PEDECIBA Informática
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Tesis de Maestría
en Informática

Redes de Contenido:
Taxonomía y Modelos de evaluación y diseño de los mecanismos de descubrimiento de contenido

Pablo Rodríguez-Bocca

Octubre 2005

Orientador de Tesis: Dr. Héctor Cancela Bosi
Supervisor: Dr. Héctor Cancela Bosi

Redes de Contenido: Taxonomía y Modelos de evaluación
y diseño de los mecanismos de descubrimiento de
contenido.
Pablo Rodríguez-Bocca.

ISSN 0797-6410
Tesis de Maestría en Informática, PEDECIBA.
Reporte Técnico INCO-RT-05-xx (número en trámite).
PEDECIBA.
Instituto de Computación - Facultad de Ingeniería
Universidad de la República.

Montevideo, Uruguay, Octubre de 2005.

Agradecimientos

En primer lugar quiero agradecer a mi tutor, Héctor Cancela Bosi, por su apoyo y estímulo permanente. Al área de Informática del PEDECIBA por la admisión al programa de Maestría en Informática. Al programa de jóvenes investigadores de la Comisión Sectorial de Investigación Científica (CSIC), al proyecto RMS del Programa de Desarrollo Tecnológico (PDT) y al proyecto PAIR del programa de Equipos Asociados del INRIA (Francia) por su apoyo económico en distintas etapas de la maestría.Al Departamento de Investigación Operativa del Instituto de Computación y a la Administración Nacional de Telecomunicaciones (ANTEL) por permitir mi dedicación al proyecto.

A mis padres, que con su esfuerzo me permitieron realizar las carreras de Ingeniería en Computación y Eléctrica, siempre fomentando en mi el ingenio y el trabajo. A mi hermana y amigos por ayudarme y disimular interés en estos temas académicos; y a mi novia por su incondicional apoyo y paciencia.

Resumen

En este trabajo se estudia a las Redes de Contenido en su conjunto, desde el punto de vista de su diseño y arquitectura. Una Red de Contenido es una red virtual que se monta sobre la infraestructura IP, donde se soportan búsquedas y enrutamiento en base exclusivamente al contenido. Estas redes virtuales de contenido tienen el atractivo y la flexibilidad de adecuarse a los requerimientos de cada aplicación específica en cuanto a confiabilidad, performance, anonimato, seguridad, etc. En los últimos años, se han desarrollado en distintos contextos gran cantidad de redes de contenido, incluyendo redes peer-to-peer, cooperative Web caching, content delivery network, subscribe-publish networks, content-based sensor networks, backup networks, distributed computing, collaboratives networks, instant messagings, multiplayers games y search engines.

En la primera parte del trabajo se determina una taxonomía y clasificación sobre la arquitectura y diseño que presentan las redes de contenido. Esto permite especificar una serie de problemas relacionados, desde un enfoque de optimización. En particular, se concentra el estudio sobre una clase de problemas de diseño: respecto al descubrimiento del contenido en la red, donde existe un compromiso entre la publicación y la búsqueda de información. Estas redes intentan, mediante distintas técnicas, maximizar la disponibilidad del contenido presente en la red. Al encontrarse los nodos participantes limitados en su consumo de comunicación, se debe decidir si invertir en publicar los nuevos contenidos o esperar a las búsquedas de los solicitantes. Dependiendo de la dinámica del contenido en la red (tanto en las solicitud como en el alojamiento del contenido) y del costo de transmisión de una publicación o una búsqueda, se encuentra el compromiso de diseño de la red.

En la segunda parte del trabajo, se presentan dos modelos para el problema del descubrimiento del contenido, uno aplicable al contexto general y otro a los nodos más comprometidos de la red: los nodos agregadores (o cache). En la amplia mayoría de las redes de gran escala existen los nodos agregadores, estos nodos tienen como finalidad disminuir el costo de participar en la red a los solicitantes y fuentes de contenido. Una de las técnicas más aplicadas en estos nodos, para minimizar el consumo de ancho de banda y la latencia, es reutilizar temporalmente consultas previas de algún contenido (el cache).

Para el problema en los nodos agregadores se presenta una metodología de resolución, la cual se aplica a dos casos de estudios concretos: una red P2P para compartir archivos y el sistema DNS.

Palabras clave*: Redes de contenido, peer-to-peer, optimización no lineal, caching, clustering, DNS.*

Contenido

Introducción General

1

En los últimos años, con el crecimiento de Internet se han expandido las redes de aplicaciones de contenido específico, como las redes entre pares (peer-to-peer), las redes para distribución de contenido (content delivery network), los sistemas de mensajería y colaboración, los juegos en línea, etc.
Estas redes, conocidas en su conjunto como Redes de Contenido, son redes virtuales que se montan sobre la infraestructura IP de la Internet o de una red corporativa. Las redes virtuales, como su nombre lo indica, tienen la flexibilidad de crear una topología virtual propia, no importando la red física IP subyacente (es decir, dos nodos conectados en la red virtual pueden estar muy alejados en la red física). Para acceder al contenido no es necesario mantener continuamente un vínculo fijo entre el contenido y el host donde está alojado. Más aún, para muchas redes de contenido, en un momento dado, parte del contenido puede ser alojado, movido o replicado en algún otro nodo de la red, con el fin de mejorar la disponibilidad, tiempo de acceso, anonimato, etc.

Las Redes de Contenido han jugado un papel protagónico en el desarrollo de la sociedad de la información, brindando infraestructura para la oferta de contenido especializado y/o masivo. La misma Internet puede verse bajo este paradigma como una red de contenido, donde el contenido son las direcciones de red IP y los protocolos de enrutamiento el mecanismo que tiene la red para descubrir el contenido. Otro ejemplo claro es el sistema de nombres de dominio DNS (por sus siglas en inglés: Internet Domain Name System), donde el contenido son los nombres completos de los host, siendo un sistema vital para la mayoría de servicios que se ofrecen en Internet.

En este capítulo se presentan las motivaciones del estudio (sección 1.1), los principales resultados del trabajo realizado (sección 1.2), y la estructura general del Informe (sección 1.3).

1.1 Motivación

Dentro de las Redes de Contenido se encuentran las redes peer-to-peer, cooperative Web caching, content delivery network, subscribe-publish networks, content-based sensor networks, backup networks, distributed computing, collaboratives networks, instant messagings, multiplayers games, search engines, etc. Cada una de estas redes varía en el público objetivo que las utiliza y principalmente en el tipo de contenido que alojan (variando su contenido desde simples archivos de sistema a capacidad de cómputo, capacidad de almacenamiento e inclusive interacción humana).
Esta amplia variedad de aplicaciones ha provocado que las redes de contenido presenten en la actualidad múltiples arquitecturas, a pesar que en muchos casos enfrenten problemas de diseño en común. Debido a esta disgregación, las redes de contenido permiten una adaptabilidad muy grande a las necesidades particulares de

nuevas aplicaciones con requerimientos específicos en cuanto a confiabilidad, disponibilidad, redundancia, performance, tiempo de acceso, anonimato, seguridad, etc. Esto sumado a que son desarrolladas bajo las nuevas tendencias de la Informática (como ser tecnologías de comunicación interoperables basadas en XML, conexiones transparentes a través de proxies, etc.) ha permitido la proliferación de las redes de contenido principalmente en el ámbito de Internet.

Otros aspectos de la actualidad determinan un contexto favorable para la utilización masiva de redes de contenido.
En los últimos años a crecido enormemente la penetración del acceso a Internet en su modalidad de banda ancha, permitiendo a los usuarios compartir la gran cantidad de recursos (procesamiento, almacenamiento, transmisión) ociosos que disponen.
La independencia entre el contenido y una dirección de alojamiento (IP) fija que utilizan estas redes, determina menos vulnerabilidad a diversos ataques, en particular los del tipo de denegación de servicio (Denial of Service: DoS), además de permitir nuevas aplicaciones entre pares o computación móvil.
Desde el punto de vista empresarial, debido a que las redes de contenido utilizan aplicaciones propias del lado del cliente, estas permiten mayor fidelización por parte del mismo. A su vez, al ser diseñadas pensando en su escalamiento, estas redes en general reducen la carga administrativa (operación y mantenimiento) y el costo de los elementos de red, respecto a las soluciones tradicionales para ofrecer contenido.

Por estas razones, en los últimos años ha crecido enormemente la utilización de las redes de contenido, lo cual es de fuerte interés para los actuales proveedores de servicios en Internet (ISPs), dado el impacto que tienen estas redes emergentes en su infraestructura de Internet, así como en posibles ofertas de valor agregado que surjan con su despliegue.

Con las redes de contenido surge una serie de consideraciones relacionadas a su arquitectura y diseño. El proyecto se enmarca en el área de interés del Departamento de Investigación Operativa, Instituto de Computación, Facultad de Ingeniería, Universidad de la República. Hace más de una década que el Departamento realiza trabajos en el diseño de redes de comunicaciones, con buenas propiedades en cuanto a su confiabilidad y performance[24][25][26][27][28]. El resultado de haber definido este área como de prioritario interés ha permitido conformar un grupo humano de alto rigor académico y experticia, además de un destaque a nivel mundial de sus investigaciones. En la actualidad, se está realizando un proyecto marco con el equipo ARMOR del INRIA, Rennes, Francia, que engloba distintas actividades en el tema. Se trata del proyecto PAIR, Planificación de Arquitectura e Infraestructura de Redes.

La flexibilidad en el diseño de estas redes virtuales muchas veces presenta un impacto negativo en la infraestructura existente de Internet, debido a que dos nodos conectados en la red virtual pueden estar muy alejados en la red física.
Al no contemplarse a la red de Internet subyacente cuando se arman sus topologías, se provoca un mayor retardo a los clientes y un fuerte impacto sobre la lógica de negocios de los proveedores de Internet (con un consumo elevado en los costosos enlaces internacionales) [187][188][200][205].
Esto también importa a los proveedores de equipamiento de redes, viendo la oportunidad de negocio al brindar soluciones a los ISPs. En la actualidad, el tráfico de las aplicaciones P2P representa un porcentaje importante del consumo en los enlaces internacionales de acceso a Internet en cada ISP, varias empresas ofrecen equipamiento que garantiza una reducción del consumo en los enlaces internacionales debido a aplicaciones P2P. Ejemplos de equipamiento: PacketShaper de Packeteer[186], Peer-to-Peer-Element de Sandvine[209] y PeerCache de Joltid[191].

1.2 Resultados y Contribuciones

Las contribuciones presentadas en este trabajo están enfocadas a brindar un mayor entendimiento de las redes de contenido en su globalidad. Estudiando en detalle lo que refiere a la arquitectura y diseño de las mismas, y a la medición y evaluación de sus performances.

Las contribuciones realizadas pueden ser agrupadas dentro de los siguientes puntos:

- Características y taxonomía de las redes de contenido y las redes P2P.
- Modelos para el problema del descubrimiento del contenido en estas redes.
- Metodología para la resolución del problema del descubrimiento en los nodos agregadores.

1.2.1 Características y Taxonomía de las Redes de Contenido y las Redes P2P.

Se ha recopilado y analizado la literatura referente a redes de contenido y redes P2P de forma detallada.
Son estudiados en profundidad dos conjuntos de características de estas redes: uno relacionado con la arquitectura o el diseño, y otro relacionado con el comportamiento o la performance, presentando ejemplos de redes reales.
A partir de dichas características se tiene como síntesis y extensión de las propuestas existentes en la literatura, una propuesta de taxonomía para redes de contenido y redes P2P.

1.2.2 Modelos para el Problema del Descubrimiento del Contenido en estas redes.

Se identificaron algunos problemas y relaciones de compromisos que ocurren en el diseño de las arquitecturas de las redes de contenido. Un tipo de problema presente en la mayoría de las redes resalta sobre el resto y es estudiado en profundidad: es el problema del descubrimiento de contenido en redes distribuidas.

En una red de contenido el enrutamiento y alojamiento del contenido se basan en la descripción del contenido en lugar de su ubicación, por tanto en estas redes no existe un vínculo fijo entre el contenido y el host que lo aloja. De esta forma, toda red de contenido es en realidad una red de conocimiento, donde el conocimiento es la información de alojamiento de cada contenido específico. Por información de alojamiento debe entenderse la meta-información que relaciona el contenido con el nodo que lo posee.

La meta-información se encuentra distribuida entre los nodos de la red, por lo que en general no existe una vista global de toda la meta-información. El objetivo de la red es poder brindar a los solicitantes de cada contenido la mayor cantidad de nodos que lo alojan, es decir, tener la vista global más completa posible de la meta-información.
Básicamente existen dos formas de descubrir la meta-información, estas formas son la publicación y la búsqueda. Por publicación debe entenderse que un nodo de la red voluntariamente decide informarle a otros de la meta-información que él conoce. Por búsqueda debe entenderse el solicitarle a otros la meta-información que posean.
El carácter dinámico del contenido y los nodos de la red hace muy difícil la tarea de mantener la meta-información en la red, se considera que cuanto más dinámica es la red más debe descubrirse la información mediante búsquedas, puesto que el costo

de la publicación sería injustificado dado que la meta-información descubierta rápidamente perdería su validez.

El problema de descubrimiento de contenido es especificado mediante una combinación de un modelado de la evolución discretizado en el tiempo con una formulación de programación entera. En el documento a este problema se le da el nombre de ***CDP* (Content Discovery Problem)**, y básicamente trata:

> Establecido un patrón de conexión, de alojamiento de contenido y de solicitud de contenido, el problema consiste en maximizar la localización del contenido solicitado, permitiendo variar la política de descubrimiento (publicación y búsqueda) y expiración de la información.

Otros dos modelos son presentados como casos particulares del problema general. Consideran el problema del compromiso entre la publicación y la búsqueda solo en un tipo de nodos de la red: los nodos agregadores (o nodos caches). El nodo agregador debe determinar que contenido alojar en su cache y que contenido debe buscar en la red de contenido, de forma de maximizar las fuentes de contenido respondidas a los solicitantes, es decir, deben definir una política óptima de fijación de tiempos de expiración de la información.
Se define el ***CCP* (Content Caching Problem)** de la manera siguiente:

> Establecido un patrón de alojamiento de contenido y de solicitud de contenido en la red, el problema consiste en maximizar la localización del contenido solicitado, permitiendo variar la política de expiración de la información en el nodo agregador.

Y el ***CCCP* (Content Class Caching Problem)**:

> Establecidas clases de contenido en una red, donde cada clase comparte un patrón de alojamiento y de solicitud común de contenidos, el problema consiste en maximizar la localización del contenido solicitado, permitiendo variar la política de expiración de la información en el nodo agregador.

El *CCCP* es una generalización del problema *CCP*. Introduciendo la noción de clases de contenido, esta generalización tiene como objetivo la simplificación en la búsqueda de soluciones. Esto se logra reduciendo la cantidad de variables de decisión del problema, para los casos de estudio del orden del millón a un orden de unos cientos (las variables de decisión del *CCP* son los tiempos de expiración de cada contenido, mientras que las del *CCCP* son los tiempos de las clases).
Ambos problemas se expresan mediante modelos de programación no lineal (en particular, problemas de maximización de función objetivo y restricciones convexas).

1.2.3 Metodología para la Resolución del Problema del descubrimiento en los nodos agregadores.

Para los problemas *CCP* y *CCCP* se especifica una metodología de resolución, estudiando su validez en dos casos de estudio:

- ***P2P***: Se estudia el comportamiento del modelo en una red de intercambio de archivos En la actualidad, el tipo de red más utilizado para estos fines (según la taxonomía presentada) pertenecen a las redes sintácticas insensibles con distribución jerarquizada (donde se incluyen por ejemplo la red FastTrack de KaZaA[127]). Los valores utilizados en las instancias de este caso de prueba se extrajeron de la bibliografía.

- ***DNS***: También se muestra la utilidad del modelo particular en la red DNS (Domain Name System)[157][158], que es una red semántica sensible de distribución jerarquizada. Los valores de las instancias de este caso de prueba se extrajeron de la realidad, utilizando datos de la empresa de telecomunicaciones ANTEL[2].

1.3 Organización del Documento

A continuación se explica brevemente la estructura del Informe.

El capítulo 2 es una introducción detallada a las Redes de Contenido, en particular lo que refiere a la arquitectura y diseño de las mismas, y a la medición y evaluación de sus performances. De esta forma, son estudiados en profundidad dos conjuntos de características de las redes de contenido: uno relacionadas con la arquitectura y otro relacionado con el comportamiento. Siempre que es posible el capítulo se nutre de ejemplos de redes reales.
A partir de dichas características se tiene como síntesis y extensión de las propuestas existentes en la literatura, una propuesta de taxonomía para redes de contenido y redes P2P.
El capítulo culmina con un estudio más detallado de dos tipos de redes de contenido que acompañaran el trabajo en los siguientes capítulos: las Redes de Pares (peer-to-peer) y el Sistema de Nombres de Dominio (DNS).

Del análisis del estado del arte, se extraen como conclusiones del capítulo 2 problemáticas y relaciones de compromiso a nivel del diseño que presentan las redes de contenido. El capítulo 3 especifica el modelado de uno de los problemas (llamado *CDP*) que se considera más importante para la mayoría de las redes: el del compromiso entre la publicación y la búsqueda de la información.
Este modelo detallado del comportamiento de los nodos de una red de contenido combina un modelado de la evolución discretizado en el tiempo con una formulación de programación entera. A pesar de no haberse implementado en este trabajo, este modelo puede ser base de estudios futuros para su simulación o eventual optimización.

En el capítulo 4 se detallan dos modelos (el *CCP* y el *CCCP*) de programación no lineal para el caso específico de la fijación de tiempos de expiración de contenidos en un nodo agregador (o nodo cache). Este es un caso particular del problema de compromiso entre publicación y búsqueda de información, donde el nodo agregador debe determinar que contenido alojar en su cache y que contenido debe buscar en la red de contenido, de forma de maximizar las fuentes de contenido respondidas a los solicitantes.

El capítulo 5 especifica una metodología para resolver los problemas *CCP* y *CCCP* a través de nuestros dos casos de estudio: una red P2P y el sistema DNS.

Por último, en el capítulo 6 se presentan las principales conclusiones y contribuciones del trabajo, conjuntamente con algunas posibles líneas de trabajo futuro.

Contexto Tecnológico: Redes de Contenido

2

Las redes de contenido son unos de los componentes de mayor despligue en la actual infraestructura tecnológica que soporta el desarrollo de la sociedad de la información, dada su capacidad para brindar acceso a grandes volúmenes de contenido especializado, manejando eficientemente la masividad tanto respecto a la cantidad de usuarios como al volúmen de información disponible. Decenas o cientos de estas redes se han prototipado y puesto en producción, entre las más nombradas en los medios de comunicación podemos mencionar las aplicaciones peer-to-peer, los sistemas de mensajería instantánea y las redes de distribución de contenido (CDNs); incluso por su esquema de funcionamiento la propia Internet y el sistema DNS son ejemplos de redes de contenido.

La variedad de diseños presentes en las Redes de Contenido les permite adaptarse exitosamente a las necesidades particulares de cada aplicación. Con el crecimiento de Internet, en los últimos años se han desarrollado gran cantidad de redes de contenido, con objetivos muy variados como el intercambio de archivos, el respaldo de la información, la mensajería, el entretenimiento, etc.

El capítulo se divide en tres secciones, en la sección 2.1 se estudian las redes de contenido en general, luego más específicamente las redes de pares (peer-to-peer) en la sección 2.2 y la red de sistemas de nombre de dominios en la sección 2.3.
Para las redes de contenido se presenta una definición formal, las principales características, así como una taxonomía de clasificación según el diseño y arquitectura de la red.
Cuando se estudian las redes de pares, se definen y enmarcan dentro de la generalidad de las redes de contenido, además de ofrecer una taxonomía más específica. De misma forma se estudia el sistema de nombres de dominios.
El capítulo concluye con un resumen de los problemas comunes que enfrentan las redes de contenido.

2.1 Redes de Contenido

2.1.1 Introducción

En [137] se define una ***Red de Contenido*** como una red donde el direccionamiento y el enrutamiento del contenido se basan en la descripción del contenido en lugar de su ubicación
Las redes de contenido son redes virtuales que se montan sobre la infraestructura IP de la Internet o de una red corporativa. Las redes virtuales, como su nombre lo indica, tienen la flexibilidad de crear una topología virtual propia, no importando la red física IP subyacente (es decir, dos nodos conectados en la red virtual pueden estar muy alejados en la red física).

Para acceder al contenido no es necesario mantener continuamente un vínculo fijo entre el contenido y el host donde está alojado. Más aún, para muchas redes de contenido, en un momento dado, parte del contenido puede ser alojado, movido o replicado en algún otro nodo de la red, con el fin de mejorar la disponibilidad, tiempo de acceso, anonimato, etc.

Estas características, sumado a una libertad de diseño producto del continuo desarrollo en el que se encuentran sus protocolos y arquitecturas, representan un atractivo muy importante para nuevas aplicaciones con requerimientos específicos en cuanto a confiabilidad, disponibilidad, redundancia, performance, tiempo de acceso, anonimato, seguridad, etc.

El capítulo sigue con una introducción a las redes de contenido según su utilización, desde un punto de vista histórico. A posteriori se especifican las bases conceptuales, una definición para las redes de contenido y los nodos que la componen. La sección siguiente presenta las principales características de las redes de contenido, por un lado aquellas referentes a la arquitectura, y por otro las que tienen en cuenta el comportamiento de la red. Finalmente se presenta una taxonomía de las redes de contenido (en base a las características más relevantes para la especificación del diseño y arquitectura de este tipo de redes), y discute cada categoría de esta taxonomía, presentando diversos ejemplos de aplicaciones existentes, extraídas de la literatura.

2.1.2 Fundamentos y Antecedentes

Antecedentes

El desarrollo de redes de contenido [137] no es nuevo, la misma Internet puede verse bajo este paradigma como una red de contenido, donde el contenido son las direcciones de red IP. En sus comienzos la Internet surge como una red de pares (peer-to-peer), donde cada nodo cooperaba en el enrutamiento de los paquetes que se realizaba a través de ellos mismos, sin embargo luego de enfrentar un crecimiento inicial, la misma perdió el diseño original de redes de pares por un diseño jerarquizado. Este diseño jerarquizado, consolidó en la actualidad una estructura cliente-servidor, donde unos pocos nodos servidores (routers) se encargan de ofrecer servicio de enrutamiento a la mayoría de nodos que son clientes. El paradigma cliente-servidor se extrapoló a las aplicaciones, donde los servicios más usados de Internet, como ser el Web, e-mail, etc. funcionan actualmente.
Sin embargo, múltiples factores desafían al paradigma cliente-servidor en la actualidad, resaltándose la gran disponibilidad de recursos a compartir por los extremos de la red.

Esta nueva realidad, ha permito proliferar una serie de redes de contenido, en especial las redes de pares (peer-to-peer), donde por ejemplo en Uruguay, el principal proveedor de acceso de banda ancha a Internet, Antel[2], tiene un consumo residencial cercano a un 10% de sus enlaces exclusivamente en señalización de las más exitosas redes peer-to-peer.

Actualmente existen gran cantidad de redes de contenido, incluyendo redes peer-to-peer, cooperative Web caching, content delivery network, subscribe-publish networks, content-based sensor networks, backup networks, distributed computing, collaboratives networks, instant messagings, multiplayers games y search engines.

Las redes de contenido, ahora pensadas como redes virtuales montadas sobre la infraestructura existente de Internet, son sumamente atrayentes dada su flexibilidad a adecuarse a los requerimientos de cada aplicación específica en cuanto a confiabilidad, performance, anonimato, seguridad, etc. sin embargo, en general no contemplan al armar sus topologías la red física de Internet (es decir dos nodos conectados en la red virtual pueden estar muy alejados en la red física) provocando mayor retardo a los clientes y un fuerte impacto sobre la lógica de negocios de los proveedores de Internet [187][188][200][205].

Motivación y Utilidad

Las motivaciones para construir redes de contenido son muy variadas; básicamente permiten una adaptabilidad muy grande a las necesidades particulares de nuevas aplicaciones con requerimientos específicos en cuanto a confiabilidad, disponibilidad, redundancia, performance, tiempo de acceso, anonimato, seguridad, etc. Además de ser desarrolladas bajo las nuevas tendencias de la Informática (como ser tecnologías de comunicación basadas en XML, conexiones transparentes a través de proxies, etc.).

La gran flexibilidad en las redes de contenido es posible gracias a que:

- Los clientes presentan muchos recursos a compartir (procesamiento, almacenamiento, transmisión) con computadores modernos muy potentes; logrando mayor disponibilidad de contenido, entrega más rápida y distribución más eficiente.
- El consumo del ancho de banda ha aumentado enormemente, gracias al incentivo por parte de los proveedores de Internet (ISPs) y sus políticas de tarifas planas.
- Dada la reciente aparición de la mayoría de las redes de contenido actualmente utilizadas, en su diseño ha sido posible aplicar diversas técnicas informáticas modernas, en especial la estandarización y las técnicas de Interoperabilidad (por ejemplo el uso de Web-Services [236]).
- Las redes de contenido son redes virtuales, por tanto se puede adaptar su topología y en particular la cantidad y tipo de conectividad de sus nodos, logrando mayor o menor redundancia y mayor o menor latencia.
- La independencia entre el contenido y la dirección de alojamiento (IP) fija determina menos vulnerabilidad a diversos ataques, en particular los del tipo de denegación de servicio (Denial of Service: DoS).
- Esta misma característica permite soportar nuevos tipos de aplicaciones de cliente donde no se tiene una dirección IP fija, como redes peer-to-peer o computación móvil.
- Se reduce la carga administrativa (operación y mantenimiento) de la red, puesto que cuando se requieren características de alta disponibilidad, balanceo de carga, redundancia y/o defensa contra ataques, estas características están incorporadas nativamente al diseño de la propia red y por tanto no requieren elementos extras de red.
- Desde el punto de vista empresarial, las redes de contenido al utilizar aplicaciones propias del lado de cliente, permiten una fidelización por parte del mismo. Esto es la situación contraria a la de la Web, donde los wrappers son comúnmente utilizados por portales que ofrecen más funcionalidades y servicios sin importar el verdadero proveedor.
- El “efecto de red” [171], que consiste en que la utilidad crece muy rápidamente con el número de usuarios que utilizan un servicio, en este tipo de redes es muy fuerte, y potencializado por la globalización de la Internet, permite que algunos mercados con determinadas exigencias de servicios encuentren natural la utilización de las redes de contenido para satisfacer su demanda.

Características motivadoras para construir redes de contenido:

- ***Disponibilidad de recursos***: almacenamiento, transmisión, procesamiento.
- Mayor ***eficiencia*** en utilización de recursos.
- ***Red virtual*** adecuada a las necesidades.
- Utilización de ***tecnologías modernas***.
- ***Menor vulnerabilidad*** frente a ataques.
- Permiten mantener ***anonimato***.
- Propicias para ***aplicaciones móviles***.
- ***Reducción de costos*** de administración y elementos de red.
- ***Fidelización*** del cliente y ***efecto de red***.

En la actualidad las redes virtuales de contenido son utilizadas en contextos muy variados. Por excelencia las aplicaciones peer-to-peer [19][38][58][64][68][88][127] [166][213] son las más conocidas, presentando una serie de características especiales (además de su popularidad) que las hace motivo de un estudio más detallado en la siguiente sección del capítulo.

Otras aplicaciones de las redes de contenido son: Cooperative Web caching [6][80][122][200][203][222], content delivery network [4][19][37][56][70][130][174] [176], subscribe-publish networks [32][234], content-based sensor networks [69][105][106], backup networks [61][207][208][136], distributed computing [11][74][82][213], collaboratives networks [20][21][100][175], instant messagings [114][121], multiplayers games [60], y search engines [94][241].

Utilización actual de las redes de contenido:

- ***Peer-to-peer.***
- ***Cooperative Web caching.***
- ***Content delivery network.***
- ***Subscribe-publish networks.***
- ***Content-based sensor networks.***
- ***Backup networks.***
- ***Distributed computing.***
- ***Collaborative networks.***
- ***Instant messaging.***
- ***Multiplayers games.***
- ***Search Engines***

Muchas veces en la bibliografía se utilizan los conceptos de redes de contenido y redes peer-to-peer como sinónimos. En realidad es muy difícil delimitar ambos conceptos puesto que no se tiene una definición clara de ninguno de los dos; en este documento se hace un intento de especificar mejor las redes peer-to-peer, brindando una definición y mostrando sus mejores características. Es de esperarse entonces que algunas de las aplicaciones que aquí se detallan como aplicaciones de las redes de contenido en realidad deban incluirse (según la futura definición) dentro del contexto más específico de redes peer-to-peer.

Hasta aquí se ha introducido a las redes de contenido según su utilización, desde un punto de vista histórico, resaltando algunas de las características más atrayentes.
A continuación se realiza una formalización y clasificación de los distintos tipos de redes de contenido según sus atributos o propiedades relevantes a la investigación presentada en este documento (no su utilidad por parte de los usuarios o la lógica de negocio de las empresas que las respaldan), de forma de poder especificar una taxonomía de las mismas.

2.1.3 Bases Conceptuales

Red de Contenido

En este documento se empleará la definición siguiente:

> Una ***Red de Contenido*** (Content Network) es una red donde el descubrimiento, el direccionamiento y el enrutamiento del contenido se basan en la descripción del contenido en lugar de su ubicación.

Esta definición de red de contenido incorpora una pequeña variante a la presentada por H. T. Kung y C. H. Wu en [137]: *"una red de contenido es una red donde el direccionamiento y el enrutamiento del contenido se basan en la descripción del contenido en lugar de su ubicación…"*. La definición original no incorpora de forma explicita el descubrimiento del contenido como parte esencial de una red de contenido. En este trabajo se hace un estudio especial sobre la técnica de descubrimiento de contenido, por considerarla de relevante importancia en la arquitectura de las redes de contenido. Por tanto, buscando enfatizar su importancia, se incorpora a la definición de la red.
En este contexto, el descubrimiento de contenido refiere a la comunicación necesaria entre los nodos de la red para distribuir la información de alojamiento que vincula el contenido con el host que lo posee (la información de alojamiento es llamada meta-información en este documento). En general dos mecanismos de comunicación de meta-información son utilizados: las búsquedas de contenido iniciadas por los solicitantes y las publicaciones de contenido iniciadas por los nodos que alojan contenido.

Las redes de contenido son ***redes virtuales*** que se montan sobre la infraestructura IP de la Internet o de una red corporativa. Las redes virtuales, como su nombre lo indica, tienen la flexibilidad de crear una topología virtual propia, no importando la red física IP subyacente (es decir, dos nodos conectados en la red virtual pueden estar muy alejados en la red física).

> Una red de contenido es una ***Red Virtual*** que se monta sobre la infraestructura IP.

Nodos de Contenido

Los nodos que pertenecen a dicha red virtual son llamados ***nodos de red*** o nodos de contenido. En base a su funcionalidad, pueden clasificarse básicamente en tres tipos: *fuentes de contenido*, que tienen algún tipo de contenido puesto a disposición para el resto de la red, *solicitante de contenido*, que desean obtener algún contenido en la red; y *enrutadores de contenido*, que intercambian mensajes de control con otros nodos de la red (de cualquier tipo).

Esta división es conceptual, puesto que un mismo host físico puede corresponder a varios nodos de contenido de diverso tipo.

Las funcionalidades básicas de un nodo *fuentes de contenido* es la entrega de contenido a sus solicitantes y la publicación de su contenido mediante descriptores en mensajes de control. Por otro lado el *solicitante de contenido* envía mensajes de control solicitando algún contenido en base a un descriptor. Los *enrutadores de contenido* se encargan de circular de la manera más inteligente posible los mensajes de control, dirigiendo el descubrimiento del contenido dentro de la red de manera de lograr que los solicitantes encuentren fuentes que puedan responder a sus pedidos.

En algunos tipos de redes de contenido existen algunos casos especializados de nodos que vale la pena tener presente en esta etapa inicial:

- *Crawlers*, son enrutadores que tienen como único fin descubrir la topología virtual de la red, para esto utilizan distintos mensajes de control.
- *Control Caches*, algunos enrutadores tienen la capacidad de almacenar mensajes de control, guardando una correspondencia temporal entre fuentes y contenido con el fin de minimizar la carga de la red y mejorar las búsquedas de contenido.
- *Relays,* los nodos relay son un tipo especial de enrutadores, capaces de guardar los mensajes de control destinados a fuentes que temporalmente no pertenecen a la red, hasta que dicha fuente reaparezca.
- *Content Caches*, a diferencia de los anteriores, algunos enrutadores (y fuente al mismo tiempo) pueden guardar una copia local del contenido, con el fin de disminuir los tiempos de entrega de contenidos que percibe como muy solicitados.
- *Content Replicators*, similar a los anteriores, algunas fuentes pueden guardar una copia de algún contenido, con el fin de aumentar la disponibilidad de contenidos que se perciben potencialmente propensos a fallas (como desaparición).
- *Content Migrators*, como extensión y mejora a los Content Caches, una funcionalidad especial de algunos enrutadores es migrar o replicar el contenido de una fuente en otras fuentes (por supuesto que esto es aplicable solo para algún tipo especial de redes de contenido, donde las fuentes permiten colocación externa de contenido).
- *Gateways*, tipo particular de enrutador con la capacidad de traducir a otro protocolo de control los mensajes de la red de contenido.
- *Proxies*, juntando funcionalidades de fuente y enrutador tienen la posibilidad de mostrarse como fuente del contenido que se enruta a través de él.

Un ***Nodo*** de la red de contenido es de alguno de los siguientes tipos, según su funcionalidad:

- ***Fuente de contenido*** (content source): entrega (delivery) y publicación (registration) de contenido.
- ***Solicitante de contenido*** (content requestor): solicitante (request) de contenido.
- ***Enrutadores de contenido*** (router): enrutar mensajes de control.
- *Casos particulares*: ***Crawlers, Control Caches, Relays, Content Caches, Content Replicators, Content Migrator, Gateways y Proxies.***

2.1.4 Características Generales

En esta sección se consideran un conjunto bastante amplio de características distintivas de las redes de contenido, basándose principalmente en el análisis de los trabajos de H. T. Kung y C. H. Wu [137], y D. S. Milojicic et.al. [162], complementados con una extensa bibliografía adicional.
Muchos de estos aspectos se retoman en la siguiente sección, cuando se estudian en profundidad las redes de pares (ver sección 2.2 de redes Peer-to-Peer).

Las diversas características abordadas en la literatura se agrupan en dos grandes conjuntos: por un lado las características ligadas a la arquitectura de las redes de contenido, y por otro las características ligadas a su comportamiento. El primer conjunto se refiere a la forma de construcción e implementación de las redes, abarcando las propiedades siguientes: Descentralización, Autonomía, Agrupación lógica del Contenido, Colocación física del Contenido, Heterogeneidad, Auto-Organización, Interoperabilidad. El segundo conjunto en cambio se refiere a aspectos del funcionamiento tal como percibido por el usuario y abarca las siguientes características: Anonimato, Performance, Escalabilidad, Transparencia y Usabilidad, Seguridad, Robustez y Resistencia a Fallas. En muchos casos, decisiones ligadas a la arquitectura tienen efecto directo en las características ligadas al comportamiento; es importante tenerlas en cuenta, ya que desde el punto de vista del usuario, suelen ser las más importantes.

Las características consideradas son:

Características Generales de las redes de contenido:

- **Relacionadas con la Arquitectura:**
  - ***Descentralización.***
  - ***Agrupación lógica del Contenido.***
  - ***Colocación física del Contenido.***
  - ***Autonomía.***
  - ***Heterogeneidad.***
  - ***Auto-Organización.***
  - ***Interoperabilidad.***
- **Relacionadas con el Comportamiento:**
  - ***Anonimato.***
  - ***Performance.***
  - ***Escalabilidad.***
  - ***Transparencia y Usabilidad.***
  - ***Seguridad.***
  - ***Robustez y Resistencia a Fallas.***

A continuación se define cada uno de estos aspectos de las redes de contenido, discutiendo brevemente los posibles casos que se presentan comúnmente.

Características relacionadas con la arquitectura

Descentralización [149][162][209]

En una red de contenido la descentralización se encuentra determinada por la capacidad que tiene la red de funcionar con varios nodos de un mismo tipo (fuentes, solicitantes y enrutadores). Los principales modelos de descentralización son el modelo cliente-servidor, el de distribución jerarquizada, y el completamente distribuido.

En el modelo tradicional de ***cliente-servidor***, la información se encuentra localizada en servidores centralizados y distribuida a través de la red a los clientes finales. De esta forma se facilita mucho el protocolo de comunicación y descubrimiento del contenido, pero aparecen serios problemas de escalabilidad a nivel del servidor central, resultando en un costo importante de su mantenimiento.
En general, la información de control es la que se encuentra centralizada (es decir, la red funciona con un único enrutador) y el contenido se encuentra distribuido entre una cantidad arbitraria de fuentes. Ejemplos de este modelo son Napster[166][167] (un único nodo enrutador), Seti@home[213] (un único nodo enrutador/solicitante), Avaki [11] (un único nodo enrutador).

En un sistema ***completamente distribuido*** cada nodo tiene la misma participación en la red, ofreciendo potencialmente la misma cantidad de contenido (es decir, existen una cantidad arbitraria de nodos fuentes, solicitantes y enrutadores).
De esta forma se reduce el costo de recursos y mantenimiento asociado al escalado de la red, puesto que no se necesita un crecimiento vertical en ningún servidor central. Pero este tipo de redes tienen que resolver el problema de la vista global; al presentarse el contenido de forma distribuida es necesario consolidan dicho contenido para ofrecérselo a los nodos solicitantes, esto trae aparejado la utilización de protocolos de comunicación más complejos y mayor utilización de la red subyacente (alto ancho de banda), que también puede representar un problema de escalamiento al crecer el número de participantes.
Para resolver el problema de la vista global, básicamente existen dos filosofías posibles (que pueden ser combinadas):

- La más comúnmente usada, pero que presenta mayor consumo de ancho de banda (y por tanto mayor limitante al escalamiento), es el ***broadcast***, donde cada nodo solicitante inunda la red solicitando el contenido deseado.
- Otra alternativa es que los nodos fuentes se encarguen de mantener informados a los nodos enrutadores, para que cuando algún contenido sea solicitado, sepan donde se encuentra. Esta alternativa presenta múltiples formas de implementarse, destacándose la utilización de ***tablas de hash distribuidos*** [202] o técnicas de ordenamiento estructurado (***auto-ordenamiento***) [9][12][137].

Ejemplos de sistemas completamente distribuidos son Gnutella [88] (y sus variantes Limewire[141], BearShare[15], Gnucleus[86]), Morpheus[79], Freenet[38][79] (todos son nodos fuente/enrutador/solicitantes). Algunos de estos ejemplos serán descriptos más adelante, mostrando en detalle el funcionamiento típico de los tres estilos de redes completamente distribuidas (broadcast, tabla de hash distribuida y auto-ordenamiento).

Entre los dos modelos antagónicos anteriores (cliente-servidor y completamente distribuido) existe una serie muy variada de modelos con ***distribución jerarquizada***, donde la idea es agrupar en niveles según la capacidad de contenidos de cada participante. Esta variante es la de mejor escalamiento, pues distribuye mejor la carga tanto de procesamiento como de consumo de ancho de banda (debe recordarse que

las redes cliente-servidor presentan una limitante en escalamiento debido al importante costo de su servidor central, mientras que las redes completamente distribuidas presentan una limitante debido al consumo exponencial de ancho de banda necesario por todos los nodos para participar en la red).
Algunos ejemplos son DNS[157][158], KaZaA[127], iMesh[115], Grokster[101], Groove[100], Jabber[121], OpenNap[177] (todos presentan una cantidad arbitraria de nodos fuente y solicitantes, y una cantidad proporcionalmente mucho menor de enrutadores).

La ***descentralización*** se encuentra determinada por la capacidad que tiene la red de funcionar con varios nodos de un mismo tipo (fuentes, solicitantes y enrutadores).

Modelos de Descentralización en las redes de contenido:

- ***Cliente-Servidor.***
- ***Distribución Jerarquizada.***
- ***Completamente Distribuido (broadcast, distribución de índices de hash, auto-ordenamiento, etc.).***

La descentralización es quizás una de las características más importantes de las redes de contenido. Puesto que el verdadero valor de la red no se encuentra en la capacidad de contenido de un único componente central, sino en la posibilidad que tiene la red de utilizar los recursos de los bordes. Esto potencializado por un buen "efecto de red" [171], permite una reducción de ***costos de propiedad*** significativo y mejor servicio a los participantes.

Agrupación lógica del Contenido [137]

Otra característica, que está muy ligada al modelo de descentralización de la red, es la agrupación lógica del contenido. Por agrupación lógica del contenido entendemos la estrategia para asignar los contenidos individuales dentro de grupos de contenido.

Se pueden identificar tres etapas separadas en el proceso de agrupación:

- ***Mapeo***: Un primer paso, desarrollado en general por los nodos fuente, es clasificar el contenido en valores de un espacio de contenido bien definido (distintas estrategias son utilizadas dependiendo del contenido, existe mucho trabajo realizado para cuando el contenido son archivos).
- ***Agrupamiento***: Un segundo paso, en general desarrollado por los nodos enrutadores, es mantener lógicamente agrupados en la red los contenidos que presentan valores similares (o idénticos).
- ***Consolidación***: En la mayoría de las redes de contenido se explota la existencia de múltiples fuentes del mismo contenido, ofreciéndole todas las fuentes posibles al solicitante. Si este es el caso, el solicitante debe consolidar el contenido brindado por todas las fuentes, para tener una vista única del mismo.

Existen dos filosofías bien distintas respecto a las variantes de agrupación lógica. La agrupación puede ser semántica o sintáctica según como sea el espacio de contenido al que se mapea en el primer paso:

- ***Semántica***: El espacio de contenido es una taxonomía, donde los valores tienen significado. Este tipo de redes permiten una búsqueda por proximidad semántica (es decir se puede buscar contenido parecido al solicitado) lo cual puede resultar muy útil para cumplir el objetivo de la red de ofrecer contenido relevante a los nodos participantes.

- ***Sintáctica*** En este caso el espacio de contenido no tiene ningún significado, por ejemplo las redes basadas en tablas de hash distribuidas simplemente transforman cualquier archivo en un valor numérico, por tanto se desconoce la semejanza semántica entre valores, impidiéndose la búsqueda por proximidad. Una ventaja de este enfoque es que se obtienen en general sistemas más escalables debido a que no existen criterios o limitaciones de diseño externas para la agrupación del contenido (en estas redes se tienen problemas de escalado en la implementación de búsquedas semánticas de contenido).

Siguiendo la nomenclatura impuesta por H. T. Kung y C. H. Wu en [137], se dice que una red de contenido es semántica o sintáctica según como se realice su agrupación lógica. Asimismo hay redes particulares donde no es necesario ningún tipo de agrupación lógica, estas se clasificaran como sintácticas.

Por ***agrupación*** lógica del contenido debe entenderse la estrategia utilizada para obtener una vista global del contenido.
Se dice que una red de contenido es semántica o sintáctica según como se realice su agrupación lógica.

La ***agrupación*** en las redes de contenido puede ser:

- ***Semántica.***
- ***Sintáctica.***

La agrupación lógica determina las posibilidades de la red de ofrecer una vista global del contenido. En redes con servidores centrales es relativamente sencillo obtener esta vista global. En cambio, en las redes descentralizadas (no cliente-servidor), la misma es el resultado del conjunto de los esfuerzos individuales de los nodos para mantener el conocimiento del contenido existente.

Colocación física del Contenido [137]

Dependiendo de la naturaleza del contenido y de la arquitectura de la red, hay situaciones en que el contenido puede migrarse o replicarse entre fuentes para lograr una mejor performance y/o mayor anonimato. La característica de colocación física del contenido determina el método utilizado para elegir el nodo fuente donde debe ser alojado un contenido. Dos grandes posibilidades se presentan:

- ***Colocación Sensible al Contenido***: En este caso se elige el nodo fuente de un contenido específico en función del contenido en si mismo. Este método facilita enormemente la tarea a los nodos enrutadores, ya que a partir de la descripción del contenido buscado, es posible encontrar la ubicación del mismo.
- ***Colocación Insensible al Contenido***: Cada contenido es colocado sin tomar en cuenta que contenido es en si mismo. En este método las fuentes deben periódicamente anunciar su contenido o los nodos enrutadores inundar la red en busca del mismo, aumentando el consumo de recursos de transmisión. Este método funciona mejor cuando el modelo de distribución es centralizado (cliente-servidor).

Se dice que una red de contenido es sensible o insensible según como se realice la colocación física del contenido. Asimismo, hay redes donde no es posible modificar la colocación física de contenido, por tanto estas redes son siempre insensibles al contenido.

Con el fin de diferenciar los conceptos de agrupamiento lógico y colocación física es importante hacer notar que ambos tipos de agrupamientos (semántico y sintáctico) pueden utilizar una colocación física sensible o insensible.
Por ejemplo, el sistema de DNS[157][158] presenta una agrupación semántica (jerarquía arborescente por nombre de dominio) y la colocación del contenido es sensible a dicha jerarquía dado que cada nombre debe ser colocado en el servidor que tiene asignado ese dominio en la jerarquía. Por otro lado las redes basadas en tabla de hash distribuidas donde el contenido es colocado en los nodos según los valores de hash, como OceanStore (Tapestry)[136] o Past (Pastry)[61][207][208], son un ejemplo de redes sintácticas sensibles.
Respecto a redes insensibles los ejemplos más claros son Gnutella[88] y TRIAD[99]. Gnutella es una red P2P sin agrupación lógica, ni colocación física del contenido, mientras que TRIAD es una propuesta para un nuevo protocolo de red, basado en contenidos, que sustituya al método actual basado en enrutamiento IP, búsqueda de hosts por DNS y recuperación de contenidos vía HTTP. Este protocolo permite una red donde el contenido es mapeado al espacio (con significado) de las URL. Una agrupación lógica se realiza según el dominio de las URL (semántica), pero no hay colocación física porque el contenido no debe ubicarse arbitrariamente en ninguna fuente en particular.

La ***colocación*** física del contenido determina el método utilizado para elegir el nodo fuente donde debe ser alojado un contenido.
Se dice que una red de contenido es sensible o insensible según como se realice la colocación física del contenido.

La ***colocación*** en las redes de contenido puede ser:

- ***Sensible al Contenido.***
- ***Insensible al Contenido.***

Como se mencionó anteriormente, la colocación física de contenido no siempre es posible o conveniente, en particular, las fuentes pueden no permitir alojar un contenido arbitrario puesto que es una forma de violar su propia autonomía de participación (la autonomía se discute a continuación).

Autonomía [22][65]

La autonomía en una red de contenido está vista como la capacidad de "independencia" de los nodos participantes.

Según los objetivos que tiene la red de contenido y lo que se le "ofrece" a los nodos participantes en cada caso, existe una mayor o menor autonomía por parte de los nodos. En los estudios clásicos de interoperabilidad [22][65] se definen cuatro tipos de autonomía para un sistema formado por varias componentes:

- ***Autonomía de Diseño***: Libertad en la elección de su propio: modelo de datos, lenguaje de consulta, implementación, restricciones, gerenciamiento de datos a ser usado, funcionalidades soportadas, etc.
- ***Autonomía de Comunicación***: Capacidad de los componentes de decidir cuándo y cómo responder a requisitos de otros componentes.
- ***Autonomía de Ejecución***: Cada componente puede ejecutar sus operaciones sin interferir en la ejecución de operaciones no locales.
- ***Autonomía de Participación***: Capacidad de los componentes de decidir cuánto de sus funciones, operaciones y datos compartir.

Para las redes de contenido, la autonomía de participación es la más importante, y que en general se encuentra presente, en este contexto significa que un nodo puede libremente elegir cuando pertenecer o no pertenecer a una red de contenido[1] y para el caso de las fuentes que y cuanto contenido ofrecer[2].
Cuando la red determina que contenido debe alojar una fuente entonces tenemos colocación de contenido, en este caso se limita fuertemente la autonomía de participación del nodo fuente, provocando que este tipo de redes sean de poco atractivo.

Las autonomías de comunicación y ejecución son prácticamente un requerimiento dado la heterogeneidad de recursos de los nodos participantes.
Muchas de las redes de contenido presentan protocolos de comunicación abiertos y conocidos (otras son propietarios), en ellas existe libertad de diseño excepto en el protocolo.

La ***autonomía*** en una red de contenido está vista como la capacidad de "independencia" de los nodos participantes.

Autonomía en las redes de contenido:

- ***Autonomía de Diseño.***
- ***Autonomía de Comunicación.***
- ***Autonomía de Ejecución.***
- ***Autonomía de Participación.***

Dado que una red de contenido es una red virtual que se monta sobre la infraestructura IP existente, la misma utiliza los recursos de los nodos participantes según las características de cada red. Los recursos que presentan los nodos son de tres tipos: procesamiento, almacenamiento y transmisión.
Teniendo en cuenta que el contenido que tienen para ofrecer los nodos fuentes se clasifican dentro de algunos de sus recursos (procesamiento, almacenamiento o transmisión), entonces la autonomía de comunicación, ejecución y participación pueden verse simplemente como la autonomía de un nodo fuente a decidir que y cuanto contenido ofrecer, es decir su grado de ***compartir***.

La autonomía es un requerimiento muy importante en algunos de los tipos de redes de contenido mencionadas, especialmente para las redes de pares (peer-to-peer) que presentan siempre una completa autonomía de participación. Se vuelve sobre este punto cuando se estudian en detalle las redes de pares (ver sección siguiente: Redes de Pares).

Heterogeneidad [22][65][162]

La heterogeneidad puede presentarse a tres niveles, según la teoría de interoperabilidad entre componentes [22][65]:

[1] la posibilidad de conexión/desconexión en las redes de contenido es conocida como **Conectividad Temporal**.
[2] Por supuesto que existen casos particulares, por ejemplo el sistema de DNS es una red de contenido donde los nodos fuentes estan obligados a siempre estar conectados. Otro caso bastante frecuente es, que en pro de una mayor performance, la red de contenido no permite a los nodos fuente manejar completamente su propio contenido (por ejemplo FreeNet).

- ***Heterogeneidad de sistemas***: Los nodos presentan diferentes plataformas de hardware y software, diferentes sistemas operativos y/o diferentes protocolos de comunicación.
- ***Heterogeneidad sintáctica***: Diferentes modelos de datos del contenido de en cada nodo.
- ***Heterogeneidad semántica***: Mismos contenidos en distintos nodos representados diferente en cada uno.

En las redes de contenido, es común encontrar heterogeneidad de sistemas, donde los nodos presentan distinta capacidad de brindar sus recursos, además de distinto hardware, sistema operativo y versión del software de red de contenido. Esta última posibilidad (distintas versiones de software) es muy frecuente, en cuyo caso puede aparecer también heterogeneidad sintáctica o semántica.

Heterogeneidad en las redes de contenido:

- ***Heterogeneidad de Sistemas.***
- ***Heterogeneidad Sintáctica.***
- ***Heterogeneidad Semántica.***

Auto-Organización [9][107][162]

En cibernética un sistema auto-organizado es aquel que tiene la habilidad de organizar sus componentes en un framework de trabajo sin la necesidad de control externo [107].
Las redes de contenido presentan un extremo dinamismo en escala, topología, contenido y carga, por tanto, algunas propiedades de auto-ordenamiento son muy deseadas, sino necesarias, para tener buena performance y escalamiento.

Dependiendo del tipo de descentralización utilizado por la red, es posible alcanzar distintos niveles de auto-ordenamiento:

- En los sistemas centralizados el problema es fácil de abordar, buscando soluciones en el sistema central.
- En los sistemas completamente distribuidos la solución no es sencilla y distintas técnicas son utilizadas, resaltándose principalmente la utilización de hipercubos para la topología y búsqueda de contenido; replicación y migración de contenido; protocolos de conexión/desconexión de nodos, etc. Ejemplos son Past (Pastry)[61][207][208], CAN[201], Chord[225], OceanStore (Tapestry)[136], etc.
- En los sistemas jerarquizados, la propia jerarquía impone un nivel de organización. Además se utilizan muchas veces algunos de los métodos de las redes centralizadas o completamente distribuidas para lograr mayor ordenamiento. Un ejemplo son las redes SuperPeer como KaZaA[127].

Es necesario resaltar que para los sistemas distribuidos no existe una vista global centralizada del sistema, y por tanto el auto-ordenamiento solo puede ser alcanzado mediante criterios locales (de auto-ordenamiento). Un estudio detallado sobre la ***localidad del auto-ordenamiento*** se encuentra en [9].

Un ***sistema auto-organizado*** es aquel que tiene la habilidad de organizar sus componentes en un framework de trabajo sin la necesidad de control externo.
Dependiendo del tipo de descentralización utilizado por la red, es posible alcanzar distintos niveles de auto-ordenamiento. Para los sistemas distribuidos no existe una vista global del sistema, y por tanto la auto-ordenación solo puede ser alcanzada mediante ***criterios locales de auto-ordenamiento***.

Algunas otras características derivan (o están fuertemente asociadas) a la auto-organización, por ejemplo: auto-configuración, auto-mantenimiento, auto-reparación, auto-estabilización, etc. En este documento no se profundiza al respecto.

Interoperabilidad [22][65][162]

El concepto de Interoperabilidad clásico [22][65] refiere al acceso uniforme a múltiples fuentes de datos heterogéneas y autónomas. La intención de la interoperación es intercambiar datos y funcionalidades cooperando con un fin común. Los problemas básicos que se enfrentan cuando se realiza un sistema integrado es resolver la heterogeneidad de las fuentes, respetar la autonomía de las mismas, y diseñar de forma que el sistema sea gerenciable y mantenible frente a cambios en la cantidad de las fuentes (escalabilidad) o cambios en las estructuras de las fuentes (evolución). La arquitectura de un sistema interoperable debe entonces, adaptarse a la naturaleza de las fuentes y los objetivos específicos de integración en cada caso.

Las redes de contenido pueden verse como sistemas interoperables en si mismos, montado sobre la infraestructura IP, en general de gran escala, donde los sistemas a integrar son cada nodo y los datos intercambiados son contenidos de distinto tipo (recursos como procesamiento, almacenamiento y transmisión).

Al momento de ser escrito este documento existe una gran explosión de redes de contenido, donde los usuarios que quieren pertenecer a determinada red deben instalarse el software que los habilite a ser algún nodo de la misma. Cuando esta situación madure, comenzará a ser muy útil la creación de software como sistemas integrados de distintas redes de contenido. Los problemas que tendrán que resolver estos futuros softwares de integración son justamente los planteados por la interoperabilidad.
Se considera que la capacidad de una red de contenido para interoperar con otras redes va a ser un factor fundamental para la permanencia de la misma en un futuro.

Distintos métodos son utilizados tradicionalmente para alcanzar sistemas interoperables:

- ***Estándares***: RFCs, IEEE, ANSI, etc.
- ***Código abierto***: GNU, Linux, etc.
- ***Estándares de facto***: Windows, Java, etc.
- ***Especificaciones en común***: ODMG, OMG, etc.

En el nuevo mundo de las redes de contenido poco se ha hecho, resaltándose:

- ***Estándares***: SOAP, Gnutella, Jabber, etc.
- ***Código abierto***: JXTA, eMule, OpenNap, Jabber, etc.
- ***Estándares de facto***: JXTA, .NET, etc.
- ***Especificaciones en común***: Peer-to-Peer Working Group, Peer-to-Peer Working Group at Internet2, O'Reilly P2P, etc.

El concepto de ***Interoperabilidad*** clásico refiere al acceso uniforme a múltiples fuentes de datos heterogéneas y autónomas. Una red de contenido es en si misma un sistema interoperable entre nodos.
La ***interoperabilidad*** entre distintas redes de contenido puede alcanzarse con:

- ***Estándares.***
- ***Código abierto.***
- ***Estándares de facto.***
- ***Especificaciones en común.***

Características relacionadas con el comportamiento

Anonimato [57][162]

Muy relacionado con el concepto de autonomía, aparece el anonimato. Un atractivo de algunas redes de contenido es brindarle a los nodos participantes cierto nivel de anonimato, evadiendo problemas legales (debido a posibles afectaciones al derecho de autor del contenido ofrecido), una potencial censura (contenido que se considere no adecuado) y otras ramificaciones (por ejemplo utilizar recursos del trabajo para juegos multi-jugador).

En Freehaven [57] se identifican las siguientes clases de anonimato:

- ***Anonimato de Autor***: El autor o creador del contenido no puede ser identificado.
- ***Anonimato de Editor***: La persona que publica el documento en la red no puede ser identificado.
- ***Anonimato de Lectura***: Las personas que leen o consumen información no pueden ser identificados.
- ***Anonimato de Servidor***: Los servidores de un contenido no pueden ser identificados a partir del contenido.
- ***Anonimato de Información***: Los servidores no saben realmente cual contenido están sirviendo.
- ***Anonimato de Consulta***: Los servidores no saben que contenido tienen que responder a una consulta determinada.

Aplicando estas clases de anonimato al modelo presentado de redes de contenido se obtiene:

- ***Anonimato de Fuente***: No se conoce las fuentes de un contenido dado.
- ***Anonimato de Editor***: No se conoce quien introdujo el contenido a la red. A veces puede ser aceptable conocer las fuentes, pero no la primera fuente.
- ***Anonimato de Solicitante***: No se conoce realmente quien solicita los contenidos.
- ***Anonimato de Entrega***: No se conoce a quien se le entrega determinado contenido (en muchos sistemas se pueden hacer las consultas anónimas, pero si se quiere obtener el contenido es necesario identificarse).
- ***Anonimato de Contenido***: Las fuentes no saben exactamente que contenidos poseen (solo es posible en redes con baja autonomía de fuente de contenido).

Distintas técnicas son utilizadas por las redes de contenido para mejorar alguno de los aspectos del anonimato:

- ***Puertos Dinámicos***: La mayoría de las redes de contenido utilizan puertos TCP/IP dinámicos, de esta forma no es posible identificar con los métodos tradicionales que poseen los administradores de redes y los ISPs quien y cuanto se utiliza una red de contenido. Esta técnica mejora (en la actualidad) la mayoría de los anonimatos.
- ***Comunicación Encriptada***: Muchos de los protocolos de comunicación entre nodos de una red de contenido presenta algún nivel de encriptación. Considerando que se poseen herramientas más potentes para la administración de la red, del tipo *sniffers*, es posible identificar quien y cuanto utiliza una red de contenido, pero no se sabe qué contenido se solicita u ofrece.
- ***Multicast***: Los nodos solicitantes forman grupos, los nodos fuente le envían el contenido al grupo. Por tanto no se conoce con exactitud cuales de los nodos solicitantes realmente solicitaron o están recibiendo realmente el contenido (anonimato de solicitud y entrega). Esto solo se puede hacer cuando la red IP subyacente acepta multicast, lo cual es poco común en la actualidad. El grupo de trabajo en P2P de Internet2 trabaja para que la nueva Internet posea estas características[193].
- ***Spoofing de Solicitante***: Es posible cambiar la identidad del solicitante de un contenido mediante el cambio de su identificación en la red. Este es un método muy básico de anonimato, puesto que siempre se puede conocer la dirección IP del solicitante, excepto cuando se está por detrás de un *proxy* donde este método es interesante para no saber de que cliente interno se produjo la solicitud. (anonimato de solicitante).
- ***Alojamiento No Voluntario***: para poder cambiar la identidad del nodo que recibe el contenido, la red tiene que permitir que otros nodos alojen temporalmente el contenido (violando la autonomía de ellos), de esta forma se tiene que el contenido pasa por una cantidad arbitraria de nodos antes de llegar al solicitante, siendo prácticamente imposible su detección (anonimato de entrega). Ejemplos son CAN[201], Past[61][207][208], Chord[225], etc.
- ***Covert Paths***: Muchas redes permiten que los nodos enrutadores mantengan una tabla entre solicitantes y solicitudes de contenido, donde al momento de retrasmitir una solicitud cambien aleatoriamente el solicitante, por tanto para conocer exactamente quien realiza una solicitud es necesario conocer el camino de todos los enrutadores intermedios (anonimato de solicitante). Este método también es útil para obtener anonimato de fuente, donde el contenido es enviado a nodos *proxies* intermedios (anonimato de fuente). Distintos grados de anonimato se pueden alcanzar si se cambia con frecuencia el camino. Este método es utilizado en redes como JAP[122], Mix[34], Anonymizing Proxy[80], Crowds[203], Herdes[214], Onion[227], etc.

