APSE: A P2P SEARCH ENGINE

JORGE ELIÉCER CAMARGO MENDOZA

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERIA
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
BOGOTÁ
2006

APSE: A P2P SEARCH ENGINE

JORGE ELIÉCER CAMARGO MENDOZA

Tesis para optar el título de
Magíster en Ingeniería de Sistemas y Computación

Asesor
Francisco Rueda Fajardo
Profesor Asociado

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERIA DE SISTEMAS Y COMPUTACIÓN
BOGOTÁ
2006

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

1 INTRODUCCION

Los motores de búsqueda se han convertido en una herramienta de uso frecuente y son una de las principales fuentes de consulta de las personas de hoy en día. Detrás de una sencilla interfase gráfica se encuentra toda una tecnología que reúne técnicas de recuperación de información, teorías matemáticas, álgebra vectorial, estadística, grandes volúmenes de capacidad en disco, memoria, redundancia en servidores, y en fin, muchos elementos que los usuarios nunca se imaginan que hay detrás para poderle traer resultados a sus consultas en fracciones de segundo. Los motores de búsqueda han seguido un modelo convencional centralizado, en donde se realiza todo el proceso de recorrido de las páginas de Internet, almacenamiento local, indexación y resolución de búsquedas. Aunque este modelo se comporta muy bien en términos de eficiencia para la resolución de consultas, tiene serios problemas de escalabilidad, y en algún momento podría ser de disponibilidad por su misma naturaleza centralizada.

Por otro lado, paralelamente han aparecido otro tipo de fuentes de información conocidos como redes P2P, en donde la arquitectura es distribuida. Tales sistemas ofrecen un alto grado de escalabilidad y disponibilidad, dado que no depende de un único repositorio de servicios e información sino que el sistema funciona aún si uno de los nodos del sistema falla por algún motivo. Bajo este tipo de redes se han creado motores de búsqueda con un alcance inferior al que ofrecen los motores de búsqueda centralizados. La diferencia radica en que un motor de búsqueda convencional indexa un alto porcentaje del contenido de un documento y en un motor de búsqueda basado en P2P se indexa en la mayoría de los casos tan sólo el nombre del documento.

1.1 Motivación

La principal motivación para desarrollar APSE consiste en explorar una alternativa a los motores de búsqueda tradicionales que se basan en arquitecturas centralizadas.
Por otro lado se desea evaluar la viabilidad de la construcción de un motor de búsqueda con la eficiencia de las aproximaciones centralizadas y con las bondades que ofrecen los sistemas distribuidos.

Algunas preguntas surgieron para los motores de búsqueda actuales como motivación para realizar el presente trabajo: cómo encontrar documentos relevantes que no fueron indexados por los motores de búsqueda centralizados, en los motores de búsqueda la cantidad de recursos computacionales que se utilizan son enormes, por qué no pensar en una alternativa en donde los recursos sean distribuidos?, por qué depender de un repositorio centralizado de información si realmente la información se encuentra distribuida en Internet?

1.2 Contribución

En esta tesis se propone un motor de búsqueda completamente distribuido, eficiente, escalable y robusto que soporte búsquedas por contenido. Toma como base las técnicas existentes de indexación, recuperación de información e índices invertidos de los motores de búsqueda tradicionales.
El número de mensajes que circulan por el sistema para resolver una consulta es completamente determinable, característica que convierte a APSE en un sistema eficiente en términos de tráfico de red.
Por otro lado se realizaron pruebas experimentales que demuestran la viabilidad del sistema para ser desarrollado en un ambiente real de producción, ya que el sistema es eficiente en términos de tiempo y de recuperación de documentos relevantes.

Este documento está organizado como sigue: en la sección 1 se presenta un marco conceptual de los motores de búsqueda y las redes P2P. En la sección 2 se revisan los trabajos relacionados. En la sección 3 se revisan algunos conceptos de los motores de búsqueda. En la sección 4 se describe el sistema propuesto. En la sección 5 se realiza una

evaluación analítica y experimental de APSE. Y por último, en la sección 6 se presentan las conclusiones y trabajos futuros.

2 MARCO CONCEPTUAL

2.1 Motores de búsqueda

Los motores de búsqueda juegan un papel importante en este proyecto dado que los servicios que ellos prestan son similares a los que se van a ofrecer en el sistema p2p que se propone.

En los motores de búsqueda el componente llamado *spider* realiza un recorrido por los hipervínculos de las páginas web, iniciando desde diferentes páginas para cubrir un mayor porcentaje de Internet. En este recorrido (*crawling*) se procede a almacenar el contenido HTML en servidores locales que posteriormente serán procesados mediante operaciones de indexación ejecutadas por el componente *indexer*. Como resultado de la indexación se obtiene una estructura de datos llamada *lexicon (*ver Figura 1*)*, un índice invertido que almacena la cantidad de ocurrencias de cada palabra en todos los documentos, documentos en donde es encontrada, cantidad de ocurrencias en cada documento y posición dentro de cada documento (La palabra árbol se encuentra en 2 documentos, 4 veces en el documento 10 y 3 veces en el documento 678). Esta estructura se mantiene en memoria principal para que las consultas sean resueltas muy rápido.

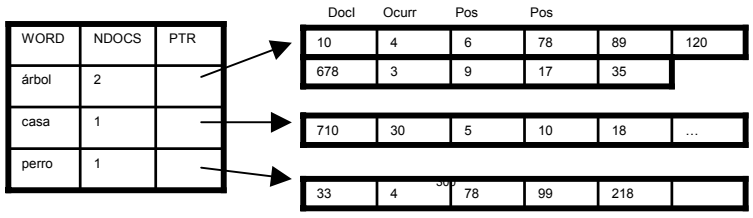

Figura 1 Lexicon. Estructura de índice invertido utilizada en los motores de búsqueda

Otro componente es el *retriever*, interfase del sistema con el cual se interactúa para realizar consultas por palabras clave en todos los documentos indexados.

2.1.1 Arquitectura de Google [1]

Por razones de eficiencia Google está implementado en C y C++, y puede correr tanto en Solaris como en Linux. La descarga de páginas (*web crawling*) es realizada por varios servidores *crawler* distribuidos. El URL Server envía las listas de URLs para que sean procesadas por los *crawler*. Posteriormente son enviadas a los *Store Server* que se encargan de comprimirlas y almacenarlas en un *Repository*. Cada página tiene un identificador asociado *docID* que se asigna cada que una nueva URL es extraída de una página. La función de indexación es realizada por los servidores *Indexer* y *Sorter*. El indexer se encarga de leer y descomprimir los documentos del repositorio para luego interpretarlos. Cada documento es convertido en un conjunto de ocurrencias de palabras llamado *hits*. El hit registra la palabra, posición en el documento, una aproximación del tamaño de letra y mayúsculas. El indexer distribuye esos hits en un conjunto de servidores llamados barrels, creando un índice parcialmente ordenado. EL indexer extrae todos los link de cada página y almacena información importante acerca de ellos en un archivo de anchors (etiqueta HTML). Este archivo almacena información suficiente para determinar el texto, a dónde y desde dónde apunta cada link [1].

El URL Resolver lee el archivo de anchors y convierte las URLs relativas a absolutas y alternadamente en docIDs. También se encarga de colocar el texto del anchor en un *forward index* asociado en el docID al que el anchor apunta. Además se encarga de generar una base de datos de links que forman pares de docID. Dicha base de datos es luego utilizada para calcular el PageRank para todos los documentos [1].

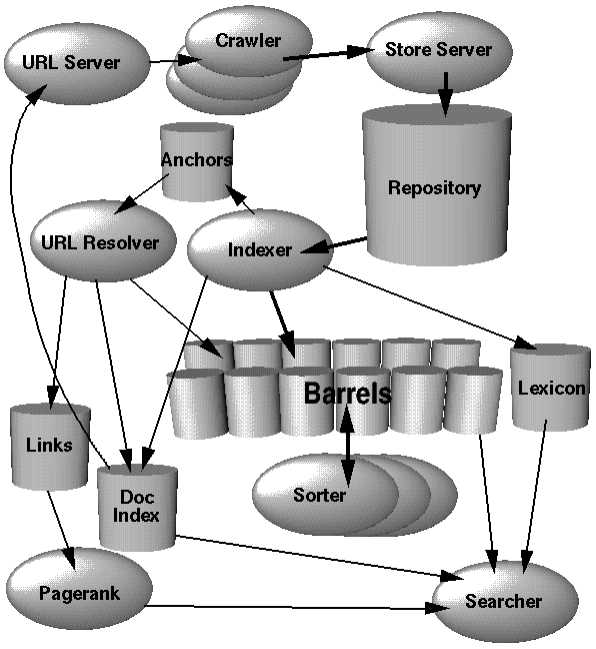

Figura 2 Arquitectura de Google a alto nivel [1]

El *sorter* toma los *barrels* que se encuentran ordenados por *docID* y los reordena por *wordID* para generar un índice invertido. También se encarga de producir una lista de *wordIDs* en dicho índice. Un programa llamado *DumpLexicon* toma esta lista junto con el lexicón producido por el *indexer* y genera un nuevo *lexicón* para ser utilizado por el *searcher*. El searcher es invocado por un servidor web que al mismo tiempo hace uso del lexicon, índice invertido y PageRank para responder a consultas [1].

2.1.2 Recuperación de información

La recuperación de información es una disciplina que estudia teorías, modelos y técnicas que tienen que ver con la representación, almacenamiento, organización y recuperación de contenido útil a los humanos.

Los documentos son representados por un conjunto de palabras clave representativas o términos índice. Un índice es un término de un documento útil para recordar un tema importante del mismo. Usualmente, los términos índice son sustantivos, dado que por sí solos tienen su propio significado. Sin embargo, los motores de búsqueda asumen que todas las palabras son términos índice.

No todos los términos son igualmente útiles para representar el contenido de los documentos: los términos menos frecuentes son generalmente quienes describen mejor un documento. La importancia de los términos índice es representada por pesos asociados a ellos [3].

Los modelos más utilizados de IR son el Booleano y el *Vector Space Model*.

