clase *TAP* para que se permita habilitar la escucha en modo promiscuo de los nodos de la red, aunque dicha funcionalidad no ha sido empleada en ninguno de los esquemas de caché evaluados.

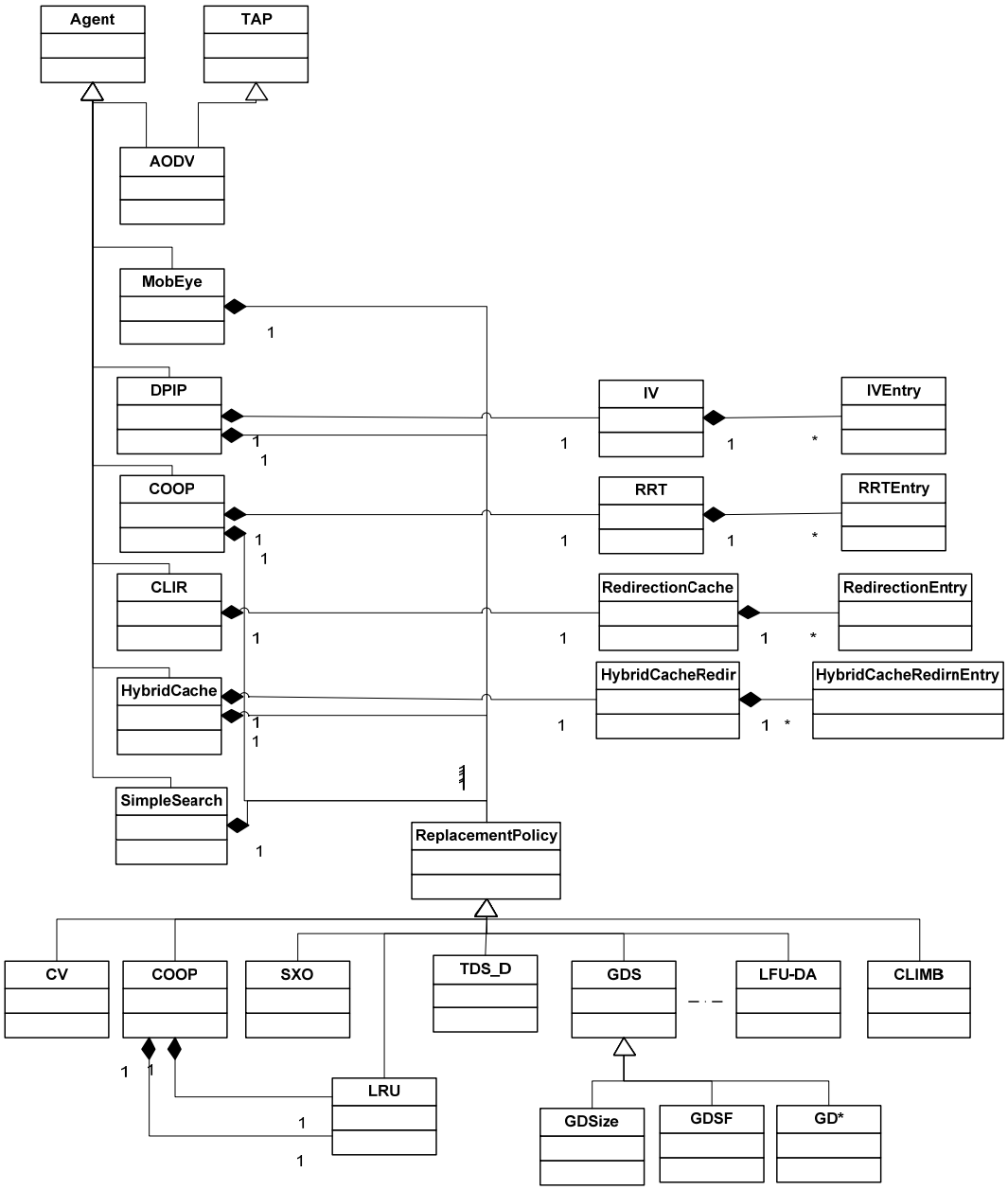


Figura A.6. Diagrama de clases de la ampliación de NS2

A.3.2 Parámetros de configuración

Para poder configurar la gran variedad de parámetros necesarios para realizar las simulaciones de los diferentes esquemas de caché se implementaron unos interfaces en lenguaje TCL (*Tool Command Language*) para configurar NS2. Estos parámetros de configuración se pueden dividir en parámetros generales y específicos de los esquemas de caché.

A continuación se enumeran los parámetros generales:

- *pathFileSizes $pathSizes* – Especifica el directorio $pathSizes donde se encuentra el fichero que contiene el tamaño de cada uno de los documentos del sistema.
- *pathFileTTL $pathTTL* – Define el directorio $pathTTL donde se encuentran los fichero que contienen el TTL inicial de cada uno de los documentos del sistema para cada media del TTL estudiada.
- *pathWaitTimeDirectory $pathWaitTimeDirectory* – Este parámetro permite especificar el directorio *$pathWaitTimeDirectory* donde se encuentran los ficheros con los tiempos de espera entre peticiones para cada uno de los nodos de la red.
- *pathRequestDirectory $pathRequestDirectory* – Similar al parámetro anterior pero especificando el directorio *$pathRequestDirectory* donde se encuentran los ficheros con la lista de peticiones que realiza cada nodo de la red.
- *nodeType CLIENT|SERVER $i* – Para cada nodo con identificador *$i* se especifica si dicho nodo es cliente o servidor.
- *servers $nservers (serverId)+* – Este parámetro informa al sistema del número de servidores *($nservers*) y el identificador de cada uno de ellos, debiendo existir al menos un servidor en la red.
- *set-cache LRU|TDS_D|DPIP|SXO|COOP $tam* – Especifica la política de reemplazo implementada y su tamaño en bytes. Si el tamaño especificado es cero, se entiende que no se implementa ninguna política de reemplazo.
- *simulationTime $time* – Establece el tiempo de duración de la simulación en segundos.
- *recvTimer $time* – Define el tiempo de duración, en segundos, del temporizador de recepción de las respuestas a las peticiones.
- *porcWarmUp $porc* – Especifica el porcentaje del tiempo de simulación que se considera de calentamiento de las cachés.

Los parámetros específicos para los esquemas de caché son:

- *remoteCacheInterception ON|OFF* – Habilita o deshabilita la intercepción de peticiones por los nodos intermedios en su ruta hacia el servidor para aquellos esquemas de caché que lo implementen.
- *AODVInterception ON|OFF* – Habilita o deshabilita la intercepción de peticiones usando la técnica *cross-layer* en AODV para el esquema de caché CLIR.
- *cacheDestMode* NO_REDIRECT|GET_ONLY|MIN_NHOPS $tam – Este parámetro permite especificar el comportamiento de la caché de redirecciones. NO_REDIRECT deshabilita la redirección. GET_ONLY define el comportamiento de la caché de redirecciones de HybridCache al redireccionar las peticiones al nodo que realizó la petición del documento a redireccionar. MIN_NHOPS define el comportamiento de la caché de redirecciones de CLIR al redireccionar las peticiones al nodo más cercano entre el nodo que ha

realizado la petición del documento y el que lo sirvió. El parámetro *$tam* define el tamaño de la caché de redirecciones en número de entradas o registros.

- *cacheDestRedirectionOnlyWithRoute ON|OFF* – Habilita o deshabilita la redirección de peticiones únicamente cuando se tiene una ruta creada hacia el nodo al cual se redirige. Se emplea en CLIR con un valor por defecto OFF.
- *useMidCache ON|OFF* – En el esquema de caché CLIR, habilita o deshabilita el almacenamiento de documentos en la caché de un nodo intermedio entre el nodo que sirve el documento y su destino.
- *maxRedirections $redir* – Permite especificar el número máximo de redirecciones que puede sufrir una petición. El valor por defecto para CLIR es de una redirección.
- *minDistServerToRedirect $hops* – Define la distancia mínima a la que se debe encontrar el servidor de documentos para que se realice la redirección de una petición en CLIR.
- *minHopsProfit2Redirect $hops* – Para los esquemas de caché HybridCache y CLIR, especifica el mínimo número de saltos de beneficio que deben obtenerse para realizar una redirección. El beneficio se define como la diferencia entre la distancia al servidor de datos y el nodo a redireccionar.
- *promiscousMode ON|OFF* – Habilita o deshabilita la escucha de los nodos en modo promiscuo de los mensajes de la red para actualizar la caché de redirecciones. Esta característica no ha sido utilizada.
- *minHopsToCache $hops* – Especifica la distancia mínima a la que debe encontrarse el nodo que ha servido un documento para que éste sea almacenado en la caché. Se emplea en SimpleSearch.