Anonimato en las redes de contenido:

- ***Anonimato de Fuente.***
- ***Anonimato de Editor.***
- ***Anonimato de Solicitante.***
- ***Anonimato de Entrega.***
- ***Anonimato de Contenido.***

Performance [149][162]

La performance esta dada por la efectividad y eficiencia con que son utilizados los recursos de los nodos de la red.

Dependiendo de la arquitectura de la red se tiene que la *efectividad* puede ser la probabilidad de encontrar las fuentes de determinado contenido, la precisión del enrutamiento, etc.; por otro lado la *eficiencia* está dada por el consumo de ancho de banda, la latencia en el acceso o consulta del contenido, la cantidad de replicas necesarias, la carga del procesador, etc.

En general los nodos especiales (crawlers, control caches, relays, content caches, content replicators, content migrator, gateways y proxies) descritos anteriormente se presentan en las redes de contenido para mejorar algún aspecto de la performance.

La ***performance*** esta dada por la efectividad y eficiencia con que son utilizados los recursos (procesamiento, almacenamiento y transmisión) de los nodos de la red.

Escalabilidad [162]

La escalabilidad de la red de contenido representa la capacidad que tiene la red de funcionar, manteniendo una performance aceptable, frente a un crecimiento en la cantidad de nodos (fuentes, solicitantes y enrutadores).

La importancia de la escalabilidad de una red de contenido depende de los objetivos de utilización de la red. Por ejemplo si la red de contenido es una red P2P para trabajo cooperativo dentro de una institución no tiene sentido pensar en un escalamiento más allá del tamaño de la empresa (en general a lo sumo unos miles de nodos). Por el contrario si la red de contenido es una red para intercambiar archivos en Internet, es importante que presente un excelente escalado a cientos de millones de nodos, puesto que de esta forma se maximiza el efecto de red y la probabilidad de éxito de la red.

La escalabilidad se encuentra determinada por las características de arquitectura de la red, principalmente por el modelo de descentralización, pero también de otras características como el tipo de colocación física de contenido y la capacidad de auto-ordenamiento. En un sistema centralizado (con un único nodo enrutador) se puede lograr una buena performance y escalabilidad si los nodos fuentes se encuentran distribuidos, este tipo de redes presentan una limitante de escalamiento en la cantidad de mensajes de control que debe manipular el servidor central (Napster [166][167] logró escalar hasta 6 millones de usuarios, cuando por litigios legales de derecho de autor tuvo que cambiar su estrategia y filtrar el contenido ilegal, Seti@home[213] tiene, al momento de escrito este documento, 5 millones de usuarios).

En una red completamente distribuida basada en broadcast la limitante de escalabilidad se presenta en la cantidad de mensajes de control que tienen que manejar los nodos, inclusive aquellos con pocos recursos de la red (como los que se conectan de forma asincrónica). Esto se debe a que en estas redes, la cantidad de mensajes de control crece exponencialmente con la cantidad de participantes, provocando en la práctica que sea costoso pertenecer a la red, además de obtener un alto grado de indeterminismo en las búsquedas. Esto se muestran en los numerosos monitoreos y estudios teóricos realizados sobre la red Gnutella [1][123][205][206][223], la cual se tiene constancia de haber escalado a 11.000 nodos simultáneos.

Una solución de compromiso entre los dos sistemas de distribución antes descritos se puede encontrar en las redes jerarquizadas o en las completamente distribuidas basadas en una estructura de ordenamiento. Estas redes distribuyen mejor la carga tanto de procesamiento como de comunicación entre los participantes según la propia capacidad de cada uno, pudiendo escalar en teoría a miles de millones de usuarios.

Las redes basadas en tablas de hash distribuidas, como Past (Pastry)[61][207][208], OceanStore (Tapestry)[136], Chord[225] y CAN[201] han sido diseñadas para alojar más de 10^{14} archivos.

Algunas variantes de auto-ordenamiento y colocación física del contenido permiten un mayor escalado utilizando básicamente el concepto de acercamiento del contenido a los potenciales nodos solicitantes, de esta forma se pueden lograr importantes mejoras en la cantidad de comunicación de la red y por tanto en su escalado.

La ***escalabilidad*** de la red de contenido, representa la capacidad que tiene la red de funcionar, manteniendo una performance aceptable, frente a un crecimiento en la cantidad de nodos (fuentes, solicitantes y enrutadores).

Transparencia y Usabilidad [29][50][162]

La transparencia en un sistema distribuido es asociada con la habilidad y simplicidad de conectar el sistema distribuido en un sistema local. Un mayor grado de transparencia determina mayor usabilidad, y por tanto mayor probabilidad de éxito de la red.

La transparencia más importante para una red virtual de contenido es la de localización, donde se resuelve una relación entre el contenido y su dirección IP. Muy relacionado con la localización se presenta la transparencia de acceso (que determina la facilidad de un nodo para conectarse a la red) y la de movilidad (que tan fácil es para un nodo cambiar de dirección IP). En [29] y [50] se presentan diversos aspectos en los que es deseable la transparencia:

- ***Localización.***
- ***Acceso.***
- ***Movilidad.***
- ***Concurrencia.***
- ***Replicación.***
- ***Falla.***
- ***Escalamiento.***
- ***Administración.***
- ***De dispositivo.***

La ***transparencia*** en un sistema distribuido es asociada con la habilidad y simplicidad de conectar el sistema distribuido en un sistema local. Un mayor grado de transparencia determina mayor ***usabilidad***.

Seguridad [47][162]

Algunos requerimientos de seguridad son pertinentes a todos los sistemas distribuidos, como ser: encriptación, intercambio de claves, firmas de contenido, firmas de pares, etc.

Algunas técnicas y aspectos a tener en cuenta, de especial relevancia en este contexto son:

- ***Firewalls***: A muchas personas les gustaría pertenecer a una red, pero su conexión es desde una intranet a través de un *firewall*. Es muy difícil que este tipo de aplicaciones funciones a través de un firewall, puesto que presentan conexiones bidireccionales. Distintas alternativas son utilizadas resaltándose los nodos Relays, Gateways y Proxies antes mencionados.

- ***Sandboxing***: Las aplicaciones de redes de contenido que requieren ejecutarse en los nodos participantes son en realidad servidores con alta vulnerabilidad. Para evitar potenciales ataques de virus o hackers es una práctica común utilizar un ambiente restringido o maquina virtual, donde se ejecute de forma segura la aplicación.
- ***Trusting***: Al compartir contenido entre nodos, por lo general anónimos, es difícil evitar la participación de nodos maliciosos que mientan sobre el tipo de contenido ofrecido [47]. El método más utilizado para redes de contenido masivas es utilizar una especie de puntaje penalizando a los malos nodos. Otros métodos por ejemplo pueden ser un chequeo cíclico CRC del contenido cuando se trata de archivos o la utilización de claves.

Robustez y Resistencia a Fallas [24][27][26][25][28][162]

Las fallas en una red de contenido pueden resumirse como la incapacidad del sistema a poder ofrecer a un solicitante algún contenido que ha pertenecido a la red. Una red es más robusta si resiste mejor a las fallas que se le presentan.

Las fallas pueden deberse entonces a:

- ***Una solicitud mal realizada***: los protocolos de comunicación muchas veces son propietarios y cambiantes con nuevas versiones del software que se instala en el nodo solicitante. Es posible que por una falla de software o de versión, las solicitudes se encuentren mal formuladas.
- ***Una contenido inalcanzable***: Debido a fallas en la red IP subyacente, problemas de particionamiento de la red, o simplemente por problemas de saturación debido a escala, los nodos enrutadores no pueden resolver las solicitudes de contenido, es decir, desconocen y no pueden alcanzar un contenido deseado. Frente a una falla de la red subyacente difícilmente se pueda encontrar solución a nivel de red de contenido; distinto es para el resto de las fallas donde múltiples técnicas son abordadas.
- ***Un contenido no disponible***: Un problema bien diferente es cuando el contenido puede ser alcanzado por los nodos enrutadores, pero en el momento de la solicitud el contenido no se encuentra disponible, debido a una desconexión de la fuente o porque dejó de compartir ese contenido. Para solucionar este problema dependiendo de los objetivos de la red se utilizan distintas técnicas: la más común es la redundancia de contenido, que significa que al solicitante se le ofrecen todas las fuentes del contenido solicitado (disminuyendo la probabilidad de falla por disponibilidad), también la replicación o migración de contenido es posible en redes donde se permite la colocación de contenido.

Las ***fallas*** en una red de contenido pueden resumirse como la incapacidad del sistema a poder ofrecer a un solicitante algún contenido que ha pertenecido a la red. Una red es más ***robusta*** si resiste mejor a las fallas que se le presentan.

Las ***fallas*** en las redes de contenido pueden ser:

- ***Solicitud mal realizada.***
- ***Contenido inalcanzable.***
- ***Contenido no disponible.***

Uno de los objetivos de la descentralización es eliminar el punto central de falla. A diferencia de los sistemas centralizados, donde la responsabilidad de las fallas se le adjudica al sistema central, en un sistema descentralizado la responsabilidad radica

de distinta forma en todos los nodos participantes, determinando el éxito de la red aquellas arquitecturas que ofrecen mayor robustez.

2.1.5 Clasificación y Taxonomía

La temática de redes de contenido tiene en la actualidad un gran desarrollo, pero se encuentra en gran medida en una etapa de propuesta de diversas arquitecturas y soluciones técnicas, y recién aparecen los primeros trabajos que apuntan a abstraer las propiedades comunes a las diversas propuestas, y a clasificar las redes de acuerdo a estas propiedades.
Para el caso de las redes peer-to-peer, algunos trabajos que desarrollan taxonomías son [116], [162], [192] y [229] (ver sección 2.2).

Para las redes de contenido en general, H. T. Kung y C. H. Wu [137] proponen una taxonomía, basada en las propiedades de *Agrupación lógica del Contenido* (Content Aggregation), y de *Colocación física del Contenido* (Content Placement).

Dimensiones de Clasificación

En esta sección se presenta una extensión de esta taxonomía [137], considerando además de las dos propiedades mencionadas, la del modelo de descentralización de la red. La descentralización es quizás una de las características más importantes de las redes de contenido, puesto que el verdadero valor de la red no se encuentra en la capacidad de contenido de un único componente central, sino en la posibilidad que tiene la red de utilizar los recursos de los bordes, lo que potencializado por un buen *"efecto de red"* [171], permite una reducción de *costos de propiedad* significativa y mejor servicio a los participantes.

Por lo tanto, la taxonomía propuesta tendrá las siguientes tres dimensiones (descritas con anterioridad):

- ***Agrupación lógica del Contenido (Content Aggregation)***: semántica o sintáctica.
- ***Colocación física del Contenido (Content Placement)***: sensible o insensible al contenido.
- ***Descentralización***: cliente-servidor, distribución jerarquizada, completamente distribuida.

Las tres dimensiones son independientes y fundamentales para determinar el diseño y la arquitectura de la red. Principalmente determinan quién y cómo se toman las decisiones de enrutamiento, localización y descubrimiento del contenido, lo que afecta en gran medida todas las otras propiedades de una red de contenido.

Si se quisiera una clasificación más fina, sería conveniente agregar las características de anonimato y auto-organización como dimensiones de clasificación.

Taxonomía

Hay doce tipos de redes de contenido, basadas en estas tres dimensiones (agrupación lógica del contenido, colocación física del contenido y descentralización). La Tabla 1 resume las combinaciones que se pueden presentar, en una diagramación que trata de ser fiel al planteo original de [137].

		Agrupación lógica del contenido	
		Sintáctica	Semántica
Colocación física del contenido	Insensible al contenido	Tipo A: Redes sintácticas insensibles A.1 Cliente Servidor A.2 Distribución Jerarquizada A.3 Distribución Completa	Tipo C Redes semánticas insensibles C.1 Cliente Servidor C.2 Distribución Jerarquizada C.3 Distribución Completa
	Sensible al contenido	Tipo B Redes sintácticas sensibles. B.1 Cliente Servidor B.2 Distribución Jerarquizada B.3 Distribución Completa	Tipo D Redes semánticas sensibles D.1 Cliente Servidor D.2 Distribución Jerarquizada D.3 Distribución Completa

Tabla 1. Tipos de redes de contenido según las dimensiones de estudio.

Distintas aptitudes se desprenden de cada una de las dimensiones, haciendo más o menos idónea una red para determinado cometido. Por ejemplo las redes semánticas son muy aptas para búsquedas de contenido por proximidad semántica y para aplicaciones donde se quiera lograr algún nivel de censura sobre el contenido, las sintácticas presentan en general mayor anonimato y resistencia a fallas.
La mejor forma de ver las aptitudes de cada tipo de red es describiendo algunos ejemplos de redes de contenido existentes que pertenecen a las distintas categorías de la taxonomía.

Ejemplos de Redes de Contenido según Taxonomía

Tipo A: Redes Sintácticas Insensibles

A.1 Cliente-Servidor

Los ***Web Proxy-Cache*** [222] son un ejemplo claro. En este caso el contenido son las páginas Web, los nodos fuentes son los sitios Web originales (y los propios proxies puesto que cachean contenido), los nodos solicitantes son los clientes del proxy, y el proxy es el único nodo enrutador. Ninguna agrupación lógica es necesaria, y no es posible la colocación de contenido.
Este sistema requiere fuerte disponibilidad de recursos (principalmente transmisión y almacenamiento) por parte del proxy y presenta un punto único de falla; como alternativa distribuida se utilizan los Cooperative Web caching.

Un ejemplo muy similar son los ***Mirrors*** de contenido, que intentan ofrecer otras alternativas al sitio original para la descarga de grandes archivos, por tanto el contenido son archivos, se tiene un único nodo enrutador/fuente que es el mirror, y los nodos solicitantes son los usuarios del mirror.
Para este caso también existe una variante jerarquizada: las Content Delivery Networks.

Un poco más general, dentro de esta categoría entran por definición todos aquellos sistemas centralizados donde no existe agrupación lógica (por definición es sintáctica) o la misma es realizada exclusivamente por el componente central, ni colocación física (por definición es insensible). Ejemplos comunes son los motores de búsqueda (***Search Engines***), como Google[94] y Yahoo[241]; los servidores de servicios Web

(***Web Services*** [236], SOAP, WSDL, UDDI); la primera generación de sistemas P2P para compartir archivos (***P2P file sharing***), como Napster[166][167] (las tecnologías P2P se estudian más adelante en la sección 2.2).

Napster es la primera aplicación P2P popular que utiliza este modelo. Los clientes de Napster se identifican al ingreso a la red en el servidor central, informándole los archivos que comparte. Cuando un solicitante busca un contenido, lo hace en el servidor central, el cual conoce completamente los archivos compartidos reduciendo la búsqueda a una búsqueda local eficiente. La entrega del archivo se hace directamente entre pares. Cuando una fuente cambia el contenido ofrecido o se desconecta de la red, esto se le informa al servidor central.

Estos sistemas tienen un punto único de falla y vulnerabilidad, además de altos costos de mantenimiento y utilización de recursos en el servidor central.

A.2 Distribución Jerarquizada

Como alternativa a un Web Proxy-Cache central se utilizan los ***Cooperative Web Caching*** [6][200], básicamente tienen el mismo objetivo que los Web Proxy convencionales pero el sistema se encuentra formado por un conjunto de Proxies, dividiendo el costo de propiedad y mantenimiento entre varios equipos. Esta distribución además brinda cierto anonimato de fuente y editor, puesto que si los Proxies son autónomos potencialmente no se conoce quien está habilitando determinado contenido.
Es necesaria la agrupación lógica de contenido con el fin de que los solicitantes reciban contenido de varios proxies simultáneamente. Esta agrupación es sintáctica, porque se basa en la URL del sitio, no en un significado.

Las ***Content Delivery Network*** [4][19][37][56][70][130][174][176] son un ejemplo muy similar al anterior. Podrían pensarse como un conjunto de Mirrors que funcionan de forma cooperativa para distribuir archivos de forma menos costosa y más rápida.
Es necesaria la agrupación lógica de contenido con el fin de que los solicitantes reciban un archivo de varios mirrors simultáneamente. Esta agrupación es sintáctica, porque se basa en el nombre (o CRC-cyclical redundancy check) del archivo, no en un significado.

Dentro de la segunda generación de redes P2P para compartir archivos (***P2P file sharing***), existe una variante conocida como redes de Super-Peers. Las redes Super-Peer presentan un nivel de jerarquía: los nodos con pocos recursos de transmisión son solo fuente/solicitantes (llamados clientes) y están conectados a un único Super-Peer, los nodos Super-Peer son fuente/enrutador/solicitantes y forman entre sí una red completamente distribuida que funciona, en general, mediante el broadcast de solicitudes de contenido a todos los nodos Super-Peers.
Los nodos Super-Peer realizan una agrupación sintáctica del contenido (basada en el nombre o CRC del archivo) ofreciéndole a los clientes una vista consolidada del sistema, en ningún caso existe colocación física del contenido. El ejemplo más notorio es el protocolo FastTrack, utilizado por las redes KaZaA[127], iMesh[115] y Grokster[101] (las tecnologías P2P se estudian más adelante en la sección 2.2).

La mayoría de los sistemas de suscripción y publicación (***Subscribe-Publish Networks***), como Siena [32] y Publius [234], funcionan de forma jerarquizada. El contenido de estas redes son notificaciones y subscripciones. Los nodos fuentes son generadores de eventos, informándole a un servidor cercano de tal evento. Los servidores son nodos enrutadores que se encargan de comunicarse entre sí los eventos y anunciárselo a los subscriptores (nodos solicitantes que previamente especificaron su interés en este tipo eventos).

En estas redes, los eventos no pueden ser agrupados lógicamente, ni colocados físicamente en los nodos fuentes.
Las redes de sensores (***Content-based Sensor Networks***), como SCADDS[69][105] y SPIN [106] se manejan exactamente con la misma dinámica.

Dentro de esta categoría se encuentran también el sistema de correo electrónico y la mayoría de los sistemas de mensajería instantáneas, como Jabber [121]. El funcionamiento de estos sistemas es idéntico al presentado para las Subscribe-Publish Networks.

A.3 Distribución Completa

Dentro de la segunda generación de redes P2P para compartir archivos (***P2P file sharing***), existe una variante que funciona mediante el ***broadcast*** de solicitudes de contenido a todos los nodos fuentes, no presentando agrupación lógica, ni colocación física del contenido.

En este tipo de redes sucede que ningún contenido es publicado por parte de las fuentes, sino que los nodos solicitantes realizan búsquedas específicas de contenido a todos sus pares. Los pares que reciben las consultas en realidad actúan como nodos fuentes contestando al solicitante del cual se recibió el pedido si poseen el contenido buscado y como nodos enrutadores volviendo a propagar la consulta a sus pares. Para evitar bucles y esperas muy largas, las búsquedas se propagan con un mecanismo de tiempo de vida limitado (TTL). El ejemplo más notorio es Gnutella[88] (ver sección 2.2).

Tipo B: Redes Sintácticas Sensibles

B.1 Cliente-Servidor

La mayoría de las propuestas para computación distribuida (***Distributed Computing***) [11][74][82][213] presentan una arquitectura cliente-servidor y tienen aplicabilidad a procesos altamente paralelizables de mucho consumo de CPU. Esto se debe a la complejidad que surge del ordenamiento y sincronización de las tareas a realizar.
En este caso el contenido son las tareas a procesar, los nodos fuentes son llamados clientes y ofrecen el tiempo ocioso de su procesador con un fin común (búsqueda extraterrestre, cura del cáncer, etc.). Existe un único nodo enrutador/solicitante central encargado de administrar las tareas que realiza cada cliente (puede pensarse que este nodo central también es fuente porque inicialmente incluye todas las tareas, es decir el contenido). En general las tareas son asignadas a los clientes por parte del servidor considerando la cantidad de cómputo que requiere la tarea y la capacidad de cómputo ofrecida por la fuente, por tanto existe una colocación física sensible al contenido. La agrupación lógica es realizada por el servidor central, sin tomar en cuenta el tipo de tarea que realiza cada cliente (por esto la agrupación es sintáctica y la implementación se limita a tareas altamente repetitivas de mucho procesamiento). Distintas variantes se han presentado, por ejemplo solicitar la misma tarea a varios clientes simultáneamente como forma de redundancia. Dentro de las opciones más conocidas se encuentran Seti@Home[213], Avaki[11], Genome@Home[82], distributed.net[59], etc.

B.2 Distribución Jerarquizada

No se han encontrado ejemplos de redes en esta categoría. Se trataría de redes que agrupan lógicamente el contenido de manera sintáctica, y que modifican su lugar de almacenamiento sobre esta base, trabajando sobre la base de una estructura jerárquica.

B.3 Distribución Completa

Dentro de la segunda generación de redes P2P para compartir archivos (***P2P file sharing***), existe una variante que funciona en base a ***índice de hash distribuidos*** [202].
Sin entrar en más detalle en esta sección (las tecnologías P2P se estudian más adelante en la sección 2.2), se puede expresar que la idea básica es asignar mediante hash una identificación única a cada nodo y a cada archivo (contenido). Un archivo se aloja en el nodo activo con identificación más similar a la identificación del archivo. Los nodos son conectados con aquellos nodos que presentan una identificación con el mismo prefijo, formando una topología que se conoce como hipercubo.
Los solicitantes deben conocer exactamente la identificación del archivo, de esta forma la búsqueda es sencilla: es necesario enrutar una consulta hasta el nodo fuente con identificación más cercana, y allí debe estar el archivo.
Puede verse que existe agrupación lógica del contenido (se le asigna una identificación a los archivos), pero la misma es sintáctica porque no se basa en significado; la colocación física debe ser también sensible al identificador del archivo.
Múltiples variantes se han presentado a la propuesta original, dentro de las cuales se incluye múltiples dimensiones de hipercubos, respaldos, etc. Las redes más difundidas son Past (Pastry)[61][207][208], OceanStore (Tapestry)[136], Chord[225], CAN[201], etc.
Es de hacer notar que muchas de estas redes tienen el objetivo de almacenamiento persistente (en lugar del más conocido de compartir archivos entre pares), que se conocen con la nomenclatura de redes de respaldo (Backup Networks). El ejemplo más notorio de uso de esta tecnología para compartir archivos es la red Kademlia[153], DHT utilizada recientemente como método secundario en la red eMule[68].

Tipo C: Redes Semánticas Insensibles

C.1 Cliente-Servidor

No se han encontrado ejemplos de redes en esta categoría. Se trataría de redes que agrupan lógicamente el contenido de manera semántica, pero que no modifican su lugar de almacenamiento, y trabajan sobre una estructura cliente-servidor.

C.2 Distribución Jerarquizada

El sistema TRIAD [99] es una propuesta para un nuevo protocolo de red, basado en contenidos, que sustituya (subsumiéndolo) al método actual basado en enrutamiento IP, búsqueda de hosts por DNS y recuperación de contenidos vía HTTP.
Este protocolo permite una red donde el contenido es mapeado al espacio (con significado) de las URL. Una agrupación lógica se realiza según el dominio de las URL (semántica), pero no hay colocación física porque el contenido no debe ubicarse arbitrariamente en ninguna fuente en particular.
Los solicitantes realizan las búsquedas de contenido utilizando el protocolo INRP (Internet Name Resolution Protocol), mientras que el contenido se publica en la red utilizando NBRP (Name Based Routing Protocol).

C.3 Distribución Completa

Dentro de la segunda generación de redes P2P para compartir archivos (***P2P file sharing***), existe una variante que funciona en base al ***auto-ordenamiento*** de la topología [9] (ver la sección 2.2).
La idea básica es brindar escalabilidad a las redes completamente distribuidas, utilizando un ordenamiento en la topología según la afinidad de tipo de archivos que

tengan los nodos participantes. En una situación ideal los solicitantes estarán conectados a los nodos fuentes que con mayor probabilidad tengan el contenido buscado, de esta forma si se utiliza una estrategia de búsqueda basada en broadcast se disminuirían enormemente los tiempos de respuesta y los costos de comunicación. En general se prevén distintos planos de ordenamiento con el fin de aumentar la precisión en afinidad.
Dado que no se tiene una vista global del sistema, el auto-ordenamiento tiene que ser basado en una optimización local.

Tipo D: Redes Semánticas Sensibles

D.1 Cliente-Servidor

Como se aclaró anteriormente, las propuestas estudiadas en este documento para computación distribuida (***Distributed Computing***) se basan en repetir un único tipo de tarea muchas veces, por tanto la agrupación lógica es sintáctica ya que todas las tareas tienen el mismo "significado".
Se considera que fácilmente pueden existir variantes de redes de pares para computación distribuida donde existan varios tipos de tareas (no solo una única tarea que es necesario repetir muchísimas veces), en estos casos el servidor central debe hacer una agrupación en base al tipo de tarea y por tanto existiría una agrupación semántica. Debe quedar claro que este tipo de computación requiere por parte del servidor central una asignación, ordenamiento y sincronización de tareas mucho más complejo que el caso de un único tipo de tarea, siendo por esto que este tipo de redes tiene mucho menos éxito y difusión.

D.2 Distribución Jerarquizada

Quizás el mejor ejemplo de este tipo de redes es el sistema de nombre DNS[157][158] (Internet Domain Name System). El servicio de DNS traduce nombres de hosts o nombres de dominios en direcciones IPs, por tanto el contenido son registros de nombres y dominios. Los nodos solicitantes son todos los computadores conectados a Internet o una intranet que utilizan algún servicio tradicional de comunicación, como ser el Web, e-mail, etc. Los nodos fuente/enrutadores construyen una jerarquía de base de datos distribuida, de forma arborescente según el nombre del dominio.

Las consultas por dominios de los solicitantes las canaliza un tipo especial de servidor de nombres, llamado recursivo; este servidor es un nodo ruteador que se encarga de resolver las consultas en la jerarquía de base de datos distribuida[3] y presentar la respuesta de forma consolidada a los solicitantes.
La red DNS se estudia en profundidad más adelante en la sección 2.3.

Aquí existe una agrupación semántica del contenido, puesto que los dominios son agrupados según la propia jerarquía de los nombres de dominio. Cada nombre debe ser colocado en el servidor (fuente) que tiene asignado ese dominio en la estructura arborescente, por tanto existe colocación física de contenido sensible a la semántica (dominio).

D.3 Distribución Completa

No hemos encontrado ejemplos de redes en esta categoría. Se trataría de redes que agrupan lógicamente el contenido de manera semántica, y que modifican su lugar de almacenamiento, trabajando sobre una estructura completamente distribuida. Una

[3] Una solicitud al servidor recursivo, en general requiere de varias consultas en la jerarquía de DNS, lo cual es costoso para ser realizado por cada solicitante desde un punto de vista de ancho de banda y latencia.

propuesta aún en desarrollo, y que podría entrar en esta categoría, el proyecto Edutella.
Edutella[64] es un proyecto para construir una infraestructura de metadatos para redes P2P, basado en JXTA. En Edutella, la información es agrupada por su contenido, manteniendo de manera distribuida entre los nodos participantes un conjunto de metadatos (especificados según el estándar RDF del W3C) que permiten la búsqueda y recuperación semántica de los contenidos en sí mismos. Edutella prevé la replicación de los metadatos en varios nodos, para proveer persistencia de datos, y mejorar la disponibilidad y la carga. Los contenidos en sí mismos podrían también ser replicados, si bien en una primera etapa no se ha implementado este servicio.

2.1.6 Conclusiones y Consideraciones Generales

El área de las redes de contenido se encuentra aún en una etapa inicial de desarrollo, existiendo relativamente pocos trabajos que formalicen marcos conceptuales genéricos para el estudio de sus propiedades. Hay gran diversidad de propuestas concretas, reflejada en la existencia de redes de características muy diversas, pertenecientes a las diversas clases de la taxonomía presentada; la lista de características relevantes presentadas en este trabajo puede ser entonces de ayuda a la hora de buscar elementos para comparar las mismas y eventualmente diseñar nuevas arquitecturas.

Debe hacerse explícito que las redes jerarquizadas parecen adaptarse más a realidades diversas, siendo motivadas principalmente por un intento de reducción de costos, mejorar escalabilidad[4], resistencia a fallas y anonimato. Esto parece hacerse especialmente notorio para las redes Sintácticas Insensibles, tipo A, donde ésta es la principal forma de escalado.

Las redes Semánticas Sensibles, tipo D, parecen tener muy buenas propiedades, pero en muchos contextos no pueden implementarse debido a violaciones de autonomía o anonimato, o simplemente porque el contenido sobre el que se tiene que dar significado es muy complejo (por ejemplo en las redes de pares para colaboración el contenido es la misma interacción humana).
Las buenas propiedades de este tipo de redes se discuten en [137], resumidamente son:

- ***Jerarquización Semántica***: En estas redes existe una topología jerárquica según el significado del contenido. El nodo enrutador raíz presenta los contenidos más generales, y a medida que uno se aleja por la red del nodo raíz se tiene un conocimiento más rico sobre un tipo de contenido en particular.
- ***Rápido re-enrutamiento***: Dada que la topología es jerarquizada según contenido, cada nodo enrutador debe conocer de forma consolidada el contenido que él administra, es decir, es menos costoso el mantenimiento de la vista local del contenido. Logrando un rápido re-enrutamiento y un eficiente descubrimiento del contenido.
- ***Eficiente descubrimiento de Contenido***: El descubrimiento de contenido es muy simple, en la mayoría de los casos se sabe de forma determinista donde se encuentra un contenido (porque allí fue colocado a propósito para respetar la jerarquía semántica) y por tanto el tiempo de búsqueda de una solicitud es acotado según el tamaño de la red.

[4] Las redes cliente-servidor presentan una limitante en escalamiento debido al importante costo de su servidor central, mientras que las redes completamente distribuidas presentan una limitante debido al consumo exponencial de ancho de banda necesario por todos los nodos para participar en la red.

- ***Búsquedas por proximidad semántica***: A diferencia de la mayoría de los otros tipos de redes de contenido, esta topología jerarquizada por significado del contenido habilita a la búsqueda por proximidad semántica, es decir, por ejemplo cuando determinado contenido no se encuentra en la red es posible suministrar aquel que más se le asemeja (por ejemplo si en una red de documentación bibliográfica se busca determinado texto, es posible ofrecer otro texto sobre el mismo tema).
- ***Asignación eficiente de recursos***: Dado que las solicitudes son de contenido, es eficiente y realizable pensar en utilizar los nodos con mejores capacidades (mayor transmisión, almacenamiento o procesamiento) para los contenidos más solicitados.

La Tabla 2 resume los ejemplos encontrados de tipos de redes de contenido según la taxonomía.

		Agrupación lógica del contenido	
		Sintáctica	Semántica
Colocación física del contenido	Insensible al contenido	Tipo A Redes sintácticas insensibles **A.1 Squid, Mirrors, Napster** **A.2 Akamai, KaZaA, Grokster, iMesh, eMule, Siena, Publius, SCADDS, SPIN, Jabber** **A.3 Gnutella**	Tipo C Redes semánticas insensibles **C.1 -** **C.2 TRIAD** **C.3 -**
	Sensible al contenido	Tipo B Redes sintácticas sensibles. **B.1 Seti@Home, Avaki, Genome@Home, distributed.net** **B.2 -** **B.3 Pastry, Tapestry, Chord, CAN, Overnet, Kad**	Tipo D Redes semánticas sensibles **D.1 -** **D.2 DNS** **D.3 Edutella**

Tabla 2. Ejemplo de redes de contenido según las dimensiones de estudio.

2.2 Redes de Pares (Peer to Peer)

2.2.1 Introducción

La bibliografía no es uniforme respecto a definir que tipos de redes son de pares y cuales no. La óptica tomada en este documento se acerca fuertemente a la expresada en [162] y [216].

En esta sección se define a la red de pares y se contextualizan dentro de las redes de contenido. Se refina la taxonomía, presentando ejemplos y una reseña historia de la evolución de arquitecturas.

2.2.2 Bases Conceptuales

Red de Pares (P2P)

El término peer-to-peer refiere a la clase de sistemas y aplicaciones que emplean recursos distribuidos para realizar alguna función crítica de forma descentralizada [162]. Los recursos pueden ser: procesamiento, almacenamiento (contenido particular) y/o transmisión (ancho de banda). Las funciones críticas a realizar son: computación distribuida, compartir información, colaboración entre pares, o cualquier tipo de servicios que puede ser brindado por un sistema con estas características.

Como se adelantó existe una serie de definiciones respecto a las redes P2P. Las más destacadas (extraídas de [162]) se detallan a continuación:

- El grupo de trabajo de Intel para P2P define estas redes como: "el compartir recursos y servicios computacionales mediante la conexión directa entre sistemas" [192].
- Alex Weytsel define P2P como "el uso en modalidad no cliente de dispositivos que se encuentran en la periferia de la Internet" [231].
- Ross Lee Graham define P2P a través de tres requerimientos claves: "tienen computadoras con calidad de servidor, tienen un sistema de direccionamiento independiente del DNS, y están preparados para diferentes formas de conectividad" [96].
- Clay Shirky de O'Reilly and Associate las define como: "la clase de aplicaciones que aprovechan de los recursos (almacenamiento, ciclos, contenido, presencia humana, etc.) disponible en los bordes de Internet. Debido a que acceder a estos recursos descentralizados significa operar en un entorno de conectividad inestable con impredecible variaciones de direcciones IP, estas redes deben operar fuera del sistema de DNS y tienen importante o completa autonomía respecto a servidores centrales" [216].
- Finalmente Kindberg define los sistemas P2P como aquellos que tienen tiempo de vida independiente [128].

Para lo que resta de este documento se toma la definición brindad por Clay Shrky [216]:

> "Una ***Red de Pares*** (Peer-to-Peer) es una red donde se aprovecha de los recursos (almacenamiento, ciclos, contenido, presencia humana, etc.) disponible en los bordes de Internet. Debido a que acceder a estos recursos descentralizados significa operar en un entorno de

conectividad inestable con impredecible variaciones de direcciones IP, estas redes deben operar fuera del sistema de DNS y tienen importante o completa autonomía respecto a servidores centrales".

Por tanto las redes de contenido comparten recursos con un objetivo común:

Recursos empleados por las redes de pares:

- ***Procesamiento.***
- ***Almacenamiento.***
- ***Transmisión.***

Objetivos de las redes de pares:

- ***Computación distribuida.***
- ***Compartir información.***
- ***Colaboración entre pares.***
- ***Plataforma de servicios.***

Especificación en el Marco de las Redes de Contenido

Se considera que las redes de pares son un caso particular de redes de contenido, donde los nodos fuente de contenidos reciben el nombre de ***pares***.
Si se recuerda la definición de red de contenido (una red donde el descubrimiento, el direccionamiento y el enrutamiento del contenido son basados en la descripción del contenido en lugar de su ubicación), no es directo entender por ejemplo como puede enrutarse en la red por recursos como el procesamiento o el ancho de banda, es decir como pueden entenderse los recursos de las redes de pares como contenido.
Esto se clarifica si se consideran que las redes de pares respetan fuertemente la autonomía y el anonimato de los pares, donde a los efectos de la red no importa quien es tal nodo, sino simplemente su capacidad de recursos. En este sentido una red de pares óptimamente diseñada debe preguntar por un recurso cuando le hace falta y no por un nodo; por tanto en general debe responder a la pregunta de como llegar a determinado recurso (o contenido) y no como llegar a determinado nodo.

Ahora que se entiende porque las redes de pares son un tipo de redes de contenido, es necesario resaltar lo que diferencia a las redes de pares del resto de las redes de contenido.
La diferencia se presenta en algunas características de la arquitectura de red y principalmente en algunas características que presentan los nodos pertenecientes a la red:

- En primer lugar las redes P2P están compuestas por pares con un alto nivel de ***autonomía***. En particular se considera indispensable que los nodos participantes tengan la posibilidad de ingresar y salir de la red en cualquier momento. Además en general cada nodo determina la cantidad de recurso que desea compartir. Por tanto la característica que se considera principalmente distintiva de las redes de pares respecto a las redes de contenido es que los participantes presentan un alto nivel de autonomía, especialmente de autonomía de participación.

- Dado que los nodos pueden libremente ingresar al sistema para ***compartir*** sus recursos, es necesario un incentivo para dicha participación. Este incentivo se logra al perseguir un objetivo o necesidad común. Por tanto existe un deseo implícito de compartir recursos en la búsqueda de un objetivo común.
- Otra consecuencia de la autonomía es el ***anonimato***, muchas veces no se quiere conocer las fuentes o solicitantes de contenido (por ejemplo por razones legales como en las redes para compartir música con derechos de autor) u otras veces simplemente es completamente innecesario su conocimiento. Por tanto las redes de pares persiguen en general un objetivo de anonimato.

En conclusión:

Una ***Red de Pares*** (Peer-to-Peer) es un caso particular de redes de contenido, donde los nodos de contenidos (llamados ***pares***) presentan fuertes características de ***autonomía***, ***disposición a compartir*** y ***anonimato***.

2.2.3 Clasificación y Taxonomía

La primera clasificación de las redes de pares que aparece en la literatura es en base al tipo de aplicación que tiene la red [14][162] (es decir, el tipo de contenido que es compartido). Bajo esta idea se clasifican las aplicaciones P2P como:

- ***Computación Distribuida (distributed computing)***: Internet/intranet distributed computing, grid computing, etc. La aplicabilidad va desde búsqueda de vida extraterrestre, proyecto genoma, ataques de vulnerabilidad conjunta, calculo de precios y riesgo financiero, análisis de mercado y crédito financiero, análisis demográfico, etc.
- ***Colaboración (collaboration)***: Communication (chat, instant messaging), Co-review/edit/author/create, shared applications, Gaming, Discovery, etc.
- ***Compartir contenido (content sharing)***: File delivery, Content Distribution, Distributed Storage, Caching, Edge Services, Information Management (discover, aggregate, filter, mining, organize, etc.), etc.

Puede verse que el mercado para estas aplicaciones es muy variado, incluyendo ***consumidores*** (compartir música, películas, o contenido en general, mensajería instantánea, e-mail, juegos), ***empresa*** (biotecnología, finanzas, soluciones IT, B2B), y ***público general*** (entretenimiento, entrega de contenido).

Una clasificación basada en tipo de aplicación no es suficiente para entender los distintos tipos de diseños, arquitecturas y algoritmos que conviven entre las distintas redes P2P.
Siendo las redes de pares un caso particular de las redes de contendido, la mayoría de las características descriptas se aplican a este caso. A continuación se especifican las características que presentan algunas diferencias en la definición en el contexto de las redes de pares respecto al de redes de contenido.
Inspirada en la taxonomía realizada para las redes de contenido, se hacen aportes procedentes de [116], [192] y [229].

Dimensiones de Clasificación que varían respecto a las Redes de Contenido

Descentralización [242]

Una primera dimensión de clasificación es (al igual que para el caso general de las redes de contenido) la ***descentralización***.
Para las redes de contenido, la descentralización se encuentra determinada por la capacidad que tiene la red de funcionar con varios nodos de un mismo tipo (fuentes, solicitantes y enrutadores). Como se señaló anteriormente, una característica distintiva de las redes de pares es el deseo de compartir con un fin común, es por esto que para este tipo particular de redes siempre se van a tener muchos nodos fuentes, remitiendo el concepto de descentralización a los nodos solicitantes y (principalmente) a los enrutadores.

En el caso general, se dijo que los grados de descentralización pueden clasificarse en los modelos de cliente-servidor, distribución jerarquizada y completamente distribuida. Para las redes de pares en general se utiliza otra terminología:

- ***Modelo Puro (Pure P2P)***: El modelo completamente distribuido recibe el nombre de puro porque expresa fielmente la filosofía de una red de pares, donde todos los nodos participantes de la red presentan iguales roles (fuente, enrutador, solicitante) y responsabilidades. A los efectos de este documento, esto significa que existe una cantidad arbitraria de nodos fuente/enrutador/solicitante (donde importa que todos los nodos son enrutadores). En este caso los solicitantes pueden consultar a cualquier nodo por un contenido, y este nodo tiene que saber como enrutarlo hasta alguna fuente. Ejemplos son Gnutella [88] y Freenet[38][79].
- ***Modelo Híbrido (Hybrid P2P)***: El modelo cliente-servidor recibe el nombre de híbrido, puesto que se considera que no es un sistema P2P puro. En este caso se tiene un único nodo central de enrutamiento (podría haber redundancia en este nodo central). Los solicitantes consultan al servidor de enrutamiento central, que les informa que fuentes tienen el contenido buscado. Pueden haber muchos solicitantes o uno solo. Ejemplos comúnmente utilizado son Napster[166][167] (un nodo enrutador, muchos nodos fuente/solicitantes) y Seti@home[213] (un nodo enrutador/solicitante, muchos nodos fuente).
- ***Modelo Jerarquizado***: El modelo jerarquizado, donde existen varios nodos enrutadores, aun no presenta un nombre distintivo. El primer ejemplo de notoriedad se presenta para las aplicaciones de compartir archivos, y se conoce como redes SuperPeers, en estas redes existen nodos fuente/enrutadores/solicitantes llamados superpeers y nodos exclusivamente fuente/solicitantes llamados clientes. Los nodos clientes pueden estar conectados únicamente a un superpeer (a veces se conecta a dos o tres por redundancia), siendo los superpeers una especie de red dorsal de contenido que funciona por lo general bajo un modelo puro de P2P. El ejemplo más notorio de este tipo de redes es KaZaA[127]. Este modelo tiene el potencial de combinar la eficiencia de las búsquedas de contenido de los sistemas híbridos centralizados con la autonomía, balanceo de carga y robustez frente a ataques de las búsqueda en los sistemas puros distribuidos[242][243].

La ***descentralización***, para las redes de pares, se encuentra determinada por la capacidad que tiene la red de funcionar con varios nodos de un mismo tipo (solicitantes y enrutadores). Se considera que siempre existe una cantidad arbitraria de nodos fuente, por la filosofía de compartir en la que se basan estas redes.

Modelos de Descentralización en las redes de pares (P2P):

- ***Híbrido.***
- ***Distribución Jerarquizada.***
- ***Puro (broadcast, distribución de índices de hash).***

Auto-Organización [33][149]

También llamado auto-escalamiento, el auto-ordenamiento presenta una clasificación especial cuando se tratan las redes de pares.
Para las redes P2P en general se utiliza la siguiente terminología:

- ***In-estructuradas***: donde los nodos de la topología virtual se organizan de forma aleatoria, utilizando inundación (broadcast o caminantes aleatorios) para el descubrimiento del contenido. Cuando un solicitante realiza una búsqueda, dicha búsqueda se propaga entre los pares sucesivamente. Este tipo de redes permiten búsquedas semánticamente complejas y no requieren colocación de contenido, sin embargo son poco eficientes para encontrar contenido poco distribuido, puesto que la búsqueda requiere de la consulta a muchos nodos. El ejemplo más notorio de estas redes es Gnutella[88].
- ***Estructuradas***: en general basadas en alguna técnica de distribución de tabla de hash (DHT), estas redes construyen un grafo estructurado de nodos, donde cada nodo es responsable de un conjunto de contenidos (o índices a contenido) determinado[5]. A pesar que este tipo de redes se diseña para mejorar la eficiencia del descubrimiento del contenido, es discutible su utilidad para redes con contenido o conexiones muy dinámicas. En general estas redes aseguran descubrir un contenido específico recorriendo a lo sumo una cantidad logarítmica de nodos según el tamaño de la red. Ejemplos son Past (Pastry)[61][207][208], CAN[201], Chord[225], OceanStore (Tapestry)[136], Overnet(eMule)[68], etc.

Modelos de Auto-Organización en las redes de pares (P2P):

- ***In-estructuradas.***
- ***Estructuradas.***

Taxonomía

Se profundiza la taxonomía en esta oportunidad, incluyendo las siguientes cuatro dimensiones (descriptas con anterioridad):

- ***Agrupación lógica del Contenido (Content Aggregation)***: semántica o sintáctica.
- ***Colocación física del Contenido (Content Placement)***: sensible o insensible al contenido.
- ***Descentralización***: híbrido, distribución jerarquizada y puro.
- ***Auto-Organización***: in-estructuradas o estructuradas.

En la Tabla 2 se incluyen ejemplos de redes de pares para las dos primeras características. La Tabla 3 resume los ejemplos encontrados para la descentralización y el auto-ordenamiento.

[5] En general cada nodo tiene un identificador y es responsable de los contenidos con un identificador que surge de la función de hash aplicada al identificador del nodo.

		Descentralización		
		Híbrido	Jerarquizada	Puro
Auto-Organización	In-estructuradas	**Napster, Seti@Home, Avaki, Genome@Home**	**FastTrack (KaZaA), eDonkey, eMule OpenNap**	**Gnutella**
	Estructuradas	**-**	-	**FreeNet, Edutella, Pastry, Tapestry, Chord, CAN, Overnet**

Tabla 3. Ejemplo de redes de pares según las nuevas dimensiones de estudio.

Por definición es imposible la existencia de redes estructuradas hibridas.
A medida que las redes in-estructuradas crecieron en cantidad de nodos, surgió la necesidad de un modelo que contemplara mejor la diversidad de los nodos participantes. En esta situación, la inexistencia de redes jerarquizadas estructuradas se supone a la poca difusión en la actualidad de las redes estructuradas en si mismas. A continuación se presenta una cronología de los diseños en las redes de pares.

2.2.4 Evolución Histórica

Las redes de pares son la forma más natural de abordar los problemas que enfrentan las aplicaciones distribuidas. Por tanto, las primeras aplicaciones surgen en general bajo este paradigma, pero luego de un crecimiento inicial pierden el modelo original y pasan a una arquitectura cliente-servidor.
De forma general se puede decir que a fines de la década de los 80's, el modelo cliente-servidor se impone entre las arquitecturas de las aplicaciones distribuidas existentes, puesto que es la forma más rápida y menos costosa de ofrecer servicio a una gran cantidad de usuarios inexpertos.
Con la Internet altamente difundida, a fines de los 90's, los potenciales usuarios de un servicio son tantos que solo pueden ser manejados por sistemas centralizados de forma muy costosa, es entonces donde parece natural, en la medida de las posibilidades, utilizar los recursos ociosos del borde de la red (es decir, los PC de los usuarios) y resurge el modelo de redes de pares (P2P) para una variedad importante de aplicaciones.

La misma Internet puede verse bajo este análisis, considerando que es una red de contenido, donde el contenido son las direcciones de red IP. En sus comienzos la Internet surge como una red de pares (peer-to-peer), donde cada nodo cooperaba en el enrutamiento de los paquetes que se realizaba a través de ellos mismos. UUNet y Fidonet son dos ejemplos de redes descentralizadas de conexión dial-up (módem), los nodos pertenecientes a esta red se conectaban periódicamente de forma asincrónica a otro nodo intercambiando paquetes (e-mails o foros de discusión), luego de varios saltos entre los nodos participantes los mensajes llegaban a destino.
Sin embargo luego de enfrentar un crecimiento inicial, se perdió el diseño original de redes de pares por un diseño jerarquizado. Este diseño jerarquizado, consolidó en la actualidad una estructura cliente-servidor, donde unos pocos nodos servidores (routers) se encargan de ofrecer servicio de enrutamiento IP a la mayoría de nodos que son clientes de Internet.

Con el fin de observar la evolución de las tecnología de las redes de pares a lo largo de la historia, resaltando aciertos y errores, se presenta como han cambiado las

aplicaciones P2P según su utilización (compartir contenido, colaboración, computación distribuida).

Compartir Contenido

Las aplicaciones pensadas para compartir contenido, en este contexto refieren a compartir archivos (files). Múltiples redes con distintos objetivos son utilizadas con este fin. Entre ellas se resaltan: File delivery, Content Distribution, Distributed Storage, Caching, Edge Services, Information Management (discover, aggregate, filter, mining, organize, etc.), etc.
En esta sección se resume la evolución histórica de este tipo de redes de pares.

Primera generación

La primera generación de redes P2P utiliza un modelo centralizado cliente-servidor (modelo híbrido) para resolver el enrutamiento y descubrimiento de contenido (un único nodo enrutador). De esta forma se simplifica enormemente el problema de las búsquedas de contenido, puesto que el enrutador central tiene una vista global del sistema, conociendo exactamente quienes comparten que contenido.
Esta solución tiene muy buena performance y su escalamiento se encuentra limitado por las operaciones que realiza el router central (sincronización y coordinación).

Napster[166][167] es la primera aplicación extremadamente popular que utiliza este modelo. Los clientes de Napster se identifican al ingreso a la red en el servidor central, informándole los archivos que comparten. Cuando un solicitante busca un contenido, lo hace en el servidor central, el cual conoce completamente los archivos compartidos reduciendo la búsqueda a una búsqueda local eficiente. La entrega del archivo se hace directamente entre pares. Cuando una fuente cambia el contenido ofrecido, esto se le informa al servidor central.
Este sistema logró escalar hasta 6 millones de usuarios, cuando por litigios legales de derecho de autor tuvo que cambiar su estrategia y filtrar el contenido ilegal. En realidad el problema, desde un punto de vista tecnológico de red, surge porque el modelo centralizado tiene un punto único de vulnerabilidad y falla.

Segunda generación

Como respuesta a la primera generación de P2P, basadas en un nodo enrutador central vulnerable, costoso y poco robusto, aparece la segunda generación de P2P con una arquitectura completamente distribuida, donde todos los nodos son fuente, enrutador y solicitante.
Una serie de variantes a esta idea original se presentan a lo largo del tiempo. Estas variantes se basan principalmente en la forma que la red permite descubrir, localizar y enrutar el contenido. En este documento las variantes se agrupan como: broadcast, superpeers, índice de hash distribuida y auto-ordenamiento.

Cuando la red se basa en ***broadcasts*** sucede que ningún contenido es publicado por parte de las fuentes, sino que los nodos solicitantes realizan búsquedas específicas de contenido a todos sus pares. Los pares que reciben las consultas en realidad actúan como nodos enrutadores volviendo a propagar la consulta y como nodos fuentes contestando al solicitante que les realizó la consulta si poseen el contenido buscado. Para evitar bucles y esperas muy largas, las búsquedas se propagan con un mecanismo de tiempo de vida limitado (TTL). El ejemplo más notorio es Gnutella[88].
Este tipo de redes presenta un gran consumo de ancho de banda y procesamiento únicamente en información de control (es decir, procesando consultas), esto en la práctica lleva a que este tipo de redes tengan una escalabilidad limitada, haciendo

muy costoso, en términos de ancho de banda, pertenecer a la red y obteniendo un alto grado de indeterminismo en las búsquedas. Esto lo muestran los numerosos monitoreos y estudios teóricos realizados sobre Gnutella [1][123][205][206][223].

Como solución a las limitaciones de escalabilidad, surgen las primeras versiones de redes jerarquizadas. Esta variante, conocida como redes de Super-Peers, presentan un nivel de jerarquía donde los nodos con pocos recursos de transmisión son solo fuente/solicitantes (llamados clientes) y están conectados a un único Super-Peer, los nodos Super-Peer [6] son fuente/enrutador/solicitantes y forman entre sí una red completamente distribuida que funciona, en general, mediante el broadcast de solicitudes de contenido a todos los nodos Super-Peers.
Los nodos Super-Peer realizan una agrupación del contenido ofreciéndole a los clientes una vista consolidada del sistema, en ningún caso existe colocación física del contenido. El ejemplo más notorio es la red FastTrack utilizada por KaZaA[127], a pesar que no ha sido publicada en todo detalle su arquitectura.

En paralelo, otra respuesta al problema de escalabilidad se estaba diseñando: redes completamente distribuidas basadas en ***índice de hash distribuidos*** [202].
En Chord[225] se asigna una identificación única a cada nodo y a cada archivo ofrecido mediante un algoritmo de hash. Existe el concepto de distancia entre los identificadores. Los índices de un archivo se alojan en el nodo activo con identificación más similar (menor distancia) a la identificación del archivo. Los nodos son conectados con aquellos nodos que presentan una identificación con el mismo prefijo, formando una topología de anillo (y de forma general un hipercubo para otras DHTs). Los solicitantes deben conocer exactamente la identificación del archivo, de esta forma la búsqueda es trivial: es necesario enrutar una consulta hasta el nodo fuente con identificación más cercana, y allí deben estar los índices del archivo.
Puede verse que existe agrupación lógica del contenido (se le asigna una identificación a los archivos), pero la misma es sintáctica porque no se basa en significado. Además para que el método funcione es necesario colocación física sensible al identificador del archivo.
Múltiples variantes se han presentado a esta propuesta, dentro de las cuales se incluye múltiples dimensiones de hipercubos, respaldos, etc. Las redes más difundidas son: Past[61][207][208] (Pastry), OceanStore[136] (Tapestry), Chord[225], CAN[201], etc.
Es de hacer notar que muchas de estas redes tienen el objetivo de almacenamiento persistente (no compartir archivos entre pares), que en la nomenclatura presentada se llamaron redes de respaldo (Backup Networks). Para redes de compartir archivos la experiencia más notoria se encuentra en la red eMule[68] que en la actualidad incluye como método complementario de descubrimiento a la red Overnet (también conocida como Kad) que utiliza el algoritmo de DHT Kademlia[153].

Los estudios indican que estos sistemas escalan a billones de usuarios, millones de servidores y 10^{14} archivos.
Como resumen se puede decir que este tipo de redes son muy atrayentes porque garantizan una cota superior logarítmica, según el tamaño de la red, al problema de encontrar una fuente particular de contenido; sin embargo presentan dos problemas que las limitan a contextos particulares: requieren colocación física de contenido (violando la autonomía deseada por los usuarios) y no es fácil realizar búsquedas porque en general es necesario conocer el nombre del archivo o alguna otra propiedad que determine el identificador del archivo (es decir carecen de un método robusto para el descubrimiento de contenido).

[6] Los nodos Super-Peers también son llamados en la literatura ultrapeers.

La última propuesta dentro de esta generación es aquella basada en el ***auto-ordenamiento*** de la topología [9]. La intención es lograr una performance similar a la obtenida con las redes de índices de hash distribuidos, sin violar la autonomía de los usuarios (no permitiendo colocación física de contenido).
El ordenamiento de la topología se encuentra basado en afinidad de tipo de archivos que tengan los nodos participantes. En una situación ideal los solicitantes estarán conectados a los nodos fuentes que con mayor probabilidad tengan el contenido buscado, de esta forma si se utiliza una estrategia de búsqueda basada en broadcast se disminuirían enormemente, inclusive asegurando un tiempo promedio acotado de respuesta muy inferior a una red desordenada.
En general se prevén distintos planos de ordenamiento con el fin de aumentar la precisión en afinidad, contemplando los varios tipos de contenido que pueden ser solicitados por un nodo.
Dado que no se tiene una vista global del sistema, el auto-ordenamiento tiene que ser basado en una optimización local.

Tercera generación

Luego del gran avance realizado sobre las redes puras, considerando algoritmos complejos para el auto-ordenamiento o la distribución de índices de hash, se observa que los usuarios optaron por las aplicaciones que preservan la mayor autonomía y anonimato en cada nodo.

Importantes mejoras se ofrecen actualmente respecto al anonimato en algunas de las redes de segunda generación. Los casos más notables son: Freenet[38][79], I2P[111], GNUnet[87], Entropy[67], MUTE[164[, Napshare[165], etc.

En la actualidad Freenet[38][79] asigna una identificación única a cada nodo y a cada archivo ofrecido mediante hash. Muy similar a una DHT, Freenet utiliza un método de enrutamiento heurístico basado en los identificadores que preserva enormemente el anonimato de los participantes.

Muchos de las redes de esta generación se conocen como F2F (Friend-to-friend), en lugar de P2P, porque basan parte de su solución de anonimato en permitir al usuario la comunicación solo con otros usuarios previamente conocidos.

Preservar el anonimato y la autonomía presenta un costo en recursos de comunicación, posiblemente este sea el principal inhibidor para la masificación de estas redes en un futuro.

Colaboración

Dentro de las aplicaciones P2P para colaboración se resaltan los sistemas de mensajería instantánea y charla, los juegos on-line y las aplicaciones compartidas en general (***instant messaging, chat, multiplayer games, shared applications***, etc).

El tipo de contenido ofrecido en este tipo de redes no es tan simple de modelar como en los otros casos porque básicamente se trata de interacción humana.
Un conjunto de desafíos técnicos deben ser enfrentados por este tipo de redes. En general las aplicaciones de colaboración son orientadas a eventos, donde un grupo de nodos quieren realizar una tarea de forma conjunta y es necesario mantener informado a cada par de lo hecho por los otros. Cualidades especiales son imprescindibles para una aceptación de estas aplicaciones en los distintos mercados (consumidores, empresas, público en general), por ejemplo:

- ***localización***: debe saberse exactamente quienes se encuentran en la red,

- ***restricciones transaccionales***: muchas veces se debe asegurar propiedades transaccionales complejas (como las ofrecidas en los sistemas de base de datos),
- ***restricciones de tiempo real***: la mayoría de las veces no es posible tener un gran retardo en la propagación de eventos, puesto que sería fácilmente notado por los otros pares. Esto depende directamente de la red subyacente, pero algunas técnicas pueden usarse. Por ejemplo esto es imprescindible en los juegos multijugador como DOOM [60] o comunicaciones telefónicas como Skype[218].
- ***tolerancia a fallas***: los eventos no pueden perderse, puesto que de otra forma los participantes tendrían distintas vistas de la información.

Estos problemas han sido resueltos satisfactoriamente en los últimos años. Algunos buenos ejemplos de software colaborativo entre pares son Groove[100], Magi[20][21] y OpenCola [175]. Para mensajería instantánea, el sistema de telefonía Skype[218], inicialmente concebido por los creadores de KaZaA[127] y hoy propiedad de eBay[62], tiene aproximadamente 4 millones de usuarios concurrentes.

Computación Distribuida

La computación distribuida (***Distributed Computing***), considerada como el método para alcanzar alta performance de cómputo con un sistema formado por varios computadores estándares y una conexión de red, no comienza con las redes de pares.

Una serie de proyectos a principios de los 90's mostraron que los sistemas de computación distribuida son realizables y útiles [13][16][143][144]. Con el experimento I-WAY [75][76] en 1995 surge la computación en grilla (***Grid Computing***), que rápidamente es difundida y estudiada en el mundo académico. En la actualidad existen una serie de productos comerciales y esfuerzos académicos que han llevado la computación en grilla a escalas mucho mayores, gran parte de la información se concentra en Globus [85], Gridbus[97], Grid Forum [98] y Mersenne Gimps [155], mientras que una serie de empresas ofrecen el servicio bajo distintos modelos de negocio [156][197][228].

Muchos de los desarrollos realizados en la computación en grilla no están pensados para el entorno de Internet, donde una enorme cantidad de usuarios quieren compartir sus recursos de cómputos manteniendo su autonomía.
A veces llamada la computación en Internet o computación de pares (***Internet and Peer-to-Peer Computing***) intenta lidiar con los problemas de heterogeneidad y autonomía de los usuarios en este entorno. La heterogeneidad se presenta en la cantidad y calidad del procesamiento ofrecido por los usuarios, y en la gran variedad de hardware y sistemas operativos coexistentes. La autonomía determina que las conexiones/desconexiones de los usuarios se realicen con mucha frecuencia, además de cambios impredecibles en el procesamiento ofrecido.
Entonces, en este entorno la computación distribuida debe realizarse bajo el paradigma de las redes de pares (P2P), y estos casos son los discutidos en este documento.

La mayoría de las propuestas para computación distribuida P2P (llamadas en este documento simplemente ***Distributed Computing***) [11][59][66][74][82][213] presentan una arquitectura cliente-servidor y tienen aplicabilidad a procesos altamente paralelizables de mucho consumo de CPU. Esto se debe a la complejidad que surge del ordenamiento, asignación y sincronización de las tareas a realizar.
En este caso el contenido son las tareas a procesar, los nodos fuentes son llamados clientes y ofrecen el tiempo ocioso de su procesador con un fin común (búsqueda

extraterrestre, cura del cáncer, etc.). Existe un único nodo enrutador/solicitante central encargado de administrar las tareas que realiza cada cliente. Las tareas son asignadas a los clientes por parte del servidor, por tanto existe una colocación física del contenido. La agrupación lógica es realizada por el servidor central, sin tomar en cuenta el tipo de tarea que realiza cada cliente (por esto la agrupación es sintáctica y la implementación se limita a tareas altamente repetitivas de mucho procesamiento).
Distintas variantes se han presentado, por ejemplo solicitar la misma tarea a varios clientes simultáneamente como forma de redundancia. Dentro de las opciones más conocidas se encuentran Seti@Home[213], Avaki[11], Genome@Home[82], distributed.net[59], Entropia[66], etc.
El proyecto Seti@home logró un pico de 3,5 millones de usuarios y cerca de 25 Tflop/s (miles de billones de operaciones de punto flotante por segundo), y como pionero en su genero logró grandes adeptos.