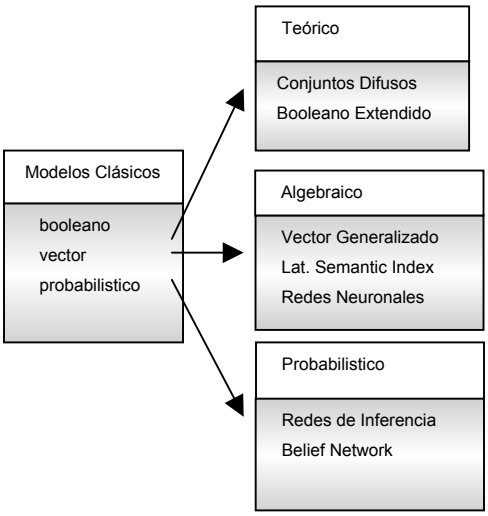

Figura 3 Modelos clásicos de IR [3]

2.1.2.1 Modelo Booleano

La lógica booleana, también conocida como álgebra booleana es un método utilizado para describir objetos o ideas. Al insertar palabras clave u operadores (OR, AND o NOT) entre los términos de una frase se puede describir la relación entre los mismos. Mediante este mecanismo se puede comunicar exactamente al motor de búsqueda lo que se quiere encontrar. Sin un lenguaje estándar como éste, dos búsquedas pueden tener resultados diferentes para una misma consulta [3].

Con este modelo la relevancia de un documento es binaria, ya que es relevante si contiene la palabra buscada. En consultas que tienen únicamente el operador AND se retornarán únicamente los documentos que tengan todas las palabras.

Ejemplo: Lo mejor de Maradona

Maradona **AND** Mundial **AND** ((Mexico'86 **OR** Italia'90) **BUT NOT** U.S.A.'94) [3]

Este modelo es considerado como primitivo, pero es muy utilizando en algunos motores de búsqueda (Altavista, Hotbot, OpenText y Lycos).

Este modelo presenta desventajas ya que no discrimina la relevancia de los documentos, la cantidad de veces que aparece un término no es tenida en cuenta, para un usuario normal es complicado formular una consulta y no existe un criterio para ordenar los resultados obtenidos.

Como ventajas se puede mencionar que es útil para usuarios avanzados que puedan expresar consultas complejas, en los sistemas de bases de datos son parte primordial (SQL), es fácil de formalizar e implementar y es útil en combinación con otro modelo (Google lo utiliza como filtros avanzados).

2.1.2.2 Vector Space Model

En VSM los documentos son modelados como vectores en donde cada componente del vector corresponde al peso asignado a cada término dentro del documento.

Un documento d_i se modela como un vector así

$$d_i \longrightarrow \vec{d}_i = (w(t_1, d_i), \ldots, w(t_k, d_i)) \quad (1)$$

en donde $w(t_n,d_i)$ es el peso del término t_n en el documento d_i.

El peso de cada término se calcula así

$$w(t_n, d_i) = w_{n,i} = \frac{tf_{n,i} * idf_n}{\left|\vec{d}_i\right|} \quad (2)$$

en donde $tf_{n,i}$ es la frecuencia del término t_n en el documento d_i. La inversa de la frecuencia del término *idf* se calcula mediante $1/tf_{n,i}$ y en algunas ocasiones como el $log_2(tf_{n,i})$.

Se puede deducir entonces que si un término aparece varias veces en un documento es importante para dicho documento, es decir *tf* crece. Pero si dicho término aparece en muchos documentos entonces no es relevante para distinguir un documento de los otros, es decir *idf* decrece. Es necesario normalizar la frecuencia para castigar los documentos con palabras repetidas muchas veces, lo cual se logra mediante $tf_{norm} = tf_{n,d} / tf_{max,d})$. Lo que se busca es identificar qué tanto ayuda un término a distinguir un documento de los demás.

La similaridad coseno es una medida que indica qué tan similar es un documento con respecto a otro, por ejemplo, la similaridad entre un documento Q y D se define como:

$$sim(Q, D_j) = \frac{Q \bullet D_j}{|Q| \cdot |D_j|} = \cos(Q, D_j) \quad (3)$$

En donde Q representa el vector de la consulta y D el vector del documento j. Cuando el ángulo entre el documento D y la consulta Q es pequeño entonces los documentos son más similares.

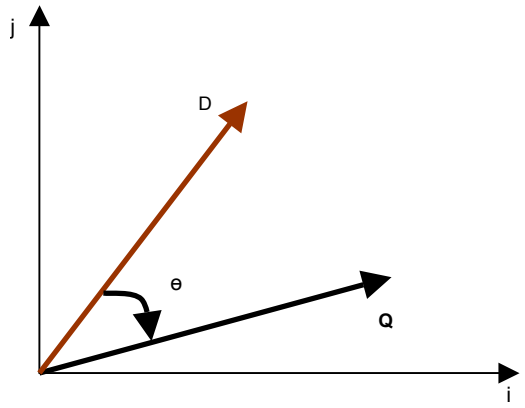

Figura 4 Angulo formado entre los vectores de una consulta Q y un documento D

2.1.2.3 PageRank

El hipervínculo es considerado el elemento más importante de una página web para los motores de búsqueda, dado que a través de él son enlazadas las páginas web. Normalmente son almacenados en una estructura llamada *Map* que permite calcular rápidamente la importancia de dichas páginas. El *PageRank* es una medida de qué tan "importante" es determinada página y se utiliza para priorizar los resultados de una búsqueda [1].

El cálculo está determinado por la cantidad de hipervínculos que apuntan a una página web y se calcula de la siguiente manera:

$$PR(A) = (1-d) + d(\frac{PR(T_1)}{C(T_1)} + \cdots + \frac{PR(T_n)}{C(T_n)}) \quad (4)$$

Se asume que una página *A* tiene *n* páginas que apuntan a ella (T_1, T_2, ..). El parámetro *d* es un factor que puede ser asignado entre 0 y 1. Google define en 0.85 el valor de este factor. *C(A)* se define como el número de links salientes de la página *A* [1].

Analizando la fórmula se puede decir que una página puede tener alto *PageRank* si hay muchas páginas que apuntan a ella, o si hay algunas páginas con alto *PageRank* que apuntan a éstas.

2.1.2.4 Latent Semantic Indexing

Los esquemas de recuperación de información literales sufren de sinonimia[1], polisemia[2] y ruido en los documentos. LSI ha sido propuesto para resolver dichos problemas. Este

---

[1] La sinonimia es una relación de semejanza de significados entre determinadas palabras (llamadas sinónimos) u oraciones. http://es.wikipedia.org/wiki/Polisemia

[2] Se llama polisemia a la capacidad que tiene una sola palabra para expresar muy distintos significados. http://es.wikipedia.org/wiki/Polisemia

modelo usa una descomposición de valores singulares (SVD) para transformar y truncar una matriz de vectores de documentos producidos por el VSM para descubrir la semántica inmersa en los términos y los documentos. LSI transforma un vector del documento altamente-dimensional en un vector semántico medio-dimensional proyectando el anterior en un subespacio semántico medio-dimensional. La base del subespacio semántico es calculada usando SVD. Los vectores semánticos son normalizados y sus similaridades son calculadas como en VSM [5].

Tanto VSM como LSI representan los documentos y consultas como vectores y la similaridad entre ellos como el coseno del ángulo entre dichos vectores.

2.2 Redes P2P

Desde hace algunos años han tomado auge un nuevo tipo de aplicaciones que funcionan sobre la red Internet denominadas redes Peer to PeerR (P2P). El principal objetivo de este tipo de aplicaciones es de el compartir recursos computacionales mediante un sistema de comunicación de intercambio de mensajes, y en especial se han orientado más a compartir archivos. Dentro de los sistemas que se consideran como P2P se encuentran:
Mensajería: ICQ, Microsoft y Yahoo Messenger
Intercambio de archivos: Napster, Emule, Edonkey y Gnutella
Trabajo en grupo: Groove (http://www.groove.net)

2.2.1 Características

La principal característica de una red P2P consiste en que cada nodo hace las funciones tanto de servidor como de cliente. Cada nodo tiene autonomía para decir hasta qué punto participa en el sistema. Los nodos no necesariamente deben tener una vista global del sistema. Los nodos ingresan y salen del sistema de forma arbitraria. La ubicación de los recursos es dinámica y depende del estado actual del sistema. Los nodos presentan características físicas heterogéneas [2].

2.2.2 Arquitectura

Las redes P2P se conectan lógicamente sobre una red física utilizando protocolos de red como UDP y TCP, ver Figura 6. En algunas ocasiones, cada peer escucha físicamente por varios puertos de acuerdo a los mensajes con los que se comunica con otros nodos o también de acuerdo a la actividad que realice en un momento determinado.
Los servicios básicos que un sistema P2P son:

Servidor. Provee acceso a datos y recursos del nodo en donde se ejecuta. Procesa peticiones de otros nodos y da accesos a sus recursos.
Cliente. Provee la interfaz de usuario para interactuar con el sistema P2P, ver Figura 5.

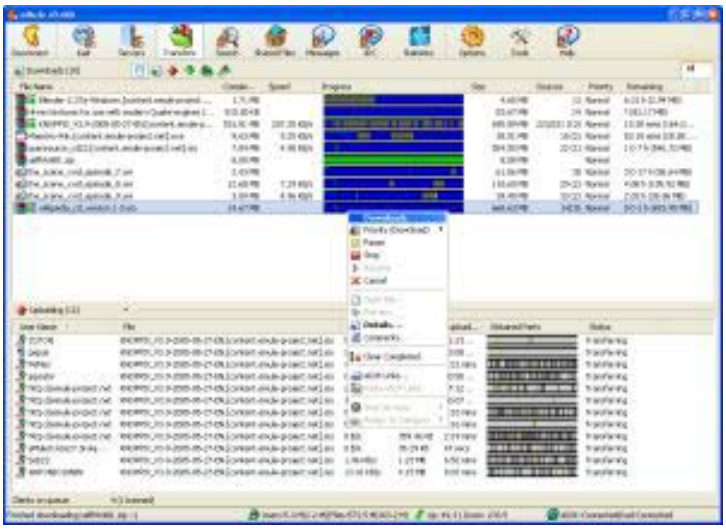

Figura 5 Interfaz gráfica del cliente P2P Emule, uno de los sistemas más populares

Datos. Provee información del estado de la red.
Enrutamiento. Provee los mecanismos de propagación de mensajes.
Comunicaciones. Provee los mecanismos de comunicación a los servicios cliente y servidor.
Seguridad. Provee los mecanismos para garantizar la seguridad del nodo.

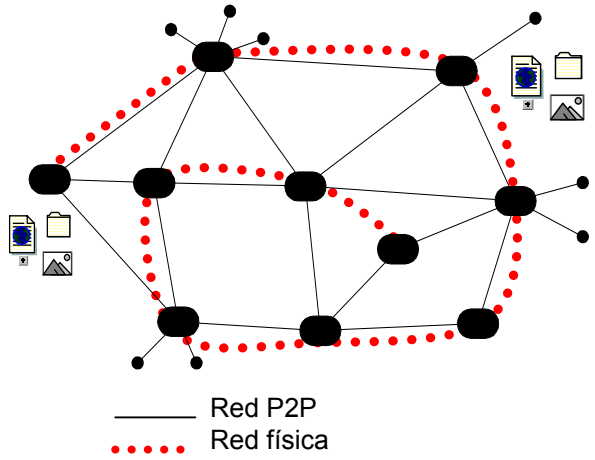

Figura 6 Esquema de una red P2P [13]. Los nodos se interconectan físicamente de una manera y lógicamente de otra

2.2.3 Búsquedas en sistemas P2P

2.2.3.1 Búsqueda basada en inundación de mensajes

La técnica de inundación de mensajes consiste en que el nodo que origina una consulta Q la envía a sus nodos más cercanos y éstos a su vez la reenvían a sus nodos vecinos y así sucesivamente. Cada que un nodo recibe la consulta busca localmente y si tiene algún documento que corresponda con Q, se devuelve una respuesta al nodo inicial notificando dicho evento.