A.4 Visualizador de redes ad hoc con caché

Con el objetivo de disponer de una herramienta que permitiera visualizar, no solo una red ad hoc, el tráfico y los movimientos de los nodos, sino también el estado de las cachés implementadas en CLIR, se desarrolló una herramienta visualizadora de redes ad hoc con caché.

A.4.1 Otras herramientas disponibles

Previamente al desarrollo de la herramienta se comprobó si existía alguna herramienta ya disponible que realizara dichas funciones. Así, se ha de mencionar las siguientes:

- NAM (*Network AniMator*) [NAM,2011] es la herramienta de visualización por defecto de NS2. NAM fue diseñado para crear una interfaz de usuario para configurar las simulaciones de topologías de redes cableadas, mostrando los

enlaces creados y el flujo de paquetes. Más tarde fue ampliado para la visualización de redes móviles.

- iNSpect (*interactive NS2 protocol and environment confirmation tool*) [Kurkowski, 2005] es una herramienta de visualización desarrollada en C++ que permite el análisis de redes inalámbricas simuladas en NS2. Su característica principal es que es capaz de gestionar diferentes ficheros de registro, aceptando los ficheros de traza y movilidad de NS2. Sin embargo, requiere de un tipo especial de fichero de muestra de tráfico denominado *vizTrace*. Además, esta herramienta calcula parámetros estadísticos como el número de paquetes procesados, tamaño, las direcciones IP origen y destino de cada salto.
- Huginn [Scheuermann, 2005] es otra herramienta de visualización de redes inalámbricas para NS2 implementada en C++. Su característica principal es el uso de gráficos 3D para representar la red, además de un sofisticado método de representar la información. Añadidamente, el usuario puede especificar el tipo de datos en los que la aplicación debe concentrarse.
- EXAMS (*EXtensible Animator for Mobile Simulations*) [Livathinos, 2009] es un completo simulador escrito en Java que añade funcionalidades no disponibles en las herramientas anteriormente mencionadas, como la posibilidad de representar el estado interno de los nodos, ver el área de cobertura de los mismos e incluso calcular datos estadísticos del rendimiento de la red.
- MobiSim [Mousavi,2007] es una aplicación gratuita desarrollada con el ánimo de ayudar en el estudio de la movilidad en redes MANET. El usuario puede crear escenarios de movilidad y simularlos de forma gráfica. Soporta una gran variedad de patrones de movilidad y es capaz, además, de mezclarlos. Sin embargo, se limita únicamente a la generación y visualización del movimiento de los nodos.

Como se puede apreciar, ninguna de las herramientas analizadas admiten esquemas de caché o la visualización de las cachés locales o de redirección. Como consecuencia, se decidió implementar una herramienta que diera respuesta a esta necesidad.

A.4.2 Funcionalidad de la herramienta de visualización

La herramienta de visualización ha sido desarrollada en el lenguaje Java y la API Java 3D. Dado que el lenguaje Java es multiplataforma, la aplicación puede ejecutarse en cualquier plataforma que lo soporte. Actualmente estas plataformas son Microsoft Windows, Linux, Solaris y Mac OS X. La ventana principal de la aplicación puede observarse en la Figura A.7.

La herramienta acepta dos ficheros de entrada:

- Fichero de movilidad – Este fichero incluye información sobre la posición de los nodos, movimientos y velocidad, así como las dimensiones del área de

simulación. El fichero de movilidad debe seguir el formato de los ficheros de movilidad de NS2, los cuales pueden crearse con cualquier generador compatible, como por ejemplo BonnMotion [Aschenbruck,2010].

- Fichero de muestra de tráfico – Este fichero especifica la información del tráfico en la red, así como los cambios de estado de las cachés locales y de las cachés de redirección. Las especificaciones de este fichero pueden consultarse en [González-Cañete,2011] y es generado por la implementación del esquema de caché CLIR en NS2.

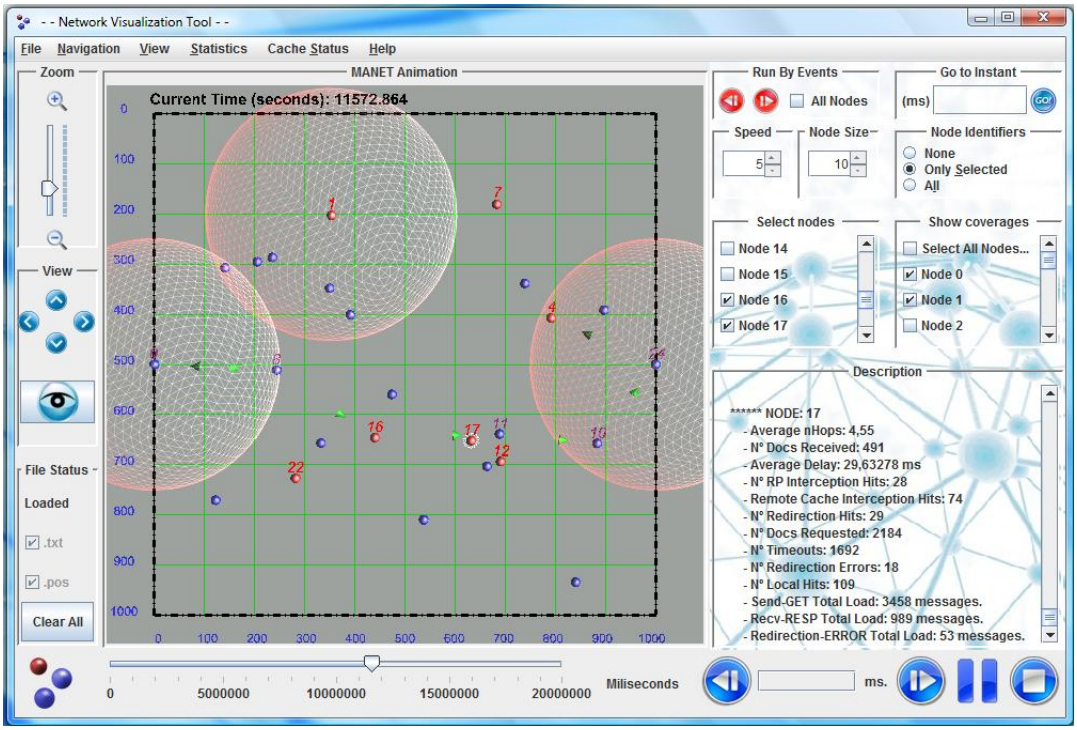


Figura A.7. Ventana principal de la aplicación de visualización

Cuando se carga el fichero de movilidad, el panel denominado "*MANET Animation*" representa el escenario de simulación usando una rejilla de 100 metros. Los nodos móviles se representan como esferas que pueden ser seleccionadas para visualizar su número de identificación y área de cobertura. La herramienta puede ser usada para validar los escenarios generados ya que es posible visualizar y animar la posición de los nodos en la red de acuerdo con el fichero de movilidad sin necesidad de especificar ningún fichero de traza. Por otro lado, cuando se carga el fichero de traza junto al fichero de movilidad, la aplicación está preparada para visualizar el tráfico y movimiento de los nodos en la red.