Como se expresa en "The Anatomy of the Grid" [77] los sistemas de computación P2P y computación en grilla presentan mucha similitud y actualmente se diferencian únicamente en los objetivos que cada comunidad enfrenta. La computación P2P brinda una solución vertical a problemas específicos, mientras que la computación en grilla intenta atacar la interoperabilidad e infraestructura de las arquitecturas y protocolos utilizados.
Se considera natural que en el futuro exista una convergencia entre ambos enfoques, cuando los objetivos y el mercado se uniformicen.

2.2.5 Arquitecturas Actuales

Como se vio hasta ahora una de las principales limitaciones que tienen los sistemas actuales es la incompatibilidad de protocolos y la falta de interoperabilidad. Esto es típico de los nichos de mercado emergentes, tal cual hoy se encuentran las tecnologías P2P en todas sus aplicaciones.

Como se dijo cuando se hablo de la característica de interoperabilidad de las redes de contenido, los métodos tradicionales para alcanzar sistemas interoperables son:

- ***Estándares***: SOAP, Gnutella, etc.
- ***Código abierto***: JXTA, eMule, OpenNap, etc.
- ***Estándares de facto***: JXTA, .NET, etc.
- ***Especificaciones en común***: Peer-to-Peer Working Group, Peer-to-Peer Working Group at Internet2, O'Reilly P2P, etc..

A continuación se realiza una introducción a los frameworks existentes para redes P2P y dos aplicaciones P2P muy populares en la actualidad: eMule[68] y KaZaA[127].

Frameworks

A continuación se hace un resumen de los frameworks o plataformas que actualmente se perfilan para ser los estándares. Es importante señalar que en realidad estas plataformas deben oficiar como un middleware. Un ***Middleware***[7] es una tecnología de software diseñada para ayudar a manejar la complejidad y heterogeneidad inherente a un sistema distribuido. Por tanto, se plantea una arquitectura de tres capas (***Three-Tier Model***) para las aplicaciones P2P, donde la capa intermedia brinda todas las funcionalidades o lógica del sistema distribuido, ofreciendo a la capa de aplicación una vista parcial y consolidada de los contenidos. La capa superior, o de presentación, solo se encarga de ofrecer una vista consolidada de los contenidos a los solicitantes, y la capa más inferior, en este caso son las fuentes crudas de contenido.

[7] Se puede encontrar mayor especificación de los middleware en la RFC 2768.

El software de middleware en generar tiene que ejecutarse en los tres tipos de nodos que pertenecen a la red (fuente, enrutador y solicitante) y es el encargado de la identificación y autenticación de los nodos pertenecientes a la red, del descubrimiento del contenido ofrecido, autorización de solicitudes y funciones particulares de cada tipo de aplicación.

Múltiples frameworks se han desarrollado con el fin de resolver los problemas comunes de heterogeneidad, distribución y persistencia de los sistemas P2P.
Algunas tecnologías como CORBA[48] o DCOM[53] son middlewares y estándares de facto pero no se adaptan completamente a la naturaleza y masividad de Internet, por tanto parece que no serán las utilizadas para el desarrollo de futuras aplicaciones P2P interoperables.
Dentro de la gama de posibilidades parecen sobresalir claramente JXTA de Sun[126] y .NET de Microsoft[221] como plataformas de facto.
Anthill[12] es un framework pionero, desarrollado específicamente como plataforma P2P, muchas de sus ideas se encuentran hoy incluidas dentro de JXTA.

eMule [68]

El proyecto eMule[68] surge inicialmente como una opción de código abierto para sustituir al cliente de la red eDonkey[63]. La red eDonkey surge como una red P2P para compartir archivos de gran tamaño, las primeras versiones de eMule aparecen en 2002. En la actualidad el cliente eMule es más popular que el original eDonkey, incluyendo mejoras y apartamientos del protocolo original. El programa eMule ha sido descargado por aproximadamente 85 millones de personas.

Arquitectura

La arquitectura de la red presenta una distribución jerarquizada, incluyendo clientes y servidores. Los clientes son fuentes/solicitantes, mientras que los servidores son enrutadores. Los clientes utilizan la aplicación de código abierto descargable de [68] para conectarse a la red. Para los servidores no existe software de código abierto, siendo la opción más popular Lugdunum[148].
Cada cliente se conecta a un único servidor, informándole en todo momento de los archivos que es fuente y solicitante. Entonces, cada servidor conoce a todos los clientes que tiene conectados y los archivos que cada uno comparte, por tanto puede responder de forma eficiente a consultas de contenidos de todos sus clientes. Existen muchos servidores disponibles en Internet, su popularidad varía notablemente, mientras que algunos tienen millones de usuarios, otros solo unas pocas decenas.

eMule utiliza puertos TCP y opcionalmente UDP para su comunicación (los puertos habituales de control pertenecen al rango 4661-4665, sin embargo puede utilizar puertos variables), dependiendo de las opciones de seguridad disponibles para los clientes, estos tienen la capacidad de aceptar conexiones de otros pares o solamente iniciarlas.

Existen tres identificadores en la red:

- ***File ID***: es un hash del archivo utilizando MD4. Básicamente se divide al archivo en partes de 9,28Mb, para cada parte se hace un hash y luego se combinan en el resultado.
- ***User ID***: La identificación de usuario es un número de 128 bits, basado en una serie de números aleatorios creado al momento de la instalación del programa.
- ***Client ID***: 32 bits que identifican la sesión de un usuario. Asignado por el servidor, existen dos posibilidades: asignar un *high ID* o un *low ID*. Cuando un cliente tiene la capacidad de aceptar conexiones TCP por parte de sus pares

entonces el servidor le asigna un *high ID*, en otro caso le asigna un *low ID*. Las sesiones con *high ID* tiene una situación preferencial respecto a las *low ID* para la descarga de archivos. Un *high ID* se forma a partir de la dirección IP del cliente (si la *IP=A.B.C.D* entonces $highID=A+2^{8}B+2^{16}C+2^{24}D$), la asignación de un *low ID* parece ser arbitraria y siempre es un numero menor a 16777216. Un cliente con *low ID* puede comunicarse solamente con clientes con *high ID*, utilizando un sistema de *callback* para iniciar conexiones.

Cuando un cliente realiza una búsqueda, ésta se envía al servidor al cual se encuentra conectado, dicho servidor responde los archivos y clientes que conoce con esa descripción. El cliente puede realizar una consulta global en la red consultando secuencialmente a los servidores conocidos de forma de obtener más respuestas (con un costo alto en consumo de ancho de banda). Los servidores se intercomunican de forma continua para conocer el estado de la red y mantener la red conexa.
Existe claramente una red de acceso formada por la conexión de los clientes a los servidores y una red dorsal formada por la comunicación entre servidores, vea la Ilustración 1.

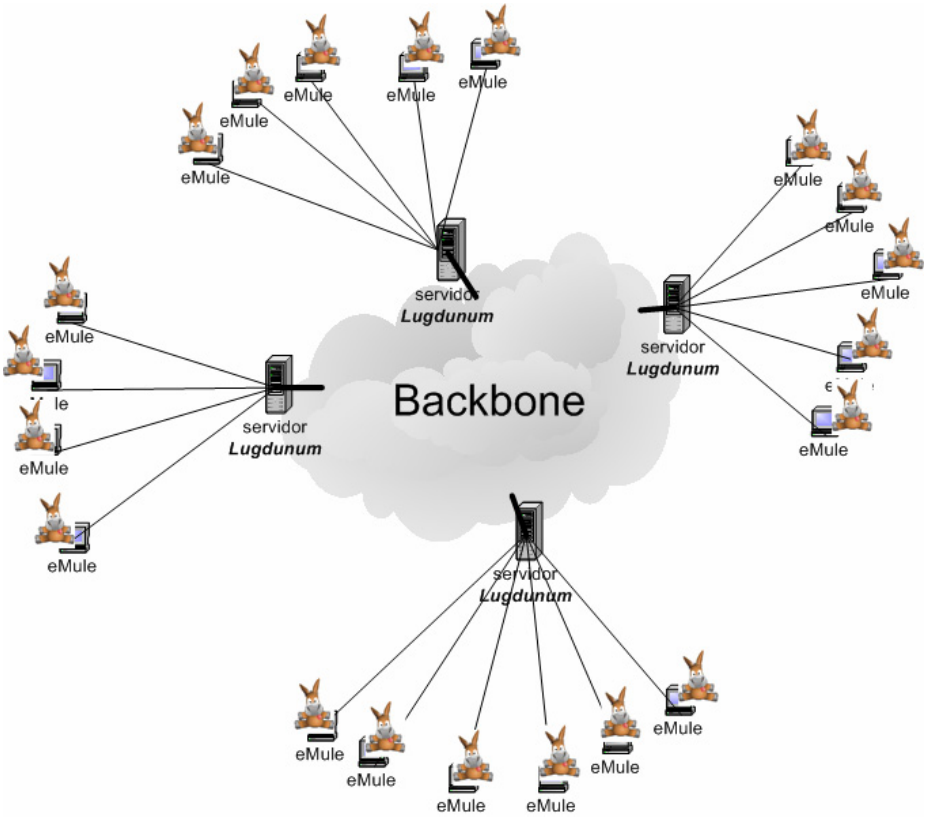


Ilustración 1. Arquitectura de la red eMule.

En la actualidad la red presenta un mecanismo complementario para realizar las búsquedas y para descubrir las fuentes de un contenido: conocido como Overnet o red Kad. En este mecanismo los clientes eMule y eDonkey además implementan el algoritmo Kademlia[153], utilizando un mecanismo de tabla de hash distribuida, asignando responsabilidad a cada cliente sobre un rango de archivos (este algoritmo elimina la necesidad de servidores en la red).

Luego de recibir las fuentes de un contenido, el cliente comienza con una comunicación directa con cada fuente en busca de la descarga de dicho archivo. Este método es sumamente complejo, incluye:

- credenciales para una segura identificación de los pares,
- un sistema de puntajes para clasificar a los solicitantes según su grado de participación en la red,
- una política de encolamiento por parte de la fuente para tener criterios justos de elección entre solicitantes de un mismo archivo,
- un sistema de créditos entre pares para agilitar el acuerdo en las descargas,
- un sistema para la recuperación de descargas corruptas (actualmente dos opciones en uso: ICH-Intelligent Corruption Handling y recientemente AICH).

La descarga de los archivos se realiza de a partes (un archivo se divide en partes de 9,28Mb), cuando una fuente le permite a un solicitante descargar el archivo, este le debe solicitar que partes requiere (hasta 3 partes pueden ser solicitadas). Primero son descargadas las partes que no se encuentran muy disponibles en la red (de forma de asegurar la persistencia del archivo frente a desconexiones de las fuentes) y luego las partes útiles para vistas previas (por ejemplo con las primeras partes de un archivo de música puede escucharse un fragmento de la canción).
Este trabajo hace hincapié en el descubrimiento del contenido en las redes de contenido, por tanto no es necesario mayor conocimiento sobre los mecanismos complejos de descarga del contenido, por mayor información sobre este importante aspecto del protocolo referirse a [68].

KaZaA [127]

Otro ejemplo de arquitectura de red de pares es KaZaA[127], esta red es objeto de estudio en capítulos siguientes.

KaZaA Media Desktop es una aplicación P2P para compartir archivos introducida en 2001. En sus comienzos captó los usuarios de Napster y Morpheus, compartiendo principalmente archivos de música MP3 (en la actualidad se utiliza también para archivos de gran tamaño como videos o películas).
De descarga gratuita, es una aplicación altamente popular a pesar de incluir en su instalación aplicaciones maliciosas (adwares, spywares y malwares).

El protocolo de red utilizado por KaZaA es conocido como FastTrack. Algunas otras aplicaciones utilizan esta red (K-Lite, KaZaA Lite, iMesh Light, Grokster Lite, Mammoth, Grokster, iMesh, Morpheus, MLDonkey, etc.) Shareman[127], la empresa creadora de KaZaA, considera ilegales a algunas de estas aplicaciones.

Arquitectura

La arquitectura de FastTrack es la siguiente. Existen dos tipos de nodos: los peers y los superpeers. Cada peer se conecta a aproximadamente tres superpeers por redundancia, informando de los archivos que desea compartir.
La búsqueda de archivos y fuentes la realiza el cliente sobre los superpeers que tiene directamente conectados, estos responden las respuestas conocidas y difunden la consulta a otros superpeers de forma de obtener más respuestas.
La aplicación KaZaA puede funcionar en las dos modalidades: peer y superpeer, no siendo necesario otro software para sustentar la red. Según la disponibilidad de recursos de un peer este puede convertirse en superpeer de forma inadvertida o manual.
La aplicación funciona utilizando el puerto TCP o UDP 1214, aunque en las versiones más recientes pueden utilizar puertos aleatorios.

Al igual que para la red eMule, existe una red de acceso formada por la conexión entre los peers y los superpeers y una red dorsal formada por la comunicación entre superpeers, vea la Ilustración 2. La principal diferencia con la red eMule es que la red

dorsal tiene una mayor carga de comunicación, debido a que es la responsable de difundir las solicitudes a toda la red (en la red eMule esto quedaba a cargo del cliente en la red de acceso)

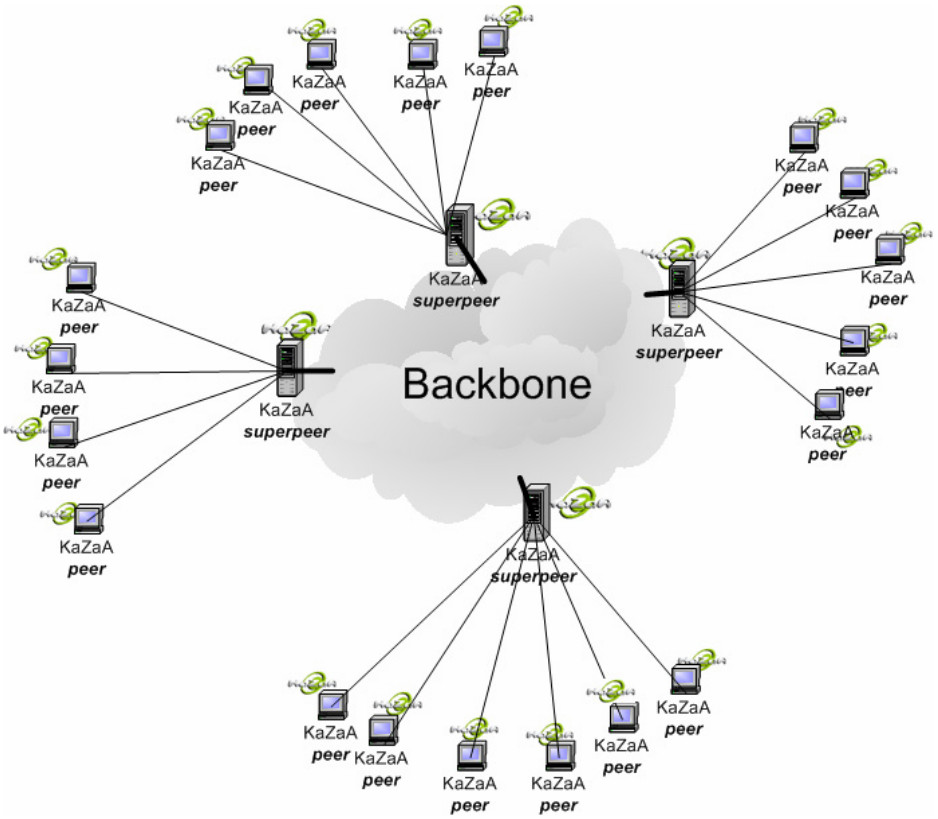


Ilustración 2. Arquitectura de la red KaZaA.

Esta red también presenta mecanismos complejos para la descarga de archivos luego de descubierta las fuentes.
La descarga entre pares se hace utilizando protocolo HTTP, las comunicaciones son encriptadas, la recuperación debido a descarga de partes corruptas se realiza utilizando el algoritmo UUHash y presenta un mecanismo para medir la participación de los pares distinto al sistema de créditos utilizado por eMule.

KaZaA en el marco de las redes de contenido

La red KaZaA presenta agrupación sintáctica y colocación insensible al contenido, su distribución es jerarquizada como se explicó anteriormente. Presenta agrupación sintáctica porque los identificadores de contenido se basan en un hash del archivo y no en el significado del mismo. No existe colocación física del contenido porque la red presenta autonomía total en las fuentes (estas puede alojar el contenido que desee, el tiempo que lo deseen).

2.3 Sistema de Nombre de Dominios

En esta sección se introduce al funcionamiento y arquitectura de la red de contenido DNS (Domain Name System)[157][158]. En capítulos posteriores se utiliza el sistema DNS como caso de estudio. Por mayor detalle sobre DNS puede consultarse [5] y [145].

2.3.1 Introducción

El objetivo del sistema de nombres de dominios es traducir nombres en direcciones IPs (y viceversa). Esta necesidad surge simplemente del hecho que para las personas es más sencillo recordar nombres que números[8]:

www.fing.edu.uy <-> 164.73.32.3

Por esta razón, el sistema de nombre de dominios acompaña el despliegue de la Internet prácticamente desde sus orígenes (la mayoría de los servicios de Internet utilizan el DNS: por ejemplo e-mail, HTTP, TELNET, FTP, etc.).

Historia

Respecto a la Internet:

- En los años 60´s surge la red ARPANET, un proyecto del Departamento de Defensa de E.E.U.U. pensado para mejorar las comunicaciones entre los grupos de investigación y desarrollo de las Universidades y centros militares.
- En los años 80's el protocolo TCP/IP se convierte en el estándar de ARPANET, además de crecer notoriamente el número de redes y máquinas conectadas en red. ARPANET se convierte en el backbone de las redes locales y regionales basadas en TCP/IP, comienza a llamarse a esta red: Internet. En 1988, ARPA decide acabar con el proyecto y la NSFNET reemplaza a la red ARPANET para convertirse en la nueva red dorsal de Internet.
- En 1995, NSFNET es desplazada por backbones comerciales: PSINet, UUNET, etc.

Respecto al Sistema de Nombres de Dominios:

- En los años 60's el sistema de nombres se encuentra basado en la distribución periódica de un archivo *HOSTS.TXT* (archivo estándar Unix que incluye la traslación entre direcciones IPs y nombres de dominios). La entidad encargada de mantener este archivo es el SRI-NIC de Standford, permitiendo solicitar cambios por correo electrónico y la descarga de la última versión actualizada por FTP. Es de imaginarse que este sistema no ofrece ninguna posibilidad de escalamiento: presenta sobrecarga en el sitio central para distribuir archivo maestro y colisiones e inconsistencias en la asignación de nombres de dominios.
- Paul Mockapetris se encarga de rediseñar completamente el servicio de nombres de dominios. En 1983 se crean las RFCs: 882, 883. Sustituidas en 1987 por las actualmente vigentes 1034, 1035. Hasta la actualidad se han estandarizado muchos más aspectos sobre el sistema de nombres, ejemplos importantes de algunas clarificaciones y extensiones son las RFCs 2181 y 2535.

Para resolver los problemas de sobrecarga, inconsistencia y colisiones del sistema precario basado en el archivo *HOSTS.TXT*, el actual Sistema de Nombres de Dominio (DNS por sus siglas en inglés - Domain Name System) incluye:

- un ***sistema distribuido***, capaz de dividir la carga de las solicitudes.
- presenta ***delegación de administración*** en subdominios, de forma que no se permite colisiones entre asignación de dominios puesto que cada entidad administrativa se encarga de distintos nombres.
- es un ***sistema on-line*** de fácil actualización, disminuyendo la inconsistencia entre distintas versiones del contenido.

[8] ¿Que sucedería si por razones de administración es necesario cambiar la dirección IP de un servicio?

¿Como es un nombre de dominio?

Es un conjunto de etiquetas separadas y (opcionalmente) finalizadas por el delimitador punto ¨.¨. Por ejemplo: *"www.fing.edu.uy."*.
Según la RFC1034, las etiquetas se encuentra formadas por caracteres: letras (ASCII), números y "-". Una etiqueta debe comenzar siempre con una letra, y cada etiqueta puede llevar hasta 63 caracteres. El nombre de dominio en total puede tener hasta 255. Y puede haber hasta 127 niveles [9].
Algunos ejemplos inválidos de dominios pueden ser:

- *españa.es.*
- *-elguion.com.*
- *www.elguión-.com.*
- *3com.com.*
- *www.3com.com.*
- *08002030.com.uy.*

2.3.2 Bases Conceptuales

Como se dijo previamente el DNS es una base de datos distribuida de nombres de dominios. Todos los dominios posibles forman una estructura jerárquica en forma arborescente conocida como ***espacio de nombres***.

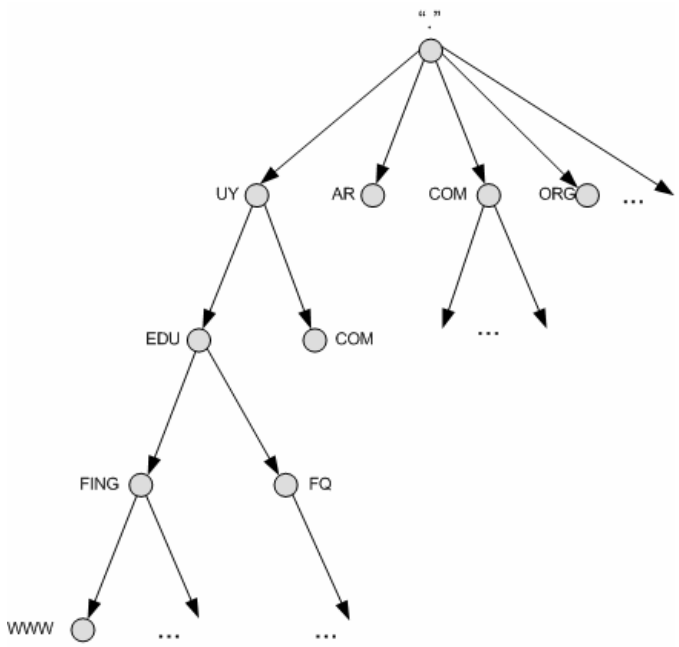


Ilustración 3. Estructura jerárquica del espacio de nombres.

El espacio de nombres es un árbol invertido donde los nodos son etiquetas. A la raíz se llama root y tiene etiqueta ¨.¨. Un nombre de dominio se lee de abajo para arriba, en la Ilustración 3 *www.fing.edu.uy*.

Un ***dominio*** es un subárbol del espacio de nombres de dominios. En la Ilustración 4 toda la zona gris es el dominio *fing.edu.uy*.

[9] Algunas extensiones al protocolo flexibilizan estas reglas actualmente.

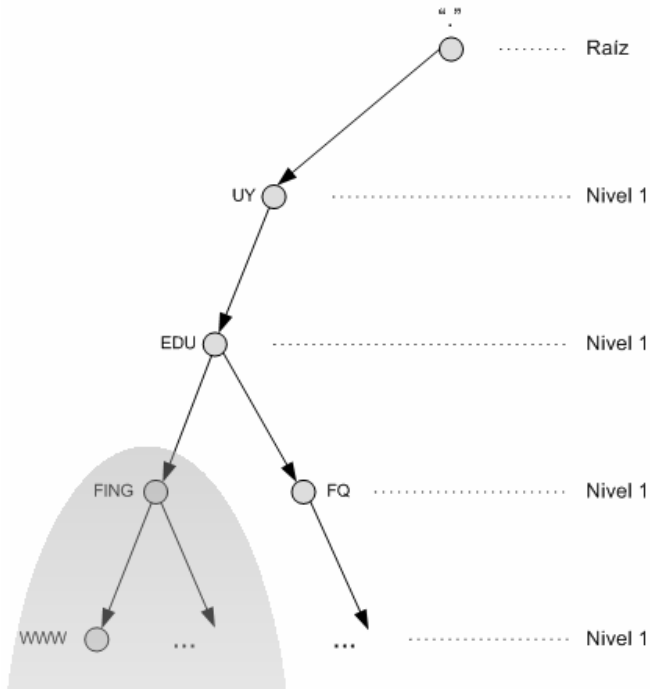


Ilustración 4. Dominio *fing.edu.uy*.

Cada etiqueta ocupa un determinado nivel en el árbol:

...tercer_nivel.segundo_nivel.primer_nivel.

Por ejemplo en la Ilustración 4: fing.edu.uy.

- *uy* dominio de primer nivel.
- *edu.uy* dominio de segundo nivel.
- *fing.edu.uy* dominio de tercer nivel.

Se dice que *fing* es hijo de *.edu.uy*. Se dice que *.edu.uy.* es padre de *fing*.

En la Ilustración 5 se observa como distintos dominios son administrados por distintas instituciones. Por ejemplo ICANN[113] es el encargado del dominio root, mientras que el SeCIU[212] del dominio *.edu.uy*. y la Facultad de Ingeniería[71] del *.fing.edu.uy*. Cada entidad administrativa cuenta con sus propios servidores de nombres, alojando la porción de la base de datos que le corresponde. De esta forma cuando una nueva consulta se realiza sobre el sistema, cada entidad responde por sus propios dominios, dividiendo de forma justa la carga de las consultas entre los servidores. También es una forma sencilla de impedir las colisiones puesto que dos instituciones pueden asignar el mismo nombre a un host (en el ejemplo: *flor*), representando distintos dominios (*flor.fing.edu.uy* y *flor.fq.edu.uy*).

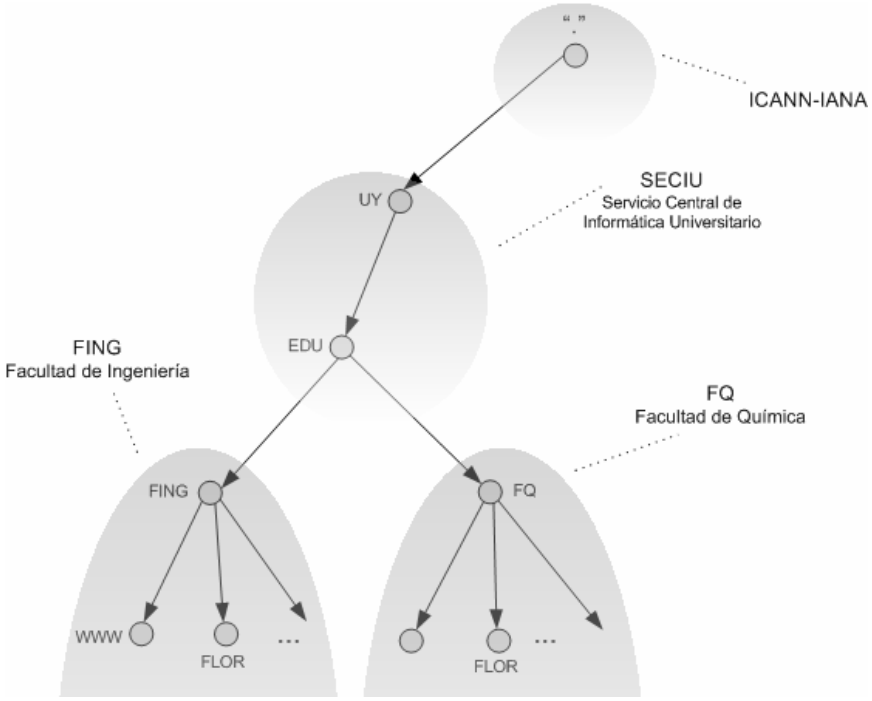


Ilustración 5. Implicancias de la estructura jerárquica.

Espacio de nombres en Internet

El espacio de nombres en Internet presenta algunas restricciones además de las dichas hasta el momento.

Domino raíz:

En sus comienzos el dominio raíz era administrado por Jon Postel[199], luego se asigno esta importante responsabilidad a IANA[112] y actualmente es ICANN la entidad que se encarga de delegar a distintas instituciones los 13 servidores que responden al dominio ".".
Los servidores raíz son identificados por las letras: A, B, C, D, E, F, G, H, I, J, K, L, M (ej. a.root-servers.net); y su ubicación es preponderantemente en E.E.U.U.:

- 8 en USA.
- 1 en Tokio.
- 4 distribuidos:
    - 1 en Europa.
    - 1 en USA y Europa.
    - 1 en Europa y Asia.
    - 1 en USA, Europa, Asia, Oceanía, África y América del Sur.

Dominios de primer nivel:

Los dominios de primer nivel son también llamados TLDs (siglas en inglés – Top Level Domain). Existen básicamente tres tipos bien difundidos:

- ***ccTLD*** dominios de cada país (country code TLD). Ejemplos *.uy, .ar*, etc. Definidos según ISO3166. Actualmente, más de 240 dominios.
- ***gTLD*** dominios de uso genérico (generic TLD): *.com, .edu, .gov, .int, .mil, .net, .org, .biz, .info, .name, .pro, .aero, .coop, y .museum.*
- ***arpa*** El subdominio más conocido es *in-addr.arpa.* utilizado para la resolución inversa de dominios.

Dominios de segundo nivel:

Los subdominios de los *gTLD* son conocidos como planos, puesto que toda etiqueta de segundo nivel es permitida. En cambio los subdominio *ccTLD* presentan dos posibilidades según decisión de la entidad administradora de cada país:

- planos, al igual que los subdominios del *gTLD*. Ejemplo *.fr*.
- replicando exclusivamente los posibles *gTLD*s. Ejemplo *.uy*. Es decir, en Uruguay se permiten los dominios *.com.uy*., *.edu.uy*., etc. y no se permite por ejemplo *empresa.uy.*

La Ilustración 6 brinda una idea del espacio de nombres posibles en Internet.

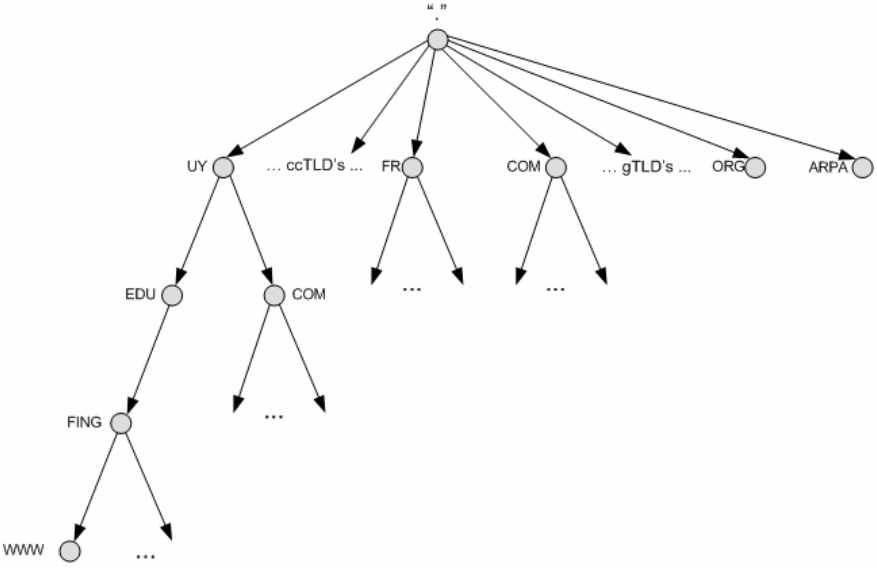


Ilustración 6. Espacio de nombres en Internet.

¿Cómo se administra el árbol?

La estructura jerárquica se administra delegando los subárboles. ***Delegar*** es asignar responsabilidades de administración de un subdominio a otra organización.
Una organización es ***autoritativa*** de un dominio si la organización de nivel superior se lo delegó. Quien administra un dominio es responsable por los subdominios (y a su vez puede delegarlos).

La delegación determina como se distribuyen los datos en la base de datos distribuida. Los servidores de nombres guardan la información de los dominios en ***zonas***. Un domino se divide en zonas para posibilitar su administración independiente. Mientras que un dominio es todo el subárbol del espacio de nombres una zona es la parte de un dominio administrada por un solo organismo.
Un servidor de nombres es autoritativo solo para los dominios que se encuentran en las zonas que administra.

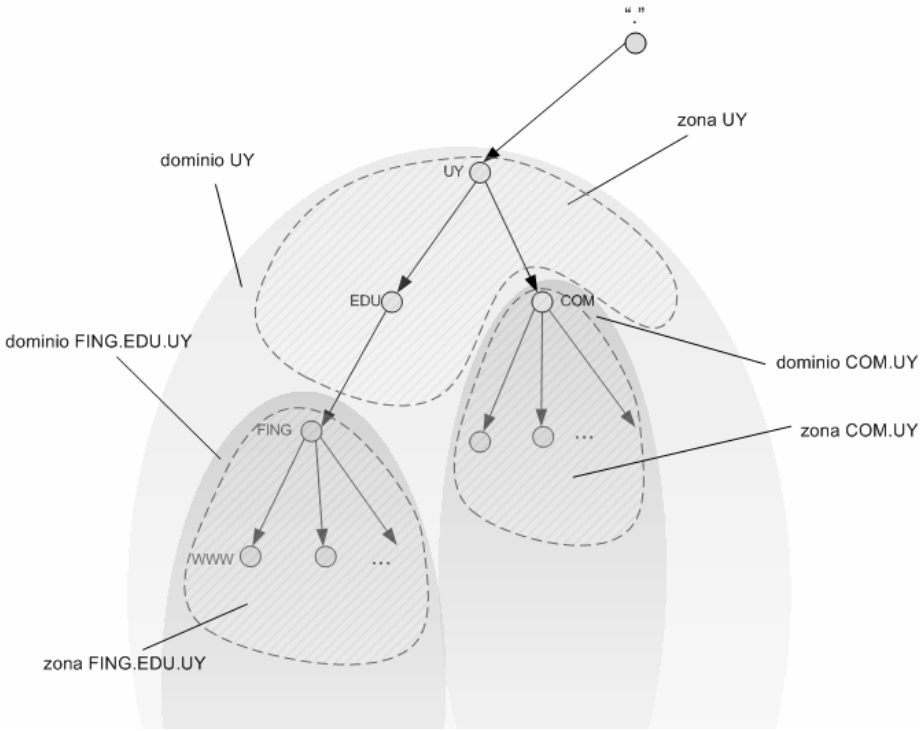


Ilustración 7. Dominio vs. zona.

El organismo SeCIU tiene delegado un subdominio *.com.uy*. a ANTEL[2]. Entonces la zona *.uy*. de SeCIU mantienen punteros de la delegación al servidor de ANTEL. Y el servidor de ANTEL mantiene la zona del subdominio *.com.uy*. Lo mismo sucede con la delegación a FING.

Arquitectura

Los servidores que componen la estructura jerárquica se llaman servidores autoritativos (debido a que responden autoritativamente a alguna zona).
Los datos asociados a un dominio se guardan en ***resource records*** (RRs) dentro de cada zona. Existen distintos tipos de RR para cada tipo de información que se desea guardar de un dominio. Los RRs más comunes son:

- ***SOA (Start Of Authority).***
- ***NS (Name Server).***
- ***A (Address).***
- ***CNAME (Canonical Name).***
- ***MX (Mail eXchange).***
- ***PTR (PoinTeR).***

En particular los registros *A* se encargan de dar la dirección IP de un dominio. Por ejemplo:

www.fing.edu.uy. IN A 200.40.30.254

Mientras que los registros NS especifican una delegación de un subdominio:

.uy. IN NS seciu.uy.

Resolución de dominios

Hasta ahora se describió que el DNS es una base de datos distribuida, donde los nodos que la componen se llaman servidores autoritativos y cada uno guarda

solamente la información que administra en unidades llamadas registros. Pero ¿como se accede de forma consolidada a esta base de datos distribuida?
El proceso de buscar una traslación de nombre de dominio a dirección IP se la conoce como ***resolución de nombre***.

Cada servidor autoritativo se encuentra obligado a responder a las consultas por la zona que administra. Si se le pregunta por un subdominio administrado por otra entidad entonces debe responder qué servidor autoritativo responde por tal subdominio. De esta forma todo dominio puede ser resuelto si se comienza a consultar desde los servidores raíz. Estos servidores dirán a quién tienen delegado cada subdominio, siendo necesario luego realizar una consulta al servidor del subdominio y así sucesivamente hasta alcanzar la respuesta deseada.

Por ejemplo en la Ilustración 7 una consulta para conocer la dirección IP del dominio *www.fing.edu.uy* requiere de los siguientes pasos:
1. consultar a algún servidor raíz por el dominio *www.fing.edu.uy.* El servidor responde que no conoce la respuesta y que se le debe consultar al administrador de la zona *.uy*.
2. se consulta al servidor autoritativo de Seciu por el dominio *www.fing.edu.uy.* El servidor responde que no conoce la respuesta y que se le debe consultar al administrador de la zona *fing.edu.uy.*
3. Se consulta al servidor autoritativo de Fing por el dominio *www.fing.edu.uy.* Como este servidor administra el dominio puede responder completamente la dirección IP deseada: *164.73.32.3.*

Por tanto toda consulta por un dominio de nivel *n*, requiere aproximadamente *n* consultas[10] en la estructura jerárquica, comenzando desde la raíz y recorriendo en profundidad hasta alcanzar el dominio buscado. Se continúa la explicación de la resolución de nombres más adelante.

La red

La red del sistema de nombres de dominios se compone básicamente de tres tipos de nodos:
- ***resolvers.***
- ***servidores recursivos.***
- ***servidores autoritativos.***

Los nodos solicitantes son todas los hosts conectados a Internet que utilizan algún servicio tradicional de comunicación, como ser el Web, e-mail, etc. Estos nodos requieren trasladar los nombres de dominio a direcciones IPs antes de utilizar dichos servicios. Para que no sea necesario que todas las aplicaciones cuenten con su propio cliente que consulte el sistema de DNS, una parte del sistema operativo, llamada **resolver**, se encarga de realizar esta tarea para todas las aplicaciones del cliente.

El resolver solo conoce una lista de servidores recursivos (comúnmente solo 2). Cuando a un cliente le surge la necesidad de resolver un dominio, el resolver realiza una consulta recursiva a uno de sus **servidores recursivos** preconfigurados.
Los servidores recursivos se encargan de realizar la búsqueda en profundidad en el árbol de **servidores autoritativos** y devolver la respuesta al resolver.

[10] Exactamente es una consulta a cada entidad autoritativa de la recorrida en profundidad del árbol. En el caso de *www.fing.edu.uy* a pesar de ser un dominio de nivel 5, solo son necesarias 3 consultas porque los dominios *.uy* y *.edu.uy* son administrados por el SeCIU y los dominios *fing.edu.uy* y *www.fing.edu.uy* por Fing.

La utilización de servidores recursivos aísla al resolver de la complejidad del sistema DNS, presentándole la información de forma consolidada al cliente final.

El proceso de búsqueda en la jerarquía arborescente es relativamente costoso en recursos de comunicación. Existen técnicas para disminuir dicho costo en los servidores recursivos. Considerando que un servidor recursivo responde a muchos resolvers es lógico pensar en reutilizar consultas realizadas anteriormente, por tanto los servidores recursivos, para entregar las respuestas a las consultas que reciben, se basan en la información que poseen o en la que aprenden en el proceso de búsqueda de la respuesta. La que aprenden, la almacenan durante cierto tiempo y se conoce como ***cache***.

La Ilustración 8 muestra los mensajes transitados en el sistema DNS para resolver el dominio *www.fing.edu.uy* . Cuando el servidor recursivo conoce la respuesta a una consulta (debido a que una consulta previa alojó en el *cache* la respuesta) la comunicación se simplifica sustancialmente (ver Ilustración 9).

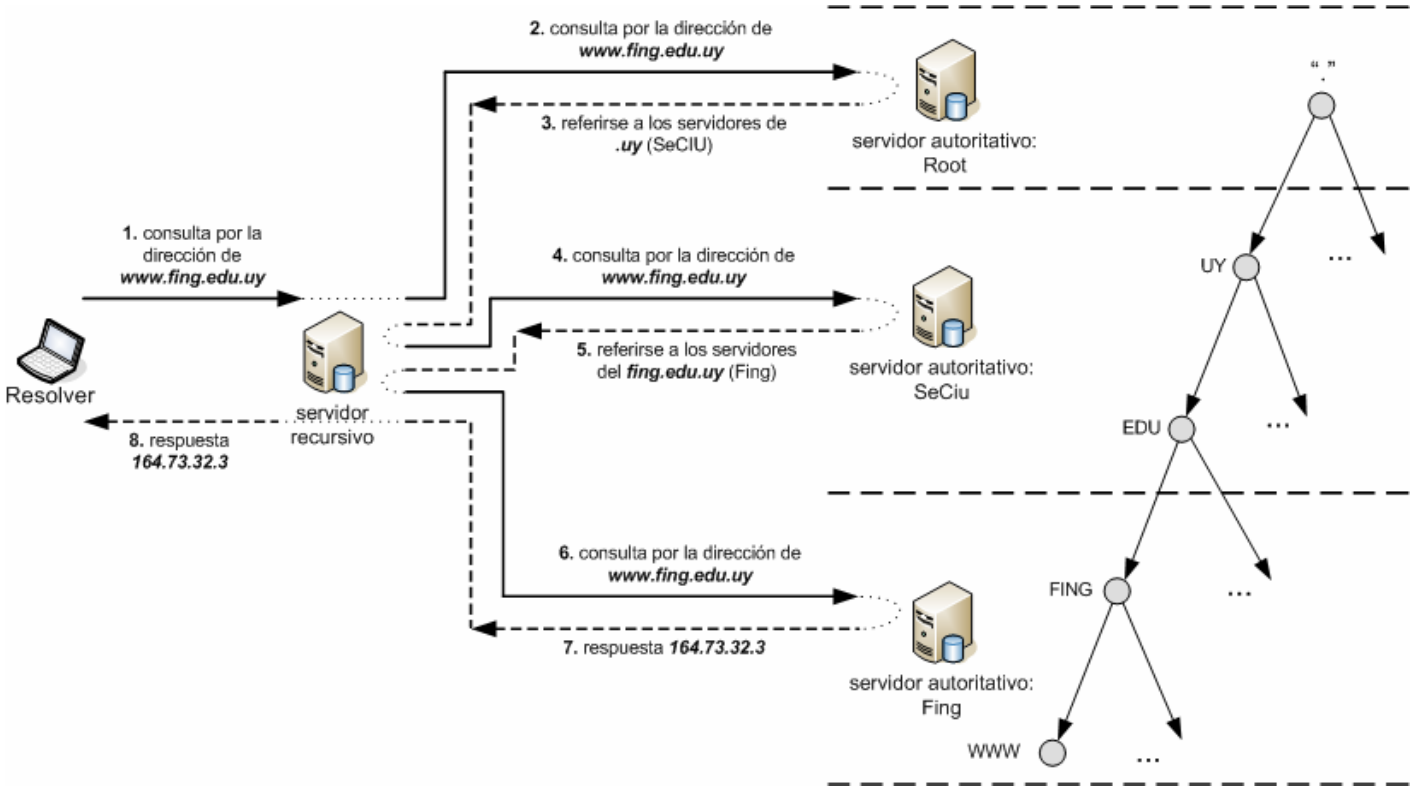


Ilustración 8. Flujo en las consultas de DNS.

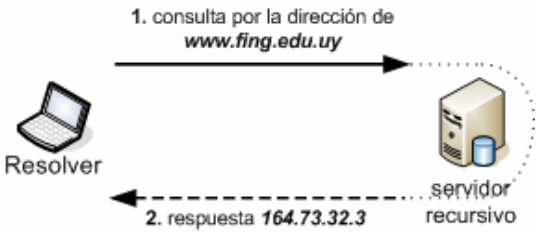


Ilustración 9. Flujo en una consulta en *cache*.

En esta sección se pretendió hacer una introducción al funcionamiento del sistema DNS debido a que es utilizado en lo que resta del trabajo como caso de estudio, a continuación se le enmarcar dentro del estudio de redes de contenido.

2.3.3 El DNS en el Marco de las Redes de Contenido

A los efectos del modelo presentado, el sistema de DNS es una red de contenido donde el ***contenido*** es simplemente la correspondencia entre los nombres de dominio y las *IPs*, es decir, los registros del *tipo A*[11].

El DNS es una red **semántica sensible** con **distribución jerarquizada**.
Existe una agrupación semántica del contenido debido a que los dominios son agrupados según la propia jerarquía de los nombres de dominio.
Cada nombre debe ser colocado en el servidor autoritativo que tiene asignado ese dominio en la estructura arborescente, por tanto existe colocación física de contenido sensible a la semántica del dominio.
Su distribución es jerarquizada por dos razones: en primer lugar los servidores autoritativos forman una estructura jerárquica repartiéndose trabajo y responsabilidades. En segundo lugar, la red claramente tiene una parte de acceso y otra dorsal; la red de acceso es la comunicación entre resolvers y servidores recursivos (comunicación simple y eficiente), mientras que la red dorsal es la estructura arborescente formada por los servidores autoritativos.

En la nomenclatura de redes de contenido los resolvers son **solicitantes**, los servidores recursivos son **enrutadores** y los servidores autoritativos son **fuentes y enrutadores** simultáneamente.
Lógicamente es posible combinar estas funcionalidades en un único host, por ejemplo es común tener servidores recursivos que a su vez sean autoritativos.
La Ilustración 10 muestra la distribución jerarquizada del sistema DNS.

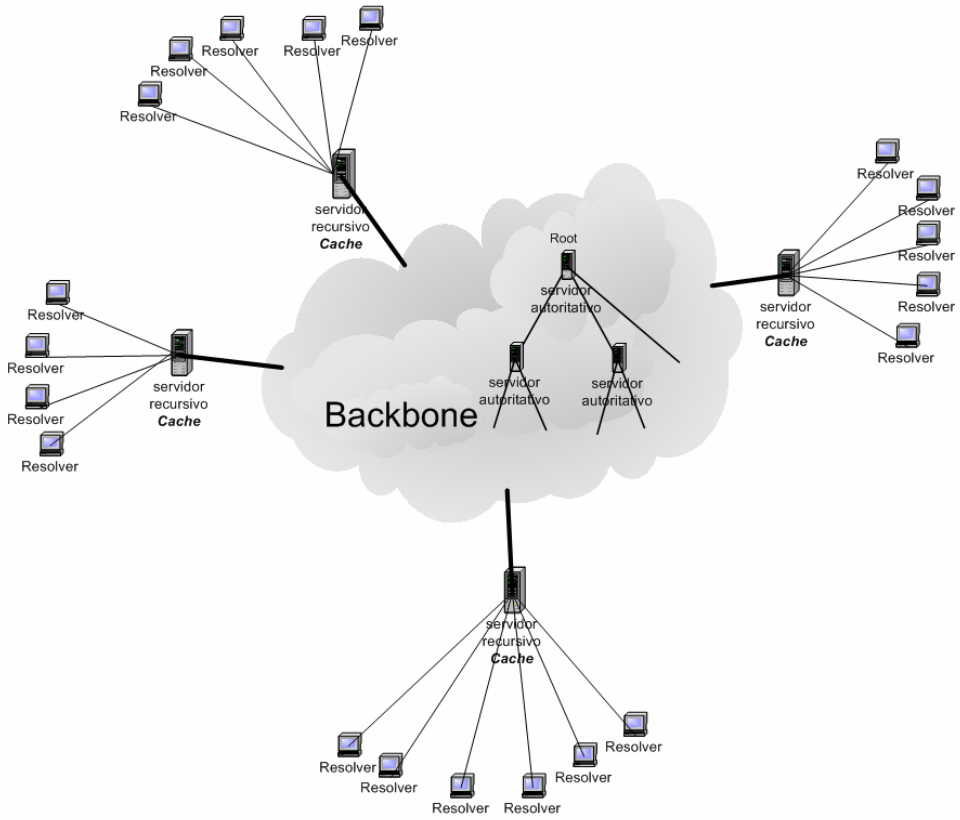


Ilustración 10. Distribución Jerarquizada del DNS.

[11] Esto elimina el problema de modelar los distintos tipos de registro y no se pierde generalidad como se verá más adelante.

2.4 Discusión y Conclusiones

En este capítulo se intenta resumir el estado del arte de la tecnología en las redes de contenido. Para ello se presenta definiciones, taxonomías, y ejemplos extraídos de la bibliografía.
Dada la amplitud del objeto de estudio, solo se profundiza en la descripción de dos redes:

- una red de pares jerarquizada para compartir archivos (como FastTrack utilizada por KaZaA[127]) y
- el sistema de nombres de Internet (DNS).

Estas redes son utilizadas en los siguientes capítulos.

Conocido el estado del arte tecnológico de las redes de contenido es posible plantearse distintas problemáticas que todas presentan en común. Algunos de los problemas identificados se presentan en 2.4.1, en particular el problema del descubrimiento del contenido en estas redes es descrito en 2.4.2 y es el motivador del resto del trabajo presentado en este documento.

2.4.1 Problemas Asociados a la Arquitectura de las Redes de Pares

Como se especificó anteriormente en una red de pares no es conveniente la colocación de contenido indiscriminada, debido a que si ese fuera el caso los nodos participantes perderían demasiada autonomía. Para las redes con colocación física de contenido existe una variedad más amplia de problemas de diseño que los presentados en esta sección.

Descubrimiento o Publicación del Contenido

Bajo la hipótesis de carencia de colocación de contenido, frente a la necesidad de algún objetivo específico de diseño solo es posible optimizar la topología de la red. En una red de contenido la topología de la red se encuentra dada por el conocimiento de alojamiento de los contenidos, es decir, la meta-información sobre que nodo fuente aloja que contenido. Esto es exactamente el método de descubrimiento o publicación del contenido que posee la red.

Las redes de contenido más difundidas presentan un modelo jerarquizado de distribución. Existen entonces dos posibilidades de modificar la topología: relajar el modelo jerarquizado o mantenerlo y modificar alguna otra propiedad.

Relajando el modelo jerarquizado

Las redes superpeer se constituyen de una red de acceso, donde los nodos solicitantes se conectan a uno o varios superpeers (por redundancia) y una red dorsal entre nodos superpeers.

En general la red dorsal maneja un tipo de distribución pura; existen varias propuestas (Freenet[38], Past [61], CAN[201], Chord[225], etc.) para cambiar este parte de la red por alguna versión estructurada.
Estudios de simulación y teóricos se han hecho sobre el comportamiento de una red con estas características, pero pocos incluyen explícitamente la optimización de alguno de sus variables de decisión.
Sin lugar a dudas este es un problema que atrae a la comunidad académica, pero por su gran estudio ha sido descartado como objeto de trabajo en esta tesis.

Pensar en un modelo jerarquizado de varias capaz (no solamente dos: acceso y dorsal) puede ser de gran interés para redes de muchas prestaciones y gran tamaño. Tampoco se explora esta opción en este trabajo.

Mejorar la red de acceso también es un desafío importante. Si la red dorsal usa un modelo puro de distribución entonces existe un *TTL* máximo para la búsqueda de contenido. Si los solicitantes se conectaran a varios superpeers, y estos se encontraran lejanos en la red dorsal, entonces se podría disminuir el *TTL* necesario para las búsquedas y de esta forma mejorar la performance de la red.

Manteniendo el modelo jerarquizado.

Cambiar el *TTL* y la cantidad de conexiones por nodo en la red dorsal ha sido de estudio en varios trabajos previos. Utilizando métodos de simulación o teóricos se puede estimar la performance de la red para distintos valores de estos parámetros.

Usando técnicas de auto-ordenamiento de la meta-información del contenido puede mejorarse enormemente la escalabilidad de la red.
Según el tipo de red de contenido, existen dos grandes opciones de auto-ordenamiento: basarse en la semántica del contenido o no.
Si existe información semántica, entonces las mejoras por auto-ordenamiento son variadas. Por ejemplo si algunos superpeers se especializan en algún tipo de contenido, entonces los solicitantes deberían conectarse a ellos cuando buscan dicho contenido. La misma situación puede darse en la elección de conectividad entre superpeers.

Por otro lado si no existe información semántica también se pueden optimizar la red mediante técnicas de auto-ordenamiento, prueba de ello son los variados estudios académicos sobre redes estructuradas (Freenet[38], Past [61], CAN[201], Chord[225], etc.).

Buscando una aplicación más general, en todos los tipos de redes de contenido estudiados existe libre colocación de meta-información de contenido en la red. Por meta-información debe entenderse el conocimiento de que nodo fuente aloja que contenido. En esta situación debe existe un equilibrio entre descubrir y publicar la información dado por la dinámica de los participantes de la red.
Dada su basta aplicación, este problema es descrito con más detalle en 2.4.2.

Distribución del Contenido

Cuando ya se ha descubierto un contenido específico entre todos los nodos solicitantes y fuentes, es decir que se ha formado un subgrafo bipartito completo entre las fuentes y los solicitantes de dicho contenido; surge el problema de lograr una distribución eficiente del contenido desde las fuentes a los solicitantes.
Por eficiencia puede entenderse: minimizar el tiempo de entrega global, minimizar el tiempo de entrega promedio, maximizar la probabilidad de que todo solicitante obtenga el contenido, etc. Las restricciones para la distribución son el ancho de banda y la posibilidad de desconexión de las fuentes.

Existen varias técnicas para resolver esta problemática, para las redes de pares en general se incorpora el concepto de grado de participación de un nodo solicitante, donde a mayor participación de un nodo mayor probabilidad de obtención de contenido. El grado de participación se mejora manteniendo largas conexiones en la red y compartiendo mucho contenido.

A su vez para las redes de distribución de archivo se acostumbra a dividir los contenidos en piezas de pequeño tamaño, para que sea menos costoso para las fuentes compartir un archivo y para aprovechar la descarga desde varias fuentes simultáneamente.

La distribución de contenido es un problema muy atractivo y desafiante para su modelado y su optimización.

Efecto de Red y Relación de Competencia

Suponiendo varias redes de contenido ofreciendo el mismo tipo de contenido es posible pensar en un juego de competencia entre dichas redes para obtener la mejor red para los solicitantes.

Este es el caso de las redes para compartir archivos, donde en la actualidad existen varias redes desconexas que compiten por los usuarios. Cuando un usuario se suma a una red como solicitante también lo hace como fuente, por tanto aumentando la disponibilidad de contenido y el valor de la red. Este efecto, llamado efecto de red en este documento, presenta un desafío interesante de modelado en el contexto de teoría de juegos.

Los usuarios tendrían un comportamiento de pertenecer a aquella red que les diera mayor beneficio (posiblemente pertenecer a más de una red sea también una opción). Lograr un equilibrio entre redes competidoras posiblemente requiera una adaptación dinámica de cada una de las redes a la cantidad de nodos y contenidos que presenta en un momento dado, de forma de mejorar la percepción al usuario final y no desaparecer. Se hace explicito en este modelo la importancia de la características de escalado de la red.

Probablemente estudiar un modelo de competencia entre redes basado en teoría de juegos; con cada red optimizándose para dar una mejor experiencia al usuario según su tamaño, sea un camino interesante de investigación para modelar este problema.

Otras Problemas Asociados

Lógicamente existen otros importantes y complejos problemas en el diseño de las redes de pares. Por ejemplo la seguridad, el anonimato, la interoperabilidad, etc. escapan a la motivación de este trabajo.

2.4.2 Problema de Estudio: Descubrimiento del Contenido en una Red de Contenido

En esta sección se introduce al lector al problema motivador del trabajo de tesis.
Este problema es la distribución eficaz y eficiente de la meta-información en una red distribuida de contenido.

Existe una topología de red subyacente conocida (Internet), esta red es modelada con un grafo completo, puesto que toda *IP* puede alcanzar a otra *IP* en la red.
De forma simplificada, en esta red subyacente no se presentan diferencia de costo, retardo o ancho de banda efectivo entre distintos enlaces, presentando a los pares de forma igualitaria frente a la red subyacente.

Una red de contenido se monta de forma virtual sobre la red Internet.
Los nodos tienen un identificador único en la red de contenido, supongamos su *IP*.
Los nodos presentan un comportamiento dinámico en la red, con libertad para

conectarse y desconectarse. Además los nodos fuentes presentan autonomía para alojar y desalojar su propio contenido también de forma dinámica.

Cada contenido tiene un identificador único, llamado *ID*. La meta-información de la red de contenido representa que nodo fuente aloja que contenido, es decir un índice que es la relación *ID-IP*.
Dependiendo del tipo de red de contenido, un *ID* puede representar a un contenido puntual o a un grupo de contenidos, por ejemplo en las redes para compartir archivos en general el identificador es un CRC (cyclical redundancy check) del archivo, para redes semánticas sensibles el identificador puede representar todos los contenidos que cumplan con una característica (por ejemplo todas las canciones de un compositor).

En esta situación, la red de contenido virtual es una red de conocimiento de ubicación del contenido, es decir, una locación dada de índices. Una instancia de red virtual esta establecida por una tabla de índices para cada nodo (lo que puede verse como una única tabla general: *"IP_tabla, ID-IP"* donde el nodo *IP_tabla* conoce el alojamiento *ID-IP*).
Para que un nodo pertenezca a la red, tiene que tener su tabla de índices no vacía, y para que sea fuente tiene que tener algún índice con su propia *IP*.

En las redes no-estructuradas no existe conocimiento global sobre la localización de los índices, quizás si sobre algunos aspectos de la topología (no existe libre colocación de contenido, solo libre colocación de índices). En las redes estructuradas o semi-estructuradas existe una función que determina la *IP* (o grupo de *IPs*) a partir del contenido *ID* (en estas redes existe libre colación de contenido o se busca una topología de red que respete la relación). Por tanto este problema modela los dos tipos de redes.

El descubrimiento de índices entre nodos se realiza mediante dos primitivas de comunicación: la publicación y la búsqueda:

- ***publicación***: envío de uno o varios índices.
- ***búsqueda***: solicitud de uno o varios índices.

Cada nodo puede mandar y recibir una cantidad limitada de primitivas por unidad de tiempo. De forma general existe un ancho de banda asimétrico y diferente entre todos los nodos.

Los nodos se caracterizan por solicitar (intentar descubrir) distinto contenido a lo largo del tiempo. La performance global de la red determina el éxito o beneficio de la misma. Una forma de medir dicha performance puede ser el porcentaje de índices existentes en la red que son descubiertos por todos los solicitantes.

En los siguientes capítulos se modela y estudia este problema de forma exhaustiva.

Modelo General: Problema del Descubrimiento de Contenido

3

Como definimos anteriormente, una ***red de contenido*** es una red donde el descubrimiento, el direccionamiento y el enrutamiento del contenido se basan en la descripción del contenido en lugar de su ubicación. Por tanto, toda red de contenido es en realidad una red de conocimiento, donde el conocimiento es la información de alojamiento de cada contenido específico. Por información de alojamiento debe entenderse la meta-información que relaciona el contenido con el nodo que lo posee.
Como el objetivo de estudio son las redes **distribuidas**, el conocimiento de la meta-información se encuentra distribuido entre los nodos de la red, por lo que en general no existe una vista global de toda la meta-información.

El **objetivo** de la red es poder brindar a los solicitantes de cada contenido la mayor cantidad de nodos que lo alojan, es decir, el objetivo final de una red de contenido es tener la vista global más completa posible de la meta-información o lo que es lo mismo descubrir de la forma más eficiente el contenido.

Básicamente existen dos formas de **descubrir** la meta-información, estas formas son la publicación y la búsqueda. Por publicación debe entenderse que un nodo de la red voluntariamente decide informarle a otros de la meta-información que él conoce. Por búsqueda debe entenderse el solicitarle a otros la meta-información que posean.
Una red estructurada presenta alguna relación explicita entre la topología de la red y la meta-información de cada contenido. Una red no estructurada no presenta ninguna limitante de donde se encuentra la meta-información.

Los nodos tienen un grado de participación en la red, este grado de participación se encuentra expresado por una limitante en la cantidad de meta-información que pueden descubrir a lo largo del tiempo.

El carácter dinámico del contenido y los nodos de la red hace muy difícil la tarea de mantener la meta-información en la red, se considera que cuanto más dinámica es la red más debe descubrirse la información mediante búsquedas, puesto que el costo de la publicación sería injustificado dado que la meta-información descubierta rápidamente perdería su validez.
Desde un punto de vista más general, este capítulo presenta un modelo para mostrar que dado un grado de dinamismo en la red existe un compromiso entre la publicación y la búsqueda de meta-información.
En primera instancia se define formalmente el problema del descubrimiento de contenido en redes distribuidas (sección 3.1). Luego se instancia el problema general a ejemplos de redes de contenido reales (sección 3.2). Finalmente se discuten algunas conclusiones sobre el modelo presentado (sección 3.3).

Como resultado se tiene un modelo muy detallado sobre el comportamiento dinámico de las redes de contenido. Útil para la simulación o evaluación de las redes, este

modelo es de mucho costo computacional y por tanto impide su aplicación a problemas de diseño donde se requiera algún método de optimización. Siendo la optimización de la red uno de los objetivos de este trabajo, este modelo es simplificado en el capítulo siguiente (observando exclusivamente el comportamiento en un tipo de nodos de la red).