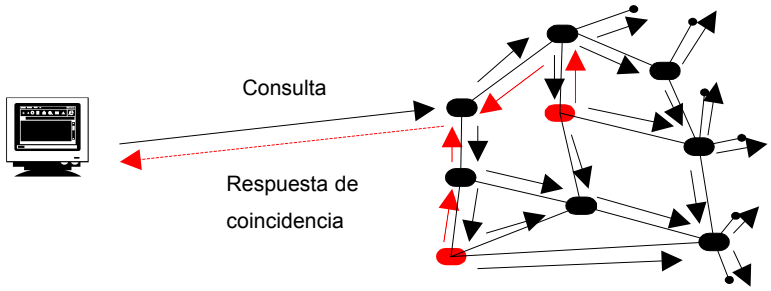

Figura 7 Búsqueda basada en inundación de mensajes

Uno de los sistemas que explora y evalúa diferentes variaciones de esta técnica es Peerware, descrito en más detalle en 3.1.

2.2.3.2 Búsqueda basada en identificadores de objetos

Sistemas de indexación distribuido de archivos como Chord [10], CAN [4] y Pastry [11] están basados en redes p2p estructuradas, las cuales permiten realizar búsquedas a través del hash de un objeto, obteniendo como respuesta la dirección IP del nodo que lo contiene.

PeerSearch y ODISSEA son sistemas que se basan en esta técnica, y se describen en más detalle en 3.2.1 y 3.2.3 respectivamente.

3 TRABAJOS RELACIONADOS

Como resultado de la investigación de trabajos relacionados en recuperación de información utilizando redes P2P, se encontró que existen dos grandes tipos de enfoques: los basados en inundación de mensajes usando una red no estructurada (tipo Gnutella) y los basados en búsqueda por identificadores bajo una red estructurada (tipo Chord).

Estos enfoques se describen en las siguientes subsecciones y se enumeran algunas ventajas y desventajas que es interesante mencionar para ser tenidas en cuenta en el sistema propuesto.

3.1 Basados en redes P2P no estructuradas

En las redes P2P no estructuradas se usa la técnica de inundación de mensajes, que consiste en inundar la red de mensajes para resolver determinada consulta. Un nodo que origina una consulta Q, la envía a sus nodos más cercanos y éstos a su vez a sus nodos vecinos, y así sucesivamente hasta ser resuelta o hasta que la consulta muera por TTL (número que disminuye en cada paso por un nodo). Cada que un nodo recibe la consulta intenta resolverla localmente, si tiene algún documento que corresponda con Q se devuelve una respuesta al nodo que la originó.

Uno de los sistemas que explora y evalúa diferentes variaciones de esta técnica es Peerware.

3.1.1 Peerware

Peerware [13] es un sistema middleware construido para realizar experimentación de diferentes algoritmos de enrutamiento sobre sistemas p2p a gran escala. El sistema es utilizado en [9] para experimentalmente mostrar que la técnica ISM propuesta es más eficiente en términos de disminución de envío de mensajes y porcentaje de documentos recuperados, en comparación con las técnicas BFS y RBFS descritas a continuación.

3.1.1.1 Técnica BFS

La técnica BFS (Breadth First Search) ha sido usada ampliamente en aplicaciones P2P para compartir archivos como es el caso de Gnutella [8]. Una consulta que se origina desde el nodo N es propagada a todos sus nodos vecinos (peers) en donde cada uno de ellos inmediatamente reenvía la consulta a sus vecinos y luego busca localmente los documentos que concuerdan con la consulta. Si la consulta concuerda en algún nodo, se envía un mensaje de retorno al nodo inicial con el resultado, en donde el mensaje contiene información del número de documentos y datos de conectividad de dicho nodo. Cuando varios nodos responden a la misma consulta, el nodo origen puede decidir de cuál de ellos descargar el archivo basado en por ejemplo quien tenga mejor ancho de banda en la conexión. En este modelo se sacrifica desempeño y utilización de la red con el objetivo de ganar simplicidad. Sin embargo cada consulta consume gran cantidad de recursos de red y procesamiento dado que una consulta es propagada por todos los enlaces de cada uno de los nodos de la red. Esta técnica evidencia una inundación de consultas por toda la red. Para aliviar un poco tal inundación se utiliza un parámetro TTL (time to live) que determina el tiempo de vida de una consulta y que se decrementa cada que es reenviado entre nodos, una vez llega a cero, el paquete es destruido [9].

3.1.1.2 Técnica RBFS

La técnica RBFS [13] es similar a la BFS, ésta utiliza un algoritmo probabilístico para determinar un subconjunto de vecinos por dónde reenviar una consulta. Con ésto se logra optimizar el uso de los recursos de red, dado que se selecciona sólo un subconjunto de nodos, pero por otro lado, puede suceder que nunca se alcancen segmentos de la red en donde puede haber documentos importantes que concuerden con la consulta.

3.1.1.3 Mecanismo Inteligente de Búsqueda (ISM)

Esta es la técnica propuesta y evaluada en Peerware [9], se basa en que un peer que recibe una consulta se la reenvía a los peer que más han respondido dicha consulta. Para esto se tienen dos componentes:

- Un mecanismo de Perfil de los nodos vecinos, en donde en una tabla de *O(Td)* se almacenan las *T* consultas más recientes para cada peer vecino *d*. Una vez completada la tabla se utiliza la política LRU para retirar las consultas menos frecuentes.
- Una función de relevancia (*RR)* que un nodo P_i utiliza para realizar un ranking en línea de sus vecinos y así determinar a cuáles reenviar una consulta *q*. Para calcular el ranking de cada peer P_i, P_m compara *q* con todas las consultas en la estructura de perfil para las cuales ha habido una respuesta y calcula la función $RR_{Pm}(P_i,q)$ así,

$$RR_{P_m}(P_i,q) = \sum_{j=\text{Consultas respondidas por } P_i} Q_{sim}(q_j,q)^{\alpha} * S(P_i,P_j) \quad (5)$$

en donde Q_{sim} es la similaridad coseno y $S(P_i,q_j)$ es el número de resultados retornados por P_i para la consulta q_j.

Ventajas

- Este sistema tiene en cada nodo un conocimiento local de sus documentos, lo cual hace que no se generen mensajes de publicación de términos a toda la red, sino que cuando una consulta llega a un nodo, éste busca en su índice local.
- A diferencia de la técnica BFS, ISM reenvía solamente a un subconjunto de sus nodos vecinos las consultas que llegan usando una función de relevancia que evalúa localmente a cuáles vecinos deben reenviarse.
- El mecanismo inteligente de búsqueda (ISM) propuesto en Peerware, logra disminuir la cantidad de tráfico que se genera por la inundación de mensajes en una red P2P, en comparación con la técnica BFS. Los resultados de la experimentación muestran que se logra obtener un 90% de los documentos relevantes en una búsqueda, utilizando sólo un 35% de los mensajes que genera la técnica BFS.
- La eficiencia de ISM mejora con el tiempo de uso del sistema, dado que los nodos almacenan conocimiento localmente producto de las consultas que han sido resueltas antes, ésto se puede ver como un tipo de “cache”.

Limitaciones

- Dado que se utiliza una red P2P no estructurada, el tráfico de red generado por una consulta crece de manera significativa, ya que el sistema es inundado de mensajes hasta que sean resuelta una consulta o hasta el os mensajes terminen su tiempo de vida (TTL).
- Es posible que una consulta nunca llegue a ciertas partes de la red en donde hay documentos relevantes.

Aunque se logra disminuir el número de mensajes que se envían por la red, no es posible garantizar un 100% de recuperación de los documentos relevantes.

3.2 Basados en redes P2P estructuradas

Sistemas de indexación distribuido de archivos como Chord [10], CAN [4] y Pastry [11] están basados en redes p2p estructuradas, las cuales permiten realizar búsquedas a través del hash de un objeto, obteniendo como respuesta la dirección IP del nodo que lo contiene.

PeerSearch y ODISSEA son sistemas que se basan en esta técnica, y se describen en más detalle en 3.2.1 y 3.2.3 respectivamente.

3.2.1 PeerSearch

PeerSearch [5] es un sistema P2P que utiliza una extensión a los modelos tradicionales Vector Space Model (VSM) [3] y Latent Semantic Indexing (LSI) [3] soportando búsquedas por contenido y semántica. Utiliza un índice distribuido en conjunto con mecanismos de enrutamiento de consultas. Se representan los documentos y consultas como vectores y se calcula la similaridad coseno entre ellos. Almacena un índice de los documentos utilizando la estructura CAN [4] con su representación vectorial en coordenadas, dando como resultado almacenamiento de índices cercanos en contenido y semántica. El sistema es descentralizado, no tiene un único punto de falla ni jerarquías complejas. Soporta búsquedas por contenido y por semántica que se expresan en lenguaje natural. Peersearch no utiliza mecanismos de inundación de índices ni de consultas como

si lo hace ISM. En este sistema, cada zona de CAN es administrada por un nodo tipo *servant* quien almacena los índices que contienen algunas palabras clave.

Dado un documento A, sus palabras clave más importantes son identificadas utilizando VSM y el índice es publicado en un determinado *servant*. Dada una consulta, ella es reenviada a los *servants* responsables por las palabras claves de dicha consulta. Los dos *servants* (documento y consulta) son utilizados para calcular la correspondencia mediante VSM [5].

En este sistema se propone una extensión a los mecanismos convencionales de Vector Space Model y Latent Semantic Indexing

Ventajas

- Es una técnica determinista puesto que se puede determinar con exactitud los documentos que son relevantes a una búsqueda.
- Necesita buscar en una menor cantidad de nodos con respecto a la técnica de inundación de consultas.
- Adiciona el soporte a búsquedas por semántica, algo que a las demás aproximaciones no le es relevante.