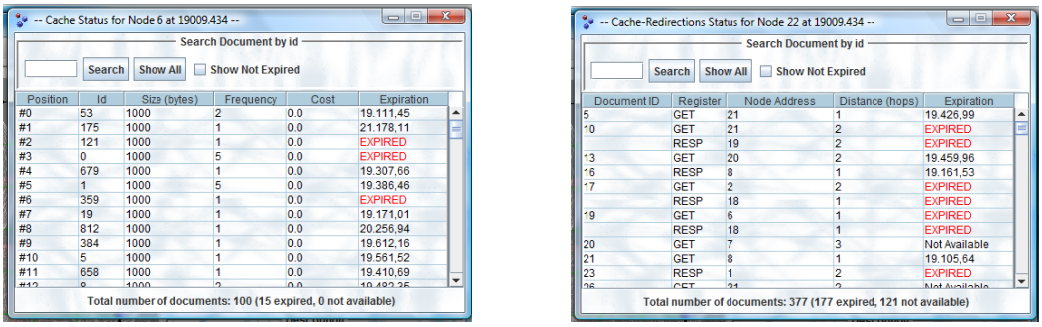
Las funcionalidades de la aplicación serán enumeradas conforme se comenten las funcionalidades de los paneles de los que se compone:

- Paneles "*Zoom*" y "*View*" – Estos paneles permiten cambiar el punto de vista de la cámara que representa el escenario. La cámara puede moverse en las cuatro direcciones básicas, además de acercarse y alejarse del escenario.

- Panel "*Playback*" – Contiene las funcionalidades de reproducción de la simulación. Con el control deslizante se puede situar la simulación en cualquier instante de tiempo determinado. Por otro lado, la visualización puede ser activada, pausada y detenida. Finalmente, la visualización puede ejecutarse a intervalos fijos de tiempo definidos por el usuario.
- Panel "*Run By Step*" – Este panel permite ejecutar la visualización y detenerla automáticamente cuando ocurre un evento en cualquiera de los nodos seleccionados. Estos eventos son: el envío, reenvío o recepción de un mensaje, un nodo alcanza las coordenadas de destino según el patrón de movilidad, o el hecho de que se modifique la caché local o de redirecciones.
- Panel "*Go To Instant*" – Este panel permite especificar un instante de tiempo para que la simulación se sitúe en dicho instante.
- Panel "*Speed*" – Por defecto la velocidad de simulación es en tiempo real, es decir, un segundo de simulación dura un segundo real. Este panel permite acelerar o decelerar la velocidad de la visualización.
- Panel "*Node Size*" – El tamaño de los nodos puede ser alterado usando este panel.
- Panel "*Node Identifiers*" – Se puede visualizar el número de identificación de todos, ninguno o solo aquellos nodos que estén seleccionados de la red MANET.
- Panel "*Select Nodes*" – Los nodos pueden ser seleccionados y deseleccionados usando la lista de identificadores de nodos de este panel.
- Panel "*Show Coverage*" – De forma similar, se puede habilitar o deshabilitar la visualización del área de cobertura de los nodos seleccionados usando este panel.
- Panel "*Description*" – Este panel lista los eventos y principales estadísticas de los nodos seleccionados de la red.
- Panel "*MANET Animation*" – Este es el panel principal de la aplicación y representa los nodos móviles y sus movimientos en el escenario que va a ser simulado. Los nodos móviles pueden seleccionarse directamente pulsando con el botón izquierdo del ratón sobre la esfera que los representa. Además, si se pulsa el botón derecho del ratón sobre un nodo, aparece un menú con opciones para mostrar estadísticas, las coordenadas, el identificador, el área de cobertura y la caché local o de redirecciones del nodo.

En cualquier instante de la visualización se puede mostrar el estado de las cachés locales y de redirección. La Figura A.8a ilustra un ejemplo de la ventana que muestra el contenido de la caché local. En esta ventana se muestra el identificador de cada documento, su tamaño, número de accesos, coste (para aquellas políticas de reemplazo basadas en coste) y el tiempo de expiración ordenado por la posición que ocupa en la caché local. El documento situado en la posición más alta de la caché local será el próximo documento en ser eliminado. Esta ventana también permite filtrar los documentos según su identificador de documento o mostrar sólo aquellos documentos que no han expirado. La figura A.8b muestra la ventana de la caché de redirecciones, donde puede verse el número

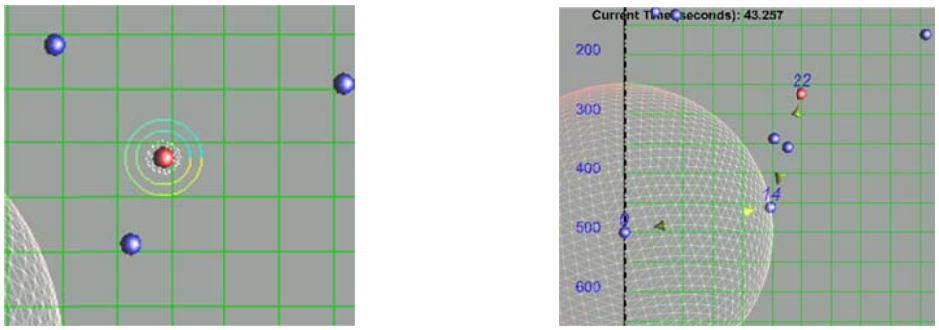
de identificación, el tipo de registro de información (GET o RESP), la identificación del nodo donde está localizado el documento, la distancia en saltos al nodo que almacena el documento y la información sobre la caducidad de la información. De forma similar a la ventana de la caché local, la ventana de caché de redirección también permite el filtrado de la información por el identificador del documento y por la caducidad de los registros.



(a) (b)

Figura A.8. Ventanas de visualización de la caché local (a) y caché de redirecciones (b)

Con el objetivo de obtener una mejor comprensión de los eventos que suceden en la red inalámbrica, la herramienta de visualización también incluye algunos indicadores visuales que facilitan la discriminación de diferentes situaciones. De esta forma, cuando un nodo móvil envía una petición se muestra un halo informando de este hecho (Figura A.9a). De modo similar, el tráfico entre nodos se representa usando flechar animadas indicando la dirección del tráfico (Figura A.9b).



(a) (b)

Figura A.9. Ayudas visuales. Halo de una petición (a) y representación del tráfico (b)

Además de las capacidades de visualización mencionadas anteriormente, la aplicación también es capaz de calcular parámetros de rendimiento y estadísticas acerca de la simulación en cualquier instante de la simulación. Estos parámetros pueden usarse para comparar el rendimiento entre configuraciones de redes o tamaños de caché. Las principales estadísticas que la aplicación calcula para cada nodo son:

- Número de peticiones enviadas y documentos recibidos.
- Número de *timeouts*.
- Retardo y número de satos medio.
- Número de aciertos y fallos en caché.

- Tráfico – Cantidad de peticiones, respuestas y errores enviados o reenviados por cada nodo.

Añadidamente, las estadísticas pueden exportarse a un fichero en formato PDF (*Portable Document Format*) donde se almacenan todos los estadísticos mencionados anteriormente para el instante actual de simulación.

Referencias[1]

[Abdulla,1997] G. Abdulla, E. Fox, M. Abrams, S. Williams, *WWW Proxy Traffic Characterization with Application to Caching*, Technical Report TR-97-03, Computer Science Department, Virginia Polytechnic Institute and State University (EEUU), Marzo 1997.

[Adamic,2002] L.A. Adamic, B.A. Huberman, *Zipf's law and the Internet*, Glottometrics, 3:143-150, 2002.

[Aggarwal,1997] C. Aggarwal, P.S. Yu, *On Disk Caching of Web Objects in Proxy Servers*, Proceedings of the 6th International Conference on Information and Knowledge Management, pp. 238-245, Las Vegas (EEUU), 1997.

[Aggarwal,1999] C. Aggarwal, J.L. Wolf, P.S. Yu, *Caching on the World Wide Web*, IEEE Transactions on Knowledge and Data Engineering", vol. 11, 1: 94-107, 1999.

[Ali,2007] W. Ali, S. M. Shamsuddin, *Artificial Life Neural Network for Optimizing Web Browser Caching*, Proceedings of the 3rd FSKSM Postgraduate Annual Research Seminar (PARS'07), Johol (Malasia), 2007.

[Almeida,1996] V. Almeida, A. Bestavros, M. Crovella, A. Oliveira, *Characterizing Reference Locality in the WWW*, Proceedings of the 1996 International Conference on Parallel and Distributed Information Systems (PDIS 96), pp. 92-103, Diciembre 1996.