En este capítulo se utiliza una notación matemática que incluye varias operaciones binarias y algunas reales.
Las operaciones binarias son:

- $\sum$ representa el OR binario entre todos los sumandos.
- la siguiente tabla define la notación del resto de las operaciones binarias utilizadas:

A	B	A + B	AB	$\neg$ A	A - B	A $\oplus$ B
0	0	0	0	1	0	0
0	1	1	0	1	0	1
1	0	1	0	0	1	1
1	1	1	1	0	0	0

Tabla 4. Operaciones Binarias.

Mientras que las operaciones reales son:

- $\overset{+}{\sum}$ representa una sumatoria real entre todos los sumandos.
- $A \cdot B$ denota la multiplicación de reales.

3.1 CDP - Content Discovery Problem

En esta sección se formaliza el problema de descubrimiento de contenido en redes distribuidas, en adelante denominado por las siglas ***CDP*** **(Content Discovery Problem)**.

3.1.1 El Tiempo

El modelo debe expresar el carácter dinámico de la red (nodos y contenido) y a su vez especificar la comunicación necesaria para descubrir la información. Por tanto la representación del tiempo juega un papel preponderante en el modelo.
Se utiliza una **discretización del tiempo** en períodos llamados tiempo de ronda y denotada por t^r.
La red a modelar es una **red sincrónica**, en el sentido que el estado de la red se encuentra perfectamente definido en los tiempo múltiplos de t^r. Por tanto, todos los eventos transcurren, para todos los nodos, exclusivamente durante los períodos de ronda.
Como se verá más adelante, un evento puede ser la conexión/desconexión de un nodo, el alojamiento/desalojamiento de un contenido por parte de un nodo, la comunicación a través de mensajes con otros nodos, etc.
La única restricción explicita que presenta la red es la limitante sobre la cantidad de mensajes que puede manejar cada nodo de la red en un período de tiempo. Esto se debe a la limitante de ancho de banda que presentan los nodos.

Una suposición con gran impacto sobre el tiempo es la de **estado estacionario**, se supone que la red no se está formando, ni desintegrando, entonces la cantidad de

mensajes manejados por cada nodo en cada ronda es muy similar y por tanto puede determinarse un período de ronda constante sin afectar la dinámica natural de la red.
Esta suposición implica una hipótesis aun más fuerte: la red es **libre de congestión**. Los nodos no se ven obligados a eliminar paquetes por sobrepasar la cantidad de mensajes que puedan manejar en una ronda.

El modelo también supone que la red subyacente no presenta pérdida de paquetes, dado que todo mensaje enviado es recibido por el destinatario.
Esta suposición de no congestión dista bastante de la realidad para algunas redes de contenido, siendo necesario un estudio detallado en algunos casos para contemplar la pérdida y eliminación de paquetes.

Debido al tratamiento que se hace del tiempo, la unidad de discretización t^r, no puede ser muy pequeña dado que el modelo no intenta describir los detalles espurios de la dinámica de la red (por ejemplo el modelo no describe el encolamiento de paquetes en los nodos, y por tanto un período de tiempo muy pequeño no permitiría mandar paquetes grandes). En el otro sentido, t^r tampoco puede ser un valor muy elevado, puesto que se perderían los detalles del protocolo que se pretende evaluar.

3.1.2 El Contenido y los Nodos

El **contenido** de la red se representa con el conjunto C.

Los **nodos** de la red pertenecen al conjunto V.
Debido a la autonomía de los nodos, en general estos tienen un comportamiento de **conexión** y **desconexión**. El estado de un nodo se expresa con la variable aleatoria:

$$X_i^t \in \{0,1\} \quad \forall t, \forall i \in V$$

Donde $X_i^t = 1$ sii el nodo *i* se encuentra conectado en el tiempo $t.t^r$. Por tanto existe un evento de conexión en el período $\left[t.t^r, (t+1).t^r\right)$ sii $\left(X_i^{t+1} - X_i^t\right) = 1$ y de desconexión sii $\left(X_i^t - X_i^{t+1}\right) = 1$.
Se considera muy improbable, e innecesario de modelar, que un nodo se conecte y desconecte dentro de una ronda (a estas conexiones se les llama espurias).

Un subconjunto de nodos $K \subseteq V$, llamados terminales, tiene la posibilidad de solicitar contenidos a la red. Un nodo **solicita un contenido** en la ronda *t* según la variable aleatoria:

$$S_{i,k}^t \in \{0,1\} \quad \forall t, \forall i \in V, \forall k \in C$$

Donde $S_{i,k}^t = 1$ sii el nodo *i* solicita el contenido *k* en el período $\left[t.t^r, (t+1).t^r\right)$.

Los contenidos son alojados físicamente en los nodos. Uno de los puntos más importantes para preservar la autonomía es que los nodos puedan elegir cuando comenzar a **alojar y terminar de alojar un contenido** dado. . El estado de un contenido en un nodo se expresa con la variable aleatoria:

$$A_{i,k}^t \in \{0,1\} \quad \forall t, \forall i \in V, \forall k \in C$$

Donde $A_{i,k}^t = 1$ sii el nodo se encuentra alojando el contenido *k* en el tiempo $t.t^r$.
Por tanto existe un evento de alojamiento en el período $\left[t.t^r, (t+1).t^r\right)$ sii $\left(A_{i,k}^{t+1} - A_{i,k}^t\right) = 1$ y de desalojamiento sii $\left(A_{i,k}^t - A_{i,k}^{t+1}\right) = 1$.

Se considera muy improbable, e innecesario de modelar, que un nodo aloje y desaloje un contenido dentro de una ronda (a estos alojamientos se les llama espurios).

3.1.3 Estado de la Red

La meta-información que cada nodo conoce es una tabla de **índices**, donde un índice es una relación nodo-contenido que representaría el conocimiento de que dicho nodo tiene dicho contenido. El **estado de la red** en un momento dado es el conjunto de todas las tablas de índices de cada nodo:

$$I^t_{i,j,k} \in \{0,1\} \quad \forall t, \forall i, j \in V, \forall k \in C$$

Donde $I^t_{i,j,k} = 1$ si en el tiempo $t.t^r$, el nodo *i* aloja el índice *j-k*, y $I^t_{i,j,k} = 0$ en caso contrario.

Lógicamente un nodo conoce perfectamente el contenido que él mismo aloja, por tanto, la tabla de índices de un nodo representa en el caso particular de los índices propios al contenido alojado, esto es:

$$I^t_{i,i,k} = A^t_{i,k} \ \forall t, \forall i \in V, \forall k \in C .$$

No existen restricciones sobre la validez de los índices alojados para el caso general, por tanto es posible (aunque lógicamente no deseable) que $I^t_{i,j,k} = 1$ a pesar de que $I^t_{j,j,k} = 0$. Esto determina que sea necesario por parte de cada nodo tomar la decisión de que un índice de la tabla expire.

El criterio de **expiración de índices** depende de la política de la red, es decir, de cada implementación en particular, esto se representa con la variable:

$$E^t_{i,j,k} \in \{0,1\} \quad \forall t, \forall i, j \in V, \forall k \in C$$

Donde $E^t_{i,j,k} = 0$ representa que en el período $\left[t.t^r, (t+1).t^r\right)$ el nodo *i* borrará el índice *j-k* de su tabla (si $E^t_{i,j,k} = 1$ entonces no se realiza acción sobre el índice).

Algunas medidas interesantes sobre el estado de la red son:

$$\begin{pmatrix} \text{cantidad de contenidos} \\ \text{alojados por el nodo } i \end{pmatrix} = \sum_k^+ I^t_{i,i,k}$$

$$\begin{pmatrix} \text{tamaño de la} \\ \text{tabla del nodo } i \end{pmatrix} = \sum_{j,k}^+ I^t_{i,j,k}$$

$$\begin{pmatrix} \text{índices válidos} \\ \text{alojados por el nodo } i \end{pmatrix} = \sum_{j,k}^+ I^t_{i,j,k} I^t_{j,j,k}$$

3.1.4 Descubrimiento del Contenido: Mensajes y Restricción de Ancho de Banda

El **descubrimiento** de índices entre nodos se realiza mediante dos primitivas de comunicación: la publicación y la búsqueda.

La **publicación** es el envío de uno o varios índices:

$$P^t_{m,i,j,k} \in \{0,1\} \quad \forall t, \forall m, i, j \in V, \forall k \in C$$

Donde $P^t_{m,i,j,k} = 1$ sii el nodo *m* le envía al nodo *i* el índice *j-k*, en el período $\left[t.t^r, (t+1).t^r\right)$ ($P^t_{m,i,j,k} = 0$ en caso contrario).

Se considera que los nodos publican solo aquellos índices que se encuentran en su propia tabla. Desde un punto de vista matemático esta restricción se expresa:

$$P^t_{m,i,j,k} \le I^t_{m,j,k} \quad \forall t,\ \forall m,i,j \in V,\ \forall k \in C$$

Esto implica que los nodos no engañan a los otros nodos (inventando indices falsos), y además determina que no existe otra forma de comunicación entre nodos que la aquí modelada.

La **búsqueda** es la solicitud de uno o varios contenidos:

$$B^t_{i,j,k} \in \{0,1\} \quad \forall t,\ \forall i,j \in V,\ \forall k \in C$$

Donde $B^t_{i,j,k} = 1$ sii el nodo *i* le solicita al nodo *j* el contenido *k*, en el período $\left[t.t^r, (t+1).t^r\right)$ ($B^t_{i,j,k} = 0$ en caso contrario).

La respuesta a una búsqueda es una publicación.

Existe una limitación en la cantidad de descubrimiento que cada nodo puede realizar en un período de tiempo t^r, esta limitante es conocida como **ancho de banda**, y en el caso general es asimétrico. Las primitivas de comunicación presentan un tamaño en bytes dado por las constantes α (bytes necesarios para publicar un índice) y β (bytes necesarios para buscar un índice).

Para el caso general la limitante se presenta con las siguientes ecuaciones:

$$BW_{IN,i} \cdot t^r \ge \alpha \cdot \sum^{+}_{\mathrm{m,j,k}} P^t_{m,i,j,k} + \beta \cdot \sum^{+}_{\mathrm{m,k}} B^t_{m,i,k} \quad \forall t, \forall m,i,j \in V, \forall k \in C$$

$$BW_{OUT,i} \cdot t^r \ge \alpha \cdot \sum^{+}_{\mathrm{j,m,k}} P^t_{i,j,m,k} + \beta \cdot \sum^{+}_{\mathrm{j,k}} B^t_{i,j,k} \quad \forall t, \forall m,i,j \in V, \forall k \in C$$

que básicamente describen que en un tiempo t^r todos los nodos deben recibir menos bytes que su ancho de banda entrante y deben enviar menos bytes que su ancho de banda saliente.

Se supone que cada nodo hace un uso eficiente de su canal de comunicaciones, es decir, el modelo no intenta representar la dinámica en un lapso muy pequeño de tiempo, sino el comportamiento de la red a largo plazo. Durante el tiempo t^r cada nodo utiliza alguna política de encolamiento para manejar los paquetes que recibe y envía de forma tal que se satisfagan las restricciones señaladas anteriormente sin ser necesario descartar paquetes por congestión.

3.1.5 Transición de Estado

Se han definido los conceptos generales para el modelo del problema de descubrimiento. Es importante determinar la **transición de estados** de forma general:

$$I^{t+1}_{i,j,k} = \begin{cases} A^{t+1}_{i,k} X^{t+1}_i & \text{si } i = j \\ \left[I^t_{i,j,k} + \sum_m P^t_{m,i,j,k} \right] E^t_{i,j,k} X^{t+1}_i & \text{si } i \neq j \end{cases}$$

Si $i = j$ entonces, la ecuación refiere al contenido alojado en el nodo, en ese caso el estado es igual al anterior a menos que el nodo *i* decida quitar el contenido $A^{t+1}_{i,k} = 0$ o desconectarse de la red $X^{t+1}_i = 0$.

Para el caso general $i \neq j$, se mantiene un índice o se descubre un índice (si algún otro nodo lo envía $\sum_m P^t_{m,i,j,k}$) siempre que el mismo no haya expirado $E^t_{i,j,k} = 0$ o el nodo no se desconecte de la red $X_i^{t+1} = 0$.

3.1.6 Objetivo de la Red

El **objetivo** de la red es maximizar el descubrimiento de contenido, esto significa encontrar la mayor cantidad de índices correctos respecto a los contenidos solicitados.
Dado que la red se encuentra en un estado estacionario basta con observar las solicitudes en una única ronda (el comportamiento entre distintas rondas debe ser muy similar).
Según el tipo de contenido y las características particulares de la red, tiene sentido esperar por la información de alojamiento una cantidad máxima de rondas $r_{\max}$.
En forma matemática el objetivo del problema es:

$$\max \mathrm{E}\left\{\sum_{i,k}^{+}\left[S^t_{i,k}\sum_{j}^{+}\left(\sum_{r=1}^{r_{\max}} I^{t+r}_{i,j,k} I^{t+r}_{j,j,k}\right)\right]\right\}$$

Para todos los nodos *i*, que solicitan algún contenido *k*, maximizar el valor esperado de la obtención de índices *j-k* validos en el período de $r_{\max}$ rondas.

3.1.7 Resumen

Definición de ***CDP*** (Content Discovery Problem):

$$\max \mathrm{E}\left\{\sum_{i,k}^{+}\left[S^t_{i,k}\sum_{j}^{+}\left(\sum_{r=1}^{r_{\max}} I^{t+r}_{i,j,k} I^{t+r}_{j,j,k}\right)\right]\right\}$$

sujeto a:

$$I^{t+1}_{i,j,k} = \begin{cases} A^{t+1}_{i,k} X^{t+1}_i & \text{si } i = j \\ \left[I^t_{i,j,k} + \sum_m P^t_{m,i,j,k}\right] E^t_{i,j,k} X^{t+1}_i & \text{si } i \neq j \end{cases}$$

$$BW_{IN,i} \cdot t^r \geq \alpha \cdot \sum_{\mathrm{m,j,k}}^{+} P^t_{m,i,j,k} + \beta \cdot \sum_{\mathrm{m,k}}^{+} B^t_{m,i,k} \quad \forall t, \forall m,i,j \in V, \forall k \in C$$

$$BW_{OUT,i} \cdot t^r \geq \alpha \cdot \sum_{\mathrm{j,m,k}}^{+} P^t_{i,j,m,k} + \beta \cdot \sum_{\mathrm{j,k}}^{+} B^t_{i,j,k} \quad \forall t, \forall m,i,j \in V, \forall k \in C$$

$$P^t_{m,i,j,k} \leq I^t_{m,j,k} \quad \forall t, \ \forall m,i,j \in V, \ \forall k \in C$$

$$X^t_i, S^t_{i,k}, A^t_{i,k} \in \{0,1\} \text{ v.a. conocidas} \quad \forall t, \forall i \in V, \forall k \in C$$

$$I^t_{i,j,k} \in \{0,1\} \ \text{ estado} \quad \forall t, \forall i,j \in V, \forall k \in C$$

$$E^t_{i,j,k}, P^t_{m,i,j,k}, B^t_{i,j,k} \in \{0,1\} \text{ variables de decisión } \forall t, \forall m,i,j \in V, \forall k \in C$$

Ecuación 1. Formulación del modelo general *CDP*.

Resumen: Establecido un patrón de conexión, de alojamiento de contenido y de solicitud de contenido, el problema consiste en maximizar la localización del contenido solicitado, permitiendo variar la política de descubrimiento (publicación y búsqueda) y expiración de la información.

Observación: Para observar los resultados de los eventos $X^t, A^t, S^t, E^t, P^t, B^t$ que suceden en el período de tiempo $\left[t.t^r, (t+1).t^r\right)$ se debe esperar al estado siguiente I^{t+1}.

Observación: Recordar que la formulación incluye operaciones binarias y reales en su notación. Ver Tabla 4 por detalles de notación.

3.2 Aplicación del CDP a Redes Reales

Para poder evaluar la expresividad del modelo en esta sección se especifican los ejemplos más citados en la literatura de redes de contenido.

3.2.1 Napster (Modelo Híbrido) [166][167]

En primer lugar todos los nodos son terminales, es decir, solicitan contenido, excepto el servidor central. Considérese el servidor central de Napster como el nodo *0*, por tanto $K = V/\{0\}$.

Los clientes de Napster se identifican al ingreso a la red en el servidor central, informándole los archivos que comparte:

$$P^t_{i,0,i,k} = I^t_{i,i,k}(X^t_i - X^{t-1}_i)$$

Cuando una fuente cambia el contenido ofrecido, esto se le informa al servidor central:

$$P^t_{i,0,i,k} = \left(A^t_{i,k} \oplus A^{t-1}_{i,k}\right)X^t_i$$

$$E^t_{0,i,k} = \neg\left((\mathrm{A}^{t-1}_{i,k} - \mathrm{A}^t_{i,k})\mathrm{X}^{\mathrm{t}}_{\mathrm{i}}\right)$$

Se supone que el costo de informar que ya no se posee un contenido es igual al costo de informar que se aloja uno nuevo, es decir una publicación.

Cuando un solicitante busca un contenido, lo hace en el servidor central, el cual conoce completamente los archivos compartidos reduciendo la búsqueda a una búsqueda local eficiente:

$$B^t_{i,0,k} = S^t_{i,k}$$

$$P^{t+1}_{0,i,j,k} = B^t_{i,0,k} I^t_{0,j,k}$$

Finalmente cuando un nodo se desconecta de la red, este le avisa al servidor central que ya no se encuentra más en la red. El contenido *k=0*, se considera un contenido especial, y es utilizado para anunciar por ejemplo el cambio de estado en un nodo.

$$P^{t}_{i,0,i,0} = (X^{t-1}_{i} - X^{t}_{i})$$
$$E^{t}_{0,i,k} = \neg(X^{t-1}_{i} - X^{t}_{i})$$

3.2.2 Gnutella (Modelo Puro) [52]

En esta sección se hace referencia al protocolo de la red Gnutella en su versión original (protocolo v0.4), actualmente la red incorpora mejoras como el concepto de nodos superpeers (conocidos en Gnutella como ultrapeers).
En esta situación $K = V$, todos los nodos son terminales, con la capacidad de solicitar, enrutar y ofrecer contenido.

En una red basada en ***broadcasts*** ningún contenido es publicado de forma proactiva por parte de las fuentes, sino que los nodos solicitantes realizan búsquedas específicas de contenido a todos sus pares. En este modelo los pares de un nodo, también llamados vecinos, son aquellos nodos de los cuales se conoce el contenido *k=0*:

$$N^{t}(i) = \left\{ j / I^{t}_{i,j,0} = 1 \right\}$$

Los pares que reciben las consultas en realidad actúan como nodos enrutadores volviendo a propagar la consulta y como nodos fuentes contestando al solicitante que les hizo la consulta si poseen el contenido buscado.
Para evitar bucles y esperas muy largas, las búsquedas se propagan con un mecanismo de tiempo de vida limitado, *TTL* (generalmente *TTL=7*).

Para entender la notación considérese que *TTL=2*, entonces para todo camino en la red de largo 2: $\forall \langle j, j_2 \rangle / \; j \in N^{t}(\mathrm{i}), j_2 \in N^{t+1}(\mathrm{j})$
Si el nodo *i* realiza una solicitud en el tiempo *t* del contenido *k*, entonces inmediatamente buscaría en sus vecinos por el contenido $B^{t}_{i,j,k} = S^{t}_{i,k}$.
Estos vecinos directos responderían si tienen el contenido en el tiempo siguiente $P^{t+1}_{j,i,j,k} = B^{t}_{i,j,k} I^{t}_{j,j,k}$ o reenviarían la búsqueda al resto de los vecinos $B^{t+1}_{j,j_2,k} = B^{t}_{i,j,k}$, los cuales podrían responder por el camino en que se realizó la búsqueda $P^{t+2}_{j_2,j,j_2,k} = B^{t+1}_{j,j_2,k} I^{t+1}_{j_2,j_2,k}$ y $P^{t+3}_{j,i,j_2,k} = P^{t+2}_{j_2,j,j_2,k}$.

En el caso general el mecanismo es el mismo, para todo camino de largo menor o igual a *TTL*
$\forall \langle j, j_2, ..., j_{TTL-1}, j_{TTL} \rangle / \; j \in N^{t}(\mathrm{i}), j_2 \in N^{t+1}(\mathrm{j}), ..., j_{TTL-1} \in N^{t+TTL-2}(\mathrm{j_{TTL\text{-}2}}), j_{TTL} \in N^{t+TTL-1}$
En este caso la comunicación se forma por los siguientes paquetes

Largo 1: $\forall j / \; j \in N^{t}(\mathrm{i})$

$$B^{t}_{i,j,k} = S^{t}_{i,k}$$
$$P^{t+1}_{j,i,j,k} = B^{t}_{i,j,k} I^{t}_{j,j,k}$$

Largo 2: $\forall \langle j, j_2 \rangle / \; j \in N^{t}(\mathrm{i}), j_2 \in N^{t+1}(\mathrm{j})$

$$B^{t+1}_{j,j_2,k} = S^{t}_{i,k}$$
$$P^{t+2}_{j_2,j,j_2,k} = B^{t+1}_{j,j_2,k} I^{t+1}_{j_2,j_2,k}$$
$$P^{t+3}_{j,i,j_2,k} = P^{t+2}_{j_2,j,j_2,k}$$

.....

Largo *TTL*:

$\forall\langle j, j_2,..., j_{TTL-1}, j_{TTL}\rangle / j \in N^{t}(\mathrm{i}), j_2 \in N^{t+1}(\mathrm{j}),..., j_{TTL-1} \in N^{t+TTL-2}(\mathrm{j_{TTL\text{-}2}}), j_{TTL} \in N^{t+TTL-1}$

$B^{t+TTL-1}_{j_{TTL-1}, j_{TTL}, k} = S^{t}_{i,k}$

$P^{t+TTL}_{j_{TTL}, j_{TTL-1}, j_{TTL}, k} = B^{t+TTL-1}_{j_{TTL-1}, j_{TTL}, k} I^{t+TTL-1}_{j_{TTL}, j_{TTL}, k}$

$P^{t+TTL+1}_{j_{TTL-1}, j_{TTL-2}, j_{TTL}, k} = P^{t+TTL}_{j_{TTL}, j_{TTL-1}, j_{TTL}, k}$

....

$P^{t+2TTL-2}_{j_2, j, j_{TTL}, k} = P^{t+2TTL-3}_{j_3, j_2, j_{TTL}, k}$

$P^{t+2TTL-1}_{j, i, j_{TTL}, k} = P^{t+2TTL-2}_{j_2, j, j_{TTL}, k}$

Como se ha dicho anteriormente Gnutella escala muy poco, para aumentar el grado de los nodos participantes y de esta forma permitir a la red que utilice un *TTL* menor se utiliza una mensajería de Ping y Pong, el Ping es una inundación de consultas de estados de conexión, mientras que el Pong es su respuesta. La lógica de comunicación es igual a la de las consultas (con *TTL* y con la respuesta recorriendo el camino inverso), solo que en este caso el contenido por el cual se pregunta es el *k=0*.

$\forall\langle j, j_2,..., j_{TTL-1}, j_{TTL}\rangle / j \in N^{t}(\mathrm{i}), j_2 \in N^{t+1}(\mathrm{j}),..., j_{TTL-1} \in N^{t+TTL-2}(\mathrm{j_{TTL\text{-}2}}), j_{TTL} \in N^{t+TTL-1}$

En este caso la comunicación se forma por los siguientes paquetes:

Largo 1: $\forall j / j \in N^{t}(\mathrm{i})$

$B^{t}_{i,j,0} = S^{t}_{i,0}$

$P^{t+1}_{j,i,j,0} = B^{t}_{i,j,0} X^{t}_{j}$

Largo 2: $\forall\langle j, j_2\rangle / j \in \mathrm{N}^{t}(\mathrm{i}), j_2 \in N^{t+1}(\mathrm{j})$

$B^{t+1}_{j,j_2,0} = S^{t}_{i,0}$

$P^{t+2}_{j_2,j,j_2,0} = B^{t+1}_{j,j_2,0} X^{t+1}_{j_2}$

$P^{t+3}_{j,i,j_2,0} = P^{t+2}_{j_2,j,j_2,0}$

.....

Largo *TTL*:

$\forall\langle j, j_2,..., j_{TTL-1}, j_{TTL}\rangle / j \in N^{t}(\mathrm{i}), j_2 \in N^{t+1}(\mathrm{j}),..., j_{TTL-1} \in N^{t+TTL-2}(\mathrm{j_{TTL\text{-}2}}), j_{TTL} \in N^{t+TTL-1}$

$B^{t+TTL-1}_{j_{TTL-1}, j_{TTL}, 0} = S^{t}_{i,0}$

$P^{t+TTL}_{j_{TTL}, j_{TTL-1}, j_{TTL}, 0} = B^{t+TTL-1}_{j_{TTL-1}, j_{TTL}, 0} X^{t+TTL-1}_{j_{TTL}}$

$P^{t+TTL+1}_{j_{TTL-1}, j_{TTL-2}, j_{TTL}, 0} = P^{t+TTL}_{j_{TTL}, j_{TTL-1}, j_{TTL}, 0}$

....

$P^{t+2TTL-2}_{j_2, j, j_{TTL}, 0} = P^{t+2TTL-3}_{j_3, j_2, j_{TTL}, 0}$

$P^{t+2TTL-1}_{j, i, j_{TTL}, 0} = P^{t+2TTL-2}_{j_2, j, j_{TTL}, 0}$

De esta forma todos los nodos que contestan (envían un PONG, que es lo mismo que publicar el contenido *k=0*) son automáticamente agregados a la tabla de vecinos. Cuando no se responde es necesario tener un criterio para eliminarlos luego de un tiempo, es decir un criterio de expiración de conexión de nodo. El caso más simple es considerar que si en las últimas ***e*** rondas no se recibió un PONG entonces el nodo ya no es vecino, esto es:

$$E^t_{i,j,0} = \sum_l \left(P^{t-e}_{l,i,j,0} + ... + P^t_{l,i,j,0} \right)$$

3.2.3 DNS (Modelo Jerarquizado) [157][158]

En la sección 2.3 se hace una introducción al funcionamiento del sistema de nombres de dominios, en esta sección se modela la red en el contexto del *CDP*.

Recordando, existen 3 tipos de ***nodos***: los resolvers (solicitantes), los servidores recursivos (enrutadores) y los servidores autoritativos (fuentes y enrutadores).
Lógicamente es posible combinar estas funcionalidades en un único host, por ejemplo es común tener servidores recursivos con delegación.

$$V = K \cup R \cup D$$

Donde K son los resolvers, R son los servidores recursivos y D son los servidores autoritativos.

$$X^t_i = \begin{cases} \text{conexión corta} & i \in K \\ \text{conexión larga} & i \in R \cup D \end{cases}$$

$$S^t_i = \begin{cases} \text{comportamiento cliente} & i \in K \\ 0 & i \in R \cup D \end{cases}$$

La tabla de índices de cada nodo representa el conocimiento de la estructura arborescente, es decir, el conocimiento de que nodo autoritativo tiene que dominio.
Los nodos resolvers y recursivos no tienen delegados dominios y solo poseen índices de contenido a nodos autoritativos, estos índices representan el *cache* de los nodos.
Los nodos resolvers tienen preconfigurado dos o tres nodos recursivos, para que resuelvan todas sus consultas, esto se modela como un índice al contenido *0*.

$$I^t_{i,j,k} = \begin{cases} 0 & j = i, i \in K \cup R \\ \text{cache} & j \neq i, i \in K \cup R, j \in D \\ j \text{ resolver de } i & k = 0, j \neq i, i \in K, j \in R \end{cases}$$

Para los nodos autoritativos la situación es muy distinta, son autoritativos de dominios y por tanto poseen índices a si mismos, además tienen índices a otros nodos autoritativos representando la delegación de subdominios a otros nodos autoritativos, es decir la estructura arborescente de delegación.

$$I^t_{i,j,k} = \begin{cases} \text{dominio que aloja} & j = i, i \in D \\ \text{delegacion de subdominio} & j \neq i, i \in D, j \in D \end{cases}$$

De lo anterior se desprende que:

$$A^t_{i,k} = 1 \Leftrightarrow i \in D, i \text{ autoritativo de } k \text{ en el tiempo } t$$

Los clientes del sistema DNS (resolvers) no informan de su conexión o desconexión a la red, solo se comunican con los servidores recursivos cuando necesitan resolver un dominio (realizan una solicitud de contenido).
Los recursivos *j*, configurados en un cliente *i*, son sus vecinos: $N^t(i) = \left\{ \mathrm{j} / \mathrm{I}^{\mathrm{t}}_{\mathrm{i,j,0}} = 1 \right\}$.

Si el nodo *i* realiza una solicitud en el tiempo *t* del dominio *k*, entonces en primer lugar busca la información en su propio *cache*, si no se tiene información valida para dicho dominio busca de forma iterativa en sus vecinos hasta lograr una respuesta satisfactoria por el contenido. Esto se modela:

$$N^t(i) = \{r_1, r_2\}$$

$$B^t_{i,r_1,k} = \left(\neg \left(\sum_{\forall j} I^t_{i,j,k} \right) \right) S^t_{i,k}$$

$$B^{t+T+1}_{i,r_2,k} = \left(\neg \left(\sum_{\substack{x \in [1..T], \\ \forall j}} P^{t+x}_{r_1,i,j,k} \right) \right) B^t_{i,r_1,k}$$

Donde *T* es un tiempo de espera máximo por la respuesta.
Ciertas simplificaciones se hicieron en este punto, en el sistema de nombres existe el concepto de *cache* negativo, que representa el conocimiento de la no existencia de un dominio. Los servidores recursivos manejan una tabla de dominios que no existen, cuando un resolver pregunta por uno de esos dominios el servidor responde con una respuesta negativa sin hacer ninguna consulta extra. Lógicamente los elementos de la tabla de inexistencia de dominios también tienen un tiempo de expiración. En este modelo no se incluye dicho conocimiento por tanto el servidor recursivo consultaría cada vez por aquellos dominios inexistentes.

La respuesta de un contenido, modelada como una publicación de dominio, siempre incluye el tiempo de validez de dicha publicación (denotado por las siglas *TTL*), por tanto se encuentra completamente especificada la política de expiración de un contenido, tanto para los nodos resolvers como para los nodos recursivos:

$$E^t_{i,j,k} = \begin{cases} 1 & i \in D \\ \text{segun } TTL \text{ de } k & j \neq i, i \in K \cup R, j \in D \end{cases}$$

Los servidores recursivos reciben las consultas de los clientes y las resuelven preguntándole a los servidores autoritativos si es que no la tiene en el *cache*.
Es común que existan varios nodos autoritativos para un dominio: el primario y sus secundarios. Se pueden tener hasta 13 servidores autoritativos para un mismo dominio. Un ejemplo de esto son los servidores raíz, donde existen 13 servidores distribuidos principalmente dentro de EEUU, y conocidos por todos los servidores recursivos del mundo.
Los servidores recursivos eligen a quien preguntarle de la lista de autoritativos según un ranking propio conocido como Round Trip Time (RTT).
Para simplificar el modelo se define una función RTT que dado un conjunto de nodos autoritativos devuelve el nodo con mejor ranking a ser preguntado.
Se comienza cuando el cliente *i* inicia una búsqueda al servidor recursivo *r* del dominio *k*, si el servidor recursivo *r* tiene la respuesta en *cache* entonces el proceso termina publicando al cliente la respuesta:

$$P^{t+1}_{r,i,d,k} = B^t_{i,r,k} I^t_{r,d,k}$$

Si el servidor recursivo no tiene la respuesta en cache, comienza una búsqueda en los servidores autoritativos comenzando por la raíz.

$$B^{t+1}_{r,RTT(0),k} = B^{t}_{i,r,k}\left(\neg\left(\sum_{\forall d} I^{t}_{r,d,k}\right)\right)$$

$$P^{t+2}_{RTT(0),r,d_1,k} = B^{t+1}_{r,RTT(0),k} I^{t+1}_{RTT(0),d_1,k}$$

$$P^{t+3}_{r,i,RTT(0),k} = P^{t+2}_{RTT(0),r,RTT(0),k} I^{t+1}_{RTT(0),RTT(0),k}$$

$$D_1 = \left\{d \,/\, P^{t+2}_{RTT(0),r,d_1,k} = 1\right\}$$

El servidor raíz *RTT(0)* puede ser autoritativo para el dominio *k*, en ese caso se devuelve la respuesta al cliente *i*, en otro caso se consulta a uno de los autoritativos de conjunto D_1 devuelto por *RTT(0)*:

$$B^{t+3}_{r,RTT(D_1),k} = P^{t+2}_{RTT(0),r,RTT(D_1),k}\left(\neg I^{t+1}_{RTT(0),RTT(0),k}\right)$$

$$P^{t+4}_{RTT(D_1),r,d_2,k} = B^{t+3}_{r,RTT(D_1),k} I^{t+3}_{RTT(D_1),d_2,k}$$

$$P^{t+5}_{r,i,RTT(D_1),k} = P^{t+4}_{RTT(D_1),r,RTT(D_1),k} I^{t+1}_{RTT(D_1),RTT(D_1),k}$$

$$D_2 = \left\{d \,/\, P^{t+4}_{RTT(D_1),r,d_2,k} = 1\right\}$$

Este proceso se repite hasta el servidor autoritativo consultado responda que contiene el contenido *k* (es decir que el mismo es autoritativo). Supongamos que esto sucede en una profundidad *L* del árbol de delegación de DNS:

$$B^{t+2L+1}_{r,RTT(D_L),k} = P^{t+2L}_{RTT(D_{L-1}),r,RTT(D_L),k}\left(\neg I^{t+2L-1}_{RTT(D_{L-1}),RTT(D_{L-1}),k}\right)$$

$$P^{t+2L+2}_{RTT(D_L),r,RTT(D_L),k} = B^{t+2L+1}_{r,RTT(D_L),k} I^{t+2L+1}_{RTT(D_L),RTT(D_L),k}$$

$$P^{t+2L+3}_{r,i,RTT(D_L),k} = P^{t+2L+2}_{RTT(D_L),r,RTT(D_L),k} I^{t+2L+1}_{RTT(D_L),RTT(D_L),k}$$

Se realiza una simplificación al protocolo, considerando que todo autoritativo se encuentra activo, en general ese no es el caso y el servidor recursivo debe consultar a más de un servidor autoritativo de un dominio para obtener la respuesta.

La comunicación entre servidores autoritativos se da cuando los secundarios de un dominio le solicitan al primario algún dominio que haya expirado. Existen varias formas de comunicación: notify, dynamic update, IXFR, TSIG, etc. con distintas característica cada una. La forma básica de comunicación implica que el servidor secundario *s* le solicite al primario *p* un dominio luego de que el mismo haya expirado:

$$B^{t}_{s,p,k} = \left(E^{t-1}_{s,s,k} - E^{t}_{s,s,k}\right) I^{t}_{s,s,k}$$

$$P^{t+1}_{p,s,p,k} = B^{t}_{s,p,k} I^{t}_{p,p,k}$$

Otros detalles del protocolo podrían ser incluidos en el modelo para darle más expresividad y precisión. Debe entenderse que el objetivo de esta sección es simplemente mostrar a través de ejemplos la expresividad del modelo, en la siguiente sección se detallan algunas desventajas que presenta el enfoque del *CDP*.

3.3 Discusión y Conclusiones

En este capítulo se presenta un modelo muy detallado sobre el comportamiento dinámico de las redes de contenido. Para mostrar su generalidad se observa el instanciamiento del modelo para las particularidad de tres redes de contenido muy distintas: Napster[166][167], Gnutella[52] y DNS[157][158].

Pero la generalidad tiene un costo en cantidad de cómputo necesario para su utilización. A pesar de que no se implementa el modelo en lenguaje de programación, se supone que presenta demasiada cantidad de estados y variables del sistema y por tanto muy difícilmente puedan desarrollarse ejecuciones para la cantidad de nodos deseados (las actuales redes de pares para compartir archivos presentan millones de usuarios simultáneos). Esto también lo evidencia los simuladores de redes P2P hasta el momento implementados, que al momento de escrito este documento escalan solo a miles de nodos. Ejemplos son: Peersim[194], P2psim[185], Simpastry[190], Anthill[12], etc.

Por tanto el modelo es útil para la simulación o evaluación de una red de contenido genérica, sin embargo al requerir tanto costo computacional lo descarta para la utilización de algún método de optimización (heurístico o no) sobre el diseño de una red así modelada.
Uno de los objetivos de este trabajo es lograr un modelo capaz de asistir en el diseño de la red mediante algunas técnicas de optimización conocida, por tanto en el siguiente capítulo se realiza una simplificación del modelo observando exclusivamente el comportamiento en un tipo de nodos de la red.

Modelo Particular: Problema del Descubrimiento de Contenido en los Agregadores

4

En una red de contenido es muy costoso mantener la información global del sistema, es decir, se requiere demasiado consumo del recurso comunicación (ancho de banda) para que todos los nodos conozcan que nodos alojan que contenido en cada instante. Por tanto en todas las redes de contenido no existe información global del contenido y en pro de que la red pueda escalar y potenciarse del efecto de red no es posible que todos los nodos conozcan exactamente en todo instante que contenido aloja cada fuente. Existen entonces principalmente dos estrategias para utilizar más eficientemente el ancho de banda de los nodos de la red:

- una jerarquización en la topología donde los nodos con más recursos puedan ser mejor aprovechados,
- y un equilibrio en el uso de la comunicación entre publicación de cambios en el alojamiento de las fuentes y una búsqueda a demanda de los contenidos solicitados.

El equilibrio entre publicación y búsqueda se encuentra dado en una red de contenido por la cantidad de cambios que presentan las fuentes respecto a las solicitudes. Se supone que existe autonomía de los nodos, donde cada nodo se conecta y desconecta de la red según su necesidad, y aloja y desaloja contenido arbitrariamente. Por tanto en una red muy dinámica, donde los nodos alojan y desalojan el contenido de forma muy acelerada parece tener menos sentido basar la estrategia de comunicación en la publicación del alojamiento (puesto que este dato sería inválido muy rápidamente) y claramente conviene potenciar la búsqueda a demanda del contenido. Por otro lado si el contenido es alojado de forma estática por las fuentes debe ser muy valioso publicar este dato a la mayor cantidad de nodos posibles y de esta forma disminuir la comunicación entre nodos al momento de una búsqueda.
Las redes de contenido actualmente no se diseñan considerando el dinamismo del contenido que piensan alojar, la hipótesis de este trabajo es que si se incluye el dinamismo del contenido para determinar el equilibrio necesario entre publicación y búsqueda del contenido se puede alcanzar una red con mayor información global y de mayor escalamiento.

Bajo esta hipótesis en el capítulo 3 se presenta un modelo muy genérico (y de difícil cuantificación), donde el equilibrio entre publicación y búsqueda es la variable de decisión. Para este modelo se muestra su aplicación a las redes de contenido más variadas, pero no se incluye una cuantificación puesto que una implementación, inclusive en base a simulación, sería muy costosa en tiempo computacional.

Para simplificar el modelo de forma que sea tratable se realizan suposiciones adicionales y se ataca el problema en solo un tipo de nodo enrutador presente en la amplia mayoría de las redes de contenido: los nodos de agregación o de cache de control.
Estos nodos son encargados de responder las consultas a los solicitantes, de forma de disminuir el costo de comunicación a los solicitantes y al backbone de contenido.
El mecanismo principal que tienen estos nodos para disminuir el costo en comunicación es el de reutilizar consultas previamente realizadas, es decir mantener un historial de las ultimas consultas y usar sus resultados frente a nuevas solicitudes. Este historial generalmente se denomina *cache*.

Considerando que la red presenta contenido dinámico no es posible conservar las consultas en el *cache* por tiempo indefinido (la información de alojamiento de la respuesta no es válida para siempre). Por tanto, el tiempo de expiración de las consultas previamente realizadas es expresión del compromiso entre publicación y búsqueda en los nodos de agregación.

En el sistema DNS los nodos agregadores son los servidores recursivos, mientras que en KaZaA son los nodos superpeers. La Ilustración 11 muestra el lugar de los nodos agregadores en una topología genérica de redes de contenido.

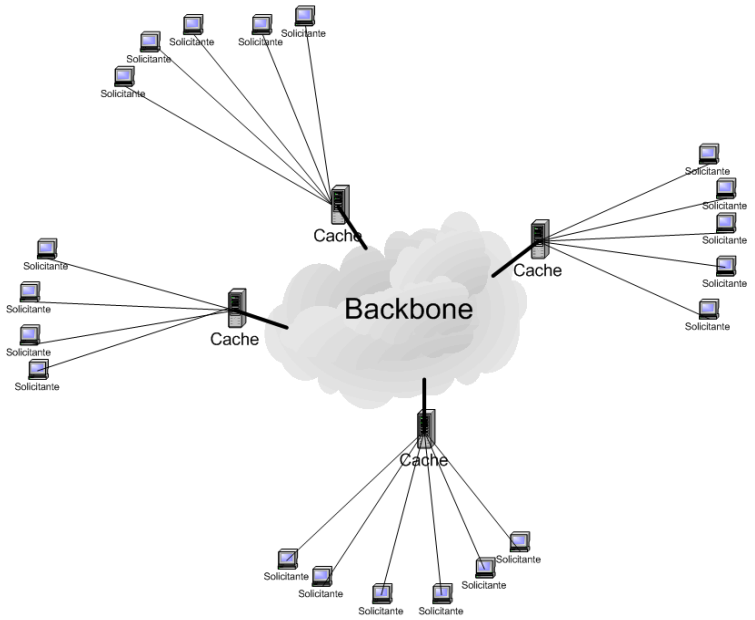


Ilustración 11. Topología de red de contenido simplificada.

4.1 CCP - Content Caching Problem

En esta sección se formaliza el problema de en un nodo agregador almacenar las solicitudes previas de contenido en redes distribuidas con el fin de disminuir el consumo de ancho de banda. De ahora en más denominado por las siglas ***CCP*** **(Content Caching Problem)**.

4.1.1 El Tiempo

Se considera el sistema en estado estacionario, por tanto no se modela el transcurso del tiempo solo se observa el comportamiento promedio de la red.

4.1.2 El Contenido y los Nodos

El **contenido** de la red se representa con el conjunto C.

Los **nodos** de la red pertenecen al conjunto V.
Para el modelo es útil detallar tres tipos de nodos: **solicitantes, fuentes** y **agregadores**.

Solicitantes

Un subconjunto de nodos $S \subseteq V$, llamados terminales o **solicitantes,** tiene la posibilidad de solicitar contenidos a la red.
Se considera que las consultas realizadas por los solicitantes llegan de acuerdo a un proceso de Poisson de tasa f_k. Esto significa que si $S_k(T)$ representan la cantidad de consultas realizadas por los solicitantes del contenido *k* en un período *T*, la probabilidad de realizar *n* consultas en ese período es:

$$p(S_k(T) = n) = \frac{(f_k T)^n e^{-f_k T}}{n!} \quad \forall k \in C, \forall n \in \aleph, \forall T \in \Re^+$$

donde f_k es la tasa o frecuencia de solicitudes del contenido *k*.

A su vez el tiempo entre dos solicitudes del contenido k: T_{S_k}, es una variable aleatoria exponencial de parámetro f_k:

$$p(T_{S_k} \le t) = \begin{cases} 1 - e^{-f_k t}, & t \ge 0 \\ 0, & t < 0 \end{cases}, \quad \overline{T_{S_k}} = \mathrm{E}\{T_{S_k}\} = 1/f_k$$

Implícitamente esta hipótesis de realización de solicitudes supone un modelo con una cantidad infinita de solicitantes potenciales y un estado estacionario del sistema.

Fuentes

Los contenidos son alojados físicamente en los nodos fuentes. Un subconjunto de nodos $F \subseteq V$, llamados **fuentes,** tiene la posibilidad de alojar contenidos a la red.
Uno de los puntos más importantes para preservar la autonomía es que los nodos puedan elegir cuando comenzar a alojar y terminar de alojar un contenido dado.
Se supone una cantidad infinita de nodos fuentes, donde para cada contenido un proceso de nacimiento y muerte de la forma $\mathrm{M/M/\infty}$ determina la cantidad de fuentes que alojan dicho contenido en un momento dado.

Es decir, cada nodo fuente comienza a alojar un contenido *k* con llegadas de Poisson de tasa λ_k y aloja dicho contenido *k* durante un período exponencial de tasa μ_k.
Esto significa que si se supone la red en estado estacionario en el instante t_0 y $A_k(t_0)$ representan la cantidad de nodos fuentes que alojan el contenido *k* en dicho

instante t_0, la probabilidad de que en dicho instante sean *n* la cantidad de nodos fuentes que alojan el contenido *k* es:

$$p\left(A_k\left(t_0\right)=n\right)=\frac{\left(\lambda_k/\mu_k\right)^n e^{-\lambda_k/\mu_k}}{n!}\quad \forall k\in C, \forall n\in \aleph, \forall t_0\in \Re^+$$

donde μ_k es la tasa de desalojamiento o validez del contenido *k* y λ_k la tasa de comienzo de alojamiento del contenido *k*.
Implícitamente esta hipótesis supone un estado estacionario del sistema.

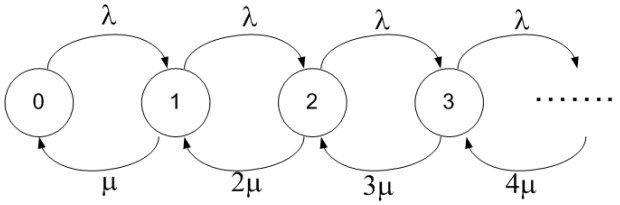


Ilustración 12. Proceso estocástico de Alojamiento de un contenido.

Un dato interesante que se desprende del modelo es el valor esperado de fuentes que alojan el contenido *k*:

$$\overline{A_k}=\mathrm{E}\left\{A_k\left(t_0\right)\right\}=\sum_{n\geq 0} n.p\left(A_k\left(t_0\right)=n\right)=\sum_{n\geq 1}\frac{\left(\lambda_k/\mu_k\right)^n e^{-\lambda_k/\mu_k}}{(n-1)!}=$$

$$=e^{-\lambda_k/\mu_k}\left(\lambda_k/\mu_k\right)\sum_{n\geq 1}\frac{\left(\lambda_k/\mu_k\right)^{n-1}}{(n-1)!}=e^{-\lambda_k/\mu_k}\left(\lambda_k/\mu_k\right)e^{\lambda_k/\mu_k}=\lambda_k/\mu_k$$

Debido a la autonomía de los nodos, en general estos tienen un comportamiento de conexión y desconexión. Por simplicidad se opta por no modelar la conexión y desconexión de los nodos de la red, y exclusivamente en los nodos fuentes la conexión y desconexión se encuentre implícitamente modelada en las tasas de alojamiento y desalojamiento de contenido.

Agregadores

Los únicos nodos enrutadores que se consideran en el modelo son los nodos de **agregación**. Todo nodo solicitante se conecta a por lo menos un nodo agregador para realizar sus búsquedas de contenido. Para la mayoría de los solicitantes realizar la búsqueda directamente en el backbone sería un proceso demasiado costoso, por tanto un nodo agregador concentra las consultas de todos los solicitantes que tiene conectados y realiza la consulta en el backbone.
Se considera que los nodos de agregación tienen una cantidad suficientemente grande de nodos solicitantes conectados de forma de poder considerarlos infinitos y mantener la formula del modelo (la probabilidad de que un nodo de agregación reciba *n* consultas por el contenido *k* en el período *T* es $p\left(S_k\left(T\right)=n\right)$).

La idea en los nodos de agregación es aprovechar las consultas previas de un contenido de forma de minimizar la cantidad de búsquedas en el backbone; es decir, la principal función de un nodo de agregación es mantener un cache de búsquedas recientes, y de allí que recibe también el nombre de nodo **cache**.
Una consulta previa de un contenido es válida solo por un período de tiempo, debido a que los nodos que alojaban el contenido al momento de la consulta previa van a ir

paulatinamente desalojando el contenido hasta que posiblemente ninguno sea válido. Supóngase que en el tiempo t_0 se consultó al backbone por el contenido *k* retornando $A_k(t_0)$ nodos fuentes que alojan dicho contenido. La información conocida por el nodo cache a partir de ese momento es que mediante un proceso de muerte pura estos nodos van a ir desalojando el contenido.

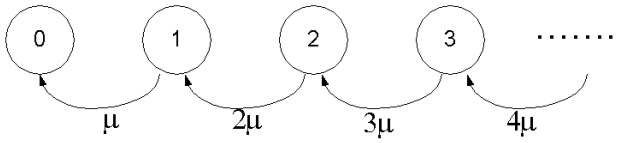


Ilustración 13. Proceso estocástico de Desalojamiento de un contenido (conocimiento del agregador).

Un dato interesante que se desprende del modelo es la cantidad promedio de alojamientos válidos que conoce un agregador luego de un tiempo *t* de haber realizado una consulta al backbone:

$$\begin{pmatrix}\text{cantidad promedio de alojamientos}\\ \text{validos luego de t tiempos de realizar}\\ \text{la consulta al backbone}\end{pmatrix} = \sum_{n\geq 0} n.p\left(A_k(t_0)=n\right)p\left(T_{V_k} > t_0 + t \middle| T_{V_k} > t_0\right) =$$

$$= \sum_{n\geq 0} n.p\left(A_k(t_0)=n\right)p\left(T_{V_k} > t\right) = \sum_{n\geq 1} \frac{\left(\lambda_k / \mu_k\right)^n e^{-\lambda_k/\mu_k}}{(n-1)!} e^{-\mu_k t} =$$

$$= e^{-\lambda_k/\mu_k}\left(\lambda_k/\mu_k\right)e^{-\mu_k t}\sum_{n\geq 1}\frac{\left(\lambda_k/\mu_k\right)^{n-1}}{(n-1)!} = e^{-\lambda_k/\mu_k}\left(\lambda_k/\mu_k\right)e^{-\mu_k t}e^{\lambda_k/\mu_k} = \lambda_k/\mu_k\, e^{-\mu_k t}$$

donde el tiempo T_{V_k} de validez del contenido *k* en un nodo es una variable aleatoria exponencial de parámetro μ_k.

Por tanto según el dinamismo que presente el alojamiento de contenido por parte de las fuentes y la cantidad de repeticiones en consultas por parte de los solicitantes es el tiempo óptimo para los nodos de agregación en guardar las consultas previas de un contenido.
Puede observarse entonces que en este modelo el equilibrio entre publicación y búsqueda para los nodos cache se encuentra en el tiempo que se decide tener guardadas las consultas previas; a este tiempo se le llamará "tiempo de cache" y se le simbolizara con d_k para el contenido *k*.

4.1.3 Estados y Transición de la Red

El funcionamiento de un nodo cache es muy simple: recibe una consulta de un solicitante por el contenido *k*, si no tiene una respuesta guardada entonces realiza una búsqueda en el backbone, retorna la respuesta y la guarda durante un período d_k. Si la respuesta se encontraba en cache simplemente retorna la respuesta sin realizar búsqueda en el backbone.
El siguiente diagrama detalla la dinámica del cache frente a varias solicitudes por un contenido *k* dado.

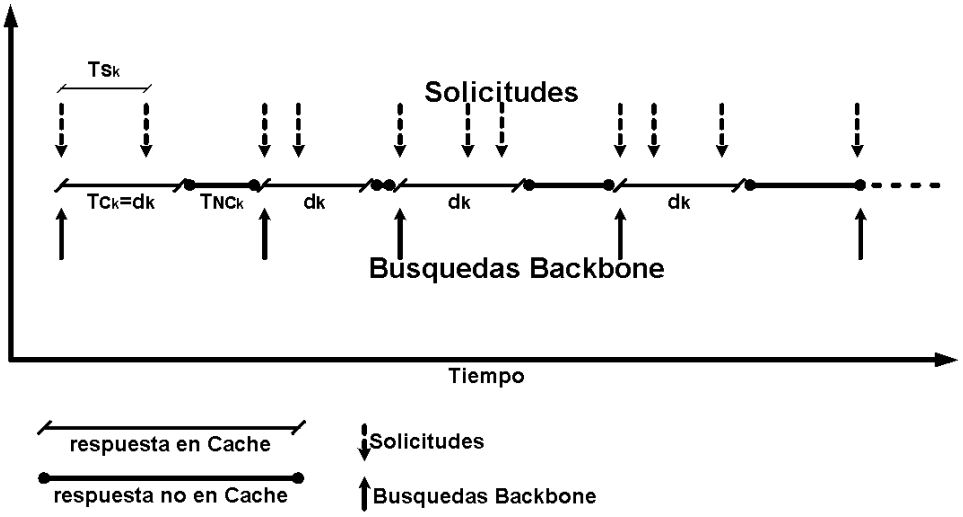


Ilustración 14. Solicitudes de un contenido a lo largo del tiempo.

Se puede observar que el tiempo en que la respuesta se guarda en cache es constante ($T_{C_k} = d_k$) y el tiempo en que el contenido no se encuentra cacheado (T_{NC_k}) responde a una variable aleatoria exponencial de parámetro f_k (f_k : frecuencia de solicitudes del contenido *k*, usando la propiedad $p\left(T_{S_k} > s + t \middle| T_{S_k} > t\right) = p\left(T_{S_k} > s\right)$).
Además, de todas las solicitudes recibidas por el agregador, solo las que recibe en estado no cacheado generan búsquedas en el backbone.

La respuesta por un contenido dado puede encontrarse en dos estados: en cache o no. Puede observarse que el sistema se comporta como un proceso estocástico regenerativo con un único ciclo (cache $\Leftrightarrow$ no cache), entonces es posible estudiar el comportamiento asintótico del sistema observando exclusivamente el comportamiento de un ciclo.
Si suponemos estados estacionarios del sistema se tienen las siguientes probabilidades para cada estado:

$$p(\text{k cacheado}) = \frac{\text{tiempo cacheado}}{\text{tiempo total}} =$$

$$= \frac{\mathrm{E}\{T_{C_k}\}}{\mathrm{E}\{T_{C_k}\} + \mathrm{E}\{T_{NC_k}\}} = \frac{d_k}{d_k + 1/f_k} = \frac{f_k d_k}{1 + f_k d_k}$$

$$p(\text{k no cacheado}) = \frac{\text{tiempo no cacheado}}{\text{tiempo total}} =$$

$$= \frac{\mathrm{E}\{T_{NC_k}\}}{\mathrm{E}\{T_{C_k}\} + \mathrm{E}\{T_{NC_k}\}} = \frac{1/f_k}{d_k + 1/f_k} = \frac{1}{1 + f_k d_k}$$

4.1.4 Descubrimiento del Contenido: Mensajes y Restricción de Ancho de Banda

La cantidad de búsquedas del contenido *k* en el backbone por unidad de tiempo en estado estacionario son:

$$\begin{pmatrix}\text{busquedas en el backbone}\\ \text{por unidad de tiempo}\end{pmatrix} = \frac{\#\,\text{busquedas}}{\text{tiempo total}} =$$

$$= 1\,\text{busqueda cada}\left(d_k + {}^{1}\!/_{f_k}\right) =$$

$$= \frac{1}{d_k + {}^{1}\!/_{f_k}} = \frac{f_k}{1 + d_k f_k}$$

La cantidad de solicitudes por unidad de tiempo del contenido *k* recibidas en estado estacionario son f_k.

Como se especificó anteriormente el valor esperado de fuentes que alojan el contenido k es $\overline{A_k} = \frac{\lambda_k}{\mu_k}$, suponiendo que el backbone responde todas las fuentes existentes, entonces la respuesta desde el backbone cuenta con $\overline{A_k}$ alojamientos.
De la misma forma la respuesta a los solicitantes cuenta con $\overline{A_k}$ alojamientos en promedio, no importando si dicha respuesta surge del cache[12] o de una búsqueda en el backbone.

Desde el punto de vista de los solicitantes, realizar una solicitud requiere β_S bytes y su respuesta requiere α_S bytes por alojamiento respondido (es decir que la respuesta varia su tamaño de acuerdo a la cantidad de nodos que se conoce que alojan el contenido).
La situación se repite en las búsquedas en el backbone, una búsqueda en el backbone requiere β_B bytes, y su respuesta α_B bytes/(alojamiento respondido).

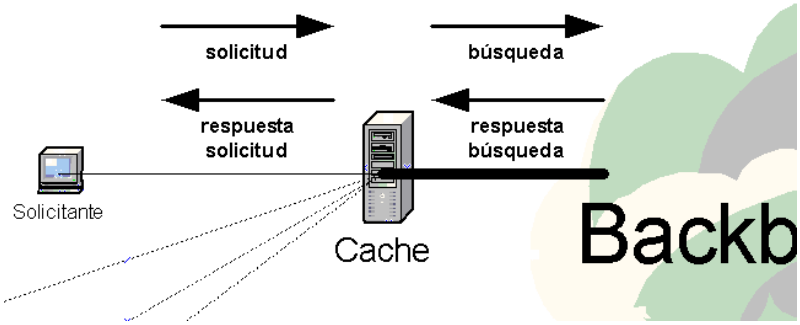


Ilustración 15. Comunicación en el agregador.

Se puede observar que la solicitud y la respuesta del backbone es información entrante al nodo cache mientras que la búsqueda y la respuesta a la solicitud es información saliente, por tanto el consumo de ancho de banda realizado en el nodo de agregación en estado estacionario es:

[12] Se verá más adelante que en este caso no todos los alojamientos van a ser válidos, pero si son todos enviados en la respuesta al solicitante.

$$\begin{pmatrix}\text{ancho de banda entrante}\\ \text{en agregador}\end{pmatrix} = \beta_S \sum_{k \in C} f_k + \alpha_B \sum_{k \in C} \frac{f_k}{1 + d_k f_k} \overline{A_k}$$

$$\begin{pmatrix}\text{ancho de banda saliente}\\ \text{en agregador}\end{pmatrix} = \alpha_S \sum_{k \in C} f_k \overline{A_k} + \beta_B \sum_{k \in C} \frac{f_k}{1 + d_k f_k}$$

Los nodos cache cuentan con un ancho de banda limitado simbolizado con BW_{IN} y BW_{OUT} (limitación en el ancho de banda de entrada y salida del nodo), por tanto el sistema presenta la siguiente restricción:

$$\beta_S \sum_{k \in C} f_k + \alpha_B \sum_{k \in C} \frac{f_k}{1 + d_k f_k} \overline{A_k} \leq BW_{IN}$$

$$\alpha_S \sum_{k \in C} f_k \overline{A_k} + \beta_B \sum_{k \in C} \frac{f_k}{1 + d_k f_k} \leq BW_{OUT}$$

4.1.5 Backbone

La red se modela como un backbone, nodos de agregación y nodos solicitantes.
El backbone se encuentra determinado por la particularidad de la red que se pretende modelar. Por ejemplo en el sistema DNS[157][158], el backbone es la estructura arborescente de servidores de nombres autoritativos, los nodos cache son los servidores recursivos y los solicitantes los resolvers. No todas las redes de contenido actuales presentan una separación clara entre los nodos enrutadores pertenecientes al backbone y los nodos de cache, más aun, algunas redes no incluyen el concepto de cache en sus arquitecturas; sin embargo la inclusión de un cache en una red de contenido masiva es altamente recomendable y finalmente es algo utilizado en la mayoría de las redes para brindar escalabilidad debido a su fácil incorporación a la mayoría de las arquitecturas.

En el modelo, el backbone presenta información perfecta, es decir, conoce en todo momento la información global de la red (en cada momento todos los contenidos alojados por cada nodo fuente). Por tanto la respuesta del backbone a las búsquedas realizadas por los nodos cache incluye a todos los nodos que en el momento alojan el contenido solicitado. Esto, más allá de ser una verdad en algunas redes (como las estructuradas DNS[157][158] o Chord[225]), se incluye solo a los efectos de aislar mejor el objeto de estudio: el cache de consultas. Se espera que estudios posteriores incluyan también la posibilidad de guardar información parcial o falsa obtenida del backbone de forma de incluir su impacto en la performance del sistema.

4.1.6 Objetivo de la Red

El **objetivo** de la red es maximizar el descubrimiento de contenido, esto significa que los solicitantes encuentren la mayor cantidad de alojamientos correctos respecto a los contenidos solicitados.