3.2.2 aDICS

aDICS [7] es un sistema que divide un índice invertido de términos en una red P2P. Dicha división es realizada por las letras del alfabeto, en donde un nodo administra el espacio del índice distribuido que corresponde a una sola letra, y existe la posibilidad de que una misma letra sea administrada por más de un nodo pero cada nodo tiene contenido diferente. Cada nodo tiene una tabla de contactos que contiene las direcciones de los nodos responsables de cada una de las 26 letras del alfabeto. De cada documento se extraen los términos más importantes y se publican en la red. Cada nodo tiene el índice de los términos que comienzan con la letra para la cual es responsable. En dicha tabla se almacena la dirección de los nodos que almacenan los documentos que contienen los

términos asociados en la lista. Esto significa que si por ejemplo un nodo *A* es responsable por la letra *m*, guardará en su tabla de índice la dirección de los nodos que almacenen documentos con por ejemplo el término "*matriz*".

3.2.2.1 Ingreso de nodos

Para que un nodo ingrese a la red debe conocer por lo menos la dirección de un nodo que ya sea participante para utilizarlo como punto de conexión. Se debe determinar si el nuevo nodo hará parte del índice distribuido. Para esto se utiliza una función que recibe como entrada los recursos del nodo (CPU y ancho de banda). En tal caso que el nodo debe ser parte del índice distribuido, se determina cuál letra del alfabeto le corresponde. Lo anterior puede verse en el siguiente algoritmo:

```
Si (ManejarIndice(P(ndx)) {
   Si (TablaIncompleta) {
          LT = SelecLetraEntradaVacía
               AsignarLetra(LT)
   }
   sino {
          LT = SelecLetraAlAzar(P(l))
   }
}
sino {
     LT = ""
}
Retornar LT
```

Figura 8 Algoritmo para que un nodo determine el índice a administrar

3.2.2.2 Publicación de términos

Cada nodo realiza un proceso de indexación local de los términos de cada uno de los documentos que almacena. Posteriormente se extraen los términos más relevantes utilizando técnicas de recuperación de información para luego ser publicados. Los términos son agrupados por la letra inicial y son publicados en el nodo responsable de administrar el índice de cada letra. La información del nodo responsable se encuentra almacenada en una tabla de enrutamiento llamada Tabla de Contactos (TC).

Letra	Dirección del nodo
A	534, 235
B	120
C	765, 543,227
. . .	
Z	140, 176

Tabla 1 Tabla de Contactos en aDICS. El nodo 120 es responsable por administrar el índice de los términos que empiezan con la letra B.

Un nodo responsable por administrar el índice de una determinada letra mantiene una Tabla Índice (TI), en la que se almacena el término como los nodos en donde se encuentra dicho término.

Letra	Dirección del nodo
acacias	221, 786, 223
américa	564, 221, 980
antioquia	564, 290
aparicio	120
arrayanes	143, 221

Tabla 2 Tabla Índice en aDICS. Las palabras acacias y américa se encuentran en uno o mas documentos del nodo 221

3.2.2.3 Resolución de consultas

En una consulta generada desde el nodo N que incluya los términos casa y árbol, se consulta la tabla local de contactos para ubicar los nodos responsables de las letras C y A. Tanto el nodo responsable por A y por C deben buscar en su índice local los nodos que almacenan los documentos que contienen los términos casa y árbol respectivamente. Posteriormente se devuelve la respuesta al nodo N que originó la consulta en donde aparecen los nodos que fueron encontrados. El nodo N realiza finalmente una

intersección de los conjuntos de nodos para casa y árbol, y el resultado de dicha intersección corresponderá a los nodos que contienen los documentos con los dos términos simultáneamente. Dado que para los términos consultados se tenga como respuesta un mismo nodo, en necesario enviar la consulta completa a dicho nodo para evaluarla completamente con los documentos que almacena.

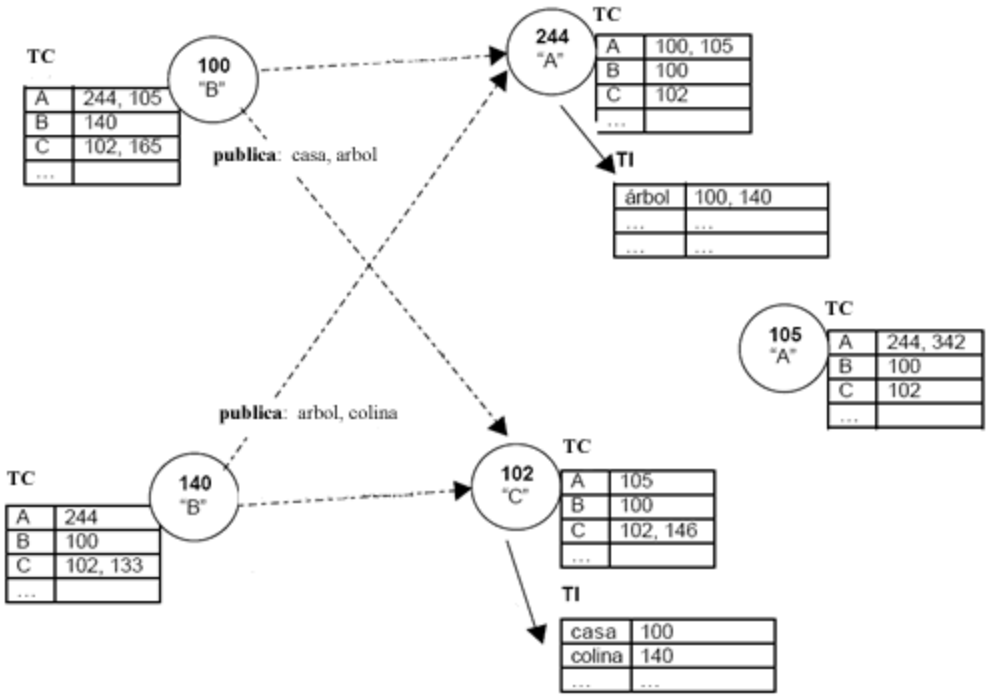

Figura 9 Estructura del sistema aDICS. Los nodos 100 y 140 publican sus términos

Por ejemplo, si el nodo 105 desea resolver la consulta Q = árbol colina, se procede a ubicar los nodos responsable de las letras A y C respectivamente.

NI_A = consulta(TC, "árbol") entonces NI_A = {244, 342}
NIc = consulta(TC, "colina") entonces NIc = {102}

Posteriormente se consulta a los nodos índices correspondientes:

R("árbol") = consulta(244, "árbol") entonces R("árbol") = {100. 140}
R("colina") = consulta(102, colina") entonces R("colina") = {140}

Ahora se determina la conjunción de los dos términos:

$$R = R(\text{“árbol”}) \text{ INTERSEC } R(\text{“colina”}) = \{140\}$$

Por lo tanto se puede deducir que el nodo 140 contiene los términos árbol y colina.

Finalmente se debe enviar la consulta completa a este nodo puesto que no se sabe hasta el momento si los dos términos se encuentran en uno o más documentos.

Ventajas

- Soporta expresiones con múltiples términos y el uso de palabras incompletas.
- Es una alternativa al uso de Tablas Hash Distribuidas (DHT) y a los mecanismos de inundación.

Limitaciones

- El algoritmo que determina si un nodo participa como índice o no, se basa en una función aleatoria. Esto hace que la carga de los nodos responsables por no sea uniforme.
- No explora otras alternativas para la determinación de si un nodo debe ser nodo índice o no para determinar cuál es más eficiente en términos de tráfico y tiempo de respuesta.

3.2.3 ODISSEA

ODDISEA [6] es un sistema p2p que utiliza un índice global distribuido de los términos indexados para los documentos que se encuentran en cada nodo. El sistema está basado en una estructura DHT (Pastry) [6,11], en donde cada objeto se identifica por el hash del nombre. Por ejemplo para el término “silla” se calcula la función hash de la cadena

indice://silla y posteriormente se asigna a un determinado lugar en la DHT. La arquitectura del sistema se divide en dos capas así:

Clientes de actualización: Esta capa se encarga de agregar al sistema documentos nuevos o actualizados mediante su indexación y almacenamiento. Un nodo de este tipo podría ser por ejemplo un crawler que inserta páginas web tejidas, un servidor web que coloca documentos en el índice o un nodo de un sistema para compartir archivos.

Clientes de consulta: En esta capa se diseñan planes de ejecución de consultas basados en estadísticas de frecuencia y correlación de términos para luego ser emitidos a la otra capa.

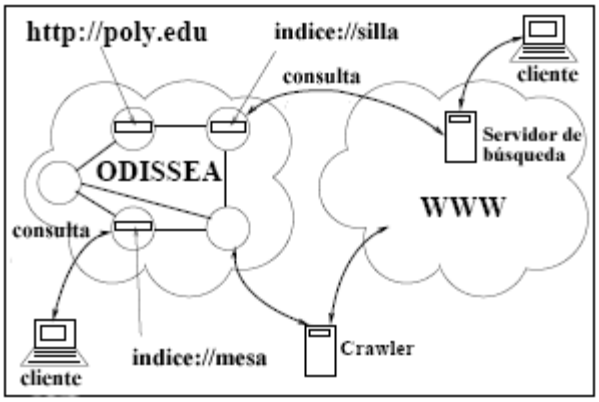

Figura 10 Arquitectura de ODISSEA

4 APSE

4.1 Arquitectura

En la capa 1 se utiliza una red P2P estructurada que suministra un API para realizar las operaciones básicas de red y permite un enrutamiento basado en identificadores. A cada peer se le calcula un identificador mediante la función de hash (SHA-1) de su dirección IP que lo identificará en la red y se le delegará la responsabilidad de manejar un índice invertido de los términos que han sido indexados en la red y que pertenecen al espacio de identificadores asociados a él.

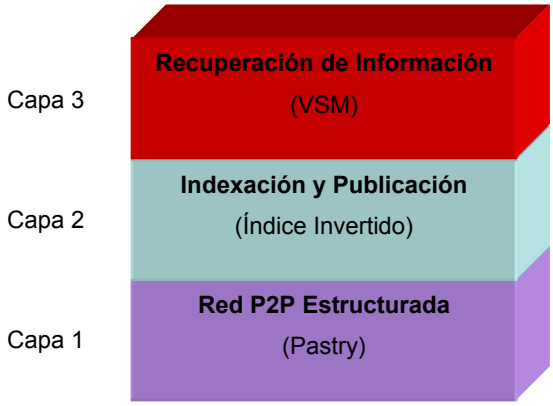

Figura 11 Arquitectura por capas del sistema APSE

En la capa 2 se realiza el proceso de indexación de los documentos y se publican los *k* términos más importantes en los nodos responsables del espacio de identificadores del hash de cada término.

En la capa 3 se da una interfase de usuario en donde se ingresan las consultas, se procesan en la red y se retorna el resultado al usuario.