[Ariza,2009] A. Ariza, A. Triviño, E. Casilari, J.C. Cano, C.T. Calafate, P. Manzoni, *Assessing the impact of Link Layer Feedback mechanisms on MANET routing protocols*, Proceedings of the IEEE Symposium on Computers Communications (ISCC 2009), pp. 770-775, Sousse (Túnez), Junio 2009.

[Arlitt,1996] M. Arlitt, *A Performance Study of Internet Web Servers*, Master's Thesis, Computer Science Department, University of Saskatchewan (Cánadá), Mayo 1996.

[Arlitt,1996b] M. Arlitt, C. Williamson, *Trace-Driven Simulation of Document Caching Strategies for Internet Web Servers*, Simulation Journal, vol. 68, 1:23-33, 1996.

[1] Se ha preferido referenciar las actas como "Proceedings of the ..." en lugar de "Actas de …" para respetar la denominación original.

[Arlitt,1996c] M. Arlitt, C. Williamson, *Web server workload characterization: The search for invariants*, Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pp. 126-137, Philadelphia (EEUU), Mayo 1996.

[Arlitt,1997] M. Arlitt, C. Williamson, *Internet Web Servers: Workload Characterization and Performance Implications*, IEEE/ACM Transaction on Networking, vol. 5, 5:631-645, Octubre 1997.

[Arlitt,1999] M. Arlitt, L. Cherkasova, J. Diley, R. Friedrich, T. Jin. *Evaluating Content Management Techniques for Web Proxy Caches*, Technical Report HPL-98-173, Hewlet-Packard Laboratories, Palo Alto, Abril 1999.

[Arlitt,1999b] M. Arlitt, R. Friedrich, R. Jin, *Workload Characterization of a Web Proxy in a Cable Modem Environment*, Hewlet-Packard Laboratories. Technical Report HPL-1999-48, 1999.

[Artail, 2005] H. Artail, H. Safa, S. Pierre, *Database Caching in MANETs Based on Separation of Queries and Responses*, Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking And Communications (WiMob'2005), Montreal (Canadá), Agosto 2005.

[Artail,2008] H. Artail, H. Safa, K. Mershad, Z. Abou-Atme, N. Sulieman, *COACS: A Cooperative and Adaptive Caching Systems for MANETs*, IEEE Transactions on Mobile Computing, vol. 7, 8:961-977, 2008.

[Aschenbruck,2010] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, M. Schwamborn, *BonnMotion – A Mobility Scenario Generation and Analysis Tool*, Proceedings of the 3rd International Conference on Simulation Tools and Techniques (SIMUTOOLS 2010), Málaga (España), Marzo 2010.

[Badford,1998] P. Badford, M. Crovella, *Generating Representative Web Workloads for Network and Server Performance Evaluation*, 1998 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pp.151-160, Philadelphia (EEUU), Mayo 1998.

[Bahn,2002] H. Bahn, K. Koh, S.H. Noh, S.L. Min, *Efficient Replacement of Nonuniform Objects in Web Caches*, IEEE Computer, vol. 35, 6:65-73, 2002.

[Bai,2003] F. Bai, N. Sadagopan, A. Helmy, *Important: a framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks*, Proceedings of the 22th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), pp. 383-403, San Francisco (EEUU), Abril 2003.

[Balamash,2004] A. Balamash, M. Krunz, *An Overview of Web Caching Replacement Algorithms*, IEEE Communications Surveys and Tutorials, vol. 6, 2: 44-56, 2004.

[Belloum,1998] A. Belloum, L.O. Hertzberger, *Dealing with One-Timer-Documents in Web Caching*, Proceeding of the 24th EUROMICRO'98 Conference and Telecommunication, Vesteras (Suecia), Agosto, 1998.

[Belloum,1998b] A. Belloum, L.O. Hertzberger, *Document Replacement Policies dedicated to Web Caching*, Proceedings of the IEEE International Symposium on Intelligent Control ISIC98, pp. 576,-581, Gaithersburg (EEUU), Septiembre 1998.

[Bolot,1996] J. Bolot, P. Hoschka, *Performance Engineering of the World Wide Web: Application to Dimensioning and Cache Design*, Proceedings of the 5th International World Wide Web Conference, París (Francia), Mayo 1996.

[Boyar,2010] J. Boyar, M.R. Ehmsen, J.S. Kohrt, K.S. Larsen, *A Theoretical Comparison of LRU and LRU-K*, Acta Informatica, vol. 47, (7-8): 359-374, 2010.

[Breslau,1999] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, *Web Caching and Zipf-like Distributions: Evidence and Implications*, Proceedings of the IEEE Conference on Computer and Communications (INFOCOM 1999), Nueva York (EEUU), Marzo 1999.

[Buchholz,2002] S. Buchholz, S. Jaensch, A. Schill, *Flexible Web Traffic Modeling for New Application Domains*, Proceeding of the IASTED International Conference on Applied Modelling and Simulation (AMS 2002), Cambridge (EEUU), Noviembre 2002.

[Busari,2002] M.Busari, C. Williamson, *ProWGen: A Synthetic Workload Generation Tool for the Simulation Evaluation of Proxy Caches*, Computer Networks, vol. 38, 6:779-794, 2002.

[Calzarossa,2003] M.C.Calzarossa, G. Valli, *A Fuzzy Algorithm for Web Caching*, Proceedings of the 2003 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'03), pp. 630-637, Montreal (Canadá), Julio 2003.

[Cao,1997] P. Cao, *Cost-Aware WWW Proxy Caching Algorithms*, Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS'97), Monterey (EEUU), Diciembre 1997.

[Cao,2011] P. Cao, Sitio Web del Wisconsin Web Cache Simulator, http://www.cs.wisc.edu/~cao/, consultado en Noviembre de 2011.

[Cardenas,2004] L.G. Cardenas, J. Sahuquillo, A. Pont, J.A. Gil, *The multikey Web cache simulator: a platform for designing proxy cache management techniques*, Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network-based Processing, pp. 390- 397, A Coruña (España), Febrero 2004.

[Chand,2006] N. Chand, R.C. Joshi, M. Misra, *Efficient Cooperative Caching in Ad Hoc Networks*, Proceedings of the 1st International Conference on Communication System Software and Middleware (Comsware'06), Delhi (India), Enero 2006.

[Chand,2007] N. Chand, R.C. Joshi, M. Misra, *Cooperative caching in mobile ad hoc networks based on data utility*, Mobile Information Systems, vol. 3, 1:19-37, 2007.

[Chand,2007b] N. Chand, R.C. Joshi, M. Misra, *Cooperative Caching in Mobile Ad Hoc Networks Based on Clusters*, International Journal on Wireless Personal Communications, 43:41-63, 2007.

[Chang,1999] C.Y. Chang, T. McGregor, G. Holmes, *The LRU* WWW proxy cache document replacement algorithm*, Proceedings of the 2nd Asia Pacific Web Conference (APWeb'99), Hong Kong (China), Septiembre 1999.

[Cheng,2000] K. Cheng, Y. Kambayashi, *LRU-SP: A Size-Adjusted and Popularity-Aware LRU Replacement Algorithm for Web Caching*, Proceedings of the 24th IEEE Computer Society International Computer Software and Applications Conference (Compsac'2000), pp. 48-53, Taipei (Taiwan), Octubre 2000.

[Cherkasova,1998] L. Cherkasova, Improving *WWW Proxies Performance with Greedy-Dual-Size-Frequency Caching Policy*, Technical Report HPL-98-69, HP Labs (EEUU), Noviembre 1998.

[Cherkasova,2001] L. Cherkasova, G. Ciardo, *Role of Aging, Frequency, and Size in Web Cache Replacement policies*, Proceedings of the 9th International Conference on High-Performance Computing and Networking, Lecture Notes in Computer Science, vol. 2110, pp. 114-123, Amsterdam (Holanda), Junio 2001.

[Chiang,1997] C.C. Chiang, H.K. Wu, W. Liu, M. Gerla, *Routing in Clustered Multihop Mobile Wireless Networks with Fading Channel*, Proceedings of the IEEE Singapore International Conference on Networks (SICON'97), pp. 197–211, Kent Ridge (Singapur), Abril 1997.