Sea R_k la variable aleatoria que representa la cantidad de respuestas correctas que recibe un solicitante al consultar por el contenido *k*.
El valor esperado de dicha variable puede estudiarse fácilmente si se observa que en estado estacionario, en cada nodo cache, todos los ciclos tienen el mismo comportamiento probabilístico: hay una solicitud que inicia el ciclo (guardando la

respuesta del backbone en cache), y luego un conjunto de *n* consultas adicionales que llegan dentro del período de validez del cache (de largo d_k); la primer consulta que llega luego del período de validez del cache marca el final del ciclo, y se considera dentro del ciclo siguiente (ver Ilustración 16).

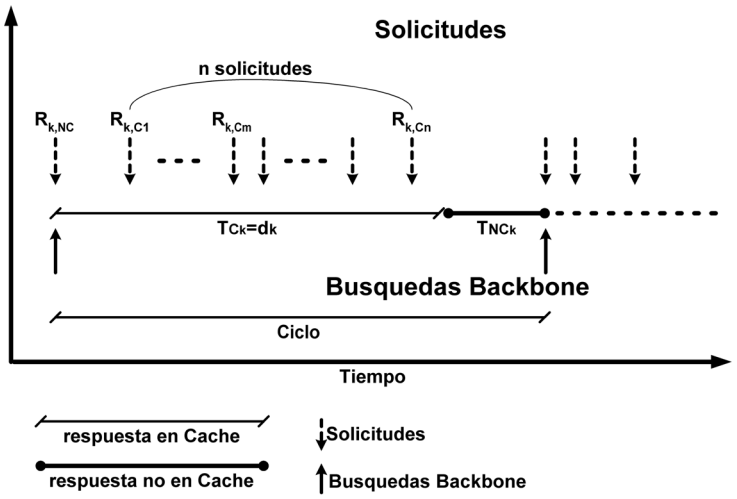


Ilustración 16. Dinámica de solicitudes y búsquedas en el backbone.

Por lo tanto, se puede calcular el valor esperado de la cantidad de respuestas correctas en estado estacionario para una consulta observando un único ciclo; y ponderando según la cantidad de solicitudes *n* que llegan dentro del período de validez del cache:

$$\overline{R_k} = \mathrm{E}\{R_k\} = \sum_{n\geq 0} \mathrm{E}\{R_k | \text{n solicitudes}\} p(\text{n solicitudes}) =$$

$$= \mathrm{E}\{R_{k,NC}\} p(0 \text{ solicitudes}) + \sum_{n\geq 1} \left(\frac{\mathrm{E}\{R_{k,NC}\} + \sum_{m=1}^{n} \mathrm{E}\{R_{k,Cm} | \text{n solicitudes}\}}{n+1} \right) p(\text{n solicitudes})$$

Siendo $R_{k,NC}$ la respuesta a la solicitud inicial (la respuestas que se guarda en cache) y $R_{k,C1} ... R_{k,Cn}$ las respuestas a las siguientes solicitudes donde se responde lo cacheado en la respuesta inicial.

Primera solicitud del ciclo, $R_{k,NC}$:

Si se considera que el backbone cuanta con información global, entonces cuando *k* no se encuentra cacheado, el agregador busca en el backbone todos los nodos fuente que alojan el contenido, el backbone responde en promedio y de forma completamente correcta que $\overline{A_k}$ nodos alojan el contenido, lo cual es devuelto al solicitante, obteniendo que: $\mathrm{E}\{R_{k,NC}\} = \overline{A_k} = \frac{\lambda_k}{\mu_k}$.

Siguientes solicitudes del ciclo, $R_{k,C1} ... R_{k,Cn}$:

Como se dijo anteriormente, el proceso de llegadas de solicitudes es un proceso de Poisson de tasa f_k . Donde si $S_k(d_k)$ representan la cantidad de consultas

realizadas por los solicitantes del contenido *k* en un período fijo $(t_0, t_0 + d_k]$, la probabilidad de realizar *n* consultas en ese período es:

$$p(S_k(d_k) = n) = \frac{(f_k d_k)^n e^{-f_k d_k}}{n!} \quad \forall k \in C, \forall n \in \aleph, \forall d_k \in \Re^+$$

donde f_k es la tasa o frecuencia de solicitudes del contenido k.

Por tanto:

$$p(\text{n solicitudes}) = \frac{(f_k d_k)^n e^{-f_k d_k}}{n!} \quad \forall k \in C, \forall n \in \aleph, \forall d_k \in \Re^+$$

Por otro lado, el valor promedio de respuestas correctas que el nodo cache devolverá en el período $(t_0, t_0 + d_k]$ corresponde al valor promedio de índices correctos en el cache durante ese intervalo. Al estudiar los agregadores se obtuvo la siguiente propiedad:

$$\begin{pmatrix} \text{cantidad promedio de alojamientos} \\ \text{validos luego de t tiempos de realizar} \\ \text{la consulta al backbone} \end{pmatrix} = \frac{\lambda_k}{\mu_k} e^{-\mu_k t}$$

Por tanto:

$$\begin{pmatrix} \text{cantidad promedio de alojamientos} \\ \text{validos en el intervalo} (t_0, t_0 + d_k] \end{pmatrix} = \frac{\int_0^{d_k} \frac{\lambda_k}{\mu_k} e^{-\mu_k t} . dt}{d_k} =$$

$$= \frac{-\frac{\lambda_k}{{\mu_k}^2} e^{-\mu_k t}\Big|_0^{d_k}}{d_k} = \frac{\lambda_k}{{\mu_k}^2 d_k}\left(1 - e^{-\mu_k d_k}\right)$$

Aplicando la siguiente propiedad bien conocida (ver por ejemplo [119]):

> **Propiedad:** Si en un proceso de Poisson se tienen *n* llegadas en un intervalo fijo de duración *T*, entonces esas llegadas se encuentran uniformemente distribuidas dentro del intervalo.

Y condicionado a conocer el número de llegadas de solicitudes, *n*; se tiene que:

$$\sum_{m=1}^{n} \mathrm{E}\{R_{k,Cm} | \text{n solicitudes}\} =$$

$$= n \begin{pmatrix} \text{cantidad promedio de alojamientos} \\ \text{validos en el intervalo} (t_0, t_0 + d_k] \end{pmatrix} =$$

$$= n \frac{\lambda_k}{{\mu_k}^2 d_k}\left(1 - e^{-\mu_k d_k}\right) \quad \forall k \in C, \forall n \in \aleph, \forall d_k \in \Re^+$$

Respuestas Correctas, $\overline{R_k}$:

Combinando los resultados anteriores, es posible calcular el número promedio de respuestas correctas a las consultas recibidas en un intervalo, condicionando respecto al número de consultas:

$$\overline{R_k} = \mathrm{E}\{R_k\} = \sum_{n\geq 0} \mathrm{E}\{R_k|\text{n solicitudes}\} p(\text{n solicitudes}) =$$

$$= \mathrm{E}\{R_{k,NC}\} p(0 \text{ solicitudes}) + \sum_{n\geq 1} \left(\frac{\mathrm{E}\{R_{k,NC}\} + \sum_{m=1}^{n} \mathrm{E}\{R_{k,Cm}|\text{n solicitudes}\}}{n+1} \right) p(\text{n solicitudes}) =$$

$$= \sum_{n\geq 0} \left(\frac{\frac{\lambda_k}{\mu_k} + n \frac{\lambda_k}{{\mu_k}^2 d_k} \left(1 - e^{-\mu_k d_k}\right)}{n+1} \right) \frac{(f_k d_k)^n e^{-f_k d_k}}{n!} =$$

$$= \frac{\lambda_k}{\mu_k} e^{-f_k d_k} \sum_{n\geq 0} \frac{(f_k d_k)^n}{(n+1)!} + \frac{\lambda_k}{{\mu_k}^2 d_k} e^{-f_k d_k} \left(1 - e^{-\mu_k d_k}\right) \sum_{n\geq 0} n \frac{(f_k d_k)^n}{(n+1)!} =$$

$$= \frac{\lambda_k}{\mu_k} e^{-f_k d_k} \frac{\left(e^{f_k d_k} - 1\right)}{f_k d_k} + \frac{\lambda_k}{{\mu_k}^2 d_k} e^{-f_k d_k} \left(1 - e^{-\mu_k d_k}\right) \left[e^{f_k d_k} - \frac{\left(e^{f_k d_k} - 1\right)}{f_k d_k} \right] =$$

$$= \frac{\lambda_k}{{\mu_k}^2 f_k d_k} \left[\mu_k \left(1 - e^{-f_k d_k}\right) + f_k \left(1 - e^{-\mu_k d_k}\right) - \frac{1}{d_k} \left(1 - e^{-f_k d_k}\right) \left(1 - e^{-\mu_k d_k}\right) \right]$$

Objetivo de la Red, ε:

Si se recuerda la característica de performance de una red de contenido, vista en la sección 2.1.4 *"La performance esta dada por la efectividad y eficiencia con que son utilizados los recursos de los nodos de la red. Dependiendo de la arquitectura de la red se tiene que la efectividad puede ser la probabilidad de encontrar las fuentes de determinado contenido, la precisión del enrutamiento, etc.; por otro lado la eficiencia está dada por el consumo de ancho de banda, la latencia en el acceso o consulta del contenido, la cantidad de replicas necesarias, la carga del procesador, etc.*
En general los nodos especiales (crawlers, control caches, relays, content caches, content replicators, content migrator, gateways y proxies) se presentan en las redes de contenido para mejorar algún aspecto de la performance..."

En este caso se estudia la performance en los *control caches*, donde se intenta maximizar la *eficacia* (entendiéndola como la probabilidad de encontrar las fuentes de los contenidos) sin descuidar la *eficiencia* (limitando el consumo de ancho de banda en los caches).

Si se supone que los nodos cache cuentan con un ancho de banda infinito, entonces no es necesario guardar en cache las respuestas, y frente a cada solicitud se puede realizar una búsqueda en el backbone, respondiendo de forma perfecta a todas las solicitudes.
La mayor cantidad de alojamientos válidos que pueden ser respondidos para cada contenido k, en promedio es $\overline{A_k} = \frac{\lambda_k}{\mu_k}$, por tanto el caso de una red ideal desde el punto de vista de la eficacia tendría un comportamiento: $\overline{R_{k,IDEAL}} = \overline{A_k} = \frac{\lambda_k}{\mu_k} \ \forall k \in C$.

Por tanto puede definirse la eficacia de la red, entendiéndola como que tan apartada se encuentra la cantidad de alojamientos correctos respondidos respecto a la red ideal:

$$\varepsilon = \begin{pmatrix} \text{eficacia de la} \\ \text{red de contenido} \end{pmatrix} = \frac{\sum_{k \in C} \overline{R_k} f_k}{\sum_{k \in C} \overline{R_{k,IDEAL}} f_k}$$

En forma matemática el objetivo del problema es maximizar la función objetivo, ε:

$$\max_{d_k \in \Re^+} \{\varepsilon\} = \max_{d_k \in \Re^+} \left\{ \frac{\sum_{k \in C} \overline{R_k} f_k}{\sum_{k \in C} \overline{R_{k,\text{IDEAL}}} f_k} \right\} =$$

$$= \max_{d_k \in \Re^+} \left\{ \frac{\sum_{k \in C} \frac{\lambda_k}{\mu_k{}^2 d_k} \left[\mu_k \left(1 - e^{-f_k d_k}\right) + f_k \left(1 - e^{-\mu_k d_k}\right) - \frac{1}{d_k} \left(1 - e^{-f_k d_k}\right)\left(1 - e^{-\mu_k d_k}\right) \right]}{\sum_{k \in C} \frac{\lambda_k}{\mu_k} f_k} \right\}$$

4.1.7 Resumen

Definición de ***CCP*** (Content Caching Problem):

$$\max_{d_k \in \Re^+} \left\{ \frac{\sum_{k \in C} \frac{\lambda_k}{\mu_k{}^2 d_k} \left[\mu_k \left(1 - e^{-f_k d_k}\right) + f_k \left(1 - e^{-\mu_k d_k}\right) - \frac{1}{d_k} \left(1 - e^{-f_k d_k}\right)\left(1 - e^{-\mu_k d_k}\right) \right]}{\sum_{k \in C} \frac{\lambda_k}{\mu_k} f_k} \right\}$$

sujeto a:

$$\beta_S \sum_{k \in C} f_k + \alpha_B \sum_{k \in C} \frac{f_k}{1 + d_k f_k} \frac{\lambda_k}{\mu_k} \le BW_{IN}$$

$$\alpha_S \sum_{k \in C} f_k \frac{\lambda_k}{\mu_k} + \beta_B \sum_{k \in C} \frac{f_k}{1 + d_k f_k} \le BW_{OUT}$$

$$d_k \in \Re^+ \ \forall k \in \text{C variables de decisión}$$

$$f_k, \lambda_k, \mu_k, \alpha_S, \alpha_B, \beta_S, \beta_B, BW_{IN}, BW_{OUT} \in \Re^+ \ \ \forall k \in \text{C}$$

Ecuación 2. Formulación del modelo *CCP*.

Resumen: Establecido un patrón de alojamiento de contenido y de solicitud de contenido en la red, el problema consiste en maximizar la localización del contenido solicitado, permitiendo variar la política de expiración de la información en el nodo agregador.

4.2 CCCP - Content Class Caching Problem

En esta sección se generaliza el problema ***CCP***, introduciendo la noción de clases de contenido. Esta generalización del problema se denomina por las siglas ***CCCP* (Content Class Caching Problem)** y tiene como objetivo la simplificación en la búsqueda de soluciones.

4.2.1 Motivación

En general, las redes de contenido manejan una cantidad suficientemente grande de contenidos para que la formulación de la Ecuación 2 sea inconveniente y de difícil tratamiento con los métodos tradicionales de resolución para problemas de optimización no lineales.

A su vez, parece demasiado costoso mantener el conocimiento de los parámetros f_k, λ_k y μ_k para cada contenido k de la red en cualquier sistema real con gran variedad de contenidos.

Por tanto se considera que en ciertos casos será útil agrupar los contenidos en clases de contenido, formándose una clase de contenido por todos aquellos contenidos que presentan parámetros f_k, λ_k y μ_k lo suficientemente similares como para poder tratarlos como idénticos.

4.2.2 Clases de Contenido

Supóngase que los contenidos $c \in C$ se agrupan en K conjuntos (**clases**), donde dos contenidos pertenecen a la misma clase si y solo si sus parámetros son idénticos:

$$C = C_1 \cup C_2 ... \cup C_K$$

$$\left.\begin{array}{l} \forall c_i, c_j \in C_k \\ \forall k \in [1..K] \end{array}\right\} \Rightarrow \begin{cases} f_{c_i} = f_{c_j} \\ \lambda_{c_i} = \lambda_{c_j} \\ \mu_{c_i} = \mu_{c_j} \end{cases}$$

El tamaño de la clase k, representado por l_k, son la cantidad de contenidos en dicha clase:

$$\|C_k\| = l_k \ \forall k \in \{1,..,K\}.$$

Por tanto la cantidad de contenido total en la red es:

$$\|C\| = \sum_{c \in C} 1 = \sum_{k \in K} \|C_k\| = \sum_{k \in K} l_k \ .$$

4.2.3 Formulación

Razonando de forma completamente análoga al problema *CCP*, puede realizarse la definición del ***CCCP*** (Content Class Caching Problem):

$$\max_{d_k \in \Re^+} \left\{ \frac{\sum_{k \in K} \frac{l_k \lambda_k}{\mu_k{}^2 d_k} \left[\mu_k \left(1 - e^{-f_k d_k}\right) + f_k \left(1 - e^{-\mu_k d_k}\right) - \frac{1}{d_k} \left(1 - e^{-f_k d_k}\right)\left(1 - e^{-\mu_k d_k}\right) \right]}{\sum_{k \in K} \frac{\lambda_k}{\mu_k} l_k f_k} \right\}$$

sujeto a:

$$\beta_S \sum_{k \in K} l_k f_k + \alpha_B \sum_{k \in K} \frac{l_k f_k}{1 + d_k f_k} \frac{\lambda_k}{\mu_k} \leq BW_{IN}$$

$$\alpha_S \sum_{k \in K} l_k f_k \frac{\lambda_k}{\mu_k} + \beta_B \sum_{k \in K} \frac{l_k f_k}{1 + d_k f_k} \leq BW_{OUT}$$

$d_k \in \Re^+ \ \forall k \in K$ variables de decisión

$l_k, f_k, \lambda_k, \mu_k, \alpha_S, \alpha_B, \beta_S, \beta_B, BW_{IN}, BW_{OUT} \in \Re^+ \ \ \forall k \in K$

Ecuación 3. Formulación del modelo *CCCP*.

Donde los parámetros tienen el siguiente significado:

l_k : cantidad de contenidos que pertenecen a la clase k ($[l_k] = 1$).

f_k : tasa de solicitud para cada contenido de la clase k ($[f_k] = 1/s$).

λ_k : tasa de comienzo de alojamiento de un contenido de la clase k realizada por cada fuente ($[\lambda_k] = 1/s$).

μ_k : tasa de validez (desalojamiento) de un contenido de la clase k realizada por cada fuente ($[\mu_k] = 1/s$).

α_S : tamaño de la respuesta a una solicitud de contenido, proporcional a la cantidad de respuestas ($[\alpha_S] = byte$).

α_B : tamaño de la respuesta a una búsqueda en el backbone de un contenido, proporcional a la cantidad de respuestas ($[\alpha_B] = byte$).

β_S : tamaño de un paquete de solicitud de contenido ($[\beta_S] = byte$).

β_B : tamaño de un paquete de búsqueda en el backbone de un contenido ($[\beta_B] = byte$).

BW_{IN}, BW_{OUT} : limitación de ancho de banda en el nodo agregador ($[BW_{IN}] = [BW_{OUT}] = byte/s$).

Resumen: Establecidas clases de contenido en una red, donde cada clase comparte un patrón de alojamiento y de solicitud común de contenidos, el problema consiste en maximizar la localización del contenido solicitado, permitiendo variar la política de expiración de la información en el nodo agregador.

4.2.4 Ejemplo Ilustrativo

En esta sección se muestra un ejemplo sencillo de aplicación del modelo *CCCP*.
Supóngase que se tiene una red de contenido con solamente 5 contenidos distintos:

$$K = \{1,..,5\}, \quad l_k = 1 \ \forall k \in K$$

La frecuencia de solicitud de contenido y la tasa de comienzo de alojamiento son inversamente proporcionales, vea la Tabla 5 para el detalle de la descripción del tipo de contenido que aloja la red.

k	l_k	f_k (1/s)	λ_k (1/s)	μ_k (1/s)
1	1	0.01	100.00	1
2	1	0.10	10.00	1
3	1	1.00	1.00	1
4	1	10.00	0.10	1
5	1	100.00	0.01	1

Tabla 5. Contenido alojado en la red de ejemplo.

Las consultas en el backbone consumen mayor ancho de banda que las realizadas por los clientes, en la Tabla 6 puede verse que una consulta en el backbone requiere tres veces el ancho de banda de una consulta de un solicitante.

β_S (bytes)	1
α_S (bytes)	1
β_B (bytes)	3
α_B (bytes)	3

Tabla 6. Comunicación en la red de ejemplo.

El problema presenta únicamente restricciones en el consumo de ancho de banda. Si el ancho de banda se encuentra holgado es lógico pensar que el agregador no guarde ninguna consulta como un método para maximizar la cantidad de respuestas correctas.
Para ver este efecto se busca la solución al problema con cinco anchos de banda distintos, $BW_{IN} = 112, 114, 117, 123 \text{ y } 127$ (se mantiene $BW_{OUT} = 1000$). La Tabla 7 muestra los resultados para cada restricción de ancho de banda (obtenidos empleando el paquete AMPL[7] con la biblioteca MINOS[224], como se discute en detalle en en el capítulo 5).

		Resultados				
		ε Función Objetivo	BW_{IN} (bytes/s)	BW_{IN} consumido (bytes/s)	BW_{OUT} consumido (bytes/s)	*d=[1,2,3,4,5]* tiempo de expiración por clase de contenido (s)
Instancias	1	0.198563	112	112	93.9766	[1307750, 8639810, 19096900, 245619, 0.0237167]
	2	0.558249	114	114	60.3430	[1075230, 8898270, 0.798618, 0.292056, 0.0551847]
	3	0.834110	117	117	40.6829	[69.0388, 6.37234, 1.16435, 0.383453, 0.0975636]
	4	0.997607	123	123	153.3330	[0.105319, 0.103796, 0.092098, 0.0564873, 0.0137929]
	5	0.999999	127	126.1097	338.3000	[0, 0, 0, 0, 0]

Tabla 7. Resultados numéricos para las instancias de ejemplo.

Puede observarse que la solución siempre alcanza la utilización máxima disponible del BW_{IN} , excepto en la instancia 5, donde sin almacenar consultas previas no se sobrepasa el ancho de banda posible. Como es de esperar, a medida que se flexibiliza la restricción del ancho de banda se van logrando mejores soluciones: la

eficiencia varía enormemente desde un 20% ($\varepsilon = 0.198563$) a un 100% (cuando no se cachea) respecto a una red con conocimiento perfecto.

Las soluciones son menos intuitivas a medida que el ancho de banda es más restrictivo. No es cierto pensar que la mejor política para recordar consultas previas es hacerlo con el contenido más solicitado, ni tampoco hacerlo con el contenido menos cambiante. Por ejemplo, en la instancia 1 (la más comprometida en ancho de banda disponible) conviene almacenar por más tiempo los contenidos intermedios (ver Ilustración 17) como compromiso entre solicitud y cambio en alojamiento.

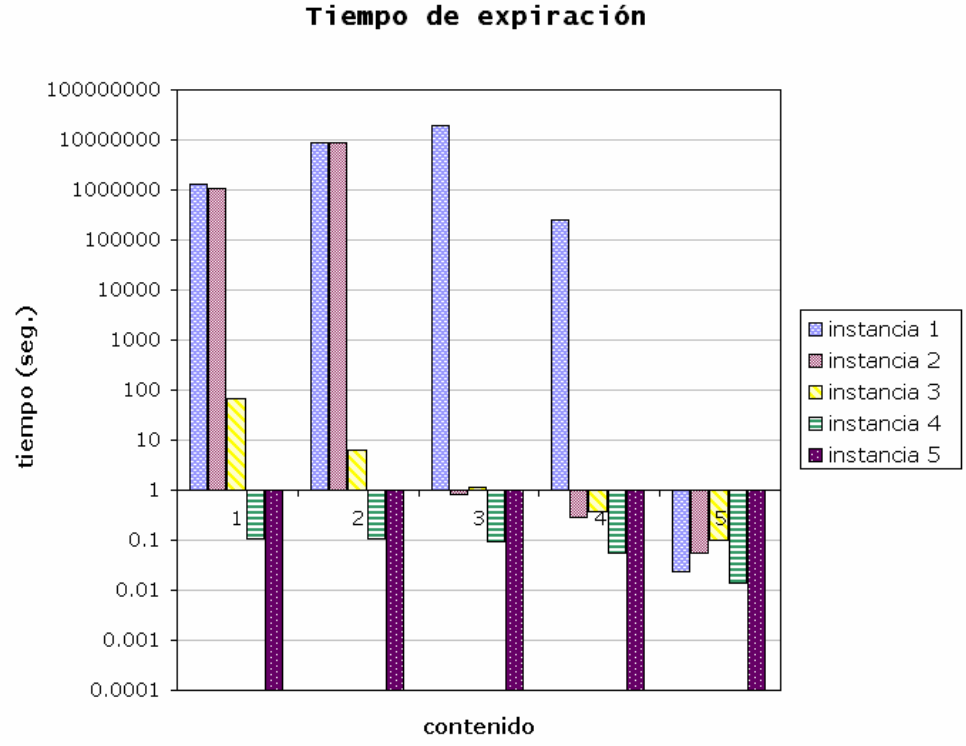


Ilustración 17. Tiempo de expiración para las diferentes instancias según clase de contenido.

4.3 Otros Enfoques: Tiempo de Expiración en Caches

Existen muchos trabajos relacionados a la estimación de tiempos de expiración en los nodos agregadores de una red de contenido. En general tiene como objetivo fundamental reducir la latencia y el tráfico en la entrega del contenido al usuario final. En esta sección se resume el trabajo relacionado en este aspecto, al cual se tuvo acceso durante el período de estudio.

En primer lugar es importante resaltar que la tecnología de caches trasciende su aplicación a las redes de contenido. Es utilizada para el acceso a memoria y/o disco rígido en los computadores [10][232], en los sistemas de archivos (distribuidos o no) [3][51][169], en los protocolos de enrutamiento, en las bases de datos distribuidas [184][211], en redes móviles[31][39], en las redes de contenido (para las consultas [17][140][147][152][183] o en el alojamiento de la información [72][84][108]), etc.

Según la aplicación se han desarrollado distintos modelos y técnicas, siendo de interés en este capítulo solo aquellas relacionadas a las redes de contenido.

4.3.1 Consistencia de la Información

En muchas redes de contenido es crucial la consistencia de la información. En los nodos agregadores la inconsistencia ocurre cuando la respuesta brindada por el cache ya no es válida porque la fuente actualizó el contenido.

Por ejemplo, en el sistema DNS es sumamente importante asegurar que las respuestas a traslaciones entre nombres de dominios e IPs sean correctas, en otro caso se enfrentarían algunos problemas de seguridad (veremos más adelante que en la actualidad no se tiene toda la consistencia deseada en esta red).

4.3.2 Mecanismo de Time-to-Live (TTL)

Cuando es necesaria la consistencia en la información, las propias redes de contenido incluyen dentro de sus protocolos algunos mecanismos para asegurarla.
En los casos en que existen nodos agregadores el método más empleado utiliza el concepto de TTL (time to live).
Cada contenido es emitido por las fuentes con un tiempo de vida, el TTL. Este tiempo de vida determina el máximo tiempo en que la fuente asegura que el contenido va a permanecer incambiado y por tanto es el máximo tiempo de expiración permitido en los caches. En la primera solicitud de un contenido, el nodo agregador lo obtiene de alguna fuente junto con un TTL, durante las próximas solicitudes incluidas en el intervalo TTL el cache puede responder por el contenido sin consultar nuevamente a la fuente, luego de ese período el nodo agregador debe borrar el contenido del cache y volver a consultar a la fuente por el contenido.

Múltiples redes utilizan el concepto de TTL, entre ellas la red de DNS y la World Wide Web. Los nodos agregadores en la red de DNS son conocidos como servidores recursivos, mientras que en la Web son los proxies Web.

World Wide Web – Proxies Web

La mayor cantidad del trabajo relacionado se encuentra en el estudio de los proxies caches para la WWW [23][35][41][43][45][103][131][132][133][134][159][215][235][237][239][238].
El mecanismo TTL asegura parcialmente la consistencia de la información en los caches de HTTP, en general los estudios previos sobre este tema se enmarcan en dos orientaciones posibles:

- elegir eficientemente que contenido alojar en el cache, suponiendo una restricción en capacidad.
- definir políticas (o algoritmos) sobre como alojar eficientemente el contenido (entre ellas elegir los tiempos de expiración).

Respecto a que contenido alojar en el cache, varias técnicas de pre-búsqueda (prefetching) han sido propuestas y probadas [43][45][103][235][237][239].

Respecto a la definición de políticas existen varios trabajos que ensayan mediante simulación o con casos reales distintos algoritmos. Algunos de estos trabajos se asemejan porque presentan modelos de optimización que tienen en cuenta la dinámica de la información en las fuentes:

- Xing Y. en su trabajo de maestría *"Caching on the Changing Web"*[240] presenta un modelo de optimización para el cache de lo proxies Web. La función a maximizar es el beneficio logrado por la disminución de latencia en los usuarios finales y el costo es el espacio de alojamiento disponible en el cache. Parte de su análisis se basa en clasificar a los contenidos en 4 clases según la tasas de solicitudes y de alojamiento.
- Gopalan P. et. al. en el trabajo *"Caching with Expiration Times"*[95] muestran que algunas políticas complejas no difieren en performance respecto a otras políticas sencillas.
- Shim J. et. al. en [215] presentan un enfoque muy similar al trabajo de Xing Y., no divide los contenidos en clases y estudia menos algoritmos para las políticas de cacheo.
- Krishnamurthy B. et. al. [131][132][133][134] realizan algoritmos para estimar el tiempo de expiración de la información en los caches a partir de la tasa de solicitudes.
- Chen X. et. al. [35] estudia el tiempo de expiración de los contenidos Web. También divide el contenido en 4 clases y presenta un algoritmo distinto a los anteriores para la política de expiración.

Sistema de DNS – Servidores Recursivos

A pesar de su importancia, no se ha encontrado mucha literatura respecto a las técnicas de almacenamiento de consultas en los servidores recursivos. La política de caching en el sistema DNS debe diferir de otras redes de contenido (como la World Wide Web) dada la naturaleza del contenido que aloja:

- Los contenidos son de tamaño muy pequeño y no resulta un problema alojarlos ni transmitirlos.
- No encontrarse una respuesta en el cache tiene un costo de latencia más elevado en comparación a otros sistemas.
- En general los tiempos de expiración son mucho menores a los tiempos entre cambios en las fuentes del contenido.
- La información se encuentra generalmente redundante, y los tiempos de acceso difieren entre las fuentes de un contenido.
- Existen muchos servidores autoritativos mal configurados (por ejemplo delegaciones LAME), generando retardos excesivos por *time-out* en las consultas.

Los servidores recursivos actualmente incorporan una política pasiva en el manejo del cache. El contenido es almacenado exclusivamente luego de que un solicitante lo consulta, y es guardado estrictamente según el TTL brindado por la fuente.
Los estudios sobre este tema analizan mejoras en distintos aspectos:

- definir políticas (o algoritmos) sobre como alojar eficientemente el contenido (entre ellas elegir los tiempos de expiración).
- relacionar las búsquedas en DNS con las utilización de otras redes (principalmente la WWW).

Dado el pequeño tamaño que tiene el contenido en la red DNS, no es necesario elegir eficientemente que contenido alojar en el cache, por tanto las técnicas relacionadas a este aspecto de la WWW no han sido trasladadas al sistema DNS.

La mayor cantidad de trabajo se concentra en la definición de políticas para almacenar eficientemente las consultas en los servidores recursivos. E. Cohen y H. Kaplan parecen reunir gran parte del trabajo en esta área, desarrollando estudios en el sistema DNS[44] y en la World Wide Web [41][45][46]. La primera mejora que se plantea es la búsqueda pro-activa (o política de renovación): para contenidos muy

solicitados se refrescan los contenidos del cache generando consultas insolicitadas, de esta forma se elimina la latencia en la primera consulta de cada contenido.
Otras mejoras se dirigen a la manipulación (en general la reducción) de los TTL almacenados en el cache. J. Jung et. al. realizan modelos sobre el comportamiento de los mecanismos de TTL [124] y su aplicación para definir políticas sobre los cache de DNS [125]. En general todos sus estudios concuerdan en que la influencia de las políticas de manejo del TTL provoca un efecto mínimo en la percepción del usuario final en la velocidad de navegación [217].

Algunos autores han discutido la utilización de políticas conjuntas entre la World Wide Web y el DNS para reducir la latencia en el usuario final. La idea subyacente es buscar la cooperación y sincronía entre los caches de DNS y los de la WWW.
La pre-búsqueda requiere de alguna información predictiva sobre el comportamiento futuro de las solicitudes. En general los caches hacen uso de información estadística, suponiendo regularidad entre los usuarios (por ejemplo se supone que un contenido muy solicitado con anterioridad seguirá siendo muy solicitado). Sin embargo conocer más sobre las aplicaciones que generan las consultas en DNS proporciona una predicción más refinada. A. S. Hughes y J. Touch estudian la utilización de las trazas de un cache Web para predecir las futuras navegaciones de los clientes y de esa forma poder realizar pre-búsquedas en el DNS de forma más eficiente [109][110]. E. Cohen y H. Kaplan también estudian esta idea, no solo analizan la pre-resolución de nombres [43], sino que también estudian la validación simultánea entre el contenido Web y el de DNS, logrando buenos resultados en entornos pequeños frente a contenido de poco cambio en el DNS [44].

¿Respetar Estrictamente el TTL?

Para las redes más difundidas que implementan el mecanismo de TTL esta pregunta tiene una única respuesta: no se respeta estrictamente el tiempo de vida.
Es decir que los clientes de una red que ofrece un mecanismo de TTL muchas veces no respetan dichos tiempos y mantienen la información como válida pasado inclusive su tiempo de vida. Esto incorpora serios inconvenientes desde el punto de vista del administrador de la red, pero algunas ventajas desde el punto de vista del usuario final: básicamente el respectar estrictamente los tiempos de TTL es una restricción sobre el problema genérico de disminuir los retardos y el tráfico en la entrega del contenido al usuario final.

Por ejemplo en la World Wide Web, todas las aplicaciones de navegación (Internet Explorer[117], Mozilla-Netscape[161], etc.) no respetan el tiempo de expiración que reciben en cada conexión HTTP sobre las páginas navegadas. Estando en competencia por ofrecer una navegación que parezca la más rápida es impensable suponer que alguna de las aplicaciones de navegación se apegue al estándar en este aspecto, puesto que esto indefectiblemente generaría mayor cantidad de consultas a las fuentes y por tanto mayor retardo de despliegue de las páginas Web.
El problema surge porque el TTL ofrecido por las fuentes rara vez responde efectivamente a un cambio en el contenido. Si cada vez que se venciera el TTL la fuente realizara un cambio en el contenido indefectiblemente los navegadores deberían apegarse al estándar porque en otro caso estarían desplegando información errónea, algo que los usuarios percibirían con gran desagrado.

Visto desde la situación de los generadores de contenido, en general las fuentes desconocen con anticipación cuando van a actualizar su contenido, por tanto solo pueden informar en el TTL del tiempo medio entre cambios y no exactamente cuando va a ocurrir el próximo. Tan clara es esta limitante que no se conocen implementaciones para servidores Web, ni para servidores autoritativos, que permitan

agendar cambios y reducir paulatinamente el TTL para informar exactamente a los solicitantes del próximo cambio. Además de que pocos administradores de estos sistemas conocen este nivel de detalle y por tanto en general se utiliza el TTL que implementa por defecto la aplicación.
Por tanto la información de TTL no especifica en la práctica el tiempo mínimo asegurado por la fuente de permanencia incambiada del contenido sino que informa simplemente del tiempo promedio entre cambios. Esto tiene una implicancia aun más grave: respetar estrictamente el mecanismo de TTL no asegura la consistencia de la información, solo brinda una hipotética cota superior al tiempo en que los solicitantes van a acceder al contenido desactualizado.

Las implementaciones para servidores recursivos y para proxies Web que pertenecen al dominio público (licenciamiento GNU o similar) respetan completamente el mecanismo de TTL (por ejemplo BIND[18] y Squid[222]), sin embargo algunas implementaciones propietarias presentan el hecho de apartarse del estándar como una ventaja comparativa argumentando mejoras en la performance y prestaciones.

4.3.3 Otras Redes y Mecanismos

Red de Pares

En las redes de pares para compartir archivos no existe el mecanismo TTL y seguramente su utilización seria ineficiente dado el dinamismo indeterminado del contenido alojado.

Varias propuestas utilizan una infraestructura P2P para ofrecer de forma descentralizada y anónima un Proxy Web, entre ellas se encuentra la aplicación Squirrel[120] que utiliza Pastry[190] como infraestructura de red.

En la actualidad, el tráfico de las aplicaciones P2P representa un porcentaje importante del consumo en los enlaces internacionales de acceso a Internet en cada ISP. Estos enlaces en general son costosos y su crecimiento una preocupación para los ISP. Varias empresas ofrecen equipamiento que garantiza una reducción del consumo en los enlaces internacionales debido a aplicaciones P2P. Este equipamiento almacena y guía el tráfico de las aplicaciones P2P más populares con el fin de un uso eficiente del enlace. A pesar de que no se cuenta con mayor información, estos equipamientos son nodos agregadores para estas redes y presentan alguna política respecto a los tiempos de expiración. Ejemplos de equipamiento: PacketShaper de Packeteer[186], Peer-to-Peer-Element de Sandvine[209] y PeerCache de Joltid[191].

4.4 Discusión y Conclusiones

En este capítulo se define el modelo *CCP* y su generalización *CCCP*. Estos modelos representan la problemática del descubrimiento del contenido en un nodo del tipo agregador.

A continuación, la sección 4.4.1 muestra explícitamente porque surge el compromiso entre publicación y búsqueda para el descubrimiento del contenido. Mientras que la sección 4.4.2 muestra una relación importante entre el modelo *CCP* y el *CCCP*, siendo posible generar una instancia del modelo *CCCP* a partir de una instancia del

modelo *CCP* (esto se estudia en más detalle con los casos de prueba en el capítulo siguiente).

4.4.1 Compromiso entre Publicación y Búsqueda

Como se adelantó al describir los agregadores (sección 4.1.2), el compromiso entre publicación y búsqueda en dichos nodos cache se encuentra dado por el tiempo que se decide tener guardadas las consultas previas; es decir en el tiempo de cache d_k para todo contenido *k*.

A continuación se muestran casos extremos de valores de d_k en el problema *CCP*.

Tiempo de cache d_k muy alto:

Puede observarse que a medida que crece el tiempo de cache d_k de un contenido, el número promedio de respuestas correctas disminuye al punto de que la red deja de cumplir su cometido (para este contenido *k*) sin encontrar ningún alojamiento válido: $\lim_{d_k \to +\infty} \overline{R_k} = 0$.

Por tanto si se supone un tiempo de cache infinito para todos los contenidos, la red no responde correctamente ningún alojamiento válido, utilizando un ancho de banda mínimo (mínima eficacia y máxima eficiencia):

$$\lim_{\substack{d_k \to +\infty \\ \forall k \in C}} \varepsilon = \inf_{\substack{d_k \in \Re^+ \\ \forall k \in C}} \{\varepsilon\} = 0$$

$$\lim_{\substack{d_k \to +\infty \\ \forall k \in C}} \left(\begin{matrix} \text{ancho de banda entrante} \\ \text{en agregador} \end{matrix} \right) = \inf_{\substack{d_k \in \Re^+ \\ \forall k \in C}} \left\{ \left(\begin{matrix} \text{ancho de banda entrante} \\ \text{en agregador} \end{matrix} \right) \right\} = \beta_S \sum_{k \in C} f_k$$

$$\lim_{\substack{d_k \to +\infty \\ \forall k \in C}} \left(\begin{matrix} \text{ancho de banda saliente} \\ \text{en agregador} \end{matrix} \right) = \inf_{\substack{d_k \in \Re^+ \\ \forall k \in C}} \left\{ \left(\begin{matrix} \text{ancho de banda saliente} \\ \text{en agregador} \end{matrix} \right) \right\} = \alpha_S \sum_{k \in C} f_k \, \overline{A_k}$$

Tiempo de cache d_k muy bajo:

Por otro lado, si el tiempo de cache d_k de un contenido se reduce a valores muy bajos, el nodo cache responde a todas las solicitudes por dicho contenido de forma perfecta (responde todos los nodos fuente que alojan el contenido): $\lim_{d_k \to 0^+} \overline{R_k} = \overline{A_k} = \frac{\lambda_k}{\mu_k}$

Por tanto si se supone un tiempo de cache nulo para todos los contenidos, la red responde perfectamente todos los alojamientos válidos, "utilizando un ancho de banda máximo" (máxima eficacia y mínima eficiencia):

$$\lim_{\substack{d_k \to 0^+ \\ \forall k \in C}} \varepsilon = \sup_{\substack{d_k \in \Re^+ \\ \forall k \in C}} \{\varepsilon\} = 1$$

$$\lim_{\substack{d_k \to 0^+ \\ \forall k \in C}} \left(\begin{array}{c} \text{ancho de banda entrante} \\ \text{en agregador} \end{array} \right) = \sup_{\substack{d_k \in \Re^+ \\ \forall k \in C}} \left\{ \left(\begin{array}{c} \text{ancho de banda entrante} \\ \text{en agregador} \end{array} \right) \right\} = \beta_S \sum_{k \in C} f_k + \alpha_B \sum_{k \in C} f_k \overline{A_k}$$

$$\lim_{\substack{d_k \to 0^+ \\ \forall k \in C}} \left(\begin{array}{c} \text{ancho de banda saliente} \\ \text{en agregador} \end{array} \right) = \sup_{\substack{d_k \in \Re^+ \\ \forall k \in C}} \left\{ \left(\begin{array}{c} \text{ancho de banda saliente} \\ \text{en agregador} \end{array} \right) \right\} = \alpha_S \sum_{k \in C} f_k \overline{A_k} + \beta_B \sum_{k \in C} f_k$$

Relación de Compromiso entre los valores d_k :

El anterior análisis deja en evidencia que la relación de compromiso entre los valores de los tiempos de cache de cada contenido es un problema de difícil solución.

Según el dinamismo que presente el alojamiento de contenido por parte de las fuentes (λ_k , μ_k) y la cantidad de repeticiones en solicitudes (f_k) es el tiempo óptimo para los nodos de agregación en guardar las consultas previas de un contenido, todo restringido a las posibilidades de ancho de banda disponible.

Las mismas conclusiones pueden extraerse en el modelo *CCCP*.

4.4.2 Relación entre los modelos CCP-CCCP. Simetría en la Solución

Agrupar los contenidos en clases, permite hacer tratable matemáticamente el problema de guardado de índices en los agregadores. Esto se debe a que cuando se agrupa el contenido en clases se está reduciendo el espacio de soluciones factibles drásticamente. El *CCCP* descarta todas las soluciones que no cumplan con un tratamiento igualitario frente a contenido idéntico:

$$\forall c_i, c_j \in C / f_{c_i} = f_{c_j}, \lambda_{c_i} = \lambda_{c_j}, \mu_{c_i} = \mu_{c_j} \Rightarrow d_{c_i} = d_{c_j}$$

Es decir, el *CCCP* presenta como soluciones factibles solo aquellas donde la red realiza el mismo tratamiento (mismo tiempo de cache d_k) para los contenidos con parámetros idénticos (f_k , λ_k y μ_k). Mientras que en el problema genérico *CCP* es posible encontrar $c_i, c_j \in C / f_{c_i} = f_{c_j}, \lambda_{c_i} = \lambda_{c_j}, \mu_{c_i} = \mu_{c_j}$ y $d_{c_i} \neq d_{c_j}$.

Esta imposición de equidad o simetría en la solución del problema *CCCP* en realidad provoca obviar algunas soluciones al problema genérico *CCP* que pueden ser útiles en algún contexto particular. Esto se ilustra en el siguiente ejemplo sencillo.

Ejemplo de Pérdida de Soluciones

Supongamos, para simplificar las formulas, una red irreal que solo cuenta con dos contenidos y que dichos contenidos pertenecen a la misma clase. Siendo los parámetros:

$C = \{c_1, c_2\}, K = \{1\}$

$l_1 = 2$

$f_1 = \lambda_1 = \mu_1 = 1$

$\alpha_S = \alpha_B = \beta_S = \beta_B = 1$

$BW_{IN} \geq BW_{OUT} = 2.25$

Simplificando las formulaciones de los problemas:

CCP

$$\max_{d_k \in \Re^+} \left\{ \left[\frac{1}{d_1}\left(1 - e^{-d_1}\right) - \frac{1}{2{d_1}^2}\left(1 - e^{-d_1}\right)^2 \right] + \left[\frac{1}{d_2}\left(1 - e^{-d_2}\right) - \frac{1}{2{d_2}^2}\left(1 - e^{-d_2}\right)^2 \right] \right\}$$

sujeto a:

$$2 + \frac{1}{1+d_1} + \frac{1}{1+d_2} \leq BW_{OUT}$$

$$d_1, d_2 \in \Re^+ \text{ variables de decisión}$$

CCCP

$$\max_{d_k \in \Re^+} \left\{ \frac{2}{d_1}\left(1 - e^{-d_1}\right) - \frac{1}{{d_1}^2}\left(1 - e^{-d_1}\right)^2 \right\}$$

sujeto a:

$$2 + \frac{2}{1+d_1} \leq BW_{OUT}$$

$$d_1 \in \Re^+ \text{ variable de decisión}$$

La función objetivo del problema *CCCP* es monótona decreciente, mientras que la limitación de ancho de banda es monótona creciente, por tanto la solución óptima se alcanza cuando el agregador se encuentra consumiendo su máximo ancho de banda: $d_1 = 7$, $\varepsilon = 0.2650827874$.

Por otro lado, el problema *CCP* presenta soluciones mejores si se buscan soluciones asimétricas, por ejemplo:

$d_1 = 3.40317$, $d_2 = 42.6857$ y $\varepsilon = 0.2668735726$.

4.4.3 Modelos CCP, CCCP y Mecanismos de TTL

Los modelos presentados en este capítulo, *CCP* y *CCCP*, no contemplan el mecanismo de TTL que varias redes de contenido incluyen para preservar la consistencia de la información.

La incorporación del mismo al problema *CCP* es inmediata agregando la restricción sobre cada contenido.

Para el *CCCP* su utilización es posiblemente una restricción demasiado fuerte, puesto que cada clase de contenido debería respeta el mismo TTL, o de forma más general el TTL menor entre todos los contenidos de la clase (en este caso es necesario estudiar en profundidad el impacto de la restricción del TTL sobre la solución).

Siguiendo con lo expresado en la sección 4.3.2, los nodos de las redes con mecanismos de TTL no siempre respetan estrictamente el tiempo de vida. Por tanto, en primera instancia, se considera adecuada la aplicación de los modelos *CCP* y *CCCP* a las redes de contenido en general (inclusive aquellas con mecanismos de TTL). Es necesario a futuro un estudio comparativo entre la consistencia de la información en las redes actuales con TTL y las que surjan de utilizar los modelos *CCP* y *CCCP*.

En el capítulo siguiente se aplican los modelos a casos de redes reales: una red P2P (sin mecanismo de TTL) y al sistema DNS (incluye TTL).

Casos de Estudio: P2P y DNS 5

Para mostrar la validez de los modelos particulares (***CCP*** y ***CCCP***) se eligieron dos casos de prueba dispares y representativos de las redes de contenido estudiadas. Estos casos de prueba son:

- ***P2P***: Se estudia el comportamiento del modelo en una red de intercambio de archivos. En la actualidad, el tipo de red más utilizado para estos fines (según la taxonomía presentada) pertenecen a las redes sintácticas insensibles con distribución jerarquizada (donde se incluyen por ejemplo la red FastTrack de KaZaA[127]). Los valores utilizados en las instancias de este caso de prueba se extrajeron de la bibliografía.
- ***DNS***: También se muestra la utilidad del modelo particular en la red DNS (Domain Name System)[157][158], que es una red semántica sensible de distribución jerarquizada. Los valores de las instancias de este caso de prueba se extrajeron de la realidad, utilizando datos de la empresa de telecomunicaciones ANTEL[2].

5.1 P2P - File Sharing System

Los sistemas para compartir archivos entre pares han sido de inspiración para gran parte de este trabajo; debido a la forma particular que tienen de encarar el descubrimiento del contenido y a su dominancia en la actualidad.
En particular las redes sintácticas insensibles con distribución jerarquizada conocidas como redes Superpeers son en la actualidad la forma más popular de compartir archivos en Internet y objeto de estudio en esta sección.

5.1.1 Objetivos

El objetivo de este caso de estudio es mostrar la utilidad de los modelos *CCP* y *CCCP* para una red del tipo Superpeer. Para ello, es necesario alcanzar los siguientes objetivos específicos:

- Obtener de la bibliografía los parámetros necesarios para generar instancias del problema *CCP*.
- A partir de una instancia del *CCP* agrupar el contenido similar en clases para generar instancias del *CCCP*.
- Especificar un mecanismo de solución para los problemas *CCP* y *CCCP*.
- Comparar el método de solución con otros posibles.
- Y comparar la utilización en un nodo agregador de una política *CCCP* con otras políticas posibles.

5.1.2 Parametrización del Modelo: Obtención de las Instancias

No existe mucha información documentada sobre trazas reales de este tipo de redes; la mayoría de las investigaciones publicadas se basan en trazas de redes más antiguas como Gnutella[52]. Recientes trabajos como los de Gummadi et.al. [102]

brindan mayor detalle sobre el comportamiento de la red y de sus participantes, pero lamentablemente no se ha encontrado un estudio que permitiera obtener todos los parámetros necesarios en el modelo *CCP*.
Escapa al alcance de este trabajo implementar una metodología para obtener los parámetros del problema a partir de una red real, por tanto se utiliza la bibliografía disponible para realizar la parametrización.

Instancias de *CCP*

Los parámetros necesarios para definir una instancia del *CCP* se encuentran listados en la Tabla 8.

Parámetros
T : tiempo de observación
C : cantidad contenidos
$\overline{f}$: tasa de solicitud promedio de un contenido cualquiera
$f_{\max}$: tasa de solicitud máxima de un contenido cualquiera
$\overline{\lambda}$: tasa de comienzo de alojamiento promedio de un contenido cualquiera
$\overline{\mu}$: tasa de validez (desalojamiento) promedio de un contenido cualquiera
$\left(\frac{\lambda}{\mu}\right)_{\max}$: límite en cantidad de respuestas de un contenido
α_S : tamaño de la respuesta a una solicitud de contenido
α_B : tamaño de la respuesta a una búsqueda en el backbone
β_S : tamaño de un paquete de solicitud
β_B : tamaño de un paquete de búsqueda en el backbone
BW_{IN} : ancho de banda entrante
BW_{OUT} : ancho de banda saliente

Tabla 8. Parámetros del CCP.

El tiempo de observación T, es simplemente un parámetro de escalamiento. A los efectos de no trabajar con números muy pequeños se utiliza una hora como parámetro T.

La cantidad de contenidos C, en una red Superpeer varía en el corto plazo debido a la conexión y desconexión de sus usuarios y en el largo plazo de acuerdo a la popularidad de la red en particular. Según el estudio de Chu J., "Availability and locality measurements of peer-to-peer file systems." [36], la red Gnutella tuvo un promedio de 878691 archivos únicos alojados durante el período de estudio, se utiliza ese valor para el parámetro C.

El tamaño de los paquetes en la red se expresan con los parámetros α_S, α_B, β_S y β_B. Yang B., en “Designing a Super-Peer Network” [242], documenta el tamaño promedio de dichos paquetes nuevamente para la red Gnutella, siendo una consulta realizada por un cliente en promedio de 94 bytes (82 bytes de encabezados y 12

bytes promedio en el string de la consulta) y una respuesta a dicha consulta de cliente de 80 bytes+28*address+76*results (address =direcciones que alojan algún resultado y results=resultados que responden a la consulta). De lo anterior se tiene directamente que β_S es 94 bytes y que α_S es aproximadamente 100 bytes.

En forma simplificada, la mayoría de las referencias bibliográficas modelan las redes Superpeers como una red pura (como Gnutella) entre los nodos superpeers y una red híbrida (como Napster) entre cada superpeer y sus clientes. Por tanto una consulta realizada por un cliente deriva en una o varias consultas a realizar por el nodo superpeer en el backbone puro. También según [242], cada nodo en la red Gnutella tiene 3.1 vecinos en promedio a los cuales realiza cada consulta que recibe, por tanto en este modelo simplificado de comunicación para las redes Superpeers se tiene que la comunicación hacia el backbone es 3.1 veces la de los clientes (siempre que el nodo superpeer no almacene consultas anteriores), concluyendo que $\beta_B = 3.1*94 = 291.4$ bytes y $\alpha_B = 3.1*100 = 310$ bytes.

En [242] se documenta que la cantidad de consultas promedios realizadas por un cliente de Gnutella es $9.26*10^{-3}$ *qps*. Si suponemos que el nodo cache atiende a 1000 clientes promedio en una hora, entonces:
$\overline{f} = 1000 * 9.26 * 10^{-3} * 3600/878691.67 = 0.03794$.

Para poder generar las frecuencias de solicitud para cada contenido, a partir de $\overline{f}$, es necesario disponer de la distribución de probabilidad de las solicitudes según el contenido.

Los estudios más difundidos sobre solicitudes en redes de intercambio de archivos muestran formas de distribución del tipo *power-law* (como ser la distribución Zipf para el caso discreto y la distribución de Pareto para el continuo).

Estudios recientes, como los de Gummadi et.al. [102], muestran que en realidad las solicitudes en las redes de intercambio de archivos se apartan de las distribuciones del tipo *power-law* en los contenidos más frecuentes, existiendo una cota mucho menor a la descripta por dichas distribuciones debido a la cantidad finita de usuarios en la red y a la cualidad de que los archivos intercambiados solo son de interés para los usuarios una sola vez (al no renovarse un contenido este deja de tener interés para un usuario que ya lo obtuvo y nunca vuelve a consultar por el mismo contenido). A este efecto se le llama en la literatura *"fetch-at-most-once"*. Considerando que todas las consultas en el nodo cache son realizadas por exclusivamente 1000 usuarios y que dichos usuarios no repiten nunca una consulta, se tiene un límite practico para la cantidad de consultas a realizar por cada contenido $f_{\max} = 1000$.

Para generar la distribución de las frecuencias de solicitud f se muestrea una distribución de Pareto con parámetro exponencial 1.01. Para incluir el efecto *"fetch-at-most-once",* se considera una frecuencia máxima de solicitud de $f_{\max}$ para todo contenido, finalmente normalizando la muestra para obtener un promedio de consultas $\overline{f} = 0.03794$.

Saroiu S. en "A Measurement Study of Peer-to-Peer File Sharing Systems" [210] determina que para la red Gnutella el tiempo promedio que un usuario se encuentra en la red es de aproximadamente 1 hora, considerando que la desconexión del usuario es la principal causa de desalojamientos de contenidos se tiene que $\overline{\mu} = 1$ hora. Estudios actuales [102] muestran que los usuarios en las redes Superpeers han aprendido a conectarse y desconectarse mucho más frecuentemente, lo cual disminuiría la tasa de validez promedio.

Chu J. [36] determina en promedio la cantidad de réplicas de todos los contenidos en la red Gnutella en 11.098. Dicho valor en nuestro modelo es $\bar{\lambda}/\bar{\mu}$, conociendo $\bar{\mu}$, se tiene que $\bar{\lambda} = 11.098$.

En estas redes existe una dependencia directa entre la cantidad de alojamientos de un contenido y la cantidad de solicitudes del mismo. Esto se debe a que cuando un usuario solicita un contenido y lo obtiene mediante la red, este automáticamente es compartido a sus pares (esta es la esencia de la red de pares y bajo distintas modalidades siempre esta presente este mecanismo).
En la literatura se utiliza mayoritariamente una dependencia lineal entre las solicitudes y el alojamiento, existiendo excepciones como [149], donde se estudia además con una dependencia cuadrática. En este trabajo se ensayaron varias posibilidades, optando finalmente por una dependencia lineal. Por tanto a partir de las frecuencias f muestreadas anteriormente se determinan linealmente los λ, de forma tal que se tenga $\bar{\lambda}$ en promedio. Se considera $\mu = 1$ para todos los contenidos.
Una restricción adicional que se suma a la generación de los λ, es no superar una cota máxima: $\left(\frac{\lambda}{\mu}\right)_{\max} = 200$. Esto no debe interpretarse como una capacidad máxima de alojamientos para un contenido, sino una limitante práctica de todas las redes de pares en no responder con más de $\left(\frac{\lambda}{\mu}\right)_{\max}$ resultados por cualquier consulta. Debe entenderse que dada la distribución *power-law* de frecuencias f, es posible tener contenidos que superen fácilmente los cientos de miles de alojamientos, no tiene sentido responder a toda nueva consulta por ese contenido con todos los usuarios que lo alojan debido a que al cliente solo le basta con conocer a unos pocos cientos.

Finalmente el ancho de banda de un agregador se limita por los parámetros BW_{IN} y BW_{OUT}. En [210] se realiza un estudio de la distribución de ancho de banda en los clientes de la red Gnutella, sin embargo el crecimiento de oferta en servicios broadband por parte de los ISPs y las características particulares de los clientes que se convierten en superpeers hace presumir que las medidas a utilizar para el ancho de banda son mayores a los promedios de estos estudios. Se toma como valor de referencia una conexión de tecnología ADSL de 2/1 *Mbps*, por tanto expresado en bytes/hora se tiene que $BW_{IN} = 2048 * 1000/8 * 3600 = 921600000$ bytes/hora y $BW_{OUT} = 1024 * 1000/8 * 3600 = 460800000$ bytes/hora.

La Tabla 9 resume los valores de los parámetros utilizados para la creación de las instancias del *CCP*. Y la Tabla 10 muestra parcialmente una instancia generada en lenguaje AMPL[7] (lenguaje algebraico de descripción de problemas de programación matemática, ver la discusión en la subsección 5.1.3).

		Instancia
Parámetros	T : tiempo de observación	1 hora
	C : cantidad contenidos	878691
	$\overline{f}$: tasa de solicitud promedio de un contenido cualquiera	0.037938251 hora^{-1}
	$f_{\max}$: tasa de solicitud máxima de un contenido cualquiera	1000 hora^{-1}
	$\overline{\lambda}$: tasa de comienzo de alojamiento promedio de un contenido cualquiera	11.09749966 hora^{-1}
	$\overline{\mu}$: tasa de validez (desalojamiento) promedio de un contenido cualquiera	1 hora^{-1}
	$\left(\frac{\lambda}{\mu}\right)_{\max}$: límite en cantidad de respuestas de un contenido	200
	α_S : tamaño de la respuesta a una solicitud de contenido	100 bytes
	α_B : tamaño de la respuesta a una búsqueda en el backbone	310 bytes
	β_S : tamaño de un paquete de solicitud	94 bytes
	β_B : tamaño de un paquete de búsqueda en el backbone	291.4 bytes
	BW_{IN} : ancho de banda entrante	921600000 bytes/hora
	BW_{OUT} : ancho de banda saliente	460800000 bytes/hora

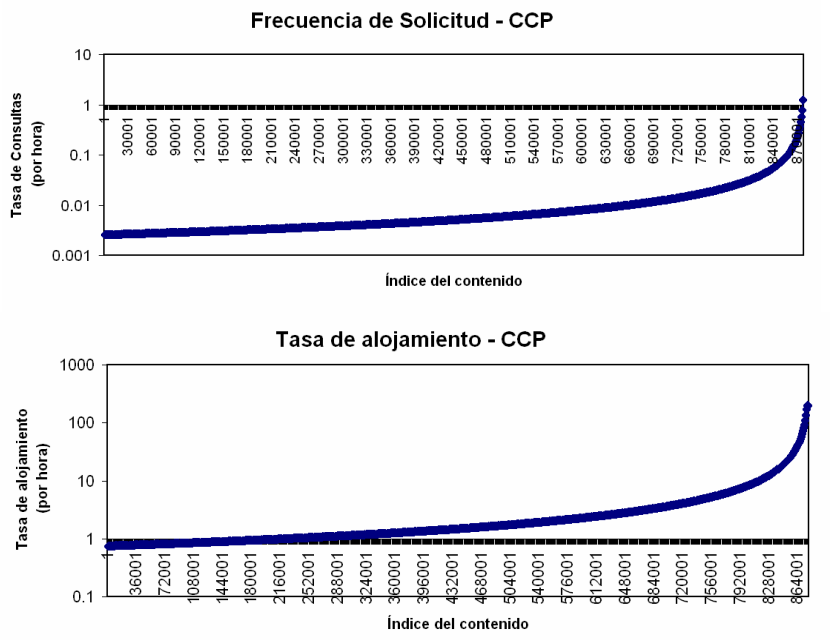
Tabla 9. Valores de los parámetros utilizados para la creación de las instancias del CCP.

```
param C        := 878691;

param alphaS   := 100.00000000;
param alphaB   := 310.00000000;
param betaS    := 94.00000000;
param betaB    := 291.40000000;
param BWin     := 921600000.00000000;
param BWout    := 460800000.00000000;

param f :=
         1 0.00215837
         2 0.00215837
         3 0.00215838
         4 0.00215838
         5 0.00215839
         6 0.00215839
         7 0.00215839
         8 0.00215839
         … …
         … …
         … …
    878683 202.55057419
    878684 386.46926398
    878685 491.59647801
    878686 515.85004203
    878687 525.51414572
    878688 1000.00000000
    878689 1000.00000000
    878690 1000.00000000
    878691 1000.00000000;
```

```
param lamda :=
         1 0.63135378
         2 0.63135619
         3 0.63135855
         4 0.63135912
         5 0.63136008
         6 0.63136040
         7 0.63136128
         8 0.63136199
         9 0.63136266
         … …
         … …
         … …
    878682 200.00000000
    878683 200.00000000
    878684 200.00000000
    878685 200.00000000
    878686 200.00000000
    878687 200.00000000
    878688 200.00000000
    878689 200.00000000
    878690 200.00000000
    878691 200.00000000;

param mu :=
         1 1.00000000
         2 1.00000000
         3 1.00000000
         4 1.00000000
         5 1.00000000
         … …
         … …
         … …
    878686 1.00000000
    878687 1.00000000
    878688 1.00000000
    878689 1.00000000
    878690 1.00000000
    878691 1.00000000;
```

Tabla 10. Instancia del CCP en AMPL. ccp-p2p.dat

La Ilustración 18 muestra de forma gráfica la distribución obtenida para las principales variables del *CCP* en el caso de estudio P2P.