4.2 Diseño de APSE

4.2.1 Componentes del Sistema

Un nodo de APSE está interconectado con los demás nodos de la red a través de un anillo generado por un espacio hash de identificadores. Un nodo está compuesto por un componente que ofrece servicios de indexación para los documentos locales y publicación de sus términos más importantes, un componente encargado de los servicios de red y comunicación, y un componente que ofrece los servicios de distribución de consultas y recepción de resultados.

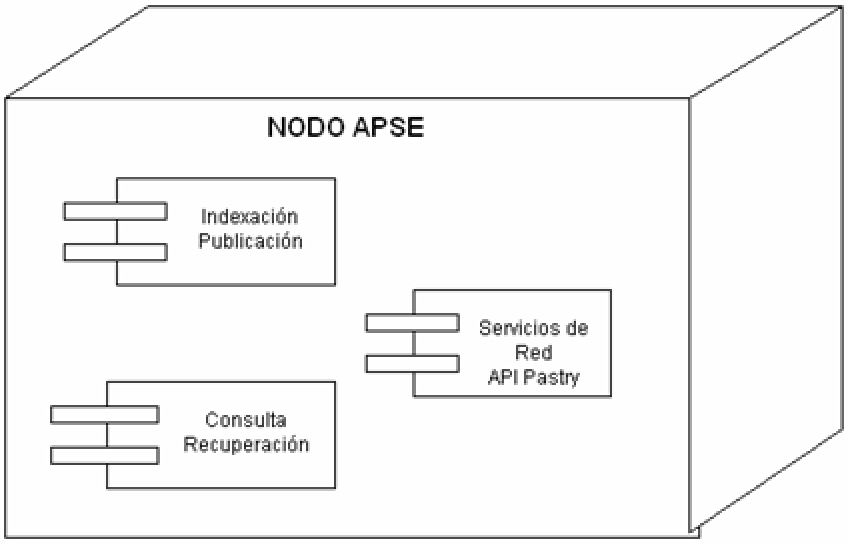

Figura 12 Diagrama de componentes de APSE

4.2.2 Componente de Indexación y publicación

El componente indexador es el responsable de extraer los términos de un documento, descartando los términos que se encuentran en una lista de *stopwords*[3]. También es responsable de calcular la función hash de cada término y publicarlo en la tabla de índices del nodo que corresponda en el espacio de identificadores.

En cada nodo del sistema se almacena un Lexicon local y otro distribuido. El Lexicón local corresponde al índice creado para los documentos que el nodo almacena. El Lexicon distribuido corresponde al índice creado para los términos que han sido publicados por otros nodos y que hacen parte del espacio hash del nodo.

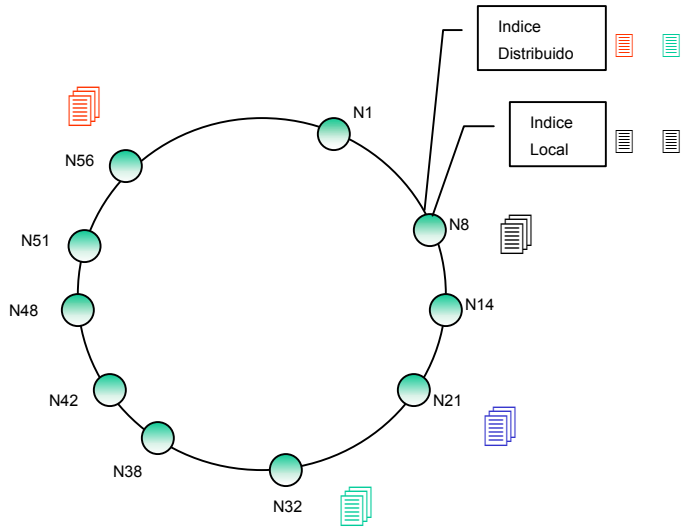

Figura 13 Estructura DHT para publicación de términos indexados en APSE

[3] Palabras usadas frecuentemente en los documentos y que no se consideran dado que no ayudan a diferenciar un documento de otro, como por ejemplo “el”, “la”, “no”, “para”, etc.

4.2.3 Componente de consulta y recuperación

Este componente es responsable por ofrecer una interfase de consulta en donde se extraen los términos de la consulta, se calcula la función de hash a cada uno de ellos y se envía la consulta a través de *FreePastry* [14]. Cuando la consulta llega a un determinado nodo *N*, éste se encarga de buscar dentro de su índice invertido el término consultado (ver sección 2.1). La estructura del índice invertido, reside en memoria para que el tiempo de respuesta sea muy corto. En caso de encontrarse el término, se le retorna al nodo que originó la consulta las direcciones IP de los nodos con sus correspondientes documentos que contienen el término. Una vez se obtiene respuesta para todos los términos, el componente hace un reenvío de la consulta completa a los nodos obtenidos en la intersección (nodos que contienen todos los términos así se encuentren en documentos diferentes) para que en cada uno de ellos se calcule qué tan relevante es cada documento con la consulta. Esta última operación se hace utilizando la medida de Similaridad Coseno descrita en 2.1.2.2. Por último, este componente ofrece un servicio de despliegue de resultados ordenados por la relevancia como lo ofrecen los motores de búsqueda convencionales.

4.2.4 Servicios de red

Este componente es encargado ofrecer el API que encapsula los procesos de enrutamiento de mensajes en la red P2P, búsqueda (*lookup*) del nodo asociado a una llave *k* (hash de un término) y los servicios de ingreso, estabilización y salida de los nodos. Tanto para la publicación como para la búsqueda, se calcula para cada término la función hash para enlutar el mensaje al nodo apropiado.

4.3 Implementación

APSE se desarrolló 100% en java en aproximadamente 2500 líneas de código, de las cuales 1500 son del paquete *co.com.uniandes.apse.pastry.** que corresponden con el motor de búsqueda distribuido y las restantes son de *co.com.uniandes.apse.google.** que

corresponden a la implementación del motor de búsqueda centralizado. La interfaz gráfica de APSE puede apreciarse en la Figura 14 y Figura 15.

Se utilizó como base un *framework* desarrollado en Java llamado *FreePastry* [14] que es una implementación del API Pastry dando soporte a una red p2p estructurada que es una de los principales componentes de APSE.

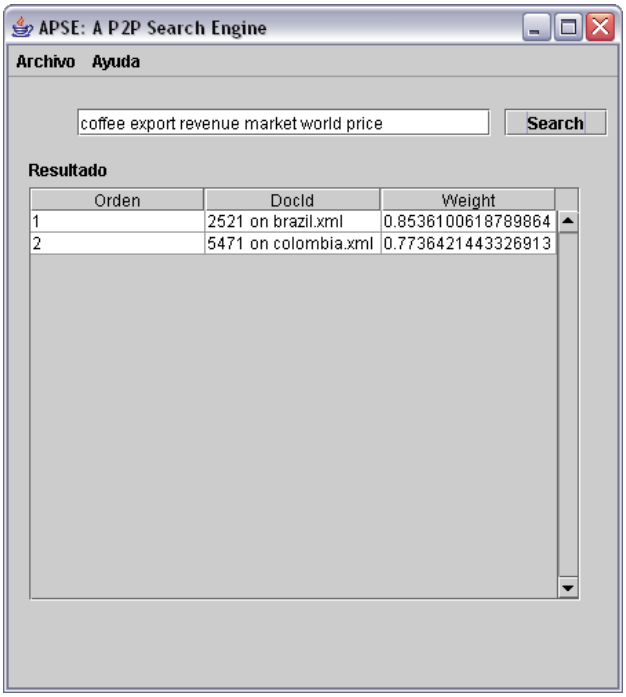

Figura 14 Interfaz gráfica de APSE para el esquema centralizado

La razón por la cual se desarrolló el prototipo en Java es por ser un lenguaje orientado a objetos que brinda suficientes herramientas para trabajar con redes, hilos, soporte a tablas hash muy utilizadas en el sistema (*HashMap, HashSet, TreeMap y TreeSet*).

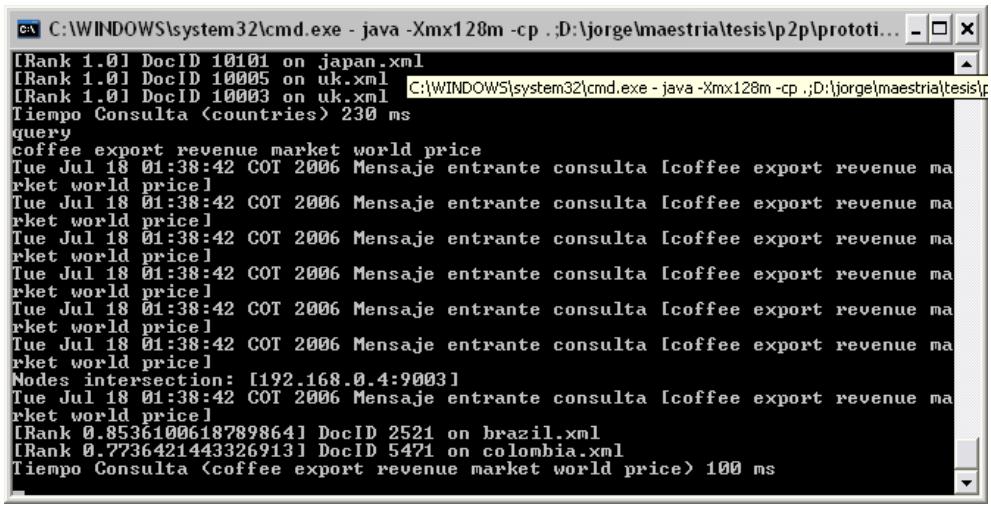

Figura 15 Interfaz de consulta de APSE para el esquema distribuido

5 EVALUACIÓN DE APSE

En este capítulo se hará una comparación de APSE con dos técnicas de implementación de motores de búsqueda: un motor de búsqueda basado en un esquema centralizado y un motor de búsqueda descentralizado basado en una optimización de la red Gnutella llamado ISM [13].

La evaluación gira en torno al desempeño del sistema en términos de *cantidad de mensajes* necesarios para resolver una consulta, *tiempo de resolución* de una consulta, *tiempo de indexación y publicación*, y por último los documentos recuperados para una determinada consulta.

Ya que una comparación directa entre estas tres aproximaciones no es posible realizarla por las diferencias sustanciales de los enfoques, se procederá a evaluar los aspectos clave que se comparten entre ellas.

5.1 Evaluación analítica

5.1.1 Desempeño del esquema centralizado

En el esquema centralizado no hay reenvío de mensajes para la resolución de una consulta. El índice sobre todos los documentos se encuentra en un único lugar convirtiéndolo en un sistema altamente eficiente en tráfico de mensajes en la red. Prácticamente se puede decir que se requieren un par de mensajes para resolver una consulta.

Con respecto al tiempo de respuesta para resolver una consulta, el esquema centralizado es también muy eficiente (por el orden de 10 milisegundos) por la misma razón de no tener que buscar en ningún otro nodo.