[Chiu,2005] G. Chiu, C. Young, F. Wu, *Efficient Cooperative Caching Schemes for Data Access in Mobile Ad Hoc Networks*, Lecture Notes in Computer Science, vol. 3824, 693-702, 2005.

[Chiu,2009] G. Chiu, C. Young, *Exploiting In-Zone Broadcast for Cache Sharing in Mobile Ad Hoc Networks*, IEEE Transactions on Mobile Computing, vol. 8, 3:384-397, 2009.

[Cho,2003] J. Cho, S. Oh, J. Kim, K.H. Lee, J. Lee, *Neighbor Caching in Multi-Hop Wireless Ad Hoc Networks*, IEEE Communications Letters, vol. 7, 11:525-527, 2003.

[Chow,2004] C.Y. Chow, H.V. Leong, A. Chan, *Peer-to-Peer Cooperative Caching in Mobile Environments*, Proceedings of the 24th International Conference on Distributed Computing Systems Workshops (ICDCSW'04), pp. 528-533, Washington (EEUU), Marzo 2004.

[Chow,2004b] C.Y. Chow, H.V. Leong, A. Chan, *Cache Signatures for Peer-to-Peer Cooperative Caching in Mobile Environments*, Proceedings of the 18th International Conference on Advanced Information Networking and Applications (AINA'04), pp. 96-101, Fukoaka (Japón), Marzo 2004.

[Chow,2004c] C.Y. Chow, H.V. Leong, A. Chan, *Group-based Cooperative Cache Management for Mobile Clients in a Mobile Environment*, Proceedings of the 33rd International Conference on Parallel Processing (ICPP'04), pp. 83-90, Montreal (Canada), Agosto 2004.

[Chow,2007] C.Y. Chow, H.V. Leong, A. Chan, *GroCoca: Group-based Peer-to-Peer Cooperative Caching in Mobile Environment*, IEEE Journal on Selected Areas in Communications, vol. 25, 1:179-191, 2007.

[Clausen,2003] T. Clausen, P. Jacquet, *Optimized Link State Routing Protocol (OLSR)*, Internet Engineering Task Force Request for Comments (RFC) 3626, Octubre 2003.

[Cobb,2008] W. Cobb, H. ElAarag, *Web proxy cache replacement scheme based on back propagation neural network*, Journal of Systems and Software, vol. 81, 9:1539–1558, 2008.

[Crovella,1996] M.E. Crovella, M.S. Taqqu, A. Bestavros, *Heavy-Tailed Probability Distributions at the World Wide Web: Traffic Self-Similarity*, Computer Science Department technical Report TR-95-015, Boston University (EEUU), Octubre 1996.

[Crovella,1997] M.E. Crovella, A. Bestavros, *Self Similarity in the World Wide Web Traffic: Evidence and Possible Causes*, IEEE/ACM Transactions on Networking, vol. 5, 6: 835-846, 1997.

[CSIM,2011] Sitio Web de CSIM: http://www.atl.lmco.com/projects/csim/, consultado en Noviembre de 2011.

[Denko,2007] M.K. Denko, *Cooperative Data Caching and Prefetching in Wireless Ad Hoc Networks*, International Journal of Business Data Communications and Networking, vol. 3, 1:1-15, 2007.

[Dodero,2006] G. Dodero, V. Gianuzzi, *Saving Energy and Reducing Latency in MANET File Access*, Proceedings of the 26th International Conference on Distributed Computing Systems Workshops (ICDCSW'06), pp. 16-20, Lisboa (Portugal), Julio 2006.

[DPD,2006] Diccionario Panhispánico de Dudas, 2ª Edición, Madrid (España) Santillana, 2006.

[Du,2005] Y. Du, S. Gupta, *COOP – A cooperative caching service in MANETs*, Proceedings of the of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services (ICAS-ICNS 2005), pp. 58-63, Papeete (Tahiti), Octubre 2005.

[Dube,1997] R. Dube, C.D. Rais, K. Wang, S.K. Tripathi, *Signal stability-based adaptive routing (SSA) for ad hoc mobile networks*, IEEE Personal Communications, vol. 4, 1:36-45, 1997.

[Dykes,2000] S.G. Dykes, K.A. Robbins, C.L. Jeffery, *Uncacheable Documents and Cold Starts in Web Proxy Cache Simulations*, Technical Report CS-2001-01, Universidad de Texas (EEUU), 2000.

[Feldman,2002] M. Feldman, J. Chuang, *Service Differentiation in Web Caching and Content Distribution*, Proceedings of the IASTED International Conference on Communication and Computer Networks (CCN 2002), Cambridge (EEUU), Noviembre 2002.

[Fiore,2007] M. Fiore, J. Harri, F. Filali, C. Bonnet, *Vehicular Mobility Simulation for VANETs*, Proceedings of the 40th Annual Simulation Symposium (ANSS'07), pp.301-309, Norfolk (USA), Marzo 2007.

[Fiore,2009] M. Fiore, F. Mininni, C. Casetti, D.F. Chiasserini, *To Cache or Not To Cache?*, Proceedings of the 28th IEEE Conference on Computer and Communications (INFOCOM 2009), pp. 235-243, Rio de Janeiro (Brazil), Abril 2009.

[Foong,1999] A.P. Foong, Y. Hu, D.M. Heisey, *Logistic Regression in an Adaptive Web Cache*, IEEE Internet Computing, vol. 3, 5:27-36, 1999.

[Gerla,1995] M. Gerla, J.T.C. Tsai, *Multicluster, mobile, multimedia radio networks*, Wireless Networks, vol. 1, 3:255-265, 1995.

[Ghosh,2007] J. Ghosh, S.J. Philipb, C. Qiao, *Sociological orbit aware location approximation and routing (SOLAR) in MANET*, Ad Hoc Networks, vol. 5, 2: 189–209, 2007.

[Gianuzzi,2003] V. Gianuzzi, *File Distribution and Caching in MANET*, Technical Report DISI-TR-03-03, DISI Tech University of Genova (Italia), 2003.

[González-Cañete,2011] Sitio Web personal de Francisco Javier González Cañete, Universidad de Málaga (España): http://pc23te.dte.uma.es

[Gourley,2002] D. Gourley, B. Totty, *HTTP: The Definitive Guide*, Sebastopol (EEUU), O'Reilly, 2002.

[Guohong, 2004] C. Guohong, L. Yin, C.R. Das, *Cooperative Cache-Based Data Access in Ad Hoc Networks*, IEEE Computer, vol. 37, 2:32-39, 2004.

[Haas,2002] Z. J. Haas, M. R. Pearlman, P. Samar, *The Zone Routing Protocol (ZRP) for Ad Hoc Networks*, Internet Engineering Task Force, Internet draft, Julio 2002.

[Havercort,2003] B.R. Haverkort, R.A. Khayari, R. Sadre, *A Class-Based Least-Recently Used Caching Algorithm for World-Wide Web Proxies*, Proceedings of the 13th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, pp. 273-290, Urbana (EEUU), Septiembre 2003.

[Heegaard,2000] P.E. Heegaard, *GenSyn - a Java based generator of synthetic Internet traffic linking user behaviour models to real network protocols*, ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management, Monterey (EEUU), Septiembre 2000.

[Hernández,2002] F. Hernández, J.S Marron, G. Samorodnitsky, F.D. Smith, *Variable Heavy Tailed Durations in Internet Traffic, Part I: Understanding Heavy Tails*, Proceedings of the 10th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunicatio Systems (MASCOTS 2002), pp. 43-52, Fort Worth (EEUU), Octubre 2002.

[Hernández,2002b] F. Hernández, J.S Marron, G. Samorodnitsky, F.D. Smith, *Variable Heavy Tailed Durations in Internet Traffic, Part II: Theoretical Implications*, Proceedings of the 40th Annual Allerton Conference on Communication, Control and Computing, Urbana-Champaign (EEUU), Octubre 2002.

[Hong,1999] X. Hong, M. Gerla, G. Pei, C. Chiang, *A Group Mobility Model for Ad Hoc Wireless Networks*, Proceedings of the ACM International Workshop on Modelling and Simulation of Wireless and Mobile Systems (MSWiM'99), pp. 53-60, Seattle (EEUU), Agosto 1999.