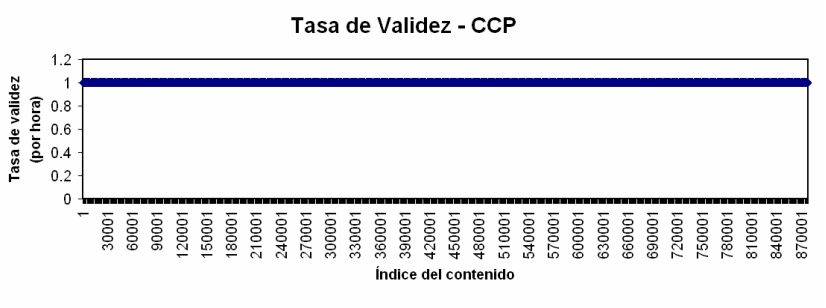


Ilustración 18. Distribución del caso de estudio P2P.

Con estos parámetros se crearon diez instancias del problema *CCP*, utilizando una semilla aleatoria distinta en cada caso.

Instancias de *CCCP*

Resolver numéricamente las instancias del *CCP*, considerando que tienen cerca de un millón de contenidos (lo que implica misma cantidad de variables de decisión), parece al menos una tarea difícil y posiblemente de resultados poco prácticos. En su lugar la estrategia de resolución es obtener a partir de una instancia del *CCP*, una instancia del *CCCP* equivalente; reduciendo la cantidad de variables del millón a la cantidad de clases elegidas.

Surgen al menos dos interrogantes en este método de solución: ¿cuántas clases son suficientes y necesarias para representar el problema original? Y ¿qué método de agrupamiento en clases se debe elegir para consolidar los contenidos?
Lógicamente ambas preguntas son interdependientes y por tanto se decide aplicar un método de consolidación y comparar resultados para varios tamaños de clases.

Para resolver el problema de la consolidación del contenido en clases se ensayaron algunos métodos de conglomerado (*clustering*), por ejemplo *k-Means* [150]. Entre las opciones se eligió por razones de eficiencia y prestaciones el método descrito en el apéndice A . Básicamente se trata de un método que garantiza un equilibrio entre la cantidad de solicitudes realizadas por cada clase (las solicitudes de una clase son la suma de las solicitudes de los contenidos que la integran), incluyendo en cada clase aquellos contenidos con mayor similitud en la frecuencia de solicitud.

El mecanismo encontrado para hallar las instancias del *CCP* impone una dependencia lineal entre λ y f, además de que $\mu = 1$ para todos los contenidos. Por tanto la única dimensión que presenta información para el método de conglomerado es la tasa de solicitud f, a esto se debe principalmente la aplicación eficaz del método.

Los tamaños de clases utilizados en este estudio son: **2**, **8**, **16**, **32** y **128**.
Partiendo de diez instancias del problema *CCP* y cinco clases posibles, se tienen cincuenta instancias del problema *CCCP* (diez instancias por cada clase).
El apéndice B incluye cinco instancias del problema *CCCP*, una instancia para cada tamaño de clase generadas a partir de un mismo *CCP*.

A continuación se muestran dos instancias del *CCCP* como ejemplo, la Tabla 11 una instancia de 2 clases y la Tabla 12 una instancia de 16 clases.

```
param K        := 2;

param alphaS   := 100.00000000;
param alphaB   := 300.00000000;
param betaS    := 94.00000000;
param betaB    := 291.40000000;
param BWin     := 921600000.00000000;
param BWout    := 460800000.00000000;

param f :=
         1 22.53685618
         2 0.01513864;
param lamda :=
         1 200.00000000
         2 4.02619446;
param mu :=
         1 1.00000000
         2 1.00000000;
param l :=
         1 590.00000000
         2 878101.00000000;
```

Tabla 11. Instancia del CCCP (2 clases) en AMPL. cccp-p2p-K002.dat

```
param K        := 16;

param alphaS   := 100.00000000;
param alphaB   := 300.00000000;
param betaS    := 94.00000000;
param betaB    := 291.40000000;
param BWin     := 921600000.00000000;
param BWout    := 460800000.00000000;

param f :=
         1 1000.00000000
         2 1000.00000000
         3 479.85748244
         4 108.05207506
         5 45.69682011
         6 20.90945385
         7 7.99243620
         8 3.11083107
         9 1.27095518
        10 0.52349239
        11 0.21990832
        12 0.09345023
        13 0.03957382
        14 0.01693693
        15 0.00731336
        16 0.00317740;

param lamda :=
         1 200.00000000
         2 200.00000000
         3 200.00000000
         4 200.00000000
         5 200.00000000
         6 200.00000000
         7 200.00000000
         8 200.00000000
         9 200.00000000
        10 149.44596626
        11 64.32643638
        12 27.33557540
        13 11.57592815
        14 4.95430221
        15 2.13926733
        16 0.92943700;
param mu :=
         1 1.00000000
         2 1.00000000
         3 1.00000000
         4 1.00000000
         5 1.00000000
```

```
         6 1.00000000
         7 1.00000000
         8 1.00000000
         9 1.00000000
        10 1.00000000
        11 1.00000000
        12 1.00000000
        13 1.00000000
        14 1.00000000
        15 1.00000000
        16 1.00000000;
param l :=
         1 2.00000000
         2 2.00000000
         3 4.00000000
         4 15.00000000
         5 35.00000000
         6 76.00000000
         7 199.00000000
         8 510.00000000
         9 1248.00000000
        10 3029.00000000
        11 7210.00000000
        12 16965.00000000
        13 40062.00000000
        14 93605.00000000
        15 216778.00000000
        16 498951.00000000;
```

Tabla 12. Instancia del CCCP (16 clases) en AMPL. cccp- p2p-K016.dat

5.1.3 Método de Resolución

Para expresar el problema y resolver el caso de prueba planteado se utiliza el lenguaje de AMPL[7]. AMPL es un paquete con un lenguaje de modelado algebraico para programación matemática, capaz de modelar una variedad importante de problemas de optimización, incluyendo problemas no lineales de variables continuas como el *CCCP*.

El lenguaje fue elegido por su poder de expresividad, lo sencillo que resulta comprender un modelo expresado en el mismo (dada su correspondencia directa con una formulación matemática clásica), y por la existencia de una cantidad muy importante de bibliotecas de optimización que aceptan como entrada descripciones en este lenguaje. Una buena introducción a AMPL. puede encontrarse en [78].

En su versión estudiantil (la utilizada en este estudio: "AMPL Student Version 20050208"), el AMPL permite utilizar varios paquetes de resolución, el paquete elegido en este caso es el MINOS[224] (versión 5.5), una muy conocida y bien probada biblioteca de optimización.

La Tabla 13 y la Tabla 14 expresan los modelos *CCP* y *CCCP* en el lenguaje AMPL respectivamente.

```
param C >=0, integer;  # cantidad de clases de contenido
set CONTENT = {1..C};  # conjunto de contenidos

param f {k in CONTENT};     #solicitudK/hora
param lamda {k in CONTENT}; #comienzoAlojamientoK/hora;
param mu {k in CONTENT};    #duracionAlojamientoK/hora;

param alphaS       >=0;     # bytes/solicitud
param alphaB       >=0;     # bytes/busquqeda
param betaS        >=0;     # bytes/respuestaSolicitud
param betaB        >=0;     # bytes/respuestaBusqueda
param BWin         >=0;     # bytes por hora
param BWout        >=0;     # bytes por hora

# variable de decisión y restricción de positividad, d=tiempo en cache
```

```
var d {k in CONTENT} >=0.0001 default 0.000001;

#función objetivo, maximizar eficacia
maximize epsilon:
         (sum {k in CONTENT} lamda[k]/mu[k]/mu[k]/d[k]*(
                  mu[k]*(1-exp(-f[k]*d[k])) +
                  f[k]*(1-exp(-mu[k]*d[k])) -
                  1/d[k]*(1-exp(-f[k]*d[k]))*(1-exp(-mu[k]*d[k]))
         ))/(sum {k in CONTENT} lamda[k]/mu[k]*f[k]);

#restricciones, limitación en ancho de banda
subject to bitsIn :
         0 <= betaS*(sum {k in CONTENT} f[k]) +
              alphaB*(sum {k in CONTENT} lamda[k]/mu[k]*f[k]/(1+d[k]*f[k]))
           <= BWin;

subject to bitsOut:
         0 <= alphaS*(sum {k in CONTENT} lamda[k]/mu[k]*f[k]) +
              betaB*(sum {k in CONTENT} f[k]/(1+d[k]*f[k]))
           <= BWout;
```

Tabla 13. Modelo del CCP en AMPL. ccp.mod

```
param K >=0, integer;  # cantidad de clases de contenido
set CLASS = {1..K};  # conjunto de clases de contenidos
param f {k in CLASS};                #solicitudK/hora
param lamda {k in CLASS};   #comienzoAlojamientoK/hora;
param mu {k in CLASS};      #duracionAlojamientoK/hora;
param l {k in CLASS};       #cantidad de contenidos en la clase;

param alphaS       >=0;      # bytes/solicitud
param alphaB       >=0;      # bytes/busquqeda
param betaS        >=0;      # bytes/respuestaSolicitud
param betaB        >=0;      # bytes/respuestaBusqueda
param BWin         >=0;      # bytes
param BWout        >=0;      # bytes

# variable de decisión y restricción de positividad, d=tiempo en cache
var d {k in CLASS} >=0.000001 default 0.000001;

#función objetivo, maximizar eficacia
maximize epsilon:
         (sum {k in CLASS} l[k]*lamda[k]/mu[k]/mu[k]/d[k]*(
                  mu[k]*(1-exp(-f[k]*d[k])) +
                  f[k]*(1-exp(-mu[k]*d[k])) -
                  1/d[k]*(1-exp(-f[k]*d[k]))*(1-exp(-mu[k]*d[k]))
                                              )
         )/(sum {k in CLASS} l[k]*lamda[k]/mu[k]*f[k]);

#restricciones, limitación en ancho de banda
subject to bitsIn :
   0 <= betaS*(sum {k in CLASS} l[k]*f[k]) +
        alphaB*(sum {k in CLASS} l[k]*lamda[k]/mu[k]*f[k]/(1+d[k]*f[k]))
     <= BWin;

subject to bitsOut:
   0 <= alphaS*(sum {k in CLASS} l[k]*lamda[k]/mu[k]*f[k]) +
        betaB*(sum {k in CLASS} l[k]*f[k]/(1+d[k]*f[k]))
     <= BWout;
```

Tabla 14. Modelo del CCCP en AMPL. cccp.mod

Al ejecutar el paquete MINOS desde el entorno AMPL, es posible invocarlo con directivas específicas que parametrizan el método de resolución. Se usan los valores por defecto para las directivas no configuradas. La Tabla 15 muestra la secuencia de comandos utilizada en AMPL para resolver la instancia cccp.dat, todas las instancias resueltas en este capítulo se ejecutaron de esta forma.

Por un detalle sobre las directivas de invocación para MINOS desde AMPL puede revisarse [8] o [224].

Es importante tener en cuenta que por las características del problema, no es posible garantizar que MINOS proveerá lo solución globalmente óptima del problema, sino que converge a un óptimo local (de buena calidad).

```
option ampl_include '.';

option solver minos;
option minos options 'crash option=0 \
        feasibility tolerance=1.0e-8 scale=no \
        summary_file=6 summary_frequency=5 \
        timing= 1';

model cccp.mod;
data cccp.dat;
solve;
display epsilon;
display bitsIn.lb, bitsIn.ub, bitsIn.body, bitsIn.slack;
display bitsOut.lb, bitsOut.ub, bitsOut.body, bitsOut.slack;
display d;
expand bitsIn, bitsOut;
```

Tabla 15. Comandos de AMPL para resolver la instancia cccp.dat.

5.1.4 Resultados

A partir de los mismos parámetros (y distinta semilla de números aleatorios) se obtuvieron 10 instancias del problema *CCP*. Para cada una de ellas se crearon 5 instancias del problema *CCCP* con distintas cantidades de clases. La cantidad de clases elegidas fueron: 2, 8, 16, 32 y 128.

La resolución de estas 50 instancias del problema *CCCP* no requieren gran capacidad de cómputo y todas se ejecutaron en un computador PIII 800 Mhz, con 320 Mb de RAM.

La Tabla 16 muestra los resultados promedios para las instancias generadas según cada cantidad de clases del *CCCP*.
Cada resultado es el promedio de diez ejecuciones del problema (instancias generadas según distintas semillas aleatorias).
En todos los casos la restricción que limita la solución es el ancho de banda entrante BW_{IN}.

		Resultados			
		ε Función Objetivo	Tiempo Ejecución (s)	BW_{IN} consumido (bytes/hora)	BW_{OUT} consumido (bytes/hora)
Cantidad de clases (*K*)	2	0.99888275	0.00625	921599933	334157325
	8	0.98845330	0.07400	921599993	406282138
	16	0.98759660	0.25800	921599993	413490851
	32	0.99659690	0.21100	921599993	415901600
	128	0.99924110	0.44900	921600793	416588930

Tabla 16. Resultados numéricos para las instancias del CCCP. Cada resultado es el promedio de 10 instancias.

Puede observarse como el tiempo de ejecución crece con el tamaño del problema, es decir la cantidad de clases.

También puede observarse como existe un error de cálculo numérico en las instancias de 128 clases, provocando una superación de la cota del ancho de banda BW_{IN} (restricción: $BW_{IN} = 921600000$ bytes/hora).

Los valores de la función objetivo ε no se pueden comparar entre problemas de distinta cantidad de clases debido a que en realidad se trata de problemas distintos. A esto se debe por ejemplo que los problemas con 2 clases tengan en promedio una función objetivo de mayor valor que los problemas de 8 clases.
Para realizar una comparación justa entre las soluciones de problemas *CCCP* es necesario convertirlos al problema original *CCP* y comparar la función objetivo del *CCP.*

Una solución del *CCCP* determina para cada clase un tiempo de expiración. El mecanismo para convertir una solución del *CCCP* a una del *CCP* es muy sencillo: conocido el tiempo de expiración de una clase, se le asigna a cada contenido del *CCP* el tiempo de expiración de la clase a la que pertenecen (la asignación de contenidos a clases se realizó mediante el método de conglomerado descrito con anterioridad, apéndice A).
La Tabla 17 muestra los resultados promedio para las instancias del problema *CCCP* y su equivalente *CCP*.

		Resultados					
		CCCP			*CCP* equivalente		
		ε Función Objetivo *CCCP*	BW_{IN} consumido (bytes/hora) *CCCP*	BW_{OUT} consumido (bytes/hora) *CCCP*	ε Función Objetivo *CCP*	BW_{IN} consumido (bytes/hora) *CCP*	BW_{OUT} consumido (bytes/hora) *CCP*
Cantidad de clases (*K*)	2	**0.99888275**	921599933	334157325	**0.86598051**	699726354	425595159
	8	**0.98845330**	921599993	406282138	**0.98248558**	897575607	415658051
	16	**0.98759660**	921599993	413490851	**0.98621038**	915414791	416016277
	32	**0.99659690**	921599993	415901600	**0.99653966**	920955186	416531221
	128	**0.99924110**	921600793	416588930	**0.99924080**	921627767	416619980

Tabla 17. Resultados numéricos para las instancias del CCCP. Cada solución del CCCP se convierte en una solución del CCP y se evalúa. Los resultados son el promedio de 10 instancias.

Puede observarse como a medida que crece la cantidad de clases se alcanzan mejores soluciones del problema *CCP* (ver ε del *CCP* en la Tabla 17). Esto se debe a una utilización más eficiente del ancho de banda entrante (BW_{IN}) que es la restricción alcanzada en las instancias analizadas.

Resolver instancias del *CCP* mediante la solución de un problema *CCCP* aumenta la sensibilidad del método a errores de cálculo numérico, puede observarse como la cota de BW_{IN} es claramente superada en el *CCP* para las instancias de 128 clases (ver Tabla 17, restricción: $BW_{IN} = 921600000$ bytes/hora).

Para instancias de muy pocas clases se hace evidente el error cometido en la simplificación del problema *CCP* a un problema *CCCP*. Por ejemplo, cuando se tienen solo dos clases la función objetivo del *CCCP* $\varepsilon_{CCCP} = 0.99888275$ es notoriamente superior a la del *CCP* $\varepsilon_{CCP} = 0.86598051$.

La Tabla 18 muestra la discrepancia promedio en la simplificación cometido para las distintas opciones de cantidad de clases (*K*). Parece claro que solo dos clases (*K=2*) no son suficientes para modelar el problema, cometiéndose una discrepancia relativa de 15% aprox. Además, con algunas pocas clases se alcanza muy buenas estimaciones, con discrepancias promedio menores al 1%.

		Análisis de Resultados			
		ε Función Objetivo *CCCP*	ε Función Objetivo *CCP equivalente*	Discrepancia absoluta $\lvert\varepsilon_{CCP} - \varepsilon_{CCCP}\rvert$	Discrepancia relativa $\frac{\lvert\varepsilon_{CCP} - \varepsilon_{CCCP}\rvert}{\varepsilon_{CCP}} * 100$
Cantidad de clases (*K*)	2	0.99888275	0.86598051	**0.132902181**	**15.4390%**
	8	0.98845330	0.98248558	**0.005967569**	**0.6085%**
	16	0.98759660	0.98621038	**0.001386349**	**0.1408%**
	32	0.99659690	0.99653966	**5.71716E-05**	**0.0057%**
	128	0.99924110	0.99924080	**3.40467E-07**	**3.408E-05%**

Tabla 18. Análisis de resultados numéricos para las instancias del CCCP y su CCP equivalente. Los resultados son el promedio de 10 instancias[13].

Parece alcanzarse una precisión muy buena si se utilizan 32 clases o más. A su vez los tiempos de cómputo son relativamente bajos para las distintas cantidades de clases estudiadas (para *K=128* la ejecución dura menos de medio segundo). Por tanto entre las opciones estudiadas parece adecuado elegir entre 32 o 128 clases.

Instancia Ejemplo de 16 clases

A continuación se analiza una instancia en particular del *CCCP*. La instancia cuenta con 16 clases de contenidos, como puede verse en la Ilustración 19, los contenidos de menor índice son los más solicitados y a su vez los más cambiantes respecto a sus fuentes de alojamiento. Por otro lado las clases con menor tasa de solicitud y alojamiento incluyen más contenidos (es típico que haya poco contenidos populares y muchos nada populares).

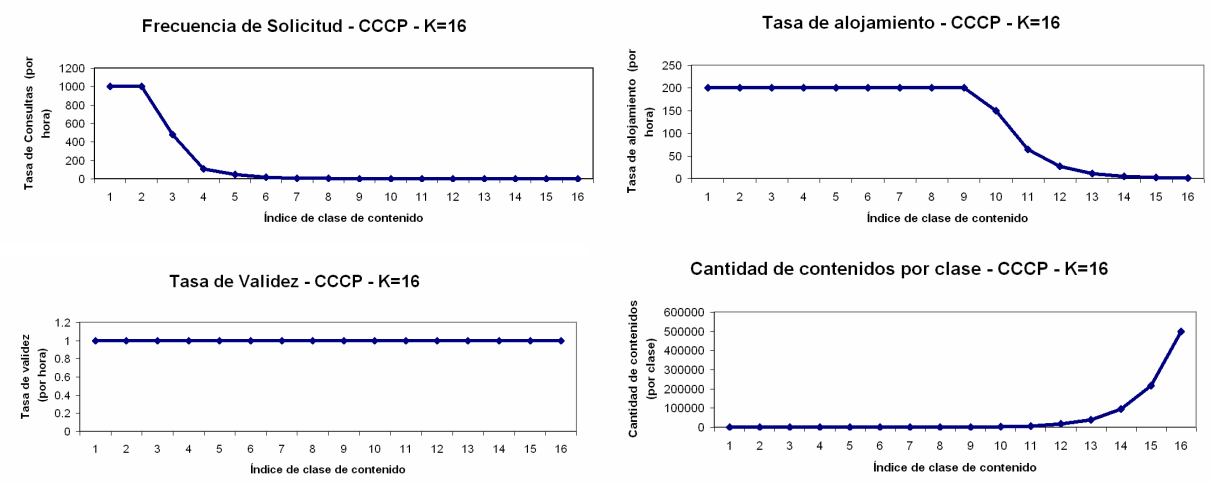


Ilustración 19. CCCP de 16 clases para el caso de estudio P2P.

[13] Para todas las instancias ejecutadas sucede que la función objetivo del CCCP es mayor a la del CCP equivalente. Por tanto la diferencia entre los promedios de las funciones objetivo es igual al promedio de la diferencia absoluta entre ellas.

La Tabla 12 muestra la instancia en detalle.

A priori es difícil decidir exclusivamente con esta información que política de alojamiento en un nodo agregador maximiza las respuestas correctas. Dos criterios intuitivos y contrapuestos pueden ser aplicados:

- alojar los contenidos más solicitados, o
- alojar los contenidos cuyas fuentes cambien lo menos posible.

La solución óptima es un compromiso entre estos criterios y la mayoría de las veces no es intuitiva.

Resultado

Siguiendo el método de resolución planteado en la sección anterior, se tiene los tiempos de expiración que maximizan la entrega de fuentes de contenido válidas a los solicitantes.

```
Time = 0.33 seg
epsilon = 0.996656
bitsIn = 921600460
bitsOut = 357952740

solution d :=
         1 0.00018695
         2 0.00018695
         3 0.00038830
         4 0.00172036
         5 0.00406624
         6 0.00888520
         7 0.02324310
         8 0.05971460
         9 0.14615600
        10 0.35482900
        11 0.84458700
        12 1.98716000
        13 0.63063300
        14 1.47309000
        15 3.41242000
        16 7.85214000;
```

Tabla 19. Solución de Instancia del CCCP (16 clases) en AMPL. cccp- p2p-K016.dat

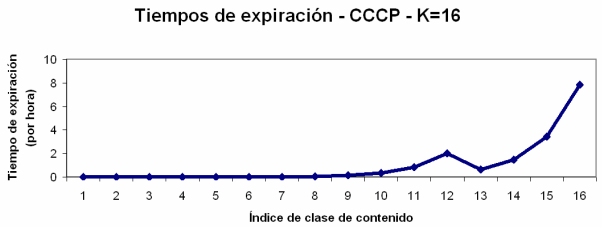


Ilustración 20. Solución del CCCP de 16 clases para el caso de estudio P2P.

La Tabla 19 y la Ilustración 20 muestran los resultados; puede observarse como la solución no mantiene un patrón, y la relación entre tasa de solicitud y alojamiento provoca la conveniencia de un alojamiento mayor de la clase de contenido 12.

5.1.5 Conclusiones

Se resumen algunas conclusiones extraídas de la aplicación de los problemas *CCP* y *CCCP* a una red de contenido P2P para compartir archivos.

Elección de la Cantidad de Clases

Se espera que cada red de contenido tenga perfiles distintos en la distribución de solicitudes y alojamientos.
Desde el punto de vista del modelo, esto determina la similitud entre los contenidos, a su vez determinando la cantidad de clases mínimas necesarias para expresar el contenido.
Por tanto, dependiendo de las características particulares del contenido de la red y de la precisión deseada es la elección de la cantidad de clases mínimas necesarias.

Por otro lado, utilizando más clases se mejora en la precisión del modelo, aumentando el tiempo de cómputo.
Esto seguramente no es relevante si el método se utiliza para el diseño inicial de la política de cache en un nodo agregador, pero es de especial importancia si se espera utilizar el método para la toma de decisiones en tiempo real en un nodo agregador.

Concluyendo que para cada red de contenido (que determina un perfil de contenido) un compromiso entre precisión y tiempo de cómputo determina la cantidad de clases a utilizar en el problema *CCCP*.

Para el caso de estudio P2P parece adecuado elegir entre 32 o 128 clases. Si se utilizan 32 clases o más se alcanza una precisión muy buena, mientras que los tiempos de cómputo son relativamente bajos para las distintas cantidades de clases estudiadas (para *K=128* la ejecución dura menos de medio segundo).

Otros Métodos de Resolución

Además de MINOS[224], se probaron otros algoritmos de resolución.
Por ejemplo, la Tabla 38 muestra los resultados de otros paquetes para el problema de ejemplo presentado en la sección anterior (ver Tabla 12). En este caso IPOPT[118] y SNORT[224] encuentran una mejor solución que el resto de los algoritmos (entre ellos MINOS).

		Resultados	
		ε Función Objetivo	Tiempo Ejecución (s)
Paquete de Resolución	MINOS	**0.99610320**[14]	0.370
	IPOPT	**0.99995470**	0.060
	KNITRO	**0.99995310**	0.410
	LANCELOT	**0.99975880**	0.130
	PENNON	**0.99955790**	0.120
	SNOPT	**0.99995470**	0.030
Máximo		**0.99995470**	

Tabla 20. Resultados numéricos de varios algoritmos para la instancia de ejemplo del CCCP de 16 clases.

IPOPT[118], KNITRO[246], LANCELOT[138], PENNON[195] y SNOPT[224] son ofrecidos para su uso desde Internet, mediante el portal NEOS[170] (este portal es una interesante "prueba de concepto" del funcionamiento de un servidor de optimización, que brinda el servicio a usuarios remotos de resolver problemas empleando localmente una batería de bibliotecas, muchas de ellas comerciales,

[14] El resultado de la ejecución de MINOS en el portal NEOS difiere ínfimamente de los obtenidos con la ejecución local.

empleando una interfaz común y lenguajes de descripción estandar, entre ellos el del propio AMPL.). La utilización de los paquetes de resolución se presenta como "caja negra" puesto que no es objeto de estudio en este trabajo. La determinación del método más adecuado para la resolución de los problemas *CCCP* y *CCP*, o el diseño de un método nuevo, escapa al alcance de este trabajo. Mediante la especificación del problema en el lenguaje estándar AMPL se permite a investigadores futuros explorar con facilidad sobre otros mecanismos de resolución.

Otras Políticas

Para comprobar la utilidad del método *CCCP* se compara la performance de las soluciones halladas con las que se obtendrían utilizando otras políticas de alojamiento en el cache.
Además de la política basada en el método *CCCP* se analizan otros dos algoritmos sencillos:

- ***Tiempo Único de Expiración***: La opción más simple para resolver el problema debe ser asignarle a todos los contenidos el mismo tiempo de expiración. Este método funciona de la siguiente forma: en el nodo agregador se define en etapa de diseño un tiempo de expiración fijo d . Todo contenido que es consultado al backbone es almacenado en el cache estrictamente por este tiempo y luego borrado.
- ***Tiempo de Expiración Proporcional a las Solicitudes***: Al obtenerse los parámetros del problema *CCP*, en la sección 5.1.2, se asumió que la única dimensión que presenta información para el método de conglomerado es la tasa de solicitud f . Puede entonces suponerse que asumir tiempos de expiración directamente proporcionales a la popularidad de un contenido es una buena política de alojamiento en el cache. Este método funciona de la siguiente forma: en el nodo agregador se define en etapa de diseño un coeficiente único e tal que el tiempo de expiración se determine linealmente según la tasa de solicitud del contenido $d = e * f$. Todo contenido que es consultado al backbone es almacenado en el cache estrictamente por este tiempo ($d = e * f$) y luego borrado.

Para especificar buenos parámetros d y e de ambas políticas se realizan modelos en el lenguaje AMPL y se resuelven de forma similar al problema *CCCP.* La Tabla 21 muestra el modelo utilizado para determinar el parámetro de la política "Tiempo Único de Expiración" y la Tabla 22 para la política Tiempo de Expiración Proporcional a las Solicitudes".

```
param K >=0, integer;        # cantidad de clases de contenido
set CLASS = {1..K};          # conjunto de clases de contenidos
param f {k in CLASS};        #solicitudK/hora
param lamda {k in CLASS};    #comienzoAlojamientoK/hora;
param mu {k in CLASS};       #duracionAlojamientoK/hora;
param l {k in CLASS};        #cantidad de contenidos en la clase;

param alphaS       >=0;      # bytes/solicitud
param alphaB       >=0;      # bytes/busquqeda
param betaS        >=0;      # bytes/respuestaSolicitud
param betaB        >=0;      # bytes/respuestaBusqueda
param BWin         >=0;      # bytes
param BWout        >=0;      # bytes

# variable de decisión y restricción de positividad, d=tiempo en cache
var d >=0.000001 default 0.000001;

#función objetivo, maximizar eficacia
maximize epsilon:
         (sum {k in CLASS} l[k]*lamda[k]/mu[k]/mu[k]/d*(
                  mu[k]*(1-exp(-f[k]*d)) +
```

```
                  f[k]*(1-exp(-mu[k]*d)) -
                  1/d*(1-exp(-f[k]*d))*(1-exp(-mu[k]*d))
                  )
         )/(sum {k in CLASS} l[k]*lamda[k]/mu[k]*f[k]);

#restricciones, limitación en ancho de banda
subject to bitsIn :
   0 <= betaS*(sum {k in CLASS} l[k]*f[k]) + alphaB*(sum {k in CLASS}
l[k]*lamda[k]/mu[k]*f[k]/(1+d*f[k])) <= BWin;

subject to bitsOut:
  0 <= alphaS*(sum {k in CLASS} l[k]*lamda[k]/mu[k]*f[k]) + betaB*(sum {k in
CLASS} l[k]*f[k]/(1+d*f[k])) <= BWout;
```

Tabla 21. Modelo de la política: Tiempo Único de Expiración.

```
param K >=0, integer;        # cantidad de clases de contenido
set CLASS = {1..K};          # conjunto de clases de contenidos
param f {k in CLASS};        #solicitudK/hora
param lamda {k in CLASS};    #comienzoAlojamientoK/hora;
param mu {k in CLASS};       #duracionAlojamientoK/hora;
param l {k in CLASS};        #cantidad de contenidos en la clase;

param alphaS       >=0;      # bytes/solicitud
param alphaB       >=0;      # bytes/busquqeda
param betaS        >=0;      # bytes/respuestaSolicitud
param betaB        >=0;      # bytes/respuestaBusqueda
param BWin         >=0;      # bytes
param BWout        >=0;      # bytes

# variable de decisión y restricción de positividad, d=tiempo en cache
var e >=0.000001 default 0.000001;

#función objetivo, maximizar eficacia
maximize epsilon:
         (sum {k in CLASS} l[k]*lamda[k]/mu[k]/mu[k]/(e*f[k])*(
                  mu[k]*(1-exp(-f[k]*e*f[k])) +
                  f[k]*(1-exp(-mu[k]*e*f[k])) -
                  1/(e*f[k])*(1-exp(-f[k]*e*f[k]))*(1-exp(-mu[k]*e*f[k]))
                  )
         )/(sum {k in CLASS} l[k]*lamda[k]/mu[k]*f[k]);

#restricciones, limitación en ancho de banda
subject to bitsIn :
   0 <= betaS*(sum {k in CLASS} l[k]*f[k]) + alphaB*(sum {k in CLASS}
l[k]*lamda[k]/mu[k]*f[k]/(1+e*f[k]*f[k])) <= BWin;

subject to bitsOut:
  0 <= alphaS*(sum {k in CLASS} l[k]*lamda[k]/mu[k]*f[k]) + betaB*(sum {k in
CLASS} l[k]*f[k]/(1+e*f[k]*f[k])) <= BWout;
```

Tabla 22 Modelo de la política: Tiempo de expiración proporcional a las solicitudes.

Para la instancia de ejemplo que venimos tratando (ver Tabla 12) se resolvieron los modelos asociados a estas nuevas políticas, obtuviendo los parámetros d y e que maximizan el descubrimiento de contenido en la red. La Tabla 23 muestra para distintos métodos de resolución el parámetro a elegir en el caso de la política "Tiempo de expiración único" y la Tabla 24 hace lo propio con la política "Tiempo de expiración proporcional a las solicitudes".

		Resultados		
		Política - Tiempo de expiración único		
		ε Función Objetivo	Tiempo Ejecución (s)	Parámetro d (tiempo de expiración) (s)
Paquete de Resolución	MINOS	**0.99995210**	0.020	**9.03529000E-04**
	IPOPT	**0.99995210**	0.040	**9.03555000E-04**
	KNITRO	0.99995190	0.140	**9.06238000E-04**
	LANCELOT	**0.99995210**	0.010	**9.03529000E-04**
	PENNON	**0.99995210**	0.030	**9.03594000E-04**
	SNOPT	**0.99995210**	0.030	**9.03529000E-04**
Máximo		0.99995210		

Tabla 23. Parámetro de la política "Tiempo de expiración único".

		Resultados		
		Política - Tiempo de expiración proporcional a las solicitudes		
		ε Función Objetivo	Tiempo Ejecución (s)	Parámetro e (coeficiente de proporcionalidad) (s)
Paquete de Resolución	MINOS	**0.99993000**	0.020	**1.30179000E-06**
	IPOPT	0.99992800	0.040	**1.32331000E-06**
	KNITRO	0.99658890	0.050	**2.60189000E-05**
	LANCELOT	0.01368548	0.030	**4.37636000E+02**
	PENNON	**0.99993000**	0.030	**1.30191000E-06**
	SNOPT	**0.99993000**	0.030	**1.30179000E-06**
Máximo		0.99993000		

Tabla 24. Parámetro de la política "Tiempo de expiración proporcional a las solicitudes".

Ahora que las políticas se encuentran completamente especificadas, es importante comparar la performance de las tres políticas.
La Tabla 25 muestra los valores de la función objetivo y los tiempos de ejecución para determinar cada una de las tres políticas. Respecto a los tiempos puede observarse que la solución al problema *CCCP* requiere más cómputo que las políticas sencillas.
Para realizar una comparación justa de la performance no es posible observar los valores de la función objetivo en cada caso puesto que se estarían comparando valores de distintos modelos. En la Tabla 26 se convierte la solución de cada política al problema *CCP* equivalente, siendo estos valores justamente comparables.

Como es de esperar, la Tabla 26 muestra que la política que logró la mejor performance fue *CCCP*. Utilizando el método de solución SNOP esta política alcanzó la mejor solución conocida, con $\varepsilon = 0.99995405$. A su vez, es importante señalar que las otras políticas no tuvieron un desempeño muy alejado de esta mejor solución.

		Resultados					
		Política - CCCP		*Política - Tiempo de expiración único*		*Política - Tiempo de expiración proporcional a las solicitudes*	
		ε Función Objetivo	Tiempo Ejecución (s)	ε Función Objetivo	Tiempo Ejecución (s)	ε Función Objetivo	Tiempo Ejecución (s)
Paquete de Resolución	MINOS	**0.99610320**	0.520	**0.99995210**	0.020	**0.99993000**	0.020
	IPOPT	**0.99995470**	0.030	**0.99995210**	0.040	**0.99992800**	0.040
	KNITRO	**0.99995310**	0.640	**0.99995190**	0.140	**0.99658890**	0.050
	LANCELOT	**0.99975880**	0.180	**0.99995210**	0.010	**0.01368548**	0.030
	PENNON	**0.99955790**	0.130	**0.99995210**	0.030	**0.99993000**	0.030
	SNOPT	**0.99995470**	0.010	**0.99995210**	0.030	**0.99993000**	0.030
Promedio		0.99921373	0.252	0.99995207	0.045	0.83499873	0.033
Máximo		**0.99995470**	0.640	**0.99995210**	0.140	**0.99993000**	0.050

Tabla 25. Comparación del CCCP con otras dos políticas simples de alojamiento en cache.
Resultados numéricos de varios algoritmos para la instancia de ejemplo del CCCP de 16 clases.

		Análisis Resultados					
		Política - CCCP		*Política - Tiempo de expiración único*		*Política - Tiempo de expiración proporcional a las solicitudes*	
		ε Función Objetivo	ε Función Objetivo *CCP equivalente*	ε Función Objetivo	ε Función Objetivo *CCP equivalente*	ε Función Objetivo	ε Función Objetivo *CCP equivalente*
Paquete de Resolución	MINOS	0.99610320	**0.99553881**	0.99995210	**0.99995200**	0.99993000	**0.99993040**
	IPOPT	0.99995470	**0.99995404**	0.99995210	**0.99995200**	0.99992800	**0.99992847**
	KNITRO	0.99995310	**0.99995215**	0.99995190	**0.99995174**	0.99658890	**0.99661043**
	LANCELOT	0.99975880	**0.99974659**	0.99995210	**0.99995200**	0.01368548	**0.01393887**
	PENNON	0.99955790	**0.99953126**	0.99995210	**0.99995199**	0.99993000	**0.99993039**
	SNOPT	0.99995470	**0.99995405**	0.99995210	**0.99995200**	0.99993000	**0.99993040**
Promedio		0.99921373	0.99911282	0.99995207	0.99995196	0.83499873	0.83504483
Máximo		0.99995470	**0.99995405**	0.99995210	**0.99995200**	0.99993000	**0.99993040**

Tabla 26. Comparación del CCP equivalente con otras dos políticas simples de alojamiento en cache.
Resultados numéricos de varios algoritmos para la instancia de ejemplo del CCCP de 16 clases.

5.2 DNS - Domain Name System

El sistema DNS (Domain Name System) es una red semántica sensible de distribución jerarquizada, donde la consistencia de la información es muy importante. El sistema que implementan la mayoría de los servidores recursivos para mantener la consistencia es el mecanismo de TTL, sin embargo, como se dijo en la sección 4.3.2, este mecanismo no asegura completamente la consistencia.

Nuestros modelos *CCP* y *CCCP* tampoco aseguran la consistencia, sus funciones objetivo intentan maximizar la cantidad de fuentes correctas en las respuestas, siendo esto una forma de mejor esfuerzo en la consistencia.

A diferencia del caso de estudio P2P, para el sistema DNS se extrae de servidores reales toda la información necesaria para crear las instancias del modelo *CCP*. Esto es posible gracias al apoyo de ANTEL[2], la principal empresa de telecomunicaciones de Uruguay.

5.2.1 Objetivos

El objetivo de este caso de estudio es mostrar la utilidad de los modelos *CCP* y *CCCP* para la red DNS. Para ello, es necesario alcanzar los siguientes objetivos específicos:

- Crear una metodología para extraer instancias del modelo *CCP* de un servidor recursivo real.
- A partir de una instancia del *CCP* agrupar el contenido similar en clases para generar instancias del *CCCP*. Validar el metodo creado para el caso de estudio P2P.
- Validar el metodo usado en el caso de estudio P2P para la solución de los problemas *CCP* y *CCCP*.

5.2.2 Medidas Reales: Obtención de las Instancias

Instancias de *CCP*

Los servidores recursivos son los nodos agregadores de la red DNS. Se utilizaron datos reales del desempeño de los servidores recursivos de ANTEL[2], que para sus servicios de Internet diariamente atienden a cientos de miles usuarios. Todos los cálculos se realizaron en dos servidores recursivos que reciben aproximadamente 1800 *qps* c/u durante el pico diario. Por claridad solo se incluye en este documento el estudio en uno de ellos.

Los parámetros necesarios para definir una instancia del *CCP* se encuentran listados en la Tabla 27.

Parámetros
T : tiempo de observación
C : cantidad contenidos
f : distribución de la tasa de solicitud
$\overline{\lambda}$: tasa de comienzo de alojamiento promedio de un contenido cualquiera
$\overline{\mu}$: tasa de validez (desalojamiento) promedio de un contenido cualquiera
α_S : tamaño de la respuesta a una solicitud de contenido
α_B : tamaño de la respuesta a una búsqueda en el backbone
β_S : tamaño de un paquete de solicitud
β_B : tamaño de un paquete de búsqueda en el backbone
BW_{IN} : ancho de banda entrante
BW_{OUT} : ancho de banda saliente

Tabla 27. Parámetros del CCP.

El tiempo de observación T, es simplemente un parámetro de escalamiento. A los efectos de no trabajar con números muy pequeños se utiliza una hora como parámetro T.

Los contenidos en la red DNS son las traslaciones entre nombres de dominio y direcciones IPs. En este estudio solo se analizan los dominios de Uruguay, es decir los terminados en *.uy*.
Para determinar la cantidad de contenidos C a considerar y la distribución de la tasa de solicitud f se analizan los logs del servidor recursivo.
El software utilizado en la infraestructura DNS de ANTEL es BIND[18], lo que facilita esta tarea gracias a la rica información que se puede obtener de sus archivos de log. Configurados los servidores para almacenar todas las solicitudes realizadas por los clientes, se procesaban los archivos de logs, rellenando una base de datos BerkeleyDB[219] con la estadística de los dominios solicitados. Todo nuevo dominio solicitado se agregaba a la base de datos, mientras que para los ya ingresados se incrementaba la cantidad de consultas realizadas.
Luego de 10 días de análisis y aproximadamente 115 horas de logs (se consideraron solo las horas donde se realizan más consultas) se obtuvo la cantidad de contenidos solicitados $C = 220107$ y la distribución de solicitudes entre los dominios.
La Ilustración 21 muestra los resultados.

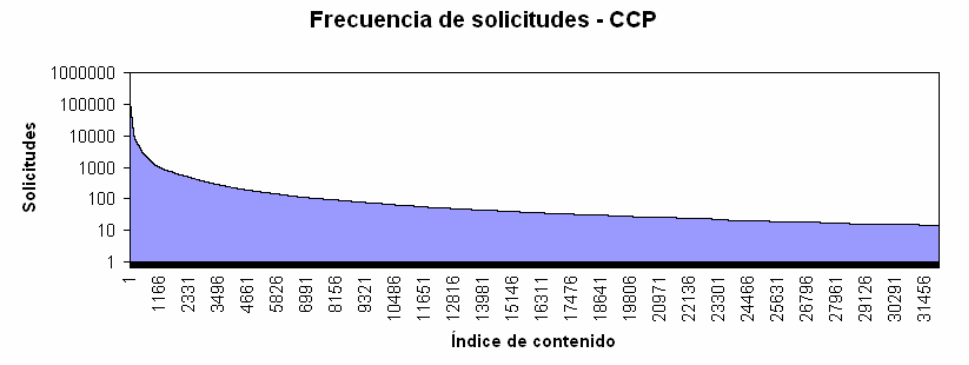


Ilustración 21. Distribución de solicitudes de dominios (se muestran solo los 30.000 dominios más solicitados).

A medida que se procesan más archivos de logs, más contenidos deben ser incorporados, difícilmente este proceso se estabilice dado que la distribución de solicitudes es de cola pesada y los dominios se encuentran en constante crecimiento y cambio. La Ilustración 22 muestra los nuevos dominios incorporados para los 700 archivos de logs procesados, puede observarse una tendencia de disminución en la cantidad de nuevos dominios.

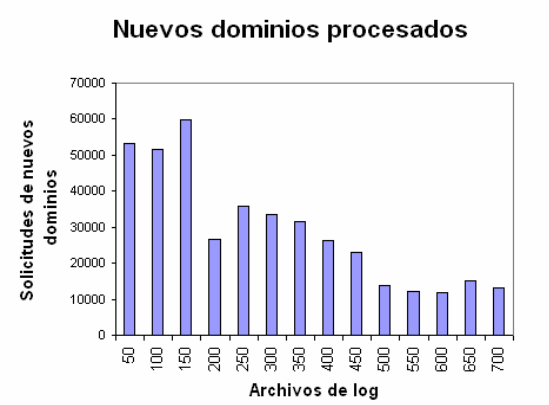


Ilustración 22. Procesamiento de nuevos contenidos en los archivos de logs.

ANTEL es responsable de la administración de la zona *.com.uy*. Esta zona cuenta con la mayor cantidad de registros de tercer nivel del dominio *.uy* (ver Ilustración 23). Para obtener las tasas de alojamiento y validez de los dominios se utiliza la información histórica de cambios en los dominios de la zona *.com.uy*.

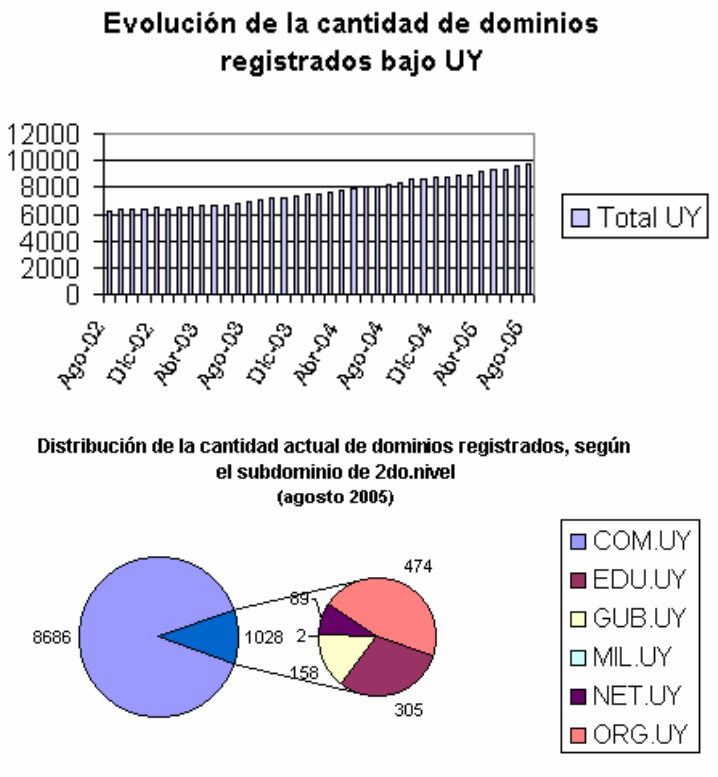


Ilustración 23. Estadísticas del dominio *.uy*. Fuente: SeCIU.

Procesando los cambios realizados entre octubre de 2003 y octubre de 2005 en la zona *.com.uy*, se obtienen los valores promedio de las tasas $\overline{\lambda}$ y $\overline{\mu}$. La tasa de comienzo de alojamiento promedio de un contenido cualquiera es $\overline{\lambda} = 0.836098208\,\text{hora}^{-1}$ y la tasa de validez (desalojamiento) promedio de un contenido cualquiera es $\overline{\mu} = 0.515815085\,\text{hora}^{-1}$. Existe una tendencia de crecimiento en $\overline{\lambda}$ y $\overline{\mu}$ con el paso del tiempo (ver Ilustración 24), por lo que se usaron exclusivamente los valores del año 2005 para hallar el promedio.

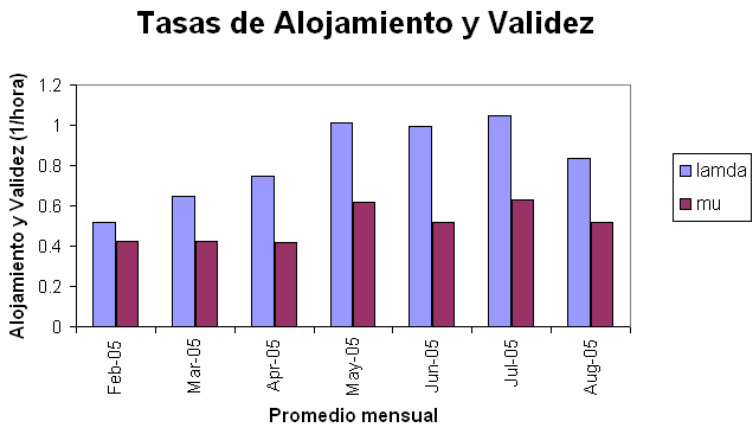


Ilustración 24. Estadísticas de tasas de alojamiento y validez en el dominio *.com.uy*.

Dado que existen muy pocas coincidencias entre los dominios solicitados (hallados mediante el análisis de logs) y los dominios de tercer nivel alojados en la zona *.com.uy* se asume la misma tasa de alojamiento $\overline{\lambda}$ y validez $\overline{\mu}$ para todos los contenidos de la instancia.

El tamaño de los paquetes en la red se expresan con los parámetros α_S, α_B, β_S y β_B. Al igual que en el caso de estudio P2P una búsqueda en el backbone requiere de varias consultas a los servidores autoritativos. Analizando los paquetes transmitidos y recibidos por el servidor recursivo se obtiene de forma estadística los valores de estos parámetros. La Tabla 28 muestra los datos obtenidos en la captura de paquetes en un lapso de 15 minutos.

	Solicitudes de cliente	Respuesta a solicitudes de cliente	Búsquedas en el backbone	Respuestas a búsquedas en el backbone
Cantidad	366148	168530	61481	56442
Bytes	29458160	40089683	5594012	10774381
Registros	402098	236386	293552	34742

Tabla 28. Estadísticas de los paquetes trasmitidos y recibidos por el servidor de nombres.

El tamaño promedio de una solicitud realizada por un cliente es (cantidad de solicitudes/bytes trasmitidos): $\beta_S = \frac{366148}{29458160} = 80.454 \text{ bytes}$.
La cantidad de fuentes promedio que se responden a una solicitud de cliente son (registros respondidos/cantidad de respuestas) : $\frac{236386}{168530} = 1.4026$. Una respuesta a cliente son en promedio $\frac{40089683}{168530} = 237.8786$ bytes. El parámetro α_S es la cantidad de bytes de una respuesta a un cliente por cada fuente respondida, por tanto $\alpha_S = \frac{237.8786}{1.4026} = 169.5942 \text{ bytes/fuente}$.

Una consulta realizada por un cliente deriva en una o varias consultas a realizar por el servidor recursivo en el backbone de servidores autoritativos.
La cantidad de solicitudes que el servidor recursivo puede responder sin consultar al backbone se encuentra dada por la cantidad de coincidencias entre las solicitudes y lo alojado en el cache. A este valor se le conoce como *hitRate* y para el servidor recursivo analizado es de 0.9604. Por tanto solo 4 de cada 100 solicitudes requieren de una búsqueda en el backbone, y en el caso de los datos capturados solo $366148 * (1 - 0.9604) = 14508.8973$ solicitudes requieren de la búsqueda.
Del siguiente análisis se obtiene el valor β_B (bytes enviados al backbone/solicitudes enviadas al backbone): $\beta_B = \frac{5594012}{14508.8973} = 385.5574 \text{ bytes}$

De forma similar se puede calcular el parámetro α_B, como los bytes recibidos del backbone/solicitudes respondidas desde el backbone/cantidad de fuentes respondidas:

$$\alpha_B = \frac{10774381}{168530 * (1 - 0.9604)} * \frac{1}{1.4026} = 1150.252 \text{ bytes/fuente}.$$

Finalmente el ancho de banda de un agregador se limita por los parámetros BW_{IN} y BW_{OUT}. Posiblemente la limitante en un servidor recursivo colocado en la infraestructura de un ISP no sea su ancho de banda, sino su procesamiento. Es posible utilizar los parámetros BW_{IN} y BW_{OUT} para describir una limitante en procesamiento. La forma más sencilla de realizar esto, es utilizar como cota de ancho de banda el ancho de banda real del servidor recursivo, que en el caso de estudio es:
$BW_{IN} = 419.5 \text{ kbps} = 193305600 \text{ bytes/hora}$ y
$BW_{OUT} = 747.6 \text{ kbps} = 344494080 \text{ bytes/hora}$.

La Tabla 29 resume los valores de los parámetros utilizados para la creación de la instancia del modelo *CCP*. Y la Tabla 30 muestra parcialmente la instancia generada en lenguaje AMPL.

		Instancia
Parámetros	T : tiempo de observación	1 hora
	C : cantidad contenidos	220107
	f : distribución de tasa de solicitud	ver Ilustración 25
	$\overline{\lambda}$: tasa de comienzo de alojamiento promedio de un contenido cualquiera	0.836098208 hora^{-1}
	$\overline{\mu}$: tasa de validez (desalojamiento) promedio de un contenido cualquiera	0.515815085 hora^{-1}
	α_S : tamaño de la respuesta a una solicitud de contenido	169.594 bytes
	α_B : tamaño de la respuesta a una búsqueda en el backbone	1150.252 bytes
	β_S : tamaño de un paquete de solicitud	80.454 bytes
	β_B : tamaño de un paquete de búsqueda en el backbone	385.557 bytes
	BW_{IN} : ancho de banda entrante	193305600 bytes/hora
	BW_{OUT} : ancho de banda saliente	344494080 bytes/hora

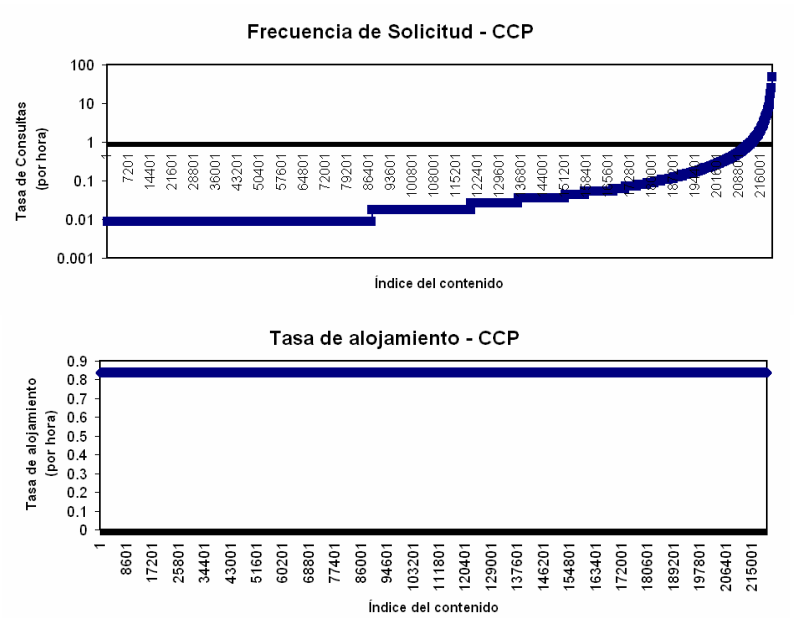
Tabla 29. Valores de los parámetros utilizados para la creación de las instancias del CCP.

```
param C        := 220107;

param alphaS   := 169.594;
param alphaB   := 1150.252;
param betaS    := 80.454;
param betaB    := 385.557;
param BWin     := 193305600;
param BWout    := 344494080;

param f :=
         1 5909.551958
         2 5310.996695
         3 3926.524960
         4 3155.834680
```

```
         5 2997.397158
         6 2660.510049
         … …
         … …
         … …
    220102 0.008686
    220103 0.008686
    220104 0.008686
    220105 0.008686
    220106 0.008686
    220107 0.008686;

param lamda :=
         1  0.836098208
         2  0.836098208
         3  0.836098208
         4  0.836098208
         … …
         … …
         … …
    220104  0.836098208
    220105  0.836098208
    220106  0.836098208
    220107  0.836098208;

param mu :=
         1  0.515815085
         2  0.515815085
         3  0.515815085
         4  0.515815085
         … …
         … …
         … …
    220103  0.515815085
    220104  0.515815085
    220105  0.515815085
    220106  0.515815085
    220107  0.515815085;
```

Tabla 30. Instancia del CCP en AMPL. ccp-dns.dat

La Ilustración 25 muestra de forma gráfica la distribución obtenida para las principales variables del *CCP* en el caso de estudio DNS.



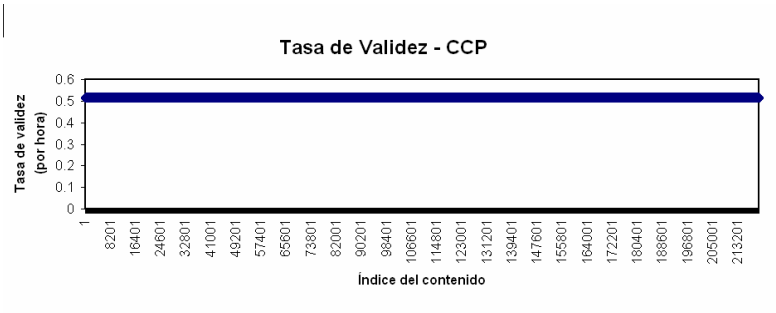


Ilustración 25. Distribución del caso de estudio DNS.

Es útil remarcar algunas diferencias con las instancias *CCP* del caso de estudio P2P. En la red DNS:

- Hay considerablemente menos contenidos.
- Todos los contenidos presentan las mismas tasas de alojamiento y validez, y por tanto son alojados en la misma cantidad de fuentes $\bar{\lambda}/\mu = 1.6209$.
- En promedio el contenido es valido la mitad de tiempo y se comienza a alojar el contenido con una tasa diez veces menor (el contenido es menos dinámico en la red de DNS).
- Las consultas en el backbone son aproximadamente tres veces más costosas en consumo de ancho de banda.
- El ancho de banda entrante al nodo agregador es cuatro veces menor.

Instancias de *CCCP*

La cantidad de contenidos en la instancia *CCP* amerita el mismo tratamiento que se realizó en el caso de estudio P2P: la estrategia de resolución es obtener a partir de la instancia del *CCP*, una instancia del *CCCP* equivalente; reduciendo la cantidad de variables de decisión de cientos de miles a la cantidad de clases elegidas.

La metodología utilizada para hallar la instancia del *CCP* fija los valores λ y μ constantes e iguales para todos los contenidos. Por tanto, al igual que para el caso de estudio P2P, la única dimensión que presenta información para el método de conglomerado es la tasa de solicitud f. Por este motivo se utiliza el mismo algoritmo de conglomerado (clustering) que para el caso P2P (ver descripción en el apéndice A).

También se utilizaron los tamaños de clases: **2**, **8**, **16**, **32** y **128**.
A continuación se muestran dos instancias del *CCCP* como ejemplo, la Tabla 31 una instancia de 2 clases y la Tabla 32 una instancia de 16 clases. El apéndice B incluye las cinco instancias del problema *CCCP*, una instancia para cada tamaño de clase generadas a partir de la instancia *CCP* obtenida de datos reales en la sección anterior.

```
param K        := 2;

param alphaS   := 169.59400000;
param alphaB   := 1150.25200000;
param betaS    := 80.45400000;
param betaB    := 385.55700000;
param BWin     := 193305600.00000000;
param BWout    := 344494080.00000000;

param f :=
         1 1059.31197749
         2 0.32706349;
```

```
param lamda :=
         1 0.83609821
         2 0.83609821;
param mu :=
         1 0.51581508
         2 0.51581509;
param l :=
         1 68.00000000
         2 220039.00000000;
```

Tabla 31. Instancia del CCCP (2 clases) en AMPL. cccp-dns-K002.dat

```
param K        := 16;

param alphaS   := 169.59400000;
param alphaB   := 1150.25200000;
param betaS    := 80.45400000;
param betaB    := 385.55700000;
param BWin     := 193305600.00000000;
param BWout    := 344494080.00000000;

param f :=
         1 5610.27432650
         2 3359.91893267
         3 2454.91953550
         4 1787.96496740
         5 1345.55070329
         6 673.74281515
         7 432.99018030
         8 348.34575252
         9 240.09216583
        10 100.73750319
        11 51.88965537
        12 25.17129564
        13 9.64498192
        14 3.67272762
        15 0.82732839
        16 0.04138581;
param lamda :=
         1 0.83609821
         2 0.83609821
         3 0.83609821
         4 0.83609821
         5 0.83609821
         6 0.83609821
         7 0.83609821
         8 0.83609821
         9 0.83609821
        10 0.83609821
        11 0.83609821
        12 0.83609821
        13 0.83609821
        14 0.83609821
        15 0.83609821
        16 0.83609821;
param mu :=
         1 0.51581508
         2 0.51581508
         3 0.51581508
         4 0.51581508
         5 0.51581508
         6 0.51581508
         7 0.51581508
         8 0.51581508
         9 0.51581508
        10 0.51581508
        11 0.51581508
        12 0.51581509
        13 0.51581508
        14 0.51581508
        15 0.51581509
        16 0.51581509;
param l :=
         1 2.00000000
```

```
 2 3.00000000
 3 4.00000000
 4 5.00000000
 5 7.00000000
 6 13.00000000
 7 20.00000000
 8 25.00000000
 9 36.00000000
10 85.00000000
11 165.00000000
12 339.00000000
13 884.00000000
14 2321.00000000
15 10300.00000000
16 205898.00000000;
```

Tabla 32. Instancia del CCCP (16 clases) en AMPL. cccp- dns-K016.dat

5.2.3 Método de Resolución

Para el caso de estudio DNS se aplica exactamente la misma metodología de resolución que para el caso P2P. Ver la sección 5.1.3.

5.2.4 Resultados

A partir de datos reales se creó una instancia del problema *CCP*. Usando ésta instancia, mediante un método de conglomerado se crearon 5 instancias del problema *CCCP* con distintas cantidades de clases[15]. La cantidad de clases elegidas fueron: 2, 8, 16, 32 y 128.