El tiempo de indexación y publicación si es uno de los aspectos en donde el sistema no es muy eficiente. Esto es porque hay un Spider que teje la red en busca de todos los documentos para extraer los términos de cada uno de ellos para luego ser indexados en el Lexicón. Este proceso toma una gran cantidad de tiempo y además no hay garantía de tener el índice actualizado al 100% en todo momento.

En términos de recuperación de documentos relevantes a una consulta, el esquema centralizado es muy eficiente ya que se puede garantizar que el 100% de los documentos son recuperados.

5.1.2 Desempeño de ISM

La técnica ISM [13] es una optimización de la técnica BFS (algoritmo base de Gnutella). Con respecto al número de mensajes que viajan por la red, esta técnica utiliza por el orden del 54% de mensajes de la técnica BFS que es:

$$\sum_{i=1}^{n}(d_i - 1) \quad (6)$$

en donde d_i es el número de conexiones que tiene cada uno de los n nodos de la red Gnutella.

Ya que la técnica está basada en la inundación de mensajes con un parámetro TTL (Time To Live) y otro parámetro d_i, el número de mensajes es significativo. Por ejemplo, para resolver una consulta de 4 términos en una red con $n=104$ nodos con un grado $d_i=8$ el número total de mensajes es *#msgs=104*(8-1)=708* [13].

El tiempo de resolución de consultas de BFS está por el orden de los 6 segundos para un TTL=4, ISM usa entre el 30%-60% de ese tiempo [13].

El tiempo de publicación e indexación es similar al del esquema centralizado, dado que cada nodo guarda únicamente el índice invertido de sus documentos.

El *recall* depende en gran medida del parámetro TTL y del grado de conexiones de los nodos. Eso significa que para alcanzar un 100% de *recall* se aumenta la cantidad de mensajes en el sistema.

5.1.3 Desempeño de APSE

El número de mensajes del sistema es determinado por la ecuación 7.

$$\#msgs = 2(n + in\sec(n,k)) \quad (7)$$

en donde *n* es el número de nodos e *insec(n)* es el número de nodos intersección para una consulta de *k* términos. Por ejemplo, para una red con *n=104* una consulta con *k=4* términos el número máximo de mensajes es *#msgs=2*(104+4)=216*.

El tiempo de respuesta del sistema a una consulta toma por el orden de los 30 milisegundos, resultado que hace al sistema completamente viable con respecto al esquema centralizado y mucho mejor que ISM.

El tiempo de publicación se ve afectado por el hecho de distribuir el índice en la red. Este tiempo es mayor a las otras dos técnicas dado que ellas almacenan localmente su índice invertido.

El *recall* del sistema es el mismo que en el esquema centralizado, dado que se garantiza que todos los nodos son alcanzables en el anillo Pastry y por otro lado el índice centralizado de la técnica centralizada se encuentra ahora distribuido en todos los nodos de la red P2P.

5.2 Evaluación experimental

La experimentación del sistema busca demostrar las premisas evaluadas en el capítulo anterior. Para lo cual se realizó la experimentación descrita a continuación.

5.2.1 Marco experimental

Los experimentos fueron realizados en una red LAN 10/100 conformada por 20 PCs, cada uno con 2 Gb de memoria RAM y procesador Intel Pentium 4 de 3.4 GHz con sistema operativo Windows XP.

Las métricas de evaluación fueron: (i) Tiempo de indexación y publicación de documentos (ii) Tiempo de resolución de consultas (iii) Número de mensajes en el sistema para resolver una consulta y (iv) fracción de documentos relevantes recuperados para una determinada consulta.

Como documentos del sistema se utilizó una clasificación del *corpus de Reuters-21578* [12] generada con *dataGen* [13], dataset que fue utilizado para evaluar Peerware. El *dataset* 21578 ha sido utilizado en varias investigaciones de recuperación de información. El criterio de clasificación del *dataset* está basado en sitios con clúster de documentos por país. Hubo 104 archivos que representan a 104 países, con un total de 22.769 documentos (elementos XML). Cada archivo corresponde con las noticias de un determinado país y se encuentra en formato XML, en donde cada registro es una noticia y representa un documento en el sistema. La estructura de cada documento es ilustrada en la Tabla 3.

```
<REUTERS ID=”213”>
   <DATE> 2-ABR-1987 08:32:34.03</DATE>
   <PLACE>COLOMBIA</PLACE>
   <TEXT>
      <TITLE>The international coffee price…</TEXT>
      <BODY>Yesterday the international…</BODY>
   </TEXT>
</REUTERS>
```

Tabla 3 Registro de ejemplo de Reuters que representa un documento en APSE

En cada configuración de prueba se inició una instancia de APSE desde una consola de comandos de Windows, escuchando por el puerto TCP 9001, como se puede observar en la Tabla 4.

java -cp .;C:\temp\prototipo\lib\FreePastry-1.4.2.jar;C:\temp\prototipo\lib\pastry-1.4.2.jar;C:\temp\prototipo\lib\xmlpull_1_1_3_4a.jar;C:\temp\prototipo\lib\xpp3-1.1.3.4d_b2.jar;C:\temp\prototipo\lib\jdom.jar co.com.uniandes.apse.pastry.ApsePeer 9001 10.2.11.16 9001

Tabla 4 Ejecución de APSE desde la línea de comandos

5.2.2 Metodología

En cada experimento se mantiene constante la red p2p de tal manera que si por ejemplo se inicia el sistema con 5 nodos, no se varía dicha cantidad durante la prueba. La idea es mantener un entorno experimental controlado, para que las mediciones puedan ser tomadas sin depender de variables como por ejemplo la salida e ingreso de nodos a la red.

Para realizar la experimentación de eficiencia del sistema se creó un *dataset* de consultas conformadas por uno hasta 10 términos (5 de cada uno) que se describen en el Anexo 1. El objetivo de este conjunto de consultas es enviar 5 veces cada experimento para tomar mediciones y posteriormente calcular su promedio. La razón por la cual se hizo necesario tomar un valor promedio de las mediciones es porque una consulta es resuelta en pocos milisegundos y se puede ver afectada por cualquier proceso que esté corriendo la máquina y también porque los valores (por ejemplo el tiempo de respuesta) pueden verse afectados por la consulta realizada (por ejemplo, no es lo mismo buscar una palabra frecuente que una que no lo es). En este experimento se realizó la ejecución de las mismas consultas en el motor de búsqueda implementado bajo el esquema centralizado para compararlo con el motor de búsqueda distribuido.

En cada configuración se inicia la red con un nodo *bootstrap* [13] que es utilizado en FreePastry para que los demás nodos ingresen al anillo Pastry a través de él.

Para cada prueba en cada nodo se enviaron a un archivo todos los mensajes de log del sistema, como por ejemplo los mensajes generados para resolver una consulta, el tiempo de resolución, tiempo de publicación y resultados obtenidos por el sistema para cada una de las consultas.

Finalmente se recolectaron los archivos generados en cada nodo para ser procesados por una serie de *scripts* de *shell* desarrollados en Linux para realizar el respectivo procesamiento.

5.2.3 Resultados

5.2.3.1 Tiempo de indexación y publicación

En el experimento realizado se eligió indexar y publicar los documentos pertenecientes a un país (*colombia.xml*) que consta de 53 documentos. Se realizó la indexación y publicación en escenarios de un sistema conformado por 3 nodos incrementando de a un nodo hasta alcanzar 20 nodos, como se puede ver en la Figura 16. Un nodo hizo las veces de indexador y publicador en todas las configuraciones.

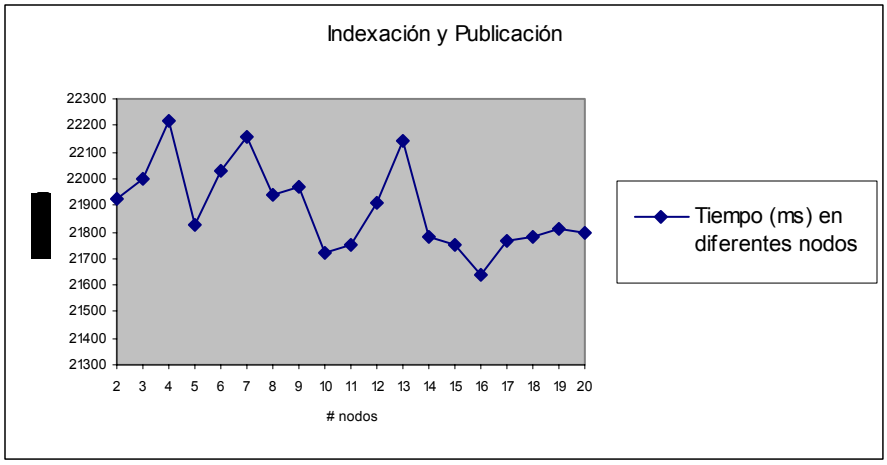

Figura 16 Tiempo utilizado por APSE para indexar y publicar los documentos en configuraciones desde 3 hasta 20 nodos

Como resultado se puede observar en la Figura 16 que a medida que se incrementa el número de nodos en el sistema, el tiempo de indexación y publicación disminuye de manera moderada. Para una configuración de 3 nodos el tiempo consumido es de 22 segundos y el tiempo para una configuración de 20 nodos fue de 21,4. Se puede observar además que hay algunos picos de tiempo que pudieron ser generados por algún proceso que se ejecutó en background en alguno de los PCs. La razón por la cual el tiempo disminuye se puede atribuir al hecho de que se distribuye la carga de publicación en todos los nodos del sistema. En este ambiente experimental hay que tener en cuenta que hay condiciones ideales con respecto a la velocidad de la red LAN, por lo tanto en un ambiente real en donde hay redes WAN con diferentes velocidades de conexión los tiempos de publicación se espera que aumentan en la medida que los nodos que conforman el sistemas se encuentren en diferentes topologías de red.

5.2.3.2 Eficiencia de resolución de consultas en términos de tiempo

Para el experimento se utilizó el *dataset* de consultas definido al comienzo de este capítulo, con configuraciones en donde se incrementaron el número de nodos y la cantidad de términos de las consultas. Se calculó el promedio del tiempo en las mediciones para las 5 consultas definidas de *n* términos. Los 104 archivos fueron distribuidos al azar de manera proporcional en cada configuración, por ejemplo, para una red de 5 nodos se asignó a cada uno de ellos aproximadamente 20 archivos. Tan pronto todos los nodos se encontraban en ejecución se procedió a realizar la indexación y publicación en el sistema. Una vez todos los nodos terminaban el proceso de publicación en cada configuración, se ejecutó desde un nodo todo el conjunto de consultas de manera secuencial con un intervalo de 5 segundos entre cada consulta.