[Hoof,2009] A. Hoof, *Top40 cache algorithm compared to LRU and LFU*, SNE Technical Report, París (Francia), Febrero 2009.

[Hosseini-Khayat,1996] S. Hosseini-Khayat, *Optimal solution of off-line and on-line generalized caching*, Technical Report WUCS-96-20, 1996.

[Hosseini-Khayat,1997] S. Hosseini-Khayat, *Investigation of generalized caching*, Ph.D. Dissertation, Washington University, St. Louis (EEUU), 1997.

[Hou,2001] W. Hou, S. Wang, *Size-Adjusted Sliding Window LFU – A New Web Caching Scheme*, Proceedings of the 12th Internatioanl Conference on Database and Expert Syetems Applications, pp.567-576 , Munich (Alemania), Septiembre 2001.

[Hsu,2007] W. Hsu, T. Spyropoulos, K. Psounis, A. Helmy, *Modeling time-variant user mobility in wireless mobile networks*, Proceedings of the IEEE Conference on Computer and Communications (INFOCOM 2007), Anchorage (EEUU), Mayo 2007.

[Hyytia,2006] E. Hyytia, P. Lassila, J. Virtamo, *Spatial Node Distribution of the Random Waypoint Mobility Model with Applications*, IEEE Transactions on Mobile Computing, vol. 5, 6:680-694, 2006.

[IANA,2011] Sitio Web del IANA, http://www.iana.org/assignments/media-types, consultado en Septiembre de 2011.

[IRCache,2011] Sitio Web del proyecto IRCache, http://www.ircache.net, consultado en Septiembre de 2011.

[Jardosh,2003] A. A. Jardosh, E.M. Belding-Royer, K.C. Almeroth, S. Suri, *Towards realistic mobility models for mobile ad hoc networks*, Proceedings of the 9th International Conference on Mobile Computing and Networking (MobiCom), pp. 217–229, San Diego (EEUU), Septiembre 2003.

[Jin,1999] S. Jin, A. Bestavros, *Popularity-aware GreedyDual-Size Proxy Caching Algorithms*, Technical Report BUCS-TR-1999-009, Boston University (EEUU), Agosto 1999.

[Jin,2000] S. Jin, A. Bestavros, *Temporal Locality in Web Request Streams. Sources, Characteristics and Caching Implications*, Poster Proceedings of the ACM SIGMETRICS 2000 Conference, Santa Clara, (EEUU), Junio 2000

[Jin,2001] S. Jin, A. Bestabros, *GreedyDual* Web Caching Algorithm: Exploiting the Two Sources of Temporal Locality in Web Request Streams*, Journal of Computer Communications, vol. 24, 2:174-183, 2001.

[Johnson,1994] T. Johnson, D. Shasha, *2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm*, Proceedings of the 20th Very Large Data Base Conference (VLDB'94), pp. 439-450, Santiago (Chile), Septiembre 1994.

[Johnson,1996] D.B. Johnson, D.A. Maltz, *Dynamic source routing in ad hoc wireless networks*, Mobile Computing, vol. 363, 5:153–181, 1996.

[Kaya, 2009] C.C. Kaya, G. Zhang, Y. Tan, V.S. Mookerjee, *An admission-control technique for delay reduction in proxy caching*, Journal Decision Support Systems, vol. 46, 2:594-603, 2009.

[Karedla,1994] R. Karedla, J.S. Love, *Caching Strategies to Improve Disk System Performance*, IEEE Computer, vol. 27, 3:38-46, 1994.

[Kelly,1999] T. Kelly, Y.M. Chan, S. Jamin, J.K. MacKie-Mason, *Biased Replacement Policies for Web Caches: Differential Quality-of-Service Aggregate User Value*, Proceedings of the 4th International Web Caching Workshop, San Diego (EEUU), Abril 1999.

[Kelly,1999b] T. Kelly, Y.M. Chan, S. Jamin, J.K. MacKie-Mason, *Variable QoS from Shared Web Caches: User-Centered Design and Value-Sensitive Replacement*, Proceedings of the MIT Workshop on Internet Service Quality Economics (ISQE 99), Diciembre 1999.

[Khayari,2003] R. Khayari, *Workload-Driven Design and Evaluation of Web-Based Systems*, Osnabrueck (Alemania), Der Andere Verlag, 2003.

[Khayari,2005] R.A. Khayari, M. Best A. Lehmann, *Impact of Document Types on the Performance of Caching Algorithms in WWW Proxies: A Trace Driven Simulation Study*, Proceedings of the 19th International Conference on Advanced Information Networking and Applications, Tamkang (Taiwan), Marzo 2005.

[Kim,2001] K. Kim, D. Park, *Least Popularity-per-Byte Replacement Algorithm for a Proxy Cache*, Proceedings of the 8th International Conference on Parallel and Distributed Systems (ICPADS 2001), pp. 780-787, KyongJu City (Korea), Junio 2001.

[Kim,2006] M. Kim, D. Kotz, S. Kim, *Extracting a mobility model from real user traces*, Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies, Barcelona (España), Abril 2006.

[Klemm, 2003] A. Klemm, C. Lindemann, P.D. Waldhorst, *A Special-Purpose Peer-to-Peer Sharing System for Mobile Ad Hoc Networks*, Proceedings of the of the IEEE Semiannual Vehicular Technology Conference (VTC 2003), pp. 2758-2763, Orlando (EEUU), Octubre 2003.

[Kumar,2010] P. Kumar, N. Chauhan, L.K. Awasthi, N. Chand, *Proactive Approach for Cooperative Caching in Mobile Adhoc Networks*, IJCSI International Journal of Computer Science Issues, vol. 7, issue 3, 8:21-27, 2010.

[Kurkowski,2005] S. Kurkowski, T. Camp, M. Colagrosso, *A Visualization and Animation Tool for NS-2 Wireless Simulations: iNSpect*, Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2005), pp. 503-506, Atlanta (EEUU), Septiembre 2005.

[Lee,2001] D. Lee, J. Choi, J. Kim, S. Noh, S.L. Min, Y. Cho, C.S. Kim, *LRFU: A Spectrum of Policies that Subsumes the Least Recently Used and Least Frequently Used Policies*, IEEE Transactions on Computers, vol. 50, 12:1352-1361, Diciembre 2001.

[Lee,2009] K. Lee, S. Hong, S.J. Kim, I. Rhee, S. Chong, *SLAW: A Mobility Model for Human Walks*, Proceedings of the 28th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2009), Rio de Janeiro (Brasil), Abril 2009.

[Liang,1999] B. Liang, Z.J. Haas, *Predictive Distance-Based Mobility Management for PCS Networks*, Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 1999), pp. 1377-1384, Nueva York (EEUU), Abril 1999.

[Lim,2006] S. Lim, W.C. Lee, G. Cao, C.R. Das, *A novel caching scheme for improving Internet-based mobile ad hoc networks performance*, Ad Hoc Networks, vol. 4, 2:225-239, 2006.

[Lim,2006b] S. Lim, C. Yu, C.R. Das, *Clustered Mobility Model for Scale-Free Wireless Networks*, Proceedings of the 31st IEEE Conference on Local Computer Networks (LCN'2006), pp. 231-238, Tampa (EEUU), Noviembre 2006.

[Lim,2007] S. Lim, W.C. Lee, G. Cao, C.R. Das, *Cache invalidation strategies for Internet-based mobile ad hoc networks*, Computer Communications, vol. 30, 8:1854-1869, 2007.

[Livathinos, 2009] S.N. Livathinos, *EXtensible Animator for Mobile Simulations: EXAMNS*, Proceedings of the 17th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2009), Londres (Reino Unido), Septiembre 2009.

[Lixin,2009] L. Lixin, Z. Huisheng, *Research on cross-layer design for MANETs*, Proceedings of the 3rd International Symposium on Intelligent Information Technology Application, pp. 225-228, Xi'an (China), Diciembre 2009.

[Lorenzetti,2000] P. Lorenzetti, L. Rizzo, L. Vicisano, *Replacement policies for a proxy cache*, IEEE/ACM Transactions on Networking, vol. 8, 2:158-170, 2000.