La resolución de estas 5 instancias del problema *CCCP* no requieren gran capacidad de cómputo y todas se ejecutaron en un computador PIII 800 Mhz., con 320 Mb. de RAM (idem. al caso de estudio P2P).

La Tabla 33 muestra los resultados para las instancias del problema *CCCP*.
En todos los casos la restricción que limita la solución es el ancho de banda entrante BW_{IN}.
Puede observarse como el tiempo de ejecución crece con el tamaño del problema, es decir la cantidad de clases.

		Resultados			
		ε Función Objetivo	Tiempo Ejecución (s)	BW_{IN} consumido (bytes/hora)	BW_{OUT} consumido (bytes/hora)
Cantidad de clases (*K*)	2	0.96962300	0.000	193305400	77163700
	8	0.98221600	0.020	193305400	77163700
	16	0.99721800	0.060	193305400	77163700
	32	0.99974200	0.120	193305400	77163700
	128	0.99991900	0.120	193305400	77163700

Tabla 33. Resultados numéricos para las instancias del CCCP. Caso de estudio: DNS.

Al igual que en el caso de estudio P2P, para realizar una comparación justa entre las soluciones de problemas *CCCP* es necesario convertirlos al problema original *CCP* y

[15] El apéndice B incluye las cinco instancias del problema CCCP, junto con sus soluciones.

comparar la función objetivo del *CCP.* La Tabla 34 muestra los resultados promedio para las instancias del problema *CCCP* y su equivalente *CCP*.

		Resultados					
		CCCP			*CCP* equivalente		
		ε Función Objetivo *CCCP*	BW_{IN} consumido (bytes/hora) *CCCP*	BW_{OUT} consumido (bytes/hora) *CCCP*	ε Función Objetivo *CCP*	BW_{IN} consumido (bytes/hora) *CCP*	BW_{OUT} consumido (bytes/hora) *CCP*
Cantidad de clases (*K*)	2	**0.96962300**	193305400	77163700	**0.88249787**	112542424.6	60462535.8
	8	**0.98221600**	193305400	77163700	**0.95332949**	179132772.1	74232842.8
	16	**0.99721800**	193305400	77163700	**0.99412569**	189667888.7	76411413.7
	32	**0.99974200**	193305400	77163700	**0.99966787**	192902705.6	77080345.8
	128	**0.99991900**	193305400	77163700	**0.99991900**	193304154.8	77163362.0

Tabla 34. Resultados numéricos para las instancias del CCCP. Cada solución del CCCP se convierte en una solución del CCP y se evalúa. Caso de estudio: DNS.

Puede observarse como a medida que crece la cantidad de clases se alcanzan mejores soluciones del problema *CCP* (ver ε del *CCP*). Esto se debe a una utilización más eficiente del ancho de banda entrante (BW_{IN}) que es la restricción alcanzada en las instancias analizadas.

Para instancias de muy pocas clases se hace evidente el error cometido en la simplificación del problema *CCP* a un problema *CCCP*. La Tabla 35 muestra la discrepancia promedio cometida en la simplificación para las distintas opciones de cantidad de clases (*K*). Solo dos clases (*K=2*) no son suficientes para modelar el problema, cometiéndose una discrepancia relativa de 10% aprox. Cuando se utilizan 32 clases o más se alcanza muy buenas estimaciones, con errores menores al 1%. A su vez los tiempos de cómputo son bajos para las distintas cantidades de clases estudiadas (para *K=128* la ejecución dura aproximadamente un décimo de segundo). Por tanto, al igual que en el caso de estudio P2P, parece adecuado elegir entre 32 o 128 clases.

		Análisis de Resultados			
		ε Función Objetivo *CCCP*	ε Función Objetivo *CCP equivalente*	Discrepancia absoluta $\lvert\varepsilon_{CCP} - \varepsilon_{CCCP}\rvert$	Discrepancia relativa $\frac{\lvert\varepsilon_{CCP} - \varepsilon_{CCCP}\rvert}{\varepsilon_{CCP}} * 100$
Cantidad de clases (*K*)	2	0.96962300	0.88249787	**0.087125127**	**9.8726%**
	8	0.98221600	0.95332949	**0.028886513**	**3.0301%**
	16	0.99721800	0.99412569	**0.003092311**	**0.3111%**
	32	0.99974200	0.99966787	**7.41270E-05**	**0.0074%**
	128	0.99991900	0.99991900	**2.15618E-09**	**2.1564E-07%**

Tabla 35. Análisis de resultados numéricos para las instancias del CCCP y su CCP equivalente. Caso de estudio: DNS.

5.2.5 Conclusiones

Se resumen algunas conclusiones extraídas de la aplicación de los problemas *CCP* y *CCCP* al sistema DNS.

Elección de la Cantidad de Clases

Como se explicó para el caso de estudio P2P, cada red presenta distintas características particulares en la distribución de las solicitudes y alojamiento del contenido; dependiendo de estas características, de la precisión deseada, y de la disponibilidad de cómputo, es la elección de la cantidad de clases mínimas necesarias para expresar el contenido.

Para el caso de estudio DNS parece adecuado elegir entre 32 o 128 clases (repitiéndose la conclusión del caso P2P). Si se utilizan 32 clases o más se alcanza una precisión muy buena (discrepancia mucho menor al 1%), mientras que los tiempos de cómputo son relativamente bajos (para *K=128* la ejecución dura menos de una décima de segundo).

Fortalezas y Debilidades en las Medidas Reales

Las instancias del problema *CCP* se obtuvieron a partir de datos reales extraídos de servidores de DNS de gran porte. Cuatro fuentes de datos fueron utilizadas:

- Archivos de logs del servidor recursivo.
- Captura de tráfico del servidor recursivo.
- Histórico de consumo de ancho de banda del servidor recursivo.
- Histórico de cambios en la zona *.com.uy* de un servidor autoritativo.

Todo el procesamiento se realizó mediante programación de scripting, en lenguaje Perl[196].

De los archivos de logs se obtuvo una basta información sobre la distribución de las solicitudes, pero no se estudió en detalle la tendencia de crecimiento de la cantidad de dominios. Con un estudio más detallado podría estimarse con más precisión la forma de la distribución de las solicitudes f y de la cantidad de contenidos C.

Respecto a la captura de tráfico, ésta se realizó en un período muy corto debido a limitantes técnicas. Ampliar esta medida permitiría una mejor estimación de los tamaños de los paquetes del modelo α_S, α_B, β_S y β_B. Además de los cálculos de los paquetes, esta medida ofrece resultados de mucha utilidad en la operación y mantenimiento de los servidores de nombre, puesto que incluye datos como el *hitRate* del servidor, la cantidad promedio de servidores autoritativos que se consultan para resolver un nombre, y otros datos estadísticos útiles.

El histórico del consumo de ancho de banda de los servidores se obtiene a partir de estadísticas obtenidas por SNMP[163][220] de los servidores. Con este método se obtienen las restricciones BW_{IN} y BW_{OUT}. Se utiliza esta estrategia para determinar el ancho de banda del servidor puesto que en realidad la limitante para estos servidores (alojados por un ISP con buena conectividad) no es la comunicación sino el procesamiento de las consultas. Debido a que la cantidad de consultas realizadas en el servidor real y las realizadas en el modelo difieren según la política de tiempos de expiración, podría ser de gran utilidad refinar el modelo incluyendo el costo en procesamiento de cada consulta.

Finalmente la obtención de las distribuciones de λ y μ son las medidas más débiles realizadas. A partir de información histórica de la zona *.com.uy* se obtuvo las tasas de cambio en el alojamiento de los dominios, al prácticamente no coincidir con los dominios consultados por los solicitantes no fue posible estimar para cada contenido solicitado cual es su tasa de alojamiento y validez. Utilizando entonces los valores $\overline{\lambda}$ y $\overline{\mu}$ para todos los contenidos. Mejorar estas estimaciones es muy importante para potenciar el modelo *CCCP* planteado en este trabajo (a pesar que no se conoce un mecanismo sencillo para lograrlo).

Otras Políticas

La comparación con otras políticas es un trabajo pendiente para el caso de estudio DNS.
La política de los servidores BIND[18] se basa en la utilización del mecanismo de TTL, dos comparaciones son necesarias: en primer lugar es necesario cuantizar la diferencia en la consistencia de la información ofrecida por cada método y en segundo lugar la performance ofrecida en cuanto a consumo de recursos (ver el detalle de la propuesta en la sección 6.2).

Conclusiones Generales

6

En este trabajo se estudia a las redes de contenido, presentando el estado del arte asociado a las mismas.
Se especifican una serie de problemas relacionados con su arquitectura y diseño desde un enfoque de optimización combinatoria, estudiando en detalle uno de estos problemas: el compromiso entre publicación y búsqueda de información.
Se reseñan algunos de los principales modelos y algoritmos existentes para la resolución del problema y se proponen dos modelos de optimización, uno aplicable al contexto general y otro a los nodos más comprometidos de la red: los nodos agregadores (o cache).
Para el problema en los nodos agregadores se presenta una metodología de resolución, la cual se aplica a dos casos de estudios concretos: una red P2P para compartir archivos y el sistema DNS.

6.1 Conclusiones

La primera contribución del trabajo es el estudio en profundidad y síntesis del estado del arte de las redes de contenido.
Se ha recopilado y estudiado la literatura referente a redes de contenido y redes P2P de forma detallada, en particular lo que refiere a la arquitectura y diseño de las mismas, y a la medición y evaluación de sus performances.
De esta forma, son estudiados en detalle dos conjuntos de características: uno relacionadas con la arquitectura y otro relacionado con el comportamiento, presentando ejemplos de redes de contenido reales.
A partir de dichas características se tiene como síntesis y extensión de las propuestas existentes en la literatura, una propuesta de taxonomía para redes de contenido y redes P2P. Dicha taxonomía, además de ser una herramienta para la comprensión de estas redes, también es útil para ver cuáles combinaciones de características son las más empleadas y cuales en cambio han sido escasamente o nada exploradas, abriendo perspectivas de desarrollo de nuevas propuestas en estas últimas categorías.

Del análisis del estado del arte y de las principales características tambíen se extraen cuáles son los problemas o compromisos esenciales a nivel del diseño que determinan la performance en estas redes. La segunda contribución del trabajo es el modelado de uno de estos problemas, el del compromiso entre la publicación y la búsqueda de la información.
En esta línea, se desarrolló un modelo detallado del comportamiento de los nodos de una red de contenido, que combina un modelado de la evolución discretizado en el tiempo con una formulación de programación entera. Este modelo no fue implementado computacionalmente pero puede ser base de estudio futuro para su simulación o eventual optimización.
También se estudió en detalle la fijación de tiempos de expiración de contenidos en un nodo agregador (o nodo cache), obteniéndose (con ciertas hipótesis simplificadoras) un modelo de programación no lineal para resolver este problema.

Este es un caso particular del problema de compromiso entre publicación y búsqueda de información, donde el nodo agregador debe determinar que contenido alojar en su cache y que contenido debe buscar en la red de contenido, de forma de maximizar las fuentes de contenido respondidas a los solicitantes.

El problema de la fijación de tiempos de expiración en los nodos agregadores fue estudiado en profundidad, planteando como tercera contribución el empleo del modelo de programación no lineal mencionado para su solución. La factibilidad práctica de su aplicación fue estudiada de manera detallada, a través del análisis del comportamiento del método en dos casos de estudio: una red P2P para compartir archivos, generándose los casos de prueba a partir de datos extraídos de la literatura; y el sistema DNS, generándose los casos de prueba a partir de un servidor recursivo (nodo agregador de la red DNS) real de gran porte. En ambos casos fue posible obtener con un esfuerzo razonable los datos necesarios para la formulación del modelo, eventualmente aprovechando algunas simplificaciones posibles por las características particulares de cada red. Asimismo, se compararon los resultados con otros métodos de solución y con otras políticas de caching. Algunos algoritmos de optimización (en lugar de MINOS) resultaron tener un comportamiento levemente superior para algunas de las instancias de los casos de estudio, suponiéndose que es posible mejorar el algoritmo de solución. Respecto a las otras políticas sencillas de caching empleadas nuestro modelo tuvo un mejor comportamiento.

6.1.1 Publicaciones

Se realizaron algunas publicaciones en base al trabajo de investigación que comprendió el período de la maestría. A continuación se detallan estos aportes.

Papers en anales de conferencias con referato:

P. Rodríguez-Bocca, H. Cancela. A mathematical programming formulation of optimal cache expiration dates in content networks. LANC'2005 (IFIP/ACM Latin American Networking Conference). Cali, Colombia - October 10 - 14, 2005.

P. Rodríguez-Bocca, H. Cancela Bosi. Redes de contenido: un panorama de sus características y principales aplicaciones. AST 2004 (5th Argentine Symposium on Computing Technology), 33 JAIIO, Córdoba - Argentina. September 20-24, 2004.

Papers en anales de conferencias con aceptación en base a resumen extendido:

B. Tuffin, P. Rodríguez-Bocca and H. Cancela Bosi. End-to-end reliability-dependent pricing of network services. XII Congreso Latino Iberoamericano de Investigación de Operaciones (CLAIO). Cuba, October, 2004.
(paper invitado para número especial del Annals of Operations Research dedicado al CLAIO, en referato).

Posters:

P. Rodríguez-Bocca, H. Cancela, B. Tuffin. A genetic algorithm for upgrading communication networks employing reliability based service pricing. ICIL 2005 (International Conference on Industrial Logistics), February 14-18, 2005, Montevideo, Uruguay. Proceedings p. 341.

Reportes técnicos:

H. Cancela, P. Rodríguez-Bocca. A mathematical programming formulation of optimal cache expiration dates in content networks. Technical Report INCO-RT-05-07. PEDECIBA Informática, Facultad de Ingeniería, UDELAR, 2005.

B. Tuffin, H. Cancela Bosi, P. Rodríguez-Bocca. End-To-End reliability-dependent pricing of network services. Technical Report PI-1639. IRISA, INRIA, Francia, 2004.

P. Rodríguez-Bocca and H. Cancela. Redes de contenido: un panorama de sus características y principales aplicaciones. Technical Report INCO-RT-03-11. PEDECIBA Informática, Facultad de Ingeniería, UDELAR, 2003.

P. Rodríguez-Bocca. Lenguajes Canónicos para la Descripción de Grafos: Estudio y Transformación entre Esquemas de distintos Modelos de Datos. Technical Report INCO-RT-03-08. PEDECIBA Informática, Facultad de Ingeniería, UDELAR, 2003.

P. Rodríguez-Bocca. Discusión y Análisis de una nueva metaheurística: SN. Technical Report INCO-RT-03-02. PEDECIBA Informática, Facultad de Ingeniería, UDELAR, 2003.

6.2 Trabajo a Futuro

Algunas extensiones al trabajo realizado son posibles y deseables:

- El modelo general *CDP* debe ser implementado y probado. Su potencialidad descriptiva se muestra en este trabajo especificando en el modelo a las redes Napster, Gnutella y DNS.

- Una característica cuestionable de los modelos particulares *CCP* y *CCCP* es que no utilizan los mecanismos de TTL disponibles en algunas de las redes para brindar consistencia en la información.
    - Estos modelos en lugar de asegurar la consistencia (como el mecanismo de TTL) realizan el mejor esfuerzo para alcanzarla (la función objetivo a maximizar representa la consistencia). Un estudio más fino sobre la consistencia de los mecanismos planteados es necesario para las redes que utilizan TTL, como el sistema DNS. De esta forma se compararían las distintas políticas de alojamiento en el cache no solo desde el punto de vista de la performance, sino también desde el impacto por respuestas erróneas a los solicitantes.
    - Otra línea de acción es extender los modelos para incorporar la restricción de TTL sobre los contenidos, esto es prácticamente inmediato para el modelo *CCP* y posible en el *CCCP*. En esta situación es posible comparar en igualdad de condiciones la performance con otros trabajos relacionados.

- En este trabajo se emplean paquetes estándar para resolver numéricamente los problemas *CCCP* y *CCP*. Existe mucho trabajo relacionado a la solución de modelos de programación no lineal, es posible comparar y mejorar la metodología con la incorporación de este conocimiento.

- Implementar la política *CCCP* (con o sin restricción de TTL) en un nodo agregador real es un camino a seguir. Existen nodos agregadores de gran

escalado e implantación que se basan en software de código abierto (por ejemplo BIND[18] y Squid[222]), en estos sistemas la incorporación de esta técnica es factible y seguramente beneficiosa para su portador. Para realizar esto seguramente sea necesario acondicionar el método de resolución y obtención de las instancias para su utilización en línea.

Bibliografía

b

[1] Adar, E., and Huberman, B. A., “Free Riding on Gnutella,” Technical report, Xerox PARC, August 2000. Available at http://www.hpl.hp.com/shl/papers/gnutella/Gnutella.pdf

[2] Administración Nacional de Telecomunicaciones – ANTEL. Home Page http://www.antel.com.uy/.

[3] Agarwal A., Simoni R., Hennessy J., Horowitz M.. An Evaluation of Directory Schemes for Cache Coherence. In Proceedings of the 15th Annual International Symposium on Computer Architecture, pages 280--289. IEEE, June 1988.

[4] Akamai Homepage, http://www.akamai.com/.

[5] Albitz P., Liu C.. DNS and BIND, 4th Edition. ISBN 0-596-00158-4. Publcado por O'Reilly, abril 2001.

[6] Almeida, J., Broder, A., Cao, P., and Fan, L., “Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol,” in Proc. of ACM SIGCOMM’98, September 1998.

[7] AMPL - A Modeling Language for Mathematical Programming. Home Page. http://www.ampl.com/

[8] AMPL - User's guide AMPL/MINOS. http://www.ampl.com/BOOKLETS/ampl-minos.pdf

[9] Anceaume E., Gradinariu M., y Roy M. “ Self-organizing Systems Case Study: peer-to-peer net-works”. Reporte interno PI-1535, IRISA, Rennes, Francia. 2003.

[10] Austin T., Pnevmatikatos D., Sohi G., Streamlining Data Cache Access with Fast Address Calculation. In Proceedings of the 22nd Annual International Symposium on Computer Architecture, pages 369--380, Santa Margherita Ligure, Italy, June 22--24, 1995.

[11] AVAKI CORPORATION. 2001. Avaki 2.0 Concepts and Architecture. White paper. www.avaki.com/papers/AVAKI_concepts_architecture.pdf.

[12] Babaoglu O., Meling H., and Montresor A.. Anthill: A framework for the development of agent-based peer-to-peer systems. Technical Report UBLCS-2001-09, University of Bologna, Italy, 2001. http://www.cs.unibo.it/projects/anthill/.

[13] Barak A., Litman A. 1985. MOS: a Multicomputer Distributed Operating System. Software – Practice and Experience, 15(8):725–737. August.

[14] Barkai D. Towards Balanced Computing : Weaving P2P Technologies into the Fabric of Computing over the Net, Internet2 P2P Workshop, Jan 2002.

[15] Bearshare Home Page http://www.bearshare.com/.

[16] Becker D.J., Stirling T., Savarese D., Dorband J.E., Ranawak U.A., Packer C.V. 1995. "Beowulf: A Parallel Workstation for Scientific Computation", Proceedings of the International Conference on Parallel Processing.

[17] Bhattacharjee B., Chawathe S., Gopalakrishnan V., Keleher P., Silaghi B., Efficient peer-to-peer searches using result-caching. In The 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03), February 2003.

[18] BIND - Berkeley Internet name domain. Internet Software Consortium, http://www.isc.org/products/BIND, June 2004.

[19] BitTorrent Home Page http://www.bittorrent.com/.

[20] Bolcer G., et al., Peer-to-Peer Architectures and the Magi Open Source Infrastructure. Endeavors technologies (www.endeavors.com). White Paper, 2000.

[21] Bolcer G., Magi: An Architecture for Mobile and Disconnected Workflow. Endeavors technologies (www.endeavors.com). White Paper, 2001.

[22] Bouguettaya, B. Benatallah y A. Elmagarmid. "Interconnecting heterogeneous information systems" Kluwer Academic, 1998.(ISBN 0-7923-8216-1).

[23] Breslau L., Cao P., Fan L., Phillips G., Shenker S.. Web caching and zipf-like distributions: Evidence and implications. In Proceedings of the IEEE INFOCOM'99 Conference, 1999.

[24] Cancela H., El Khadiri M. On the RVR Simulation Algorithm for Network Reliability Evaluation. Aceptado (enero 2002) para su publicación en IEEE Tr. on Reliability (número a aparecer en el correr de 2003).

[25] Cancela H., Petingi L. Diameter constrained network reliability: exact evaluation by factorization and bounds. In ICIL'2001 (International Conference on Industrial Logistics), pages 359-366, Okinawa, Japan, 9-12 July 2001.

[26] Cancela H., Petingi L. Polynomial time computation of the reliability of source-terminal complete networks with path length restrictions. In CLEI'2002 (XXVIII Conferencia Latinoamericana de Informática), Montevideo, Uruguay, Noviembre 2002 (anales publicados en CD-ROM).

[27] Cancela H., Rubino G., Urquhart M.E. An algorithm to compute the all-terminal reliability measure. Journal of the Indian Operational Research Society (OPSEARCH), Vol. 38, No.6, pp. 567-579, 2001.

[28] Cancela H., Rubino G., Urquhart M.E. HEIDI - Una herramienta de apoyo para el diseño de redes de comunicaciones. Technical Report Instituto de Computación 93-01, PEDECIBA Informática, Facultad de Ingeniería, UDELAR, 1993.

[29] Carpenter. B. Internet Transparency. Internet Society Request for Comments 2775. www.faqs.org/rfcs/rfc2775.html, 2000.

[30] Carpenter T., Carter R., Codfinwain M., Eiger M., Caching for Mobile Communication. ADBIS-DASFAA 2000: 37-50.

[31] Carpenter T., Carter R., Codfinwain M., Eiger M., Client-Server Caching with Expiration Timestamps. Distributed and Parallel Databases, 10(1): 5-22 (2001)

[32] Carzaniga, A., Rosenblum, D., and Wolf, A., "Design and Evaluation of a Wide-Area Event Notification Service," ACM Transactions on Computer Systems, 19(3):332-383, August 2001.

[33] Castro M., Costa M., Rowstron A. Peer-to-peer overlays: structured, unstructured, or both? Microsoft Research, Cambridge, CB3 0FB, UK. Technical Report MSR-TR-2004-73. http://www.research.microsoft.com/.

[34] Chaum, D. Untraceable Electronic Mail Return Addresses and Digital Pseudonyms, Communication of the ACM 24, 2, Feb. 1981, pp. 84-88.

[35] Chen X., Mohapatra P., Lifetime behavior and its impact on web caching, July 1999, http://citeseer.nj.nec.com/chen99lifetime.html.

[36] Chu J., Labonte K., Levine B., "Availability and locality measurements of peer-to-peer file systems," in ITCom: Scalability and Traffic Control in IP Networks. July 2002, vol. 4868 of Proceedings of SPIE, Proceedings of SPIE.

[37] Cisco, "Cisco Content Networking Architecture," http://www.cisco.com/go/cdn.

[38] Clarke, I., Sandberg, O., Wiley, B., and Hong, T. W., "Freenet: A Distributed Anonymous Information Storage and Retrieval System," in Proc. of ICSI Workshop on Design Issues in Anonymity and Unobservability, 2000.

[39] Cochinwala M., Database Performance for Next Generation Telecommunications. IODE 2001:295-298

[40] Cohen E., Kaplan H., Aging Through Cascaded Caches: Performance Issues in the Distribution of Web Content. In Proceedings of the ACM SIGCOMM 2001 Conference (SIGCOMM-01), ser. Computer Communication Review, R. Guerin, Ed., vol. 31, 4. New York: ACM Press, Aug. 27–31 2001, pp. 41–54.

[41] Cohen E., Kaplan H.. Exploiting regularities in Web traffic patterns for cache replacement. In Proc. 31st Annual ACM Symposium on Theory of Computing. ACM, 1999.

[42] Cohen E., Kaplan H., Performance Aspects of Distributed Caches Using TTL-Based Consistency. In Proceedings of ICALP'01 conference, 2001.

[43] Cohen E., Kaplan H., Prefetching the means for document transfer: A new approach for reducing Web latency. IEEE INFOCOM'00.

[44] Cohen E., Kaplan H., Proactive caching of DNS records: Addressing a performance bottleneck. 2001 Symposium on Applications and the Internet (SAINT). IEEE, 2001.

[45] Cohen E., Kaplan H., Refreshment policies for Web content caches. IEEE INFOCOM'01.

[46] Cohen E., Kaplan H., The age penalty and its effect on cache performance. USENIX Symposium on Internet Technologies and Systems (USITS). 2001.

[47] Cooper B., Crespo A., Garcia-Molina H. Protecting the pipe from maliciousus peers. Technical report, Standford University, 2002. Available at http://dbpubs.standford.edu/pub/2002-3.

[48] CORBA (Common Object Request Broker Architecture) Home Page. http://www.corba.org/

[49] Costa P., Migliavacca M., Picco G., Cugola G., Epidemic Algorithms for Reliable Content-Based PublishSubscribe. Technical report, 2003. Available at www. elet.polimi.it/picco.

[50] Couloris G.,Dollimore J., 2001. Distributed Systems. Concepts and Design. Addison Wesley.

[51] Dahlin M., Mather C., Wang R., Anderson T., Patterson D., A Quantitative Analysis of Cache Policies for Scalable Network File Systems. In Proc. of 1994 SIGMETRICS, pages 150--160, May 1994.

[52] Dan A., Towsley D., An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes. In Proceedings of the ACM SIGMETRICS, Denver, CO, 1990.

[53] DCOM: Distributed Component Object Model Technologies. http://www.microsoft.com/com/

[54] Dean J., Henzinger M. R., “Finding Related Pages in the World Wide Web,” in Proc. of the 8th World-Wide Web Conference, May 1999.

[55] Di Caro G., Dorigo M., An adaptive multi-agent routing algorithm inspired by ants behavior. Proc. PART98 - 5 th Annual Australasian Conference on Parallel and Real-Time Systems, Adelaide, Australia, September 28-29, 1998.

[56] Digital Island Homepage, http://www.digitalisland.net.

[57] Dingledine R., Freedman M.J., and Molnar D.. The free haven project: Distributed anonymous storage service. In Proceedngs of the Workshop on Design Issues in Anonymity and Unobservbility (LNCS 2009), July 2001.

[58] Direct Connect. http://www.neo-modus.com.

[59] Distributed.net Home Page. http://www.distributed.net.

[60] Doom Game Home Page http://www.idsoftware.com/games/doom/doom-final/.

[61] Druschel, P., and Rowstron, A., "PAST: A Large-Scale, Persistent Peer-to-Peer Storage Utility," in Proc. of HotOS VIII, May 2001.

[62] eBay Home Page. http://www.ebay.com/

[63] eDonkey Home Page http://www.edonkey2000.com/.

[64] Edutella Home Page. http://edutella.jxta.org.

[65] Elmagarmid A., Rusinkiewicz M., Sheth A., Management of Heterogeneous and Autonomous Database Systems. Morgan Kaufmann, 1999. (ISBN1-55860-216-X).

[66] Entropia Home Page http://www.entropia.com/.

[67] Entropy Home Page. http://entropy.stop1984.com/

[68] eMule Home Page http://www.emule-project.net/.

[69] Estrin, D., Govindan, R., Heidemann, J., and Kumar, S., "Next Century Challenges: Scalable Coordination in Sensor Networks," in Proc. of ACM MOBICOM'99, August 1999.

[70] Exodus, http://www.exodus.com.

[71] Facultad de Ingeniería de la Universidad de la República. Home page http://www.fing.edu.uy.

[72] Fan L., Cao P., Almeida J., Broder A.Z., Summary cache: A scalable wide-area web cache sharing protocol. Tech. Rep. 1361, Department of Computer Science, University of WisconsinMadison, February 1998

[73] File Rogue, File Rogue Inc. http://www.filerogue.com.

[74] Folding@Home 2001. foldingathome.stanford.edu.

[75] Foster I. Internet Computing and the emerging Grid. Nature, Dec. 7, 2000.

[76] Foster I., Kesselman C. 1999. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufman Publishers, Inc. San Francisco, California.

[77] Foster I., Kesselman C., Tuecke S.. The Anatomy of the Grid. Enabling Scalable Virtual Organizations. International Journal of Supercomputer Applications, 2001.

[78] Fourer R., Gay D., Kernighan B., AMPL: - A Modeling Language for Mathematical Programming Book. Duxbury Press / Brooks/Cole Publishing Company, 2002. Second edition, ISBN 0-534-38809-4.

[79] Freenet Protocol 1.0 Specification.
http://freenetproject.org/index.php?page=protocol/.

[80] Gabber, E., Gibbons, P., Kristol, D., Matias, Y., and Mayer, A.. Consistent, yet anonymous, Web access with LPWA. Communications of the ACM. 42, 2. February 1999. pp. 42-47.

[81] Garey M. R., Johnson D. S. Computers and intractability. A guide to the theory of NP-Completeness, Bell Laboratories, Murray, New Jersey. (1979).

[82] Genome@Home. 2001. genomeathome.stanford.edu.

[83] Gkantsidis C., Mihail M., Saberi A., "Conductance and congestion in power law graphs," in Proc. SigMetrics. ACM, 2003.

[84] Glassman S., A Caching Relay for the World Wide Web. Computer Networks and ISDN systems, First International Conference on the World-Wide Web, Elsevier Science BV. 8 pages. 1994.

[85] Globus Home Page http://www.globus.org/.

[86] Gnucleus's Home Page http://www.gnucleus.com/.

[87] GNUnet Home Page. http://www.gnu.org/software/gnunet/

[88] Gnutella Homepage, http://gnutella.wego.com/.

[89] Gnutella Protocol Specification v0.4
http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf.

[90] Gnutella hosts. http://www.gnutellahosts.com.

[91] Goelle P. Keyton-Brown K., and Mironov I. Incentives for sharing in peer-to-peer networks. In Proc. Of ACM Conference on Electronic Commerce, October 2001.

[92] Goldberg D. E. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison--Wesley, Reading MA, 1989.

[93] Goldberg D. E., Voessner S., Optimizing global-local search hybrids. See Banzhaf, Daida, Eiben, Garzon, Honavar, Jakiela, and Smith (1999), pp. 220-228, 1999.

[94] Google Home Page http://www.google.com/.

[95] Gopalan P., Karloff H., Mehta A., Mihail M., Vishnoi N., Caching with Expiration Times. To appear in Internet Mathematics. Appeared in SODA'02

[96] GRAHAM, R.L. 2001. Traditional and Non-Traditional Applications; Peer-to-Peer Networks. Lecture. www.ida.liu.se/~TDTS43/tdts43-10-peer-to-peer.pdf.

[97] Gridbus Home Page http://www.gridbus.org/.

[98] Gridforum Home Page http://www.gridforum.org/.

[99] Gritter, M., and Cheriton, D. R., “An Architecture for Content Routing Support in the Internet,” in Proc. of the 3rd USENIX Symposium on Internet Technologies and Systems, March 2001.

[100] Groove Home Page. http://www.groove.net.

[101] Grokster Home Page. http://www.grokster.com/.

[102] Gummadi K., Dunn R., Saroiu S., Gribble S., Levy H., Zahorjan J., “Measurement, Modeling and Analysis of a Peer-to-Peer File-Sharing Workload”, Proceedings of the 19th ACM Symposium of Operating Systems Principles (SOSP), Bolton Landing, NY, October 2003.

[103] Gwertzman J.,Seltzer M., World Wide Web Cache Consistency, Proc. 1996 USENIX Technical Conference, San Diego, CA, Jan 1996.

[104] Hefeeda M., Habib A., Bhargava B., Cost-profit analysis of a peer-to-peer media streaming architecture. Technical report, CERIAS TR 2002-37, Purdue University, June 2003.

[105] Heidemann, J., Silva, F., Intanagonwiwat, C., Govindan, R., Estrin, D., and Ganesan, D., “Building Efficient Wireless Sensor Networks with Low-Level Naming,” in Proc. of the 18th ACM Symposium on Operating Systems Principles, October 2001.

[106] Heinzelman, W., Kulik, J., and Balakrishnan, H., “Adaptive Protocols for Information Dissemination in Wireless Sensor Networks,” in Proc. of ACM MOBICOM'99, August 1999.

[107] Heylighen F. Principa Cybernetica Web. pespmc1.vub.ac.be/SELFORG.html, 1997.

[108] Holmedahl V., Smith B., Yang T., Cooperative Caching of Dynamic Content on a Distributed Web Server . In Proc. of Seventh IEEE International Symposium on High Performance Distributed Computing, pages 243--250, July 1998.

[109] Hughes A. S., Touch J. Cross-Domain Cache Cooperation for Small Clients. In Proceedings of NetStore ’99, Seattle, Washington, October 1999.

[110] Hughes A. S., Touch J., Expanding the scope of prefetching through inter-application cooperation. In Web Caching and Content Delivery: Proceedings of the Sixth International Web Content Caching and Distribution Workshop (WCW'01), Boston, MA, June 2001.

[111] I2P Home Page. http://www.i2p.net/

[112] IANA - Internet Assigned Numbers Authority. Home Page http://www.iana.org/.

[113] ICANN - Internet Corporation For Assigned Names and Numbers. Home Page http://www.icann.org/

[114] ICQ Home Page. http://www.icq.com/.

[115] iMesh Home Page: http://www.imesh.com/.

[116] Intel, Peer-to-peer work: The Intel(R) Philanthropic Peer-to-Peer Program http://www.intel.com/cure/.

[117] Internet Explorer. Home Page. http://www.microsoft.com/windows/ie/

[118] Ipopt (Interior Point OPTimizer, pronounced I-P-Opt) Home Page. http://projects.coin-or.org/Ipopt

[119] Iversen, V. B. "Teletraffic Engineering Handbook", Draft-version. November, 2001. www.tele.dtu.dk/teletraffic.

[120] Iyer S., Rowstron A., Druschel P., Squirrel: A decentralized peer-to-peer web cache. In Proc. 21st ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC), 2002.

[121] Jabber Home Page: http://www.jabber.com/.

[122] JAP - Java Anon Proxy Home Page: http://anon.inf.tu-dresden.de/index_en.html/.

[123] Jovanovic, M., "Scalability Issues in Large Peer-to-Peer Networks - A Case Study of Gnutella," University of Cincinnati Technical Report 2001. Available at http://www.ececs.uc.edu/~mjovanov/Research/paper.html.

[124] Jung J., Berger A. W., Balakrishnan H., Modeling TTL-based Internet caches. http://nms.lcs.mit.edu/dns/, July 2002.

[125] Jung J., Sit E., Balakrishnan H., Morris R., Dns performance and the effectiveness of caching. In Proceedings of the ACM SIGCOMM Internet Measurement Workshop '01, San Francisco, California, November 2001

[126] JXTA v1.0 Protocols Specification, 2001. SUN Microsystems. http://spec.jxta.org/v1.0/docbook/JXTAProtocols.html.

[127] KaZaA. http://www.kazaa.com.

[128] Kinddberg T. 2002. Personal Communication.

[129] Konspire Home Page. http://konspire.sourceforge.com/.

[130] Kontiki Home page. http://www.kontiki.com.
http://help.kontiki.com/enduser/group.jsp?node=1332.

[131] Krishnamurthy B., Wills C. Piggyback Cache Validation for Proxy Caches in the World Wide Web. Proceedings of the 2nd Web Caching Workshop, Boulder, CO, June 1997.

[132] Krishnamurthy B., Wills C.. Piggyback Server Invalidation for Proxy Cache Coherency. Proceedings of the WWW-7 Conference, Brisbane, Australia, pp. 185--194, April 1998.

[133] Krishnamurthy B., Wills C. Proxy cache coherency and replacement - towards a more complete picture, Proceedings of the ICDCS conference, June 1999, http://www.research.att.com/~bala/papers/ccrcp.ps.gz.

[134] Krishnamurthy B., Wills C. Study of Piggyback Cache Validation for Proxy Caches in the WWW. Proceedings of the 1997 USENIX Symposium on Internet Technology and Systems, Monterey, California, pp. 1--12, Dec 1997.

[135] Kronfol, A.Z.: FASD: A fault-tolerant, Adaptive Scalable, Distributed Search Engine. Princeton University Technical Report. (2002).
http://www.cs.princeton.edu/~akronfol/fasd/.

[136] Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R.,Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., and Zhao, B., "OceanStore: An Architecture for Global-Scale Persistent Storage, in Proc. of ASPLOS'00, November 2000.

[137] Kung, H. T., and Wu, C. H. (2002). Content Networks: Taxonomy and New Approaches, to appear in The Internet as a Large-Scale Complex System, Kihong Park and Walter Willinger (Editors), published by Oxford University Press as part of Santa Fe Institute series, 2002. www.eecs.harvard.edu/~htk/publication/2002-santa-fe-kung-wu.pdf.

[138] LANCELOT Home Page. http://www.cse.clrc.ac.uk/Activity/LANCELOT/

[139] Ledlie J., Taylor J., Serban L., Seltzer M., Self-organization in peer-to-peer systems. In 10th EW SIGOPS, September 2002.

[140] Li J., Loo B. T., Hellerstein J., Kaashoek F., Karger D., Morris R., On the Feasibility of Peer-to-Peer Web Indexing and Search. 2nd International Workshop on Peer-to-Peer Systems (IPTPS), Berkeley, CA, Feb 2003.

[141] Limewire Home Page.http://www.limewire.com/.

[142] http://www.limewire.com/index.jsp/query_cache.

[143] Litzkow M. , Livny M., Mutka M. June 1988. Condor - A Hunter of Idle Workstations. Proceedings of the 8th International Conference on Distributed Computing Systems, pages 104–111.

[144] Litzkow M. , Solomon M. 1992. Supporting Checkpointing and Process Migration outside the UNIX Kernel. Proceedings of the USENIX Winter Conference, pages 283–290.

[145] Liu C.. DNS & BIND Cookbook. ISBN 0-596-00410-9. Publcado por O'Reilly, octubre 2002.

[146] LOCKSS Home Page. http://lockss.stanford.edu/.

[147] Loo B. T., Hellerstein J. M., Huebsch R., Shenker S., Stoica I., Enhancing P2P File-Sharing with an Internet Scale Query Processor. In Proceedings of VLDB, Sep 2004.

[148] Lugdunum Home Page. http://lugdunum2k.free.fr/

[149] Lv Q., Cao P., Cohen E., Li K., and Shenker S.. Search and replication in unstructured peer-to-peer networks. In Proceedings of the 16th annual ACM International Conference on supercomputing, 2002.

[150] MacQueen J., Some methods for classification and analysis of multivariate observations. Volume 1 of Proceedings of the Fifth Berkeley Symposium on Mathematical statistics and probability, pages 281-297, Berkeley, 1967. University of California Press.

[151] Mahajan R., Castro M., Rowstron A., Controlling the cost of reliability in peer-to-peer overlays. In IPTPS, 2003.

[152] Markatos E. P., On Caching Search Engine Query Results. In the Proceedings of the 5th International Web Caching and Content Delivery Workshop, May 2000.

[153] Maymounkov P., Mazieres D. Kademlia: A peer-to-peer information system based on the XOR metric. In Proceedings of IPTPS02, Cambridge, USA, Marzo 2002.

[154] McCoy J., Mojo Nation. In httpp://www.mojonation.net/, 2001.

[155] Mersenne Gimps Home Page http://www.mersenne.org.

[156] Mithral Communications & Design Inc. http://www.mithral.com/.

[157] Mockapetris, P.V. Internet Domain Name System Standard: Domain names - concepts and facilities. Rfc-Editor Home Page, ftp://ftp.rfc-editor.org/in-notes/rfc1034.txt, 1987.

[158] Mockapetris, P.V. Internet Domain Name System Standard: Domain names - implementation and specification. Rfc-Editor Home Page, ftp://ftp.rfc-editor.org/in-notes/rfc1035.txt, 1987.

[159] Mogul J.C. Hinted caching in the web. In Proceedings of the 1996 SIGOPS European Workshop, 1996.

[160] Morpheus. http://www.musiccity.com.

[161] Mozilla Home Page. http://www.mozilla.org/

[162] Milojicic D., Kalogeraki V., Lukose R., Nagaraja K., Pruyne J., Richard B., Rollins S., Xu Z. Peer-to-Peer Computing. Techical report HPL-2002-57, HP Labs. 2002.

[163] MRTG - Multi Router Traffic Grapher. Home Page. http://people.ee.ethz.ch/~oetiker/webtools/mrtg/

[164] MUTE Home Page. http://mute-net.sourceforge.net/

[165] Napshare Home Page. http://napshare.sourceforge.net/

[166] Napster Homepage, http://www.napster.com/.

[167] Napster Messages. http://opennap.sourceforge.net/napster.txt.

[168] Nejdl W., Siberski W., Wolpers M., Schmnitz C., Routing and clustering in schema-based super peer networks. In Submitted to the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), 2003.

[169] Nelson M., Welch B., Ousterhout J., Caching in the Sprite Network File System. ACM Transactions on Computer Systems, 6(1):134--154, February 1988.

[170] NEOS Server for Optimization. Home Page. http://www-neos.mcs.anl.gov/

[171] Nicholas E., Himmelberg C., Critical Mass and Network Evolution in Telecommunications. Selected Papers from the 1994 Telecommunications Policy Research Conference, Gerard Brock (ed.), 1995.

[172] Obraczka K., Danzig P.. Finding Low-Diameter, Low Edge-Cost, Networks (1997) Way Marina del Rey, CA 90292, USA katia@isi.edu Peter Danzig University of Southern California Computer, Low Edge-Cost, Networks Katia Obraczka, Peter Danzig, usc.edu/pub/csinfo/tech-reports/papers/97-652.ps.Z.

[173] Oh-ishi T., Sakai K., Kikuma K., Kurokawa A.. Study of the Relationship between Peer-to-Peer Systems and IP Multicasting. World Telecomunications Congress 2002 (WTC 2002).

[174] Onion Tornado-Cache Home Page. http://onionnetworks.com/.

[175] OpenCola Home Page http://www.opencola.com/.

[176] Open Content Network. http://www.open-content.net.

[177] OpenNap Home Page. http://opennap.sourceforge.net/.

[178] Oram, A., ed., Peer-to-Peer: Harnessing the Power of Disruptive Technologies, Sebastopol, CA: O'Reilly & Associates, March 2001.

[179] O'Reilly Peer-to-Peer Conference Homepage, http://conferences.oreilly.com/p2p/.

[180] O'Reilly P2P http://www.openp2p.com/.

[181] Osman, I. H. An introduction to Meta-heuristics, Operational Research Tutorial Papers Series, Annal Conference OR37 - Canterbury (1995).

[182] Osman, I.H., Kelly, J.P. “Meta-heuristics : theory and applications”. Boston. Kluwer, 1996.

[183] Osokine S., Search Optimization in the Distributed Networks. Online Report http://www.grouter.net/gnutella/search.htm. 2002.

[184] Ozsu M. T., Voruganti K., Unrau R., An Asynchronous Avoidance-based Cache Consistency Algorithm for Client Caching DBMSs. In Proceedings of the 24th International Conference on Very Large Data Bases, pages 440--451, New York, NY, Aug. 1998.

[185] P2psim. A Simulator for Peer-to-Peer protocols. http://pdos.csail.mit.edu/p2psim/

[186] Packeteer's PacketShaper® Home Page
http://www.packeteer.com/products/packetshaper.cfm.

[187] Pandurangan, G., Raghavan, P., and Upfal, E. Building P2P networks with good topological properties, 2002. http://citeseer.nj.nec.com/465282.html.

[188] Pandurangan, G., Raghavan, P., and Upfal, E. Building Low-Diameter P2P Networks. In Proceedings of the 42nd Annual IEEE Symposium on the Foundations of Computer Science (FOCS) (2001).

[189] Park K., Pai V.S., Peterson L., Wang Z., CoDNS: Improving DNS Performance and Reliability via Cooperative Lookups. In Proceedings of the Sixth Symposium on Operating Systems Design and Implementation (OSDI '04). San Francisco, CA, December 2004

[190] Pastry Home Page. http://research.microsoft.com/~antr/Pastry/

[191] PeerCache de Joltid. Home Page. http://www.joltid.com/index.php/peercache

[192] Peer-to-Peer Working Group http://www.p2pwg.org/whatis/index.html.

[193] Peer-to-Peer Working Group at Internet2, Home Page, http://p2p.internet2.edu/.

[194] Peersim Peer-to-Peer Simulator Home Page. http://peersim.sourceforge.net/

[195] PENNON (PENalty method for NONlinear and semidefinite programming) Home Page. http://www2.am.uni-erlangen.de/~kocvara/pennon/

[196] Perl Home Page. http://www.perl.org/

[197] Platform's distributed and Grid computing software http://www.platform.com/ .

[198] Pointera Home Page. http://www.pointera.com/.

[199] Postel center. Home Page. http://www.postel.org/

[200] Qiu L., Padmanabham V. N., Voelker G. M., On the placement of web server replicas. In Proc. 20th IEEE INFOCOM, 2001.

[201] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S., "A Scalable Content-Addressable Network," in Proc. of ACM SIGCOMM'01, August 2001.

[202] Ratnasamy, S., Shenker, S., and Stoica, I., "Routing Algoriths for DHTs: Some Open Questions," in Proc. of the First International Workshop on Peer-to-Peer Systems, March, 2002.

[203] Reiter, M. K., Rubin, A. D.. Crowss: Anonymity for Web Transactions. ACM Transaction on Information and System Security 1, 1, November 1998, pp 66-92.

[204] Ribeiro, Celso C., Hansen, P. "Essays and surveys in metaheuristics". Boston, MA. Kluwer, 2001.

[205] Ripeanu M., Foster I., and Iamnitchi A., Mapping the Gnutella network: Properties of largescale peer-to-peer systems and implications for system design," IEEE Internet Computing Journal 6(1), 2002.

[206] Ritter, J., "Why Gnutella Can't Scale. No, Really," http://www.darkridge.com/~jpr5/.

[207] Rowstron A., Druschel P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In Proc. Of the 18th ACM Symposium on Operating Systems Principles, October 2001.

[208] Rowstron A., Druschel P. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In Proceedings of SOSP'01, 2001.

[209] Sandvine Incorporated. Home Page. http://www.sandvine.com/.

[210] Saroiu S., Gummadi K, Gribble S. "A Measurement Study of Peer-to-Peer File Sharing Systems". In Proceedings of Multimedia Computing and Networking, 2002.

[211] Scheuermann P., Shim J., Vingralek R., "WATCHMAN: A Data Warehouse Intelligent Cache Manager," Proc. VLDB Conf., Bombay, India, 1996.

[212] SeCIU - Servicio Central de Informática Universitario. Home page. http://www.rau.edu.uy/seciu/

[213] Seti@home Home Page. http://setiathome.ssl.berkeley.edu.

[214] Shields, C., Levine, B. N.. A protocol for Anonymous Communication Over the Internet. 7th ACM Conference on Computer and Communication Security (ACM CCS 2000), November 2000.

[215] Shim J., Scheuermann P., Vingralek R., Proxy cache design: Algorithms, implementation and performance, IEEE Transactions on Knowledge and Data Engineering (1999), http://www.ece.nwu.edu/~shimjh/publication/tkde98.ps.

[216] Shirky,C.2001.WhatisP2P...and what Isn't. An article published on O'Reilly Network. www.openp2p.com/lpt/a//p2p/2000/11/24/shirky1-whatisp2p.html.

[217] Sit E., A Study of Caching in the Internet Domain Name System. Masters thesis, Massachusetts Institute of Technology, 2000.

[218] Skype Home Page http://www.skype.com

[219] Sleepycat Software Inc. Home Page. http://www.sleepycat.com/

[220] SNMP - Simple Network Management Protocol. RFC1157. http://www.faqs.org/rfcs/rfc1157.html

[221] SOAP (siglas de Simple Object Access Protocol Version 1.2 Part 0: Primer. http://www.w3.org/TR/soap12-part0/

[222] Squid Web Proxy Cache Project, http://www.squid-cache.org/.

[223] Sripanidkulchai, K., "The Popularity of Gnutella Queries and Its Implications on Scalability," http://www.cs.cmu.edu/~kunwadee/research/ p2p/paper.html.

[224] Stanford Business Software's home page. http://www.sbsi-sol-optimize.com/

[225] Stoica, I., Morris, R., Karger, D., Kaashoek, F., and Balakrishnan, H., "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," in Proc. of ACM SIGCOMM'01, August 2001.

[226] Sunaga H., Takemoto M., and Iwata T. Advanced Peer-to-Peer Network Platform for Varius Services- SIONet, P2P 2002, Linkoping, Sweeden, Sept. 2002.

[227] Syverson, P. F., Goldschlag, D. M., Reed M. G.. Anonymous Connections and Onion Routing, 1997 IEEE Symposium on Security and Privacy. pp. 44-53.

[228] United Devices, Inc. http://www.ud.com (y http://www.grid.org/about/).

[229] Universal Plug and Play Forum (Microsoft) http://www.upnp.org/.

[230] Urrutia, S., Loiseau I. A new metaheuristic and its application to the Steiner Problems in graph. Tesis de Licenciatura, FCEyN, Universidad de Buenos Aires, 2001.

[231] Veytsel, A. 2001. There is no P-to-P Market... But There is a Market for P-to-P. Aberdeen Group Pre-sentation at the P2PWG, May 2001.

[232] Voelker G., Anderson E., Kimbrel T., Feeley M., Chase J., Karlin A., Levy H., Implementing Cooperative Prefetching and Caching in a Globally Managed Memory System. In Proceedings of the 1998 ACM SIGMETRICS Conference on Performance Measurement, Modeling, and Evaluation, June 1998.

[233] Voss, S., Martello, S., Osman, I.H., Roucairol, C. “Meta-heuristics : advances and trends in local search paradigms for optimization”. Norwell, Massachusetts. Kluwer, 1999.

[234] Waldman, M., Rubin, A., and Cranor, L., “Publius: A Robust, Tamper-Evident, Censorship-Resistant Web Publishing System,” in Proc. of the 9th USENIX Security Symposium, August 2000.

[235] Wang J., A Survey of Web Caching Schemes for the Internet, ACM Computer Communication Review (CCR), Vol. 29, No. 5, October 1999.

[236] Web Services Activity at W3C http://www.w3.org/2002/ws/.

[237] Wessels D. Intelligent Caching for WWW Objects, Proe. INET-95, 1995.

[238] Williams S., Abrams M., Standbridge C.R., Abdulla G., Fox E.A. Removal policies in network caches for worldwide web documents. In Proceedings of the ACM SIGCOMM Conference, pages 293–305, August 1996.

[239] Wolman A., Voelker G., Sharma N., Cardwell N., Karlin A., Levy H., On the scale and performance of cooperative Web proxy eadfiug. Prac. of the 17th ACM Symposium on Operating Systems Principles SOSP'99), December 1999.

[240] Xing Y. Caching on the Changing Web. Master Thesis. Department of Computer Science, Brown University. May, 2001.

[241] Yahoo Home Page, http://www.yahoo.com/.

[242] Yang, B., Garcia-Molina, H. “Designing a Super-Peer Network”. In: Proc. of the 19th Intl. Conf. on Data Engineering, 2003

[243] Yang B., Garcia-Molina H. Comparing hybrid peer-to-peer systems. In VLDB'2001.

[244] Zeinalipour D., Folias Y., A Quantitative Analysis of the Gnutella Network Traffic. Department of Computer Science University of California - Riverside, CA 92507, USA. Tech. Rep. at http://www.cs.ucr.edu/~csyiazti/cs204.html, 2002

[245] Zhao B. Y., Kubiatowicz J., and Joseph A.. Tapestry: An infraestructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01.1141, University of California at Berkeley, Computer Department, 2001.

[246] Ziena Optimization Inc. KNitro Home Page. http://www.ziena.com/knitro.html

[247] Zihui Ge., Figueiredo D., Jaiswal S., Kurose J., Towsley D. Z. Ge, D. R. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley, "Modeling peer-peer file sharing systems," University of Massachusetts, Dept. of Computer Science, Tech. Rep. CMPSCI 02-27, 2002.

Glosario

g

ADSL Asymmetric Digital Subscriber Line (Línea de Suscripción Asimétrica Digital). Se refiere a una tecnología de acceso de buen ancho de banda sobre los hilos de cobre del cableado telefónico convencional. Universal ADSL o G.lite, ha sido estandarizado por la ITU-TS y permite una velocidad máxima teórica de 1.5Mbps. En la actualidad otras tecnologías ADSL permiten anchos de banda cercanos a los 30Mbps.

AMPL A Modeling Language for Mathematical Programming. Lenguaje de modelado para problemas de optimización lineal y no lineal, de variables continuas o discretas. Desarrollado por los laboratorios Bell, AMPL es de fácil uso, recomendado para prototipado dada la variedad de algoritmos para la búsqueda de soluciones.

CCCP Content Class Caching Problem. Modelo para la optimización de los tiempos de expiración de clases de meta-información en los agregadores.

CCP Content Caching Problem. Modelo para la optimización de los tiempos de expiración de la meta-información en los agregadores.

CDP Content Discovery Problem (Problema del Descubrimiento de Contenido). Modelo para la optimización de la relación entre publicación y búsqueda de la meta-información en una red de contenido.

CRC Cyclical Redundancy Check. Chequeo diseñado para la detección de errores de transmisión. Un decodificador calcula el CRC de un archivo y lo compara con el generado por el codificador, si son distintos el archivo se transmitió con errores

DHT Distributed hash table (tabla de hash distribuida). Sistema descentralizado que particiona la propiedad de un espacio de claves entre nodos, y enruta mensajes de forma eficiente al nodo propietario de cada clave. Comúnmente es infraestructura para sistemas más complejos como sistemas de archivos distribuidos, P2P, DNS, etc.

DNS Domain Name System (Sistema de Nombres de Dominio). Servicio distribuido y multiplicado para localizar servidores o máquinas en la Internet. Realiza la traslación entre direcciones IPs y nombres de dominio.

FING Facultad de Ingeniería de la Universidad de la República. Institución pública de Educación Superior del Uruguay en Ingeniería.

FQ Facultad de Química de la Universidad de la República. Institución pública de Educación Superior del Uruguay en Ingeniería Química.

FTP File Transfer Protocol (Protocolo de transferencia de archivo). Primer servicio disponible en Internet para la transferencia (subida y bajada) de archivos entre un servidor y un cliente. Utiliza el puerto 21 y carece de mecanismos de encriptación en la comunicación.

HTTP Hypertext Transfer Protocol (Protocolo de transferencia de hypertexto). Protocolo que ha sido usado por los servidores World Wide Web desde su inicio en 1993, sustentando el principal servicio de Internet: la navegación.

ICANN Internet Corporation For Assigned Names and Numbers. Organización sin fines de lucro que opera a nivel internacional, responsable de asignar espacio de direcciones numéricas de protocolo de Internet (IP) y de las funciones de administración del sistema de nombres de dominio de primer nivel genéricos (gTLD) y de códigos de países (ccTLD), así como de la administración del sistema de servidores raíz.

IP Internet Protocol (Protocolo Internet). Es el estándar que regula la transmisión de datos a través de la Internet.

ISP Internet Service Provider (Proveedor de Servicios de Internet). Organización dedicada a ofrecer servicios de conectividad a Internet.

Mbps Mega bits per second (Mega bits por segundo). Medida de la cantidad de trafico en un medio de telecomunicaciones.

MD4 MD4 es un algoritmo de hash diseñado por prof. Ronald Rivest del MIT en 1990. Implementa una función de hash criptografiada para el chequeo de la integridad de los mensajes.

MP3 MPEG-1 Audio Layer-3 (Estrato de Audio 3 de MPEG-1). Sistema de compresión de sonido que permite la grabación y reproducción digital de audio con buena calidad.

MPEG Motion Picture Experts Group (Grupo de Expertos en Películas). Es una familia de estándares usada para la codificación de información audio-visual en un formato digital.

P2P Peer-to-peer (Red entre pares). Red donde se aprovecha de los recursos (almacenamiento, ciclos, contenido, presencia humana, etc.) disponible en los bordes de Internet. Debido a que acceder a estos recursos descentralizados significa operar en un entorno de conectividad inestable con impredecible variaciones de direcciones IP, estas redes deben operar fuera del sistema de DNS y tienen importante o completa autonomía respecto a servidores centrales.

QPS Queries per second (consultas por segundo). Medida de cantidad de tráfico que es manejado por un servidor de consultas en un momento dado.

SeCIU Servicio Central de Informática Universitario de la República Oriental del Uruguay. Organización que instrumenta las políticas informáticas de la Universidad y controla el cumplimiento de las mismas. Además ejecuta las actividades de desarrollo, adquisición y mantenimiento de los sistemas informáticos necesarios para la gestión universitaria.

SNMP Simple Network Management Protocol (Protocolo para el Manejo de la Red). Protocolo estándar para la administración de redes IP. Trabaja mandando mensajes, llamados Protocol Data Units (PDUs) a diferentes partes de la red. Los elementos de red que implementan SNMP, se llaman agentes, y guardan información sobre si mismos en una base de datos llamada Management Information Bases (MIBs), la cual responden frente a consultas SNMP. En la actualidad, prácticamente todos los elementos de estas redes (sistemas operativos, routers, switches, módems cable o ADSL módem, firewalls, etc.) ofrecen este servicio.

TCP Transmission Control Protocol (Protocolo de Control de Transmisión). Forma de comunicación básica de Internet la cual hace posible que cualquier tipo de información (mensajes, gráficos o audio) viaje en forma de paquetes sin que estos se pierdan y siguiendo cualquier ruta posible.

TTL Time to Live (Tiempo de Vida). El tiempo de vida es utilizado por distintos protocolos para determinar la validez de una información. Comúnmente utilizado en algoritmos de enrutamiento y en el sistema DNS.

UDP User Datagram Protocol (Protocolo de Datagramas de Usuario). Protocolo que no pide confirmación de la validez de los paquetes enviados por la computadora emisora. Aplicaciones comunes que utilizan UDP son DNS, transmisión de sonido y vídeo, y juegos en línea.

UUHash UUHash es un algoritmo de hash utilizado por los clientes de la red FastTrack. Aplicable a archivos de gran tamaño debido a su poca necesidad de computo. Solo utiliza parte de la información del archive para su resultado, por tanto presenta algunas vulnerabilidades bien conocidas.

WWW World Wide Web (WWW). Sistema de información distribuido, basado en hipertexto (HTTP), cuya función es buscar y tener acceso a documentos a través de la red mediante un navegador Web. Creada por Tim Berners-Lee (en el CERN, Suiza) a principios de los años 90. Actualmente la información puede ser de cualquier formato (texto, gráfico, audio, imagen fija o en movimiento).

XML eXtensible Markup Language (Lenguaje Extensible de Marcado). Lenguaje desarrollado por el W3 Consortium para permitir la descripción de información contenida en la Internet a través de estándares y formatos comunes de texto. El lenguaje XML es similar al HTML pero no es una extensión ni un componente de éste.

Apéndice A: Método de Agregación del Contenido en Clases

A

En este apéndice se detalla el método de consolidación de contenidos en clases de contenido.

En las redes de contenido reales puede ser muy costoso mantener el conocimiento de los parámetros f_c, λ_c y μ_c para todo contenido c, debido a que manejan una cantidad suficientemente grande de contenidos. Por otro lado, cuando hay demasiados contenidos, la formulación del problema *CCP* es inconveniente y de difícil tratamiento si se utilizan los métodos tradicionales de resolución para problemas de optimización no lineales (debido a que cada contenido agrega una variable de decisión al problema).
Por estas razones se realiza una generalización al modelo, el problema *CCCP*, donde se considera útil agrupar los contenidos en clases de contenido, formándose una clase de contenido por todos aquellos contenidos que presentan parámetros f_c, λ_c y μ_c lo suficientemente similares como para poder tratarlos como idénticos[16].

Supóngase que los contenidos $c \in C$ se agrupan en K conjuntos (**clases**), donde dos contenidos pertenecen a la misma clase si y solo si sus parámetros (f, λ y μ) son lo suficientemente similares. No existe un criterio único de similitud entre contenidos, las técnicas que estudian esto se conocen como métodos de conglomerado (*clustering*) y formalizan la similitud con alguna función de distancia entre los contenidos.
Se ensayaron distintos métodos de conglomerado como el *k-Means* [150], optándose por un método sencillo y eficiente que se describe a continuación en este apéndice.