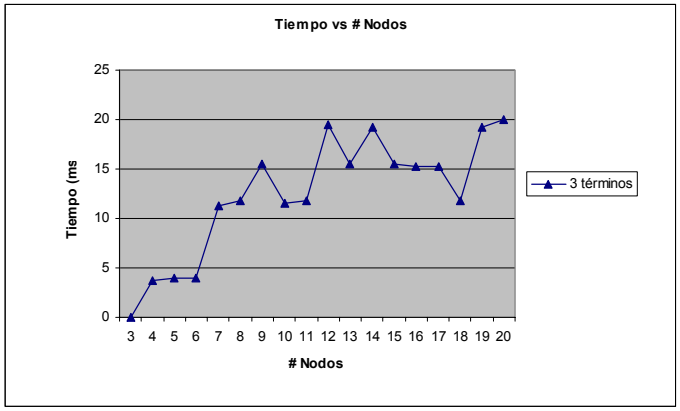

Figura 17 Tiempo utilizado para resolver una consulta de 3 términos incrementando la cantidad de nodos en el sistema.

Para resolver una consulta de 3 términos el sistema aumenta el tiempo de respuesta cuando se incrementa la cantidad de nodos, como se puede ver en la Figura 17. La razón por la cual ésto sucede es porque los documentos están cada vez más distribuidos en los nodos, haciendo que el nodo que origina la consulta tome más tiempo resolviendo la intersección de los nodos que contienen los documentos relevantes a la consulta.

Los resultados para todas las configuraciones incrementando el número de nodos pueden verse en la Figura 18.

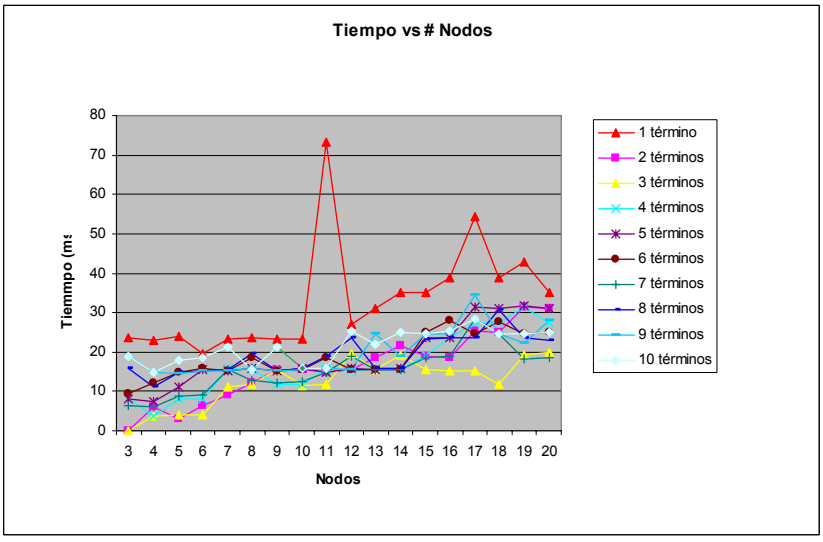

Figura 18 Tiempo utilizado para configuraciones incrementando el número de nodos.

A medida que se incrementa la cantidad de términos en las consultas con una configuración de 5 nodos el tiempo de respuesta aumenta, ver Figura 19. Esto sucede porque la intersección de documentos realizada en cada uno de los nodos es mayor a medida que aumenta la cantidad de términos, por lo tanto el procesamiento es mayor en cada nodo del sistema. Para las consultas de un solo término puede observarse que el tiempo de respuesta es superior en todas las configuraciones, esto puede atribuirse a que para las consultas de un solo término hay muchos documentos que la contienen.

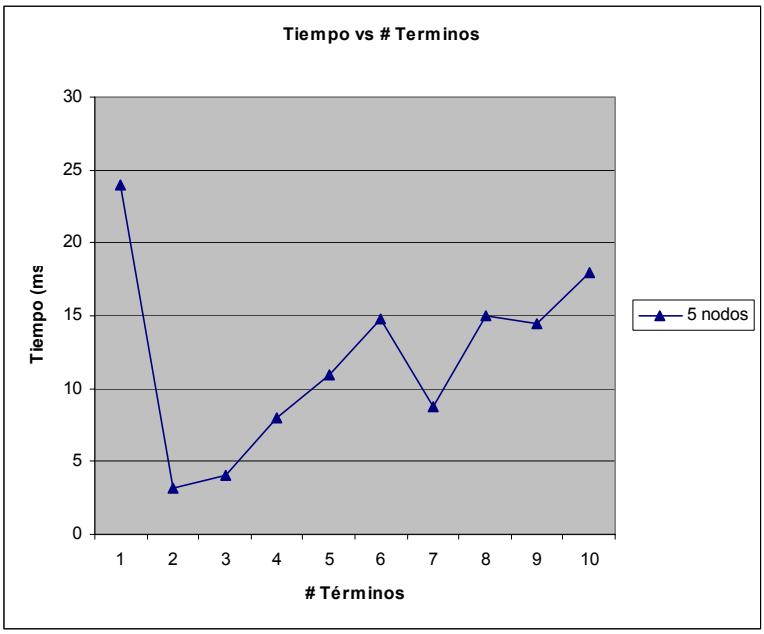

Figura 19 Tiempo utilizado para resolver una consulta en una configuración del sistema con 5 nodos incrementando la cantidad de términos de una consulta.

Los resultados para todas las configuraciones incrementando el número de términos en las consultas pueden verse en la Figura 20.

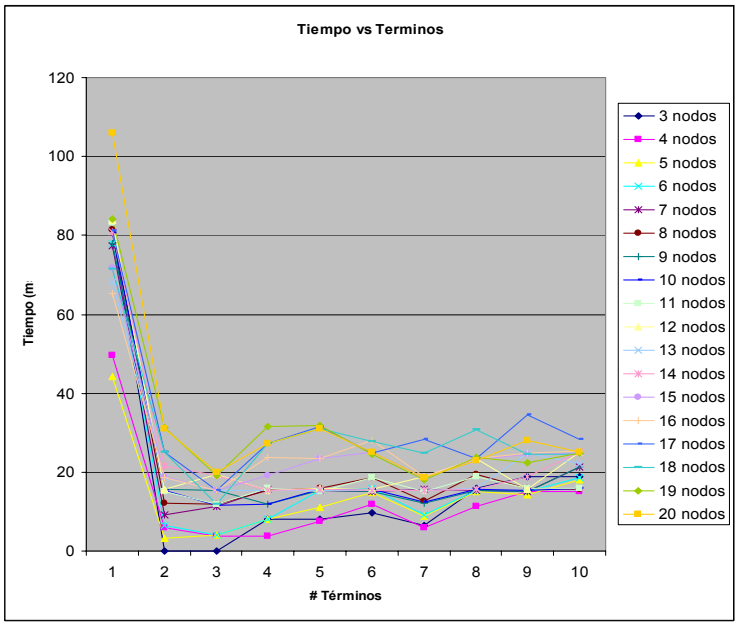

Figura 20 Tiempo utilizado para resolver consultas incrementando el número de términos.

Se realizó la ejecución del conjunto de consultas utilizando el sistema centralizado que se implementó y se obtuvieron tiempos de respuesta inferiores a 6 milisegundos, ver Figura 21. Esto se debe a que en el esquema centralizado no es necesario realizar operaciones de intersección de conjuntos de nodos ni tampoco envío de mensajes por la red, obteniendo una eficiencia en tiempo de respuesta superior a la de APSE, ver Figura 22.

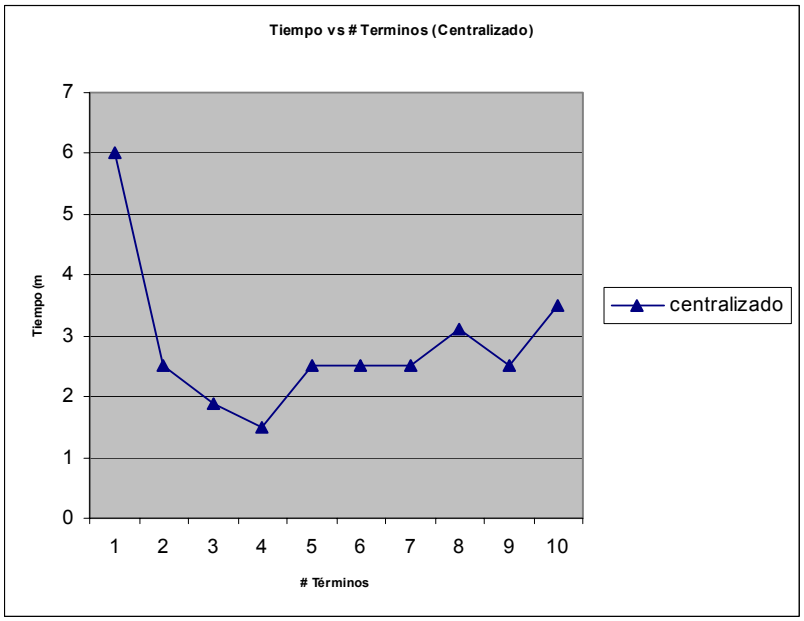

Figura 21 Tiempo utilizado por el sistema centralizado para consultas incrementando la cantidad de términos

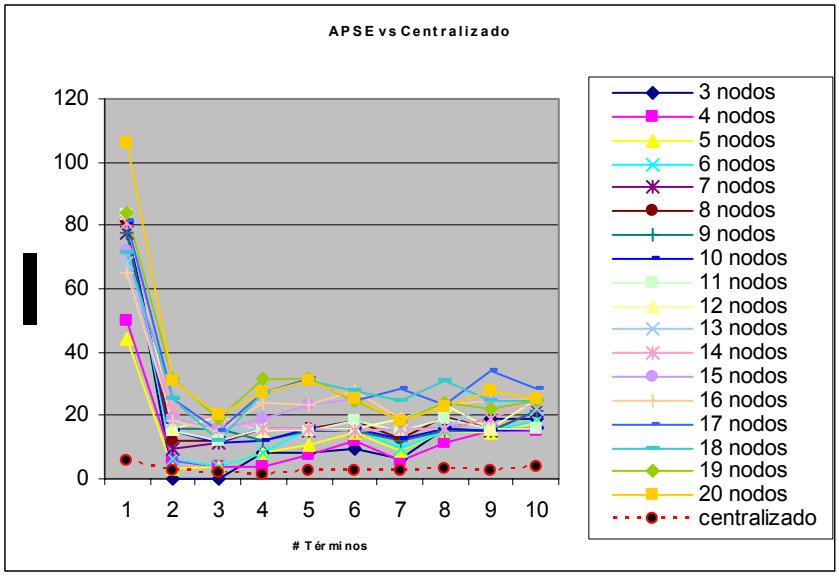

Figura 22 Tiempo de respuesta del sistema centralizado versus APSE

5.2.3.3 Eficiencia de resolución de consultas en términos de tráfico de mensajes

La medición del número de mensajes enviados en el sistema se realizó para comparar los resultados esperados que genera la ecuación 7 descrita en la sección 5.3.