[Mahanti,1999] A. Mahanti, C. Williamson, *Web Proxy Workload Characterization*, Technical Report, Department of Computer Science, University of Saskatchewan, Febrero 1999.

[Mahanti,1999b] A. Mahanti, *Web Proxy Workload Characterization and Modelling*, M.Sc. Thesis, Department of Computer Science, University of Saskatchewan, Septiembre 1999.

[Mahanti,2000] A. Mahanti, C. Williamson, y D. Eager, *Traffic Analysis of a Web Proxy Caching Hierarchy*, IEEE Network, vol. 14, 3:16-23, 2000.

[Markatchev,2002] N. Markatchev, C. Williamson, *WebTraff: A GUI for web proxy cache workload modeling and analysis*, Proceedings of the 10th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2002), pp. 356-363, Fort Worth (EEUU), Octubre 2002.

[Markatos,1996] E.P. Markatos, *Main Memory Caching of Web Documents*, Proceedings of the 5th International World Wide Web conference on Computer networks and ISDN systems, París (Francia), 1996.

[Márquez,2008] J. Márquez, J. Domènech, J.A. Gil, A. Pont, *A Web Caching and Prefetching Simulator*, Proceedings of the 16th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2008), Split (Croacia), Septiembre 2008.

[Menaud,2000] J. Menaud, V. Issarny, M. Banâtre, *Improving the Effectiveness of Web Caching*, Lecture Notes in Computer Science, vol. 1752, pp. 375-401, 2000.

[Moriya,2003] T. Moriya, H. Aida, *Cache Data Access System in Ad Hoc Networks*, Proceedings of the 57th IEEE Semiannual Vehicular Technology Conference (VTC 2003), vol. 2, pp. 1228-1232, Orlando (USA), Abril 2003.

[Mousavi,2007] S.M. Mousavi, H.R. Rabiee, M. Moshref, A. Davirmoghaddam, *MobiSim: A Framework for Simulation of Mobility Models in Mobile Ad-Hoc Networks*, Proceedings of the 3rd IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2007), Nueva York, (EEUU), Octubre 2007.

[Murta,1998] C.D. Murta, V. Almeida, W. Meira, *Analyzing Performance of Partitioned Caches for the WWW*, Proceedings of the 3rd International WWW Caching Workshop, Manchester (Reino Unido), Junio 1998.

[Murthy,1996] S. Murthy J.J. García-Luna-Aceves, *An Efficient Routing Protocol for Wireless Networks*, Mobile Networks and Applications: special issue in mobile communications networks, vol. 1, 2:183-197, 1996.

[Nagaraj,2004] S. Nagaraj, *Web Caching and its Applications*, Londres (Reino Unido), Kluwer Academic Publishers, 2004.

[NAM,2011] Sitio Web de NAM, http://www.isi.edu/nsnam/nam/, consultado en Noviembre de 2011.

[Niclausse,1998] N. Niclausse, Z. Liu, P. Nain, *A New Efficient Caching Policy for the World Wide Web*, Proceedings of the Workshop on Internet Server Performance (WISP'98), Madison (EEUU) , Junio 1998.

[NS2,2011] Sitio Web de NS-2, http://www.isi.edu/nsnam/ns/, consultado en Noviembre de 2011.

[Nuggehalli,2003] P. Nuggehalli, V. Srinivasan, C.F. Chiasserini, *Energy-Efficient Caching Strategies in Ad Hoc Wireless Networks*, Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003), pp. 25-34, Annapolis (EEUU), Junio 2003.

[O'Neil,1993] E.J. O'Neil, P.E. O'Neil, G. Weikum, *The LRU-K Page Replacement Algorithm for Database Disk Buffering*, Proceeding of the ACM SIGMOD International Conference on Management of Data, pp. 297-306, Washington (EEUU), Mayo 1993.

[Osawa,1997] N. Osawa, T. Yuba, K. Hokozaki, *Generational replacement schemes for a WWW proxy server*, Proceedings of the High-performance Computing and Networking (HPCN'97), pp. 940-949, Viena (Austria), Abril 1997.

[Park,1996] K. Park, G. Kim, M. Crovella, *On the relationship between file sizes, transport protocols and self-similar network traffic*, Proceedings of the 4th International Conference on Networks Protocols (ICNP '96), pp. 171–180, Columbus (EEUU), Octubre-Noviembre 1996.

[Park,2001] V. Park, S. Corson, *Temporally-Ordered Routing Algorithm (TORA)*, Internet Engineering Task Force, Internet draft, Julio 2001.

[Perkins,1994] C. E. Perkins, P. Bhagwat, *Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers*, ACM SIGCOMM Computer Communication Review, vol. 24, 4:234-244, 1994.

[Perkins,2003] C. E. Perkins, E. M. Belding-Royer, S. Das, *Ad Hoc On Demand Distance Vector (AODV) Routing*, Internet Engineering Task Force Request for Comments (RFC) 3561, Julio 2003.

[Pitkow,1994] J.E. Pitkow, M.M Recker, *A Simple Yet Robust Caching Algorithm Based on Dynamic Access Patterns*, Proceedings of the 2nd International WWW Conference, Chicago (EEUU), Octubre 1994.

[Podlipnig, 2003] S. Podlipnig, L Boszormenyi, *A survey of Web cache replacement strategies*, ACM Computing Surveys, vol. 35, 4:374-398, 2003.

[Psounis,2001] K. Psounis, B. Prabhakar, *A Randomized Web-Cache Replacement Scheme*, Proceeding of the 20th IEEE Conference on Computer and Communications (INFOCOM 2001), vol. 3, pp. 1407-1415, Anchorage (EEUU), Abril 2001.

[Radhika,2003] S. Radhika, R. Govindarajan, *An Efficient Web Cache Replacement Policy*, Procceding of the 9th International Symposium on High Performance Computing, Hyderabad (India), Diciembre 2003.

[Rappaport,1996] T.S. Rappaport, *Wireless communications: principles and practice*, Prentice Hall, 1996.

[Reddy,1998] M. Reddy, G.P. Fletcher, *Intelligent web caching using document life histories: A comparison with existing cache management techniques*, Proceedings of the 3rd International WWW Caching Workshop, pp. 35-50, Manchester (Reino Unido), Junio 1998.

[Ridruejo,2005] F.J. Ridruejo, A. Gonzalez, J. Miguel-Alonso, *TrGen: a Traffic Generation System for Interconnection Network Simulators*, Proceedings of the 34th International Conference Workshops on Parallel Processing, pp. 547-553, Oslo (Noruega), Junio 2005.

[Roadknight,1999] C. Roadknight, I. Marshall, D. Vearer, *File Popularity Characterization*, Proceedings of the 2nd Workshop on Internet Server Performance (WISP 99), Atlanta (EEUU), Mayo 1999.

[Robinson,1990] J.T. Robinson, M.V. Devarakonda, *Data Cache Management Using Frequency-Based Replacement*, Proceedings of the 1990 ACM Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 1990), pp. 134-142, Boulder (EEUU), Mayo 1990.

[Romano,2011] S. Romano, H. ElAarag, *A neural network proxy cache replacement strategy and its implementation in the Squid proxy server*, Neural Computing & Applications, vol. 20, 1:59-78, 2011.

[Royer,2001] E.M. Royer, P.M. Melliar-Smith, L.E. Moser, *An Analysis of the Optimum Node Density for Ad hoc Mobile Networks*, Proceedings of the IEEE International Conference on Communications (ICC'01), pp. 857-861, Helsinki (Finlandia), Junio 2001.

[Sabeghi,2006] M. Sabeghi, M. H. Yaghmaee, *Using Fuzzy Logic to Improve Cache Replacement Decisions*, International Journal of Computer Science and Network Security, vol. 6, 3:182-188, 2006.

[Sailhan,2002] F. Sailhan, V. Issarny, *Energy-aware Web Caching for Mobile Terminals*, Proceedings of the 22nd International Conference on Distributed Computing Workshops (ICDCSW'02), pp. 820-825, Viena (Austria), Julio 2002.

[Sailhan,2003] F. Sailhan, V. Issarny, *Cooperative Caching in ad hoc Networks*, Proceedings of the 4th ACM International Conference on Mobile Data Management (MDM'2003), pp. 13-28, Melbourne (Australia), Enero 2003.