Primero es necesario recordar básicamente los mecanismos utilizados para obtener las instancias del *CCP* en los casos de estudio:

- ***P2P***: para hallar las instancias del *CCP* se impuso una dependencia lineal entre λ_c y f_c. Además de que $\mu_c = 1$ es constante para todos los contenidos.
- ***DNS***: en este caso se fijaron los valores λ_c y μ_c constantes e iguales para todos los contenidos.

Por tanto para los dos casos de estudio la única dimensión que presenta información para el método de conglomerado es la tasa de solicitud f_c y en consecuencia debe

[16] Si una red tiene millones de contenidos entonces el problema *CCP* tiene la misma cantidad de variables de decisión (millones). Si sus contenidos son agrupables en clases, una instancia del *CCCP* equivalente reduce la cantidad de variables del millón a la cantidad de clases elegidas (en este trabajo se muestra que para los casos de estudio es suficiente con un ciento de variables de decisión).

ser eficiente un método que utilice únicamente este parámetro (λ_c y μ_c presentan información completamente redundante para cualquier mecanismo de conglomerado).

Basándose en esta simplificación de los casos de estudio, se utiliza un método donde todas las clases tienen la misma cantidad de solicitudes (las solicitudes de una clase son la suma de las solicitudes de los contenidos que la integran), incluyendo en cada clase aquellos contenidos con mayor similitud en la frecuencia de solicitud. Es decir, hay clases de gran tamaño formada por contenidos relativamente poco solicitados, y clases cada vez más pequeñas con contenido más solicitado.
Es importante no confundir las solicitudes de una clase (suma de las solicitudes de los contenidos que la integran) con el valor de f para la clase que es el promedio del valor de las solicitudes de los contenidos de la clase[17].

La Tabla 36 muestra el pseudo-código del algoritmo de conglomerado. En primer lugar se calcula el valor de las solicitudes que debe tener cada clase (`acumF`) y se ordena el vector de solicitudes de contenido `f` de forma decreciente.
Comenzando por el contenido de mayor solicitud este se agrega a la primera clase, luego el segundo, y así sucesivamente hasta alcanzar la cantidad de solicitudes de la clase (valor más cercano superiormente a `acumF`). Se prosigue con el resto de los contenidos en la segunda clase, repitiendo el proceso hasta completar todas las clases.

```
function cccp = clustering(ccp, K, C)
% f in decreasing order
% C number of contents
% K number of classes

acumF = sum(ccp.f([1,..,C]))/K;

start = 1;
for k = 1 to K
    finish = start;
    while ( sum(ccp.f([start,.., finish])) < acumF) & (finish <= C) )
        finish = finish + 1;
    end

    cccp.l(k)      = finish - start + 1;
    cccp.f(k)      = sum(ccp.f([start,.., finish]))/cccp.l(k);
    cccp.lamda(k)  = sum(ccp.lamda([start,.., finish]))/cccp.l(k);
    cccp.mu(k)     = sum(ccp.mu([start,..,finish]))/cccp.l(k);

    start = finish + 1;
end
```

Tabla 36. Pseudo-código del método de consolidación de clases

[17] Todas las clases tienen un valor similar $f_k.l_k \approx cte \quad \forall k \in K$.

Apéndice B: Instancias de los Casos de Estudio

B

Este apéndice incluye algunas de las instancias utilizadas en los casos de estudio del capítulo 5.

B.1 P2P - File Sharing System

Se generaron 10 instancias del problema *CCP*, y a partir de cada una de ellas se generaron 5 instancias del *CCCP* (una por cada cantidad de clases considerada), siendo un total de 50 instancias evaluadas.

B.1.1 Instancias del problema CCCP

A continuación se presentan las 5 instancias de distinta cantidad de clases, generadas a partir de una misma instancia del problema *CCP*.

```
param K        := 2;

param alphaS   := 100.00000000;
param alphaB   := 300.00000000;
param betaS    := 94.00000000;
param betaB    := 291.40000000;
param BWin     := 921600000.00000000;
param BWout    := 460800000.00000000;

param f :=
         1 22.53685618
         2 0.01513864;
param lamda :=
         1 200.00000000
         2 4.02619446;
param mu :=
         1 1.00000000
         2 1.00000000;
param l :=
         1 590.00000000
         2 878101.00000000;
```

Tabla 37. Instancia del CCCP (2 clases) en AMPL. cccp-p2p-K002.dat

```
param K        := 8;

param alphaS   := 100.00000000;
param alphaB   := 300.00000000;
param betaS    := 94.00000000;
param betaB    := 291.40000000;
param BWin     := 921600000.00000000;
param BWout    := 460800000.00000000;

param f :=
         1 1000.00000000
         2 216.15859672
```

```
         3 32.96348528
         4 5.12742726
         5 0.81727228
         6 0.14024312
         7 0.02464053
         8 0.00447485;
param lamda :=
         1 200.00000000
         2 200.00000000
         3 200.00000000
         4 200.00000000
         5 171.38934707
         6 41.02318819
         7 7.20772083
         8 1.30896076;
param mu :=
         1 1.00000000
         2 1.00000000
         3 1.00000000
         4 1.00000000
         5 1.00000000
         6 1.00000000
         7 1.00000000
         8 1.00000000;
param l :=
         1 4.00000000
         2 15.00000000
         3 98.00000000
         4 629.00000000
         5 3944.00000000
         6 22981.00000000
         7 130797.00000000
         8 720223.00000000;
```

Tabla 38. Instancia del CCCP (8 clases) en AMPL. cccp-p2p-K008.dat

```
param K        := 16;

param alphaS   := 100.00000000;
param alphaB   := 300.00000000;
param betaS    := 94.00000000;
param betaB    := 291.40000000;
param BWin     := 921600000.00000000;
param BWout    := 460800000.00000000;

param f :=
         1 1000.00000000
         2 1000.00000000
         3 479.85748244
         4 108.05207506
         5 45.69682011
         6 20.90945385
         7 7.99243620
         8 3.11083107
         9 1.27095518
        10 0.52349239
        11 0.21990832
        12 0.09345023
        13 0.03957382
        14 0.01693693
        15 0.00731336
        16 0.00317740;
param lamda :=
         1 200.00000000
         2 200.00000000
         3 200.00000000
         4 200.00000000
         5 200.00000000
         6 200.00000000
         7 200.00000000
         8 200.00000000
         9 200.00000000
        10 149.44596626
        11 64.32643638
```

```
        12 27.33557540
        13 11.57592815
        14 4.95430221
        15 2.13926733
        16 0.92943700;
param mu :=
         1 1.00000000
         2 1.00000000
         3 1.00000000
         4 1.00000000
         5 1.00000000
         6 1.00000000
         7 1.00000000
         8 1.00000000
         9 1.00000000
        10 1.00000000
        11 1.00000000
        12 1.00000000
        13 1.00000000
        14 1.00000000
        15 1.00000000
        16 1.00000000;
param l :=
         1 2.00000000
         2 2.00000000
         3 4.00000000
         4 15.00000000
         5 35.00000000
         6 76.00000000
         7 199.00000000
         8 510.00000000
         9 1248.00000000
        10 3029.00000000
        11 7210.00000000
        12 16965.00000000
        13 40062.00000000
        14 93605.00000000
        15 216778.00000000
        16 498951.00000000;
```

Tabla 39. Instancia del CCCP (16 clases) en AMPL. cccp- p2p-K016.dat

```
param K        := 32;

param alphaS   := 100.00000000;
param alphaB   := 300.00000000;
param betaS    := 94.00000000;
param betaB    := 291.40000000;
param BWin     := 921600000.00000000;
param BWout    := 460800000.00000000;

param f :=
         1 1000.00000000
         2 1000.00000000
         3 1000.00000000
         4 1000.00000000
         5 520.68209388
         6 439.03287100
         7 160.07769375
         8 82.03926571
         9 57.29910065
        10 37.96196642
        11 25.87729866
        12 17.40379923
        13 10.25258815
        14 6.44451758
        15 4.00621173
        16 2.50417805
        17 1.61129941
        18 1.03491490
        19 0.66541261
        20 0.42622249
        21 0.27674292
        22 0.18060754
```

```
        23 0.11793535
        24 0.07673251
        25 0.04974309
        26 0.03264603
        27 0.02134275
        28 0.01397504
        29 0.00919977
        30 0.00605282
        31 0.00399394
        32 0.00263596;
param lamda :=
         1 200.00000000
         2 200.00000000
         3 200.00000000
         4 200.00000000
         5 200.00000000
         6 200.00000000
         7 200.00000000
         8 200.00000000
         9 200.00000000
        10 200.00000000
        11 200.00000000
        12 200.00000000
        13 200.00000000
        14 200.00000000
        15 200.00000000
        16 200.00000000
        17 200.00000000
        18 200.00000000
        19 185.94597306
        20 124.67638142
        21 80.95139664
        22 52.83037580
        23 34.49783500
        24 22.44539273
        25 14.55058953
        26 9.54944519
        27 6.24307142
        28 4.08790445
        29 2.69106790
        30 1.77053982
        31 1.16828690
        32 0.77105724;
param mu :=
         1 1.00000000
         2 1.00000000
         3 1.00000000
         4 1.00000000
         5 1.00000000
         6 1.00000000
         7 1.00000000
         8 1.00000000
         9 1.00000000
        10 1.00000000
        11 1.00000000
        12 1.00000000
        13 1.00000000
        14 1.00000000
        15 1.00000000
        16 1.00000000
        17 1.00000000
        18 1.00000000
        19 1.00000000
        20 1.00000000
        21 1.00000000
        22 1.00000000
        23 1.00000000
        24 1.00000000
        25 1.00000000
        26 1.00000000
        27 1.00000000
        28 1.00000000
        29 1.00000000
        30 1.00000000
```

```
        31 1.00000000
        32 1.00000000;
param l :=
         1 1.00000000
         2 1.00000000
         3 1.00000000
         4 1.00000000
         5 2.00000000
         6 2.00000000
         7 5.00000000
         8 10.00000000
         9 14.00000000
        10 21.00000000
        11 31.00000000
        12 46.00000000
        13 78.00000000
        14 123.00000000
        15 198.00000000
        16 317.00000000
        17 492.00000000
        18 765.00000000
        19 1190.00000000
        20 1858.00000000
        21 2861.00000000
        22 4384.00000000
        23 6713.00000000
        24 10317.00000000
        25 15915.00000000
        26 24249.00000000
        27 37092.00000000
        28 56646.00000000
        29 86049.00000000
        30 130786.00000000
        31 198207.00000000
        32 300316.00000000;
```

Tabla 40. Instancia del CCCP (32 clases) en AMPL. cccp- p2p-K032.dat

```
param K        := 128;

param alphaS   := 100.00000000;
param alphaB   := 300.00000000;
param betaS    := 94.00000000;
param betaB    := 291.40000000;
param BWin     := 921600000.00000000;
param BWout    := 460800000.00000000;

param f :=
         1 1000.00000000
         2 1000.00000000
         3 1000.00000000
         4 1000.00000000
         5 525.51414572
         6 515.85004203
         7 491.59647801
         8 386.46926398
         9 202.55057419
        10 175.19234721
        11 174.15536215
        12 124.24509261
        13 96.07998346
        14 85.78181251
        15 78.60191435
        16 73.62110736
        17 66.06227289
        18 60.78043182
        19 56.89859283
        20 51.00451546
        21 45.65190586
        22 40.74985514
        23 37.53527000
        24 34.47303656
        25 32.12736374
        26 29.07302086
```

```
 27 25.87386272
 28 24.04515690
 29 22.91364136
 30 21.25312878
 31 19.09218255
 32 17.31321503
 33 15.52132629
 34 14.01603523
 35 12.27198766
 36 10.94142421
 37 9.78267409
 38 8.74009597
 39 7.87343296
 40 7.22406197
 41 6.63469722
 42 5.94182707
 43 5.32542787
 44 4.77351731
 45 4.31330588
 46 3.96124595
 47 3.56210356
 48 3.19248139
 49 2.90019575
 50 2.65788255
 51 2.38165305
 52 2.15545327
 53 1.96486626
 54 1.80440735
 55 1.64981885
 56 1.49567558
 57 1.36656363
 58 1.23805916
 59 1.12084960
 60 1.02590416
 61 0.94052208
 62 0.85821728
 63 0.78065731
 64 0.70940442
 65 0.64564575
 66 0.58372292
 67 0.53120093
 68 0.48358215
 69 0.44148832
 70 0.40322438
 71 0.36797029
 72 0.33530878
 73 0.30573895
 74 0.27965745
 75 0.25594566
 76 0.23356942
 77 0.21396363
 78 0.19506542
 79 0.17800907
 80 0.16264435
 81 0.14909424
 82 0.13638013
 83 0.12443994
 84 0.11349299
 85 0.10371246
 86 0.09465563
 87 0.08663626
 88 0.07915488
 89 0.07227324
 90 0.06571959
 91 0.05991720
 92 0.05473279
 93 0.05000381
 94 0.04566857
 95 0.04176996
 96 0.03825248
 97 0.03502933
 98 0.03207387
 99 0.02933778
100 0.02680667
```

```
       101 0.02446192
       102 0.02238675
       103 0.02047344
       104 0.01870196
       105 0.01710010
       106 0.01564398
       107 0.01429996
       108 0.01308381
       109 0.01198489
       110 0.01097526
       111 0.01005438
       112 0.00920707
       113 0.00842313
       114 0.00770305
       115 0.00705338
       116 0.00645908
       117 0.00591530
       118 0.00541540
       119 0.00496025
       120 0.00454400
       121 0.00416283
       122 0.00381345
       123 0.00349460
       124 0.00319995
       125 0.00293076
       126 0.00268457
       127 0.00245966
       128 0.00225327;
param lamda :=
         1 200.00000000
         2 200.00000000
         3 200.00000000
         4 200.00000000
         5 200.00000000
         6 200.00000000
         7 200.00000000
         8 200.00000000
         9 200.00000000
        10 200.00000000
        11 200.00000000
        12 200.00000000
        13 200.00000000
        14 200.00000000
        15 200.00000000
        16 200.00000000
        17 200.00000000
        18 200.00000000
        19 200.00000000
        20 200.00000000
        21 200.00000000
        22 200.00000000
        23 200.00000000
        24 200.00000000
        25 200.00000000
        26 200.00000000
        27 200.00000000
        28 200.00000000
        29 200.00000000
        30 200.00000000
        31 200.00000000
        32 200.00000000
        33 200.00000000
        34 200.00000000
        35 200.00000000
        36 200.00000000
        37 200.00000000
        38 200.00000000
        39 200.00000000
        40 200.00000000
        41 200.00000000
        42 200.00000000
        43 200.00000000
        44 200.00000000
        45 200.00000000
```

```
 46 200.00000000
 47 200.00000000
 48 200.00000000
 49 200.00000000
 50 200.00000000
 51 200.00000000
 52 200.00000000
 53 200.00000000
 54 200.00000000
 55 200.00000000
 56 200.00000000
 57 200.00000000
 58 200.00000000
 59 200.00000000
 60 200.00000000
 61 200.00000000
 62 200.00000000
 63 200.00000000
 64 199.93523743
 65 188.86093059
 66 170.74758896
 67 155.38412945
 68 141.45493097
 69 129.14186480
 70 117.94909432
 71 107.63675188
 72 98.08277887
 73 89.43316559
 74 81.80394047
 75 74.86789199
 76 68.32250917
 77 62.58752517
 78 57.05952017
 79 52.07028707
 80 47.57587714
 81 43.61226932
 82 39.89320576
 83 36.40052159
 84 33.19837719
 85 30.33742718
 86 27.68817256
 87 25.34238677
 88 23.15397207
 89 21.14099211
 90 19.22395101
 91 17.52666841
 92 16.01015142
 93 14.62685393
 94 13.35873292
 95 12.21833159
 96 11.18941655
 97 10.24659677
 98 9.38208148
 99 8.58173572
100 7.84134667
101 7.15547440
102 6.54845428
103 5.98878305
104 5.47059873
105 5.00203138
106 4.57609545
107 4.18294897
108 3.82720905
109 3.50575666
110 3.21042456
111 2.94105565
112 2.69320274
113 2.46388964
114 2.25325498
115 2.06321801
116 1.88937660
117 1.73031159
118 1.58408515
119 1.45094613
```

```
       120 1.32918657
       121 1.21768982
       122 1.11549018
       123 1.02222199
       124 0.93603397
       125 0.85729042
       126 0.78527720
       127 0.71948738
       128 0.65911563;
param mu :=
         1 1.00000000
         2 1.00000000
         3 1.00000000
         4 1.00000000
         5 1.00000000
         6 1.00000000
         7 1.00000000
         8 1.00000000
         9 1.00000000
        10 1.00000000
        11 1.00000000
        12 1.00000000
        13 1.00000000
        14 1.00000000
        15 1.00000000
        16 1.00000000
        17 1.00000000
        18 1.00000000
        19 1.00000000
        20 1.00000000
        21 1.00000000
        22 1.00000000
        23 1.00000000
        24 1.00000000
        25 1.00000000
        26 1.00000000
        27 1.00000000
        28 1.00000000
        29 1.00000000
        30 1.00000000
        31 1.00000000
        32 1.00000000
        33 1.00000000
        34 1.00000000
        35 1.00000000
        36 1.00000000
        37 1.00000000
        38 1.00000000
        39 1.00000000
        40 1.00000000
        41 1.00000000
        42 1.00000000
        43 1.00000000
        44 1.00000000
        45 1.00000000
        46 1.00000000
        47 1.00000000
        48 1.00000000
        49 1.00000000
        50 1.00000000
        51 1.00000000
        52 1.00000000
        53 1.00000000
        54 1.00000000
        55 1.00000000
        56 1.00000000
        57 1.00000000
        58 1.00000000
        59 1.00000000
        60 1.00000000
        61 1.00000000
        62 1.00000000
        63 1.00000000
        64 1.00000000
```

```
        65 1.00000000
        66 1.00000000
        67 1.00000000
        68 1.00000000
        69 1.00000000
        70 1.00000000
        71 1.00000000
        72 1.00000000
        73 1.00000000
        74 1.00000000
        75 1.00000000
        76 1.00000000
        77 1.00000000
        78 1.00000000
        79 1.00000000
        80 1.00000000
        81 1.00000000
        82 1.00000000
        83 1.00000000
        84 1.00000000
        85 1.00000000
        86 1.00000000
        87 1.00000000
        88 1.00000000
        89 1.00000000
        90 1.00000000
        91 1.00000000
        92 1.00000000
        93 1.00000000
        94 1.00000000
        95 1.00000000
        96 1.00000000
        97 1.00000000
        98 1.00000000
        99 1.00000000
       100 1.00000000
       101 1.00000000
       102 1.00000000
       103 1.00000000
       104 1.00000000
       105 1.00000000
       106 1.00000000
       107 1.00000000
       108 1.00000000
       109 1.00000000
       110 1.00000000
       111 1.00000000
       112 1.00000000
       113 1.00000000
       114 1.00000000
       115 1.00000000
       116 1.00000000
       117 1.00000000
       118 1.00000000
       119 1.00000000
       120 1.00000000
       121 1.00000000
       122 1.00000000
       123 1.00000000
       124 1.00000000
       125 1.00000000
       126 1.00000000
       127 1.00000000
       128 1.00000000;
param l :=
         1 1.00000000
         2 1.00000000
         3 1.00000000
         4 1.00000000
         5 1.00000000
         6 1.00000000
         7 1.00000000
         8 1.00000000
         9 1.00000000
```

```
        10 1.00000000
        11 1.00000000
        12 2.00000000
        13 2.00000000
        14 2.00000000
        15 3.00000000
        16 3.00000000
        17 3.00000000
        18 3.00000000
        19 3.00000000
        20 4.00000000
        21 4.00000000
        22 5.00000000
        23 5.00000000
        24 5.00000000
        25 6.00000000
        26 6.00000000
        27 7.00000000
        28 7.00000000
        29 8.00000000
        30 8.00000000
        31 9.00000000
        32 10.00000000
        33 11.00000000
        34 12.00000000
        35 14.00000000
        36 16.00000000
        37 18.00000000
        38 20.00000000
        39 22.00000000
        40 24.00000000
        41 26.00000000
        42 29.00000000
        43 32.00000000
        44 36.00000000
        45 39.00000000
        46 43.00000000
        47 47.00000000
        48 53.00000000
        49 58.00000000
        50 63.00000000
        51 71.00000000
        52 78.00000000
        53 86.00000000
        54 93.00000000
        55 102.00000000
        56 112.00000000
        57 123.00000000
        58 136.00000000
        59 150.00000000
        60 163.00000000
        61 178.00000000
        62 195.00000000
        63 215.00000000
        64 236.00000000
        65 259.00000000
        66 287.00000000
        67 315.00000000
        68 346.00000000
        69 379.00000000
        70 415.00000000
        71 455.00000000
        72 499.00000000
        73 547.00000000
        74 598.00000000
        75 654.00000000
        76 716.00000000
        77 782.00000000
        78 857.00000000
        79 940.00000000
        80 1028.00000000
        81 1122.00000000
        82 1226.00000000
        83 1344.00000000
```

```
         84 1473.00000000
         85 1612.00000000
         86 1766.00000000
         87 1930.00000000
         88 2112.00000000
         89 2313.00000000
         90 2544.00000000
         91 2790.00000000
         92 3054.00000000
         93 3343.00000000
         94 3660.00000000
         95 4002.00000000
         96 4370.00000000
         97 4772.00000000
         98 5212.00000000
         99 5698.00000000
        100 6235.00000000
        101 6833.00000000
        102 7466.00000000
        103 8164.00000000
        104 8937.00000000
        105 9775.00000000
        106 10684.00000000
        107 11688.00000000
        108 12775.00000000
        109 13946.00000000
        110 15229.00000000
        111 16624.00000000
        112 18153.00000000
        113 19843.00000000
        114 21698.00000000
        115 23696.00000000
        116 25876.00000000
        117 28255.00000000
        118 30863.00000000
        119 33695.00000000
        120 36782.00000000
        121 40149.00000000
        122 43828.00000000
        123 47826.00000000
        124 52230.00000000
        125 57027.00000000
        126 62256.00000000
        127 67949.00000000
        128 74172.00000000;
```

Tabla 41. Instancia del CCCP (128 clases) en AMPL. cccp- p2p-K128.dat

B.1.2 Soluciones a las Instancias del problema CCCP

Las soluciones de las instancias anteriores halladas mediante el método descrito en la sección 5.1.3.

```
Time = 0.00 seg
epsilon = 1
bitsIn = 843470460
bitsOut = 279035240

solution d :=
         1 0.00000100
         2 0.00000100;
```

Tabla 42. Solución de la instancia del CCCP (2 clases) en AMPL. cccp-p2p-K002.dat

```
Time = 0.06 seg
epsilon = 0.996806
bitsIn = 921600460
bitsOut = 351826810

solution d :=
         1 0.00016505
```

```
         2 0.00075936
         3 0.00497302
         4 0.03196430
         5 0.20052600
         6 1.16831000
         7 6.64391000
         8 36.43360000;
```

Tabla 43. Solución de la instancia del CCCP (8 clases) en AMPL. cccp-p2p-K008.dat

```
Time = 0.33 seg
epsilon = 0.996656
bitsIn = 921600460
bitsOut = 357952740

solution d :=
         1 0.00018695
         2 0.00018695
         3 0.00038830
         4 0.00172036
         5 0.00406624
         6 0.00888520
         7 0.02324310
         8 0.05971460
         9 0.14615600
        10 0.35482900
        11 0.84458700
        12 1.98716000
        13 0.63063300
        14 1.47309000
        15 3.41242000
        16 7.85214000;
```

Tabla 44. Solución de la instancia del CCCP (16 clases) en AMPL. cccp-p2p-K016.dat

```
Time = 0.04 seg
epsilon = 0.999461
bitsIn = 921600460
bitsOut = 359960130

solution d :=
         1 0.00027019
         2 0.00025992
         3 0.00025969
         4 0.00025953
         5 0.00049781
         6 0.00059010
         7 0.00161234
         8 0.00313917
         9 0.00449347
        10 0.00677694
        11 0.00993479
        12 0.01476690
        13 0.02511370
        14 0.03840770
        15 0.05924010
        16 0.01370610
        17 0.02130140
        18 0.03316180
        19 0.05157580
        20 0.08051740
        21 0.12400600
        22 0.19001500
        23 0.29113400
        24 0.00000100
        25 0.00000100
        26 0.00000100
        27 0.00000100
        28 0.00000100
        29 0.00000100
        30 0.00000100
        31 0.00000100
        32 0.00000100;
```

Tabla 45. Solución de la instancia del CCCP (32 clases) en AMPL. cccp-p2p-K032.dat

```
Time = 0.4 seg
epsilon = 0.999897
bitsIn = 921600460
bitsOut = 360526000

solution d :=
         1 0.00047198
         2 0.00047144
         3 0.00047145
         4 0.00047144
         5 0.00089575
         6 0.00091258
         7 0.00095757
         8 0.00121620
         9 0.00231843
        10 0.00268721
        11 0.00270757
        12 0.00378375
        13 0.00486387
        14 0.00551389
        15 0.00596971
        16 0.00633982
        17 0.00736622
        18 0.00794260
        19 0.00154868
        20 0.00878030
        21 0.00192988
        22 0.00216280
        23 0.00234934
        24 0.00256028
        25 0.00274616
        26 0.00303586
        27 0.00341068
        28 0.00366739
        29 0.00384976
        30 0.00415066
        31 0.00461851
        32 0.00507008
        33 0.00568093
        34 0.00631097
        35 0.00712434
        36 0.00811475
        37 0.00057476
        38 0.00064321
        39 0.00071389
        40 0.00077798
        41 0.00084700
        42 0.00094565
        43 0.00105499
        44 0.00117685
        45 0.00130230
        46 0.00141785
        47 0.00131057
        48 0.00000989
        49 0.00001079
        50 0.00001168
        51 0.00001292
        52 0.00000100
        53 0.00000100
        54 0.00000100
        55 0.00000100
        56 0.00000100
        57 0.00000100
        58 0.00000100
        59 0.00000100
        60 0.00000100
        61 0.00000100
        62 0.00000100
        63 0.00000100
        64 0.00000100
        65 0.00000100
        66 0.00000100
        67 0.00000100
        68 0.00000100
```

```
 69 0.00000100
 70 0.00000100
 71 0.00000100
 72 0.00000100
 73 0.00000100
 74 0.00000100
 75 0.00000100
 76 0.00000100
 77 0.00000100
 78 0.00000100
 79 0.00000100
 80 0.00000100
 81 0.00000100
 82 0.00000100
 83 0.00000100
 84 0.00000100
 85 0.00000100
 86 0.00000100
 87 0.00000100
 88 0.00000100
 89 0.00000100
 90 0.00000100
 91 0.00000100
 92 0.00000100
 93 0.00000100
 94 0.00000100
 95 0.00000100
 96 0.00000100
 97 0.00000100
 98 0.00000100
 99 0.00000100
100 0.00000100
101 0.00000100
102 0.00000100
103 0.00000100
104 0.00000100
105 0.00000100
106 0.00000100
107 0.00000100
108 0.00000100
109 0.00000100
110 0.00000100
111 0.00000100
112 0.00000100
113 0.00000100
114 0.00000100
115 0.00000100
116 0.00000100
117 0.00000100
118 0.00000100
119 0.00000100
120 0.00000100
121 0.00000100
122 0.00000100
123 0.00000100
124 0.00000100
125 0.00000100
126 0.00000100
127 0.00000100
128 0.00000100;
```

Tabla 46. Solution de la instancia del CCCP (128 clases) en AMPL. cccp-p2p-K128.dat

B.1.3 Gráficas de Todas las Instancias

Por ser impractico presentar todas las instancias de forma escrita, en esta sección se presentan de forma gráfica sus principales características (la frecuencia de solicitud y la solución de tiempo de expiración para las diez instancias de cada clase).

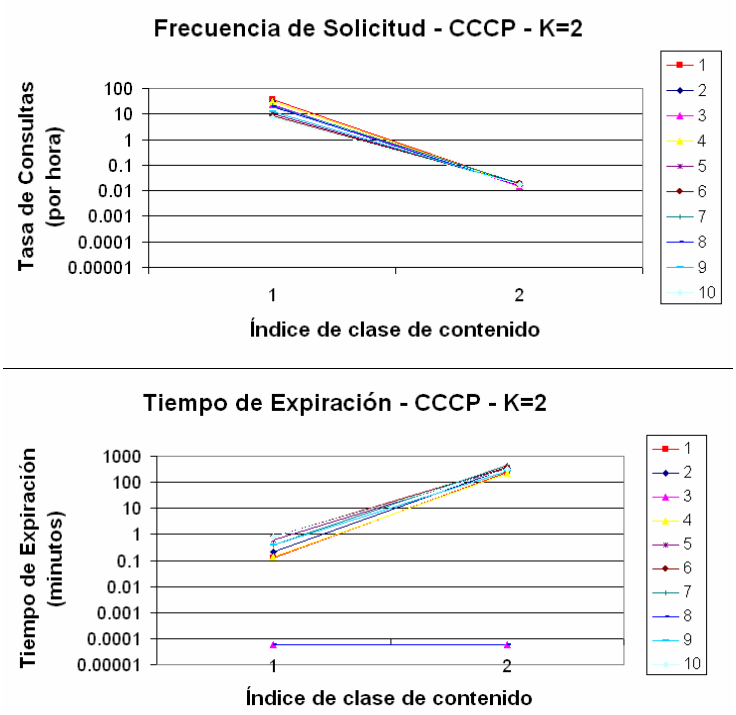


Ilustración 26. Tasa de solicitud y Tiempos de expiración de cada clase para las 10 instancias del problema CCCP de 2 clases.

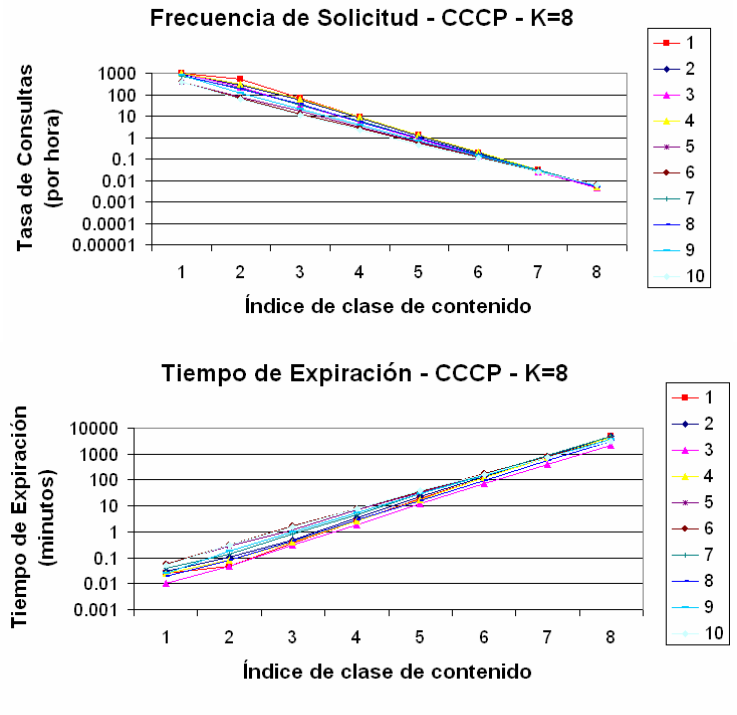


Ilustración 27. Tasa de solicitud y Tiempos de expiración de cada clase para las 10 instancias del problema CCCP de 8 clases.

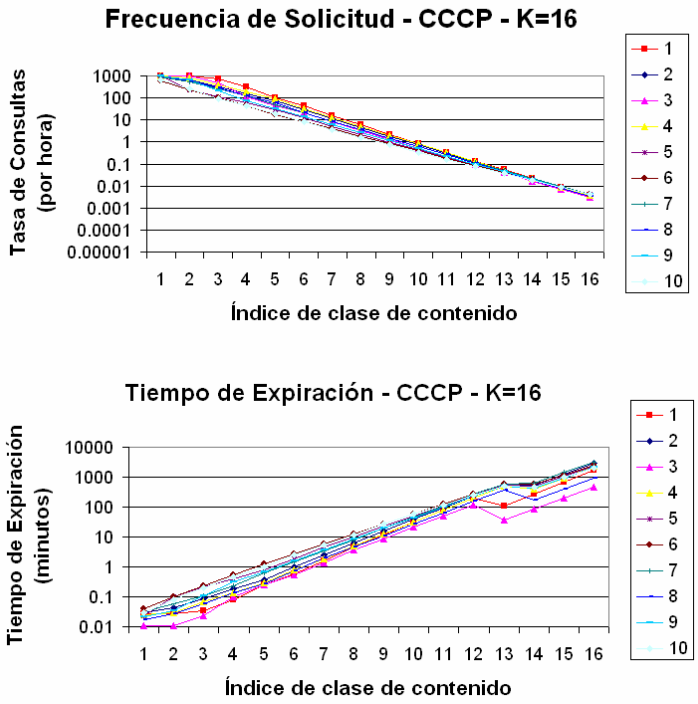


Ilustración 28. Tasa de solicitud y Tiempos de expiración de cada clase para las 10 instancias del problema CCCP de 16 clases.

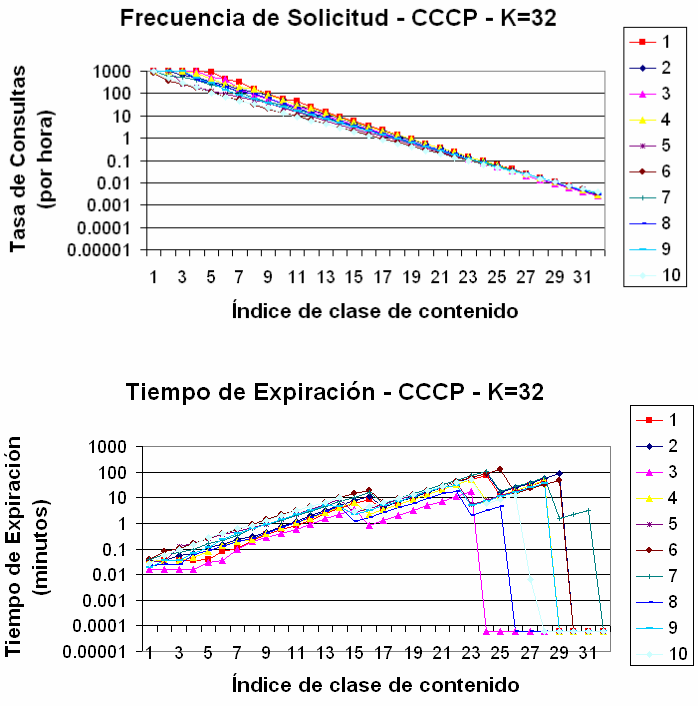


Ilustración 29. Tasa de solicitud y Tiempos de expiración de cada clase para las 10 instancias del problema CCCP de 32 clases.

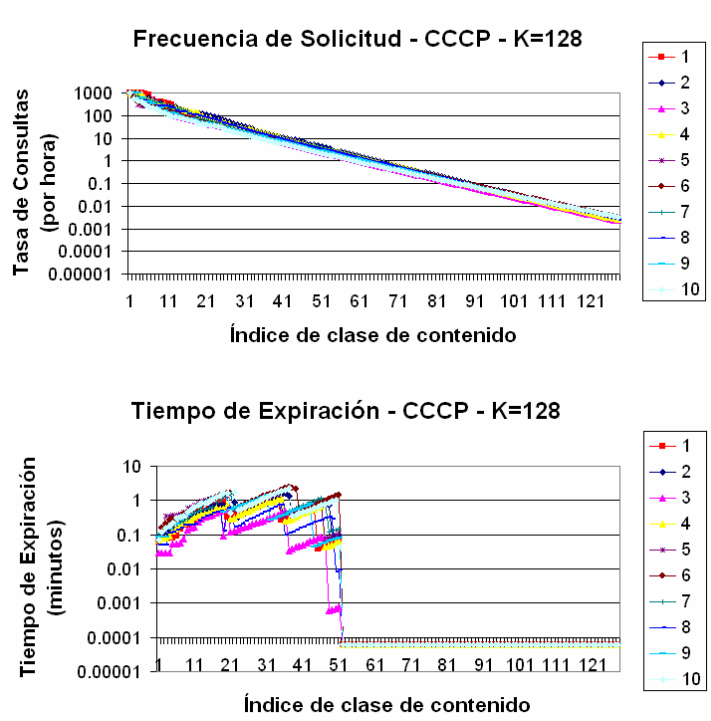


Ilustración 30. Tasa de solicitud y Tiempos de expiración de cada clase para las 10 instancias del problema CCCP de 128 clases.

B.2 DNS - Domain Name System

En el sistema DNS se realizan dos instancias del problema *CCP*, una para cada servidor recursivo al cual se tuvo acceso.
Para cada una de esas instancias del *CCP*, se crearon 5 instancias del modelo *CCCP*.

Los resultados para ambos servidores fueron muy similares y por tanto en este documento solo se incluye el estudio sobre uno de ellos.

B.2.1 Instancias del problema CCCP

A continuación se presentan las 5 instancias de distinta cantidad de clases, generadas a partir de una misma instancia del problema *CCP*.

```
param K         := 2;

param alphaS    := 169.59400000;
param alphaB    := 1150.25200000;
param betaS     := 80.45400000;
param betaB     := 385.55700000;
param BWin      := 193305600.00000000;
param BWout     := 344494080.00000000;

param f :=
          1 1059.31197749
          2 0.32706349;
```

```
param lamda :=
         1 0.83609821
         2 0.83609821;
param mu :=
         1 0.51581508
         2 0.51581509;
param l :=
         1 68.00000000
         2 220039.00000000;
```

Tabla 47. Instancia del CCCP (2 clases) en AMPL. cccp-dns-K002.dat

```
param K        := 8;

param alphaS   := 169.59400000;
param alphaB   := 1150.25200000;
param betaS    := 80.45400000;
param betaB    := 385.55700000;
param BWin     := 193305600.00000000;
param BWout    := 344494080.00000000;

param f :=
         1 4575.72707325
         2 2305.23294800
         3 1087.83816041
         4 423.95948017
         5 172.81579635
         6 38.78318038
         7 5.96143395
         8 0.08181972;
param lamda :=
         1 0.83609821
         2 0.83609821
         3 0.83609821
         4 0.83609821
         5 0.83609821
         6 0.83609821
         7 0.83609821
         8 0.83609821;
param mu :=
         1 0.51581508
         2 0.51581508
         3 0.51581508
         4 0.51581508
         5 0.51581508
         6 0.51581509
         7 0.51581508
         8 0.51581509;
param l :=
         1 4.00000000
         2 8.00000000
         3 17.00000000
         4 42.00000000
         5 103.00000000
         6 457.00000000
         7 2972.00000000
         8 216504.00000000;
```

Tabla 48. Instancia del CCCP (8 clases) en AMPL. cccp-dns-K008.dat

```
param K        := 16;

param alphaS   := 169.59400000;
param alphaB   := 1150.25200000;
param betaS    := 80.45400000;
param betaB    := 385.55700000;
param BWin     := 193305600.00000000;
param BWout    := 344494080.00000000;

param f :=
         1 5610.27432650
         2 3359.91893267
         3 2454.91953550
```

```
         4 1787.96496740
         5 1345.55070329
         6 673.74281515
         7 432.99018030
         8 348.34575252
         9 240.09216583
        10 100.73750319
        11 51.88965537
        12 25.17129564
        13 9.64498192
        14 3.67272762
        15 0.82732839
        16 0.04138581;
param lamda :=
         1 0.83609821
         2 0.83609821
         3 0.83609821
         4 0.83609821
         5 0.83609821
         6 0.83609821
         7 0.83609821
         8 0.83609821
         9 0.83609821
        10 0.83609821
        11 0.83609821
        12 0.83609821
        13 0.83609821
        14 0.83609821
        15 0.83609821
        16 0.83609821;
param mu :=
         1 0.51581508
         2 0.51581508
         3 0.51581508
         4 0.51581508
         5 0.51581508
         6 0.51581508
         7 0.51581508
         8 0.51581508
         9 0.51581508
        10 0.51581508
        11 0.51581508
        12 0.51581509
        13 0.51581508
        14 0.51581508
        15 0.51581509
        16 0.51581509;
param l :=
         1 2.00000000
         2 3.00000000
         3 4.00000000
         4 5.00000000
         5 7.00000000
         6 13.00000000
         7 20.00000000
         8 25.00000000
         9 36.00000000
        10 85.00000000
        11 165.00000000
        12 339.00000000
        13 884.00000000
        14 2321.00000000
        15 10300.00000000
        16 205898.00000000;
```

Tabla 49. Instancia del CCCP (16 clases) en AMPL. cccp- dns-K016.dat

```
param K        := 32;

param alphaS   := 169.59400000;
param alphaB   := 1150.25200000;
param betaS    := 80.45400000;
param betaB    := 385.55700000;
param BWin     := 193305600.00000000;
```

```
param BWout    := 344494080.00000000;

param f :=
         1 5909.55195800
         2 5310.99669500
         3 3541.17982000
         4 2828.95360350
         5 2553.01903600
         6 1980.80633100
         7 1683.51195500
         8 1528.11156433
         9 1208.63005750
        10 743.08972900
        11 614.30260329
        12 471.18681700
        13 401.73838664
        14 373.18266282
        15 330.50505654
        16 289.07927157
        17 225.36202856
        18 142.94925886
        19 90.79296456
        20 66.64223359
        21 50.20805689
        22 37.86501802
        23 24.85492851
        24 16.74801553
        25 10.13071777
        26 6.67813282
        27 4.34690023
        28 2.40618281
        29 1.18719598
        30 0.51453742
        31 0.16931227
        32 0.02221085;
param lamda :=
         1 0.83609821
         2 0.83609821
         3 0.83609821
         4 0.83609821
         5 0.83609821
         6 0.83609821
         7 0.83609821
         8 0.83609821
         9 0.83609821
        10 0.83609821
        11 0.83609821
        12 0.83609821
        13 0.83609821
        14 0.83609821
        15 0.83609821
        16 0.83609821
        17 0.83609821
        18 0.83609821
        19 0.83609821
        20 0.83609821
        21 0.83609821
        22 0.83609821
        23 0.83609821
        24 0.83609821
        25 0.83609821
        26 0.83609821
        27 0.83609821
        28 0.83609821
        29 0.83609821
        30 0.83609821
        31 0.83609821
        32 0.83609821;
param mu :=
         1 0.51581508
         2 0.51581508
         3 0.51581508
         4 0.51581508
         5 0.51581508
```

```
         6 0.51581508
         7 0.51581508
         8 0.51581508
         9 0.51581508
        10 0.51581508
        11 0.51581508
        12 0.51581508
        13 0.51581508
        14 0.51581508
        15 0.51581508
        16 0.51581508
        17 0.51581508
        18 0.51581508
        19 0.51581508
        20 0.51581508
        21 0.51581508
        22 0.51581508
        23 0.51581508
        24 0.51581508
        25 0.51581509
        26 0.51581509
        27 0.51581508
        28 0.51581508
        29 0.51581508
        30 0.51581509
        31 0.51581509
        32 0.51581509;
param l :=
         1 1.00000000
         2 1.00000000
         3 2.00000000
         4 2.00000000
         5 2.00000000
         6 3.00000000
         7 3.00000000
         8 3.00000000
         9 4.00000000
        10 6.00000000
        11 7.00000000
        12 9.00000000
        13 11.00000000
        14 11.00000000
        15 13.00000000
        16 14.00000000
        17 18.00000000
        18 29.00000000
        19 45.00000000
        20 61.00000000
        21 81.00000000
        22 107.00000000
        23 162.00000000
        24 240.00000000
        25 397.00000000
        26 602.00000000
        27 924.00000000
        28 1669.00000000
        29 3383.00000000
        30 7804.00000000
        31 23715.00000000
        32 180778.00000000;
```

Tabla 50. Instancia del CCCP (32 clases) en AMPL. cccp- dns-K032.dat

```
param K        := 128;

param alphaS   := 169.59400000;
param alphaB   := 1150.25200000;
param betaS    := 80.45400000;
param betaB    := 385.55700000;
param BWin     := 193305600.00000000;
param BWout    := 344494080.00000000;

param f :=
         1 5909.55195800
```

```
 2 5310.99669500
 3 3926.52496000
 4 3155.83468000
 5 2997.39715800
 6 2660.51004900
 7 2609.96742800
 8 2496.07064400
 9 2053.13002100
10 1951.40203200
11 1937.88694000
12 1735.49931200
13 1680.79619800
14 1634.24035500
15 1576.88807400
16 1520.49123000
17 1486.95538900
18 1430.13294100
19 1290.82153100
20 1126.35606900
21 987.20968900
22 926.28754800
23 747.75544650
24 687.00702100
25 662.33503050
26 637.73252550
27 598.43366250
28 582.92083850
29 535.01869850
30 484.81916700
31 456.17342650
32 436.07884750
33 413.87796467
34 404.78394100
35 399.99807000
36 393.21446667
37 386.36716800
38 380.33343800
39 370.82539167
40 345.73793033
41 342.12464100
42 334.34796267
43 327.36458633
44 313.42389067
45 304.06929333
46 294.11248100
47 287.49390767
48 278.93840300
49 274.61866967
50 265.68316150
51 244.02079750
52 206.48514975
53 184.23432340
54 169.35208820
55 155.57121133
56 142.10533950
57 132.13818686
58 116.30771143
59 104.88418950
60 95.13354422
61 87.73083690
62 81.76282970
63 76.86609618
64 72.88800364
65 69.02019450
66 64.64366915
67 61.86020708
68 58.21279493
69 54.63247047
70 51.05616812
71 48.73188871
72 46.98859906
73 43.63010221
74 41.17803460
75 39.16382024
```

```
         76 37.27152086
         77 34.59588517
         78 31.54245181
         79 27.89546759
         80 25.68164390
         81 23.73954544
         82 22.12753647
         83 20.76037805
         84 19.22703348
         85 17.92612249
         86 16.28444092
         87 14.49927067
         88 12.91717289
         89 11.64349236
         90 10.65943495
         91 9.67501840
         92 8.95778366
         93 8.06787201
         94 7.47912199
         95 6.92982657
         96 6.54248268
         97 6.08197682
         98 5.49967432
         99 5.06210146
        100 4.69114968
        101 4.27777577
        102 3.82132865
        103 3.42991005
        104 3.02769451
        105 2.67639751
        106 2.35633118
        107 2.05417840
        108 1.78095897
        109 1.55791981
        110 1.36699955
        111 1.17853639
        112 0.99785832
        113 0.86423151
        114 0.74765943
        115 0.62693369
        116 0.51746398
        117 0.42435897
        118 0.35083040
        119 0.28692750
        120 0.23040263
        121 0.17964189
        122 0.13741880
        123 0.10244741
        124 0.07285764
        125 0.05084277
        126 0.03335912
        127 0.01942898
        128 0.00885266;
param lamda :=
          1 0.83609821
          2 0.83609821
          3 0.83609821
          4 0.83609821
          5 0.83609821
          6 0.83609821
          7 0.83609821
          8 0.83609821
          9 0.83609821
         10 0.83609821
         11 0.83609821
         12 0.83609821
         13 0.83609821
         14 0.83609821
         15 0.83609821
         16 0.83609821
         17 0.83609821
         18 0.83609821
         19 0.83609821
         20 0.83609821
```

```
        21 0.83609821
        22 0.83609821
        23 0.83609821
        24 0.83609821
        25 0.83609821
        26 0.83609821
        27 0.83609821
        28 0.83609821
        29 0.83609821
        30 0.83609821
        31 0.83609821
        32 0.83609821
        33 0.83609821
        34 0.83609821
        35 0.83609821
        36 0.83609821
        37 0.83609821
        38 0.83609821
        39 0.83609821
        40 0.83609821
        41 0.83609821
        42 0.83609821
        43 0.83609821
        44 0.83609821
        45 0.83609821
        46 0.83609821
        47 0.83609821
        48 0.83609821
        49 0.83609821
        50 0.83609821
        51 0.83609821
        52 0.83609821
        53 0.83609821
        54 0.83609821
        55 0.83609821
        56 0.83609821
        57 0.83609821
        58 0.83609821
        59 0.83609821
        60 0.83609821
        61 0.83609821
        62 0.83609821
        63 0.83609821
        64 0.83609821
        65 0.83609821
        66 0.83609821
        67 0.83609821
        68 0.83609821
        69 0.83609821
        70 0.83609821
        71 0.83609821
        72 0.83609821
        73 0.83609821
        74 0.83609821
        75 0.83609821
        76 0.83609821
        77 0.83609821
        78 0.83609821
        79 0.83609821
        80 0.83609821
        81 0.83609821
        82 0.83609821
        83 0.83609821
        84 0.83609821
        85 0.83609821
        86 0.83609821
        87 0.83609821
        88 0.83609821
        89 0.83609821
        90 0.83609821
        91 0.83609821
        92 0.83609821
        93 0.83609821
        94 0.83609821
```

```
         95 0.83609821
         96 0.83609821
         97 0.83609821
         98 0.83609821
         99 0.83609821
        100 0.83609821
        101 0.83609821
        102 0.83609821
        103 0.83609821
        104 0.83609821
        105 0.83609821
        106 0.83609821
        107 0.83609821
        108 0.83609821
        109 0.83609821
        110 0.83609821
        111 0.83609821
        112 0.83609821
        113 0.83609821
        114 0.83609821
        115 0.83609821
        116 0.83609821
        117 0.83609821
        118 0.83609821
        119 0.83609821
        120 0.83609821
        121 0.83609821
        122 0.83609821
        123 0.83609821
        124 0.83609821
        125 0.83609821
        126 0.83609821
        127 0.83609821
        128 0.83609821;
param mu :=
          1 0.51581508
          2 0.51581508
          3 0.51581508
          4 0.51581508
          5 0.51581508
          6 0.51581508
          7 0.51581508
          8 0.51581508
          9 0.51581508
         10 0.51581508
         11 0.51581508
         12 0.51581508
         13 0.51581508
         14 0.51581508
         15 0.51581508
         16 0.51581508
         17 0.51581508
         18 0.51581508
         19 0.51581508
         20 0.51581508
         21 0.51581508
         22 0.51581508
         23 0.51581508
         24 0.51581508
         25 0.51581508
         26 0.51581508
         27 0.51581508
         28 0.51581508
         29 0.51581508
         30 0.51581508
         31 0.51581508
         32 0.51581508
         33 0.51581508
         34 0.51581508
         35 0.51581508
         36 0.51581508
         37 0.51581508
         38 0.51581508
         39 0.51581508
```

```
 40 0.51581508
 41 0.51581508
 42 0.51581508
 43 0.51581508
 44 0.51581508
 45 0.51581508
 46 0.51581508
 47 0.51581508
 48 0.51581508
 49 0.51581508
 50 0.51581508
 51 0.51581508
 52 0.51581508
 53 0.51581508
 54 0.51581508
 55 0.51581508
 56 0.51581508
 57 0.51581508
 58 0.51581508
 59 0.51581508
 60 0.51581508
 61 0.51581508
 62 0.51581508
 63 0.51581508
 64 0.51581508
 65 0.51581508
 66 0.51581508
 67 0.51581508
 68 0.51581508
 69 0.51581508
 70 0.51581508
 71 0.51581508
 72 0.51581508
 73 0.51581508
 74 0.51581508
 75 0.51581508
 76 0.51581508
 77 0.51581508
 78 0.51581508
 79 0.51581508
 80 0.51581508
 81 0.51581508
 82 0.51581508
 83 0.51581508
 84 0.51581508
 85 0.51581508
 86 0.51581508
 87 0.51581508
 88 0.51581508
 89 0.51581508
 90 0.51581508
 91 0.51581508
 92 0.51581508
 93 0.51581508
 94 0.51581508
 95 0.51581508
 96 0.51581508
 97 0.51581508
 98 0.51581508
 99 0.51581508
100 0.51581508
101 0.51581508
102 0.51581508
103 0.51581508
104 0.51581509
105 0.51581509
106 0.51581509
107 0.51581509
108 0.51581509
109 0.51581509
110 0.51581509
111 0.51581509
112 0.51581508
113 0.51581508
```

```
       114 0.51581508
       115 0.51581508
       116 0.51581508
       117 0.51581508
       118 0.51581508
       119 0.51581508
       120 0.51581508
       121 0.51581509
       122 0.51581509
       123 0.51581509
       124 0.51581509
       125 0.51581509
       126 0.51581509
       127 0.51581509
       128 0.51581508;
param l :=
         1 1.00000000
         2 1.00000000
         3 1.00000000
         4 1.00000000
         5 1.00000000
         6 1.00000000
         7 1.00000000
         8 1.00000000
         9 1.00000000
        10 1.00000000
        11 1.00000000
        12 1.00000000
        13 1.00000000
        14 1.00000000
        15 1.00000000
        16 1.00000000
        17 1.00000000
        18 1.00000000
        19 1.00000000
        20 1.00000000
        21 1.00000000
        22 1.00000000
        23 2.00000000
        24 2.00000000
        25 2.00000000
        26 2.00000000
        27 2.00000000
        28 2.00000000
        29 2.00000000
        30 2.00000000
        31 2.00000000
        32 2.00000000
        33 3.00000000
        34 3.00000000
        35 3.00000000
        36 3.00000000
        37 3.00000000
        38 3.00000000
        39 3.00000000
        40 3.00000000
        41 3.00000000
        42 3.00000000
        43 3.00000000
        44 3.00000000
        45 3.00000000
        46 3.00000000
        47 3.00000000
        48 3.00000000
        49 3.00000000
        50 4.00000000
        51 4.00000000
        52 4.00000000
        53 5.00000000
        54 5.00000000
        55 6.00000000
        56 6.00000000
        57 7.00000000
        58 7.00000000
```

```
 59 8.00000000
 60 9.00000000
 61 10.00000000
 62 10.00000000
 63 11.00000000
 64 11.00000000
 65 12.00000000
 66 13.00000000
 67 13.00000000
 68 14.00000000
 69 15.00000000
 70 16.00000000
 71 17.00000000
 72 17.00000000
 73 19.00000000
 74 20.00000000
 75 21.00000000
 76 22.00000000
 77 23.00000000
 78 26.00000000
 79 29.00000000
 80 31.00000000
 81 34.00000000
 82 36.00000000
 83 39.00000000
 84 42.00000000
 85 45.00000000
 86 49.00000000
 87 55.00000000
 88 62.00000000
 89 69.00000000
 90 75.00000000
 91 82.00000000
 92 89.00000000
 93 99.00000000
 94 106.00000000
 95 115.00000000
 96 121.00000000
 97 131.00000000
 98 144.00000000
 99 157.00000000
100 169.00000000
101 185.00000000
102 207.00000000
103 231.00000000
104 262.00000000
105 296.00000000
106 336.00000000
107 385.00000000
108 444.00000000
109 508.00000000
110 579.00000000
111 671.00000000
112 793.00000000
113 915.00000000
114 1058.00000000
115 1261.00000000
116 1528.00000000
117 1863.00000000
118 2254.00000000
119 2755.00000000
120 3431.00000000
121 4400.00000000
122 5752.00000000
123 7716.00000000
124 10849.00000000
125 15546.00000000
126 23694.00000000
127 40681.00000000
128 89280.00000000;
```

Tabla 51. Instancia del CCCP (128 clases) en AMPL. cccp- dns-K128.dat

B.2.2 Soluciones a las Instancias del problema CCCP

Las soluciones de las instancias anteriores halladas mediante el método descrito en la sección 5.1.3.

```
Time = 0.00 seg
epsilon = 0.969623
bitsIn = 193305400
bitsOut = 77163700

solution d :=
         1 0.00045150
         2 1.45731000;
```

Tabla 52. Solución de la instancia del CCCP (2 clases) en AMPL. cccp-dns-K002.dat

```
Time = 0.02 seg
epsilon = 0.982216
bitsIn = 193305400
bitsOut = 77163700

solution d :=
         1 0.00010547
         2 0.00020790
         3 0.00043892
         4 0.00112391
         5 0.00275507
         6 0.01227130
         7 0.07982420
         8 5.81410000;
```

Tabla 53. Solución de la instancia del CCCP (8 clases) en AMPL. cccp-dns-K008.dat

```
Time = 0.06 seg
epsilon = 0.997218
bitsIn = 193305400
bitsOut = 77163700

solution d :=
         1 0.00009815
         2 0.00016219
         3 0.00022253
         4 0.00030212
         5 0.00040067
         6 0.00080116
         7 0.00124660
         8 0.00154965
         9 0.00224743
        10 0.00535496
        11 0.01039620
        12 0.02139530
        13 0.05687810
        14 0.14559100
        15 0.15399700
        16 3.07597000;
```

Tabla 54. Solución de la instancia del CCCP (16 clases) en AMPL. cccp- dns-K016.dat

```
Time = 0.12 seg
epsilon = 0.999742
bitsIn = 193305400
bitsOut = 77163700

solution d :=
         1 0.00013915
         2 0.00017515
         3 0.00026369
         4 0.00032987
         5 0.00036724
         6 0.00046882
         7 0.00056446
         8 0.00061433
         9 0.00079602
```

```
        10 0.00128394
        11 0.00157242
        12 0.00204865
        13 0.00238609
        14 0.00257409
        15 0.00282043
        16 0.00282180
        17 0.00098665
        18 0.00155420
        19 0.00244659
        20 0.00333278
        21 0.00442327
        22 0.00586463
        23 0.00893375
        24 0.01325920
        25 0.02192250
        26 0.03324360
        27 0.02752610
        28 0.01141460
        29 0.02312980
        30 0.05336790
        31 0.16216600
        32 1.23272000;
```

Tabla 55. Solución de la instancia del CCCP (32 clases) en AMPL. cccp- dns-K032.dat

```
Time =  seg
epsilon = 0.999919
bitsIn = 193305400
bitsOut = 77163700

solution d :=
         1 0.00038073
         2 0.00046314
         3 0.00063631
         4 0.00079772
         5 0.00084492
         6 0.00094342
         7 0.00094913
         8 0.00100031
         9 0.00119840
        10 0.00126766
        11 0.00129570
        12 0.00145154
        13 0.00147242
        14 0.00150381
        15 0.00157647
        16 0.00168134
        17 0.00174745
        18 0.00182912
        19 0.00200729
        20 0.00218290
        21 0.00079928
        22 0.00087143
        23 0.00296684
        24 0.00116854
        25 0.00121673
        26 0.00126516
        27 0.00134536
        28 0.00138099
        29 0.00151021
        30 0.00163624
        31 0.00179414
        32 0.00158590
        33 0.00196639
        34 0.00199837
        35 0.00198786
        36 0.00203489
        37 0.00214039
        38 0.00222446
        39 0.00199777
        40 0.00060329
        41 0.00060996
        42 0.00062405
```

```
 43 0.00063726
 44 0.00066557
 45 0.00068583
 46 0.00070912
 47 0.00072565
 48 0.00074788
 49 0.00075944
 50 0.00078483
 51 0.00085481
 52 0.00000100
 53 0.00000100
 54 0.00000100
 55 0.00000100
 56 0.00000100
 57 0.00000100
 58 0.00000100
 59 0.00000100
 60 0.00000100
 61 0.00000100
 62 0.00000100
 63 0.00000100
 64 0.00000100
 65 0.00000100
 66 0.00000100
 67 0.00000100
 68 0.00000100
 69 0.00000100
 70 0.00000100
 71 0.00000100
 72 0.00000100
 73 0.00000100
 74 0.00000100
 75 0.00000100
 76 0.00000100
 77 0.00000100
 78 0.00000100
 79 0.00000100
 80 0.00000100
 81 0.00000100
 82 0.00000100
 83 0.00000100
 84 0.00000100
 85 0.00000100
 86 0.00000100
 87 0.00000100
 88 0.00000100
 89 0.00000100
 90 0.00000100
 91 0.00000100
 92 0.00000100
 93 0.00000100
 94 0.00000100
 95 0.00000100
 96 0.00000100
 97 0.00000100
 98 0.00000100
 99 0.00000100
100 0.00000100
101 0.00000100
102 0.00000100
103 0.00000100
104 0.00000100
105 0.00000100
106 0.00000100
107 0.00000100
108 0.00000100
109 0.00000100
110 0.00000100
111 0.00000100
112 0.00000100
113 0.00000100
114 0.00000100
115 0.00000100
116 0.00000100
```

```
      117 0.00000100
      118 0.00000100
      119 0.00000100
      120 0.00000100
      121 0.00000100
      122 0.00000100
      123 0.00000100
      124 0.00000100
      125 0.00000100
      126 0.00000100
      127 0.00000100
      128 0.00000100;
```

Tabla 56. Solución de la instancia del CCCP (128 clases) en AMPL. cccp- dns-K128.dat

Índice de Ilustraciones

Índice de Tablas