Como resultado de la experimentación en la Figura 23 se observa que la cantidad de mensajes reales generados para resolver una consulta de 1 término son inferiores a los esperados

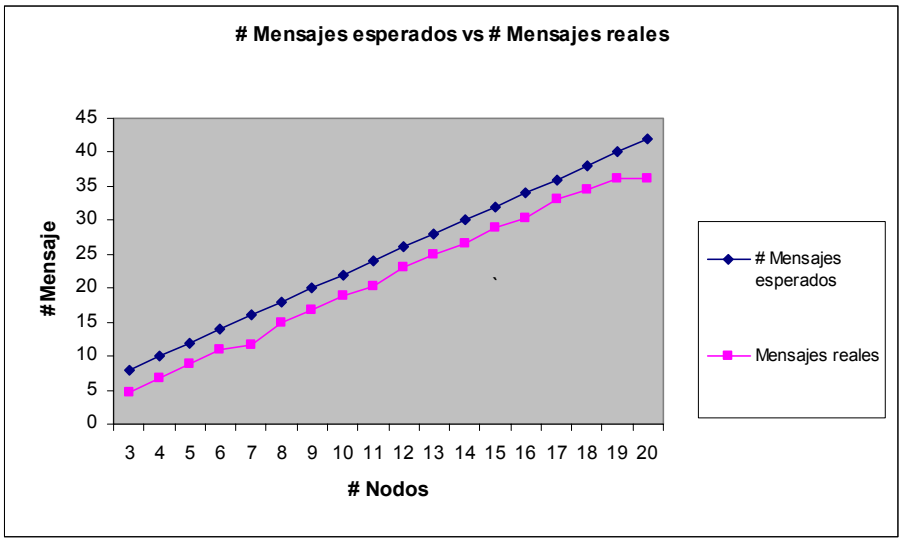

Figura 23 Cantidad de mensajes esperados versus cantidad mensajes obtenidos en la experimentación para consultas de 1 término

Como resultado de la experimentación en la Figura 24 se observa que la cantidad de mensajes reales generados para resolver una consulta de 2 términos son inferiores a los esperados.

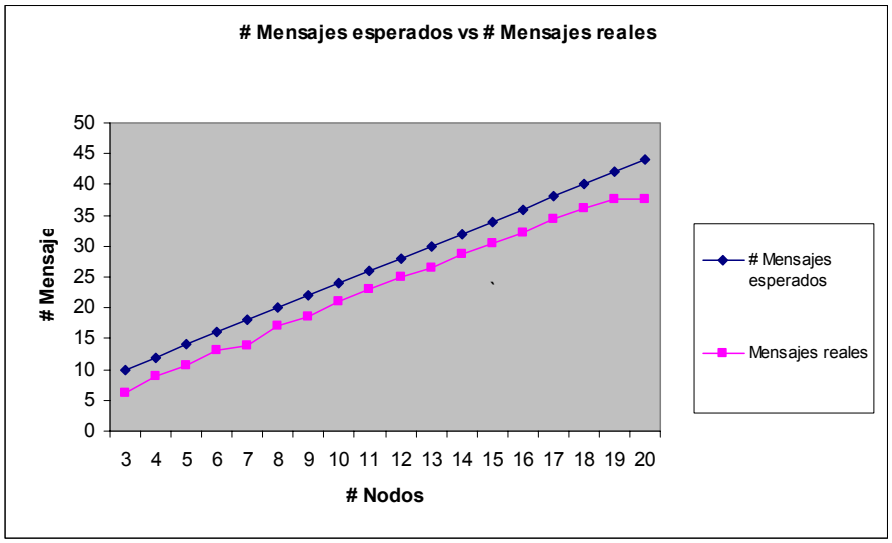

Figura 24 Cantidad de mensajes esperados versus cantidad mensajes obtenidos en la experimentación para consultas de 2 términos

Como resultado de la experimentación en la Figura 25 se observa que la cantidad de mensajes reales generados para resolver una consulta de 20 términos son inferiores a los esperados.

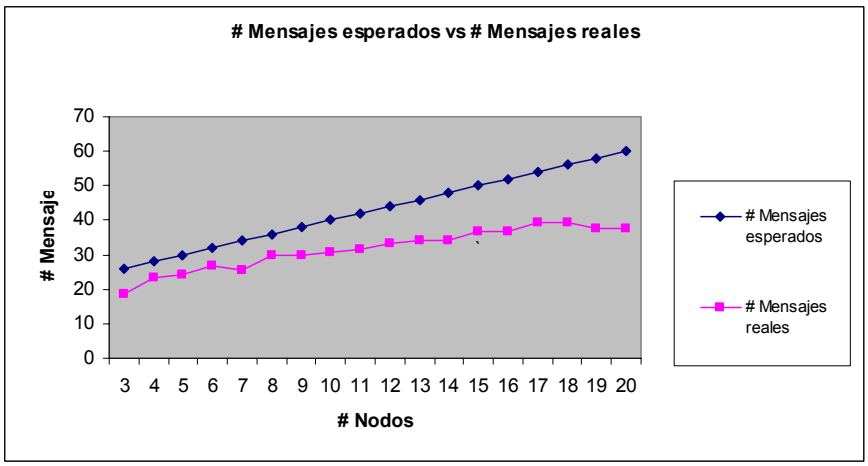

Figura 25 Cantidad de mensajes esperados versus cantidad mensajes obtenidos en la experimentación para consultas de 20 términos

Finalmente, se comprueba que la cantidad de mensajes generados por el sistema nunca supera lo esperado.

5.2.3.4 Recuperación de documentos relevantes

Para este experimento se ejecutó una serie de consultas en el motor de búsqueda centralizado y se comparó con APSE obteniendo un 100% de los documentos esperados, ver Anexo C.

6 CONCLUSIONES Y TRABAJO FUTURO

Los motores de búsqueda centralizados son eficientes en la resolución de consultas, sin embargo no garantizan un 100% de actualización en cualquier instante del tiempo, al no poder garantizar que el Lexicón ha sido refrescado por los spider que tejen la red.

Con el trabajo realizado se confirma que implementar un motor de búsqueda distribuido en una red P2P, al menos en una red local, es completamente viable. El enfoque distribuido del sistema propuesto es una alternativa a los motores de búsqueda convencionales y ofrece un sistema escalable que no tiene un único punto de falla como el esquema centralizado.

Aunque los tiempos de respuesta para resolución de consultas en el sistema centralizado son inferiores a los de APSE, se encuentran por el orden de milisegundos, lo cual hace a APSE comparable con los motores de búsqueda centralizados en términos de eficiencia en tiempo de respuesta.

Con el sistema propuesto se logra dar soporte a consultas basadas en contenido como lo hacen los motores de búsqueda centralizados. El número de mensajes generados por el sistema para resolución de consultas es inferior comparado con aproximaciones basadas en inundación de mensajes tipo Gnutella. La cantidad de documentos relevantes a una consulta es el mismo que las aproximaciones centralizadas y se garantiza el mismo porcentaje de *recall.*

Como trabajos futuros se pueden hacer experimentaciones con más nodos y en entornos más amplios como redes WAN. Hay problemáticas que están fuera del alcance del

presente trabajo y que pueden ser objeto de estudio posteriormente, como por ejemplo ingreso y salida de nodos en el sistema, replicación de los índices distribuidos para dar soporte a tolerancia a fallos, publicación de nuevos documentos en tiempo de ejecución, entre otros.

APSE podría ser por ejemplo implementado en una red de servidores web de una universidad, ofreciendo así a la comunidad universitaria un motor de búsqueda actualizado y distribuido. Otra posible implementación puede ser una red académica inter-universitaria, en donde docentes compartan sus documentos, presentaciones, material de clase y artículos a través de APSE.

BIBLIOGRAFIA

[1] Sergey Brin and Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine, 1998.

[2] Fernando Bordignon y Gabriel Tolosa. Redes compañero a compañero: conceptos y tendencias de aplicación. (2003). Novática. (España), No. 166, nov-dic 2003.

[3] Berthier Ribeiro-Neto, Ricardo Baeza-Yates. Modern Information Retrieval. Addison Wesley; 1st edition (May 15, 1999).

[4] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, y Scott Shenker. A scalable content-addressable network. In Proc. ACM SIGCOMM 2001, August 2001.

[5] Chunqiang Tang, Zhichen Xu y Mallik Mahalingam. PeerSearch: Efficient Information Retrieval in Peer-to-Peer Networks. HotNets'02, 2002.

[6] Torsten Suel, Chandan Mathur JoWen,Wu Jiangong Zhang,Alex Delis, Mehdi Kharrazi, Xiaohui Long y Kulesh Shanmugasundaram. ODISSEA: A PeertoPeer Architecture for Scalable Web Search and Information Retrieval. 6th International Workshop on the Web and Databases (WebDB), June 2003.

[7] Gabriel Tolosa, Fernando Bordignon y Jorge A Peri. “aDICS: Modelo de Índice Distribuido sobre una Red P2P para Búsquedas por Contenido”. X Congreso Argentino de Ciencias de la Computación, Octubre 2004.

[8] Gnutella, http://gnutella.wago.com

[9] D. Zeinalipour-Yazti, V. Kalogeraki y D. Gunopulos, "Exploiting Locality for Scalable Information Retrieval in Peer-to-Peer Systems", Information Systems Journal, Elsevier Publications, Volume 30, Issue 4, Pages 277-298, 2005.

[10] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications." Proc. of ACM SIGCOMM'01, San Diego CA, 2001.

[11] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November, 2001.

[12] Hettich, S. and Bay, S. D. (1999). The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science.

[13] V. Kalogeraki, D. Gunopulus and D. Zeinalipour-Yasti, "A Local Search Mechanism for Peer-to-Peer Networks". Proc. Of CIKM'02, McLean VA, USA, 2003.

[14] FreePastry: Rice University. (http://freepastry.rice.edu/FreePastry/)

ANEXOS

ANEXO A

Dataset de consultas utilizado en la experimentación

Cantidad de Términos	Consulta
1	▪ countries ▪ spending ▪ growing ▪ executive ▪ sharp
2	▪ algeria 1987 ▪ fact forecast ▪ standard extended ▪ support equipment ▪ export program
3	▪ usda general sales ▪ computer home internet ▪ computer graphics markets ▪ oil build market ▪ sharp mark buy
4	▪ switzerland austria sweden