[Scheuermann,1996] P. Scheuermann, J. Shim, R. Vingralek, *WATHMAN A Data Warehouse Intelligent Cache Manager*, Proceedings of the 22nd Very Large DataBases Conference (VLDB'96), pp. 51-62, Bombay (India), 1996.

[Scheuermann,1997] P. Scheuermann, J. Shim, R. Vingralek, *A Case for Delay-conscious Caching of Web Documents*, Proceedings of the 6th International World Wide Web Conference, pp. 8-13, Santa Clara (EEUU), Abril 1997.

[Scheuermann,2005] B. Scheuermann, H. Füßler, M. Transier, M. Busse, M. Mauve, W. Effelsberg, *Huginn: A 3D Visualizer for Wireless ns2 Traces*, Proceedings of the 8th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pp. 143-150, Montreal (Canadá), Octubre 2005.

[Shakkottai,2003] S. Shakkottai, T.S. Rappaport, P.C. Karlsson, *Cross-layer design for wireless networks*, IEEE Communications Magazine, vol. 41, 10:74-80, 2003.

[Shim,1999] J. Shim, P. Scheuermann, R. Vingralek, *Proxy Cache Algorithms: Design, Implementation and Performance*, IEEE Transactions on Knowledge and Data Engineering, vol. 11, 4:549-562, 1999.

[Shin,2003] S.W. Shin, K.Y. Kim, J.S. Jan, *LRU based Small Latency First Replacement (SLRF) algorithm for the proxy cache*, Proceedings of the IEEE/WIC International Conference on Web Intelligence (WI'03), pp. 499-502, Halifax (Canadá), Octubre 2003.

[Squid,2011] Sitio Web de Squid, http://www.squid-cache.org/, consultado en Septiembre de 2011.

[Starobinski,2001] D. Starobinski, D. Tse, *Probabilistic Methods for Web Caching*, Performance Evaluation, vol. 46, 2:125-37, Octubre 2001.

[Tabassum, 2010] K. Tabassum, A. Damodaram, *A Heuristic-based Cache Replacement Policy for Data Caching*, International Journal of Computer Science and Technology, vol. 1, 2:68,71, Diciembre 2010.

[Tang,2008] B. Tang, H. Gupta, S.R. Das, *Benefit-Based Data Caching in Ad Hoc Networks*, IEEE Transactions on Mobile Computing, vol. 7, 3:289-304, 2008.

[Tatarinov,1998] I. Tatarinov, *An Efficient LFU-Like Policy for Web Caches*, Computer Sccience Department Technical Report NDSU-CSORTR-98-01, North Dakota State University (EEUU), 1998.

[Ting,2007] Y. Ting, Y. Chang, *A Novel Cooperative Caching Scheme for Wireless Ad Hoc Networks: GroupCaching*, Proceedings of the International Conference on Networking, Architecture and Storage (NAS 2007), pp. 62-68, Guilin (China), 2007.

[Toh,1996] C.K. Toh, *A Novel Distributed Routing Protocol To Support Ad-Hoc Mobile Computing*, Proceeding of the 15th IEEE Annual International Phoenix Conference on Computer and Communications, pp. 480–86, Scottsdale (EEUU), Marzo 1996.

[Vakali,2000] A. Vakali, *LRU-based algorithms for Web cache replacement*, Proceedings of the 1st International Conference on Electronic Commerce and Web Technologies, Lecture Notes in Computer Science, vol. 1875, pp. 409-418, Londres (Reino Unido), Septiembre 2000.

[Vakali,2002] A. Vakali, *Evolutionary Techniques for Web Caching*, Journal Distributed and Parallel Databases, vol. 11, 1:93-116, 2002

[Wang,1999] J. Wang, *A Survey of Web Caching Schemes for the Internet*, ACM SIGCOMM Computer Communication Review, vol. 19, 5:36-46, 1999.

[Wang,2006] Y.H. Wang, J. Chen, C.F. Chao, C.C. Chuang, *A Distributed Data Caching Framework for Mobile Ad Hoc Networks*, Proceedings of the 2006 International conference on Wireless communications and mobile computing, pp. 1357-1362, Vancouver (Canadá), 2006.

[Wessels,1995] D. Wessels, *Intelligent caching for World Wide-Web objects*, M.S. Thesis, Colorado University (EEUU), 1995.

[Williams,1996] S. Williams, M. Abrams, C.R. Standridge, G. Abdulla, E.A. Fox, *Removal Policies in Network Caches for World-Wide Web Document*, Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'96), pp.293-305, Stanford (EEUU), Agosto 1996.

[Wong,2006] K. Wong, *Web Cache Replacment Policies: A Pragmatic Approach*, IEEE Network, vol. 20, 1:28-34, 2006.

[Wooster,1996] R.P. Wooster, *Optimizing Response Time, Rather than Hir Rates*, Master thesis, CS Department, Virginia Polytechnic Institute and State University (EEUU), Diciembre, 1996.

[Wooster,1997] R. Wooster, M. Abrams, *Proxy Caching that Estimates Page Load Delays*, Computer Networks and ISDN Systems, vol. 29, 8:977-986, 1997.

[Yang,2001] Q. Yang, H.H. Zhang, H. Zhang, *Taylor series Prediction: A Cache Replacement Policy Based on Second-Order Trend Analysis*, Proceedings of the 34th Annual Hawaii International Conference on System Science (HICSS'2001), Hawaii (EEUU), Enero, 2001.

[Yang,2001b] Q. Yang, H.H. Zhang, T. Li, *Mining web logs for prediction models in WWW caching and prefetching*, Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2001), pp. 473-478, San Francisco (EEUU), Agosto 2001.

[Yin,2006] L. Yin, G. Cao, *Supporting Cooperative Caching in Ad Hoc Networks*, IEEE Transaction on Mobile Computing, vol. 5, 1:77- 89, 2006.

[Yoon,2003] J. Yoon, M. Liu, B. Noble, *Random Waypoint Considered Harmful*, Proceedings of the 22th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003), pp. 1312-1321, San Francisco (EEUU), Abril 2003.

[Young,1998] N. Young, *Online file caching*, Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 82-86, San Francisco (EEUU), Enero 1998.

[Yu,2001] F. Yu, Q. Zhang, W. Zhu, Y. Zhang, *Network-Adaptive Cache Management Schemes for Mixed Media*, Proceedings of the 2nd IEEE Pacific-Rim Conference on Multimedia (IEEE-PCM), pp. 685-692, Pequín (China), Octubre 2001.

[Zhang,1999] J. Zhang, R. Izmailov, D. Reininger, M. Ott, *WebCASE: A simulation environment for web caching study*, Proceedings of the 4th International Web Caching Workshop, San Diego (EEUU), Marzo-Abril 1999.

[Zhang,1999b] J. Zhang, R. Izmailov, D. Reininger, M. Ott, *Web caching framework: Analytical models and beyond*, Proceedings of the IEEE Workshop on Internet Applications, pp. 132-141, Piscataway (EEUU), Julio 1999.

[Zhang,2000] X. Zhang, *Cachability of Web Objects*, Technical Report 2000-19, 2000.

[Zhao,2008] J. Zhao, P. Zhang, G. Cao, *On Cooperative Caching in Wireless P2P Networks*, Proceedings of the 28th International Conference on Distributed Computing Systems, pp. 731-739 , Pequín (China), Junio 2008.

[Zhao,2010] J. Zhao, P. Zhang, G. Cao, C.R. Das, *Cooperative Caching in Wireless P2P Networks: Design, Implementation and Evaluation*, IEEE Transaction on Parallel and Distributed Systems, vol. 21, 2:229-241, 2010.

[Zhou,2001] Y. Zhou, J.F. Philbin, *The Multi-Queue Replacement Algorithm for Second Level Buffer Caches*, Proceedings of the 2001 Annual Technical Conference (USENIX 2001), pp. 91–104, Boston (EEUU), Junio, 2001.

[Zipf,2011] Information on Zipfs law (bibliografía on-line gestionada por W. Li): http://www.nslij-genetics.org/wli/zipf/, consultado en Octubre 2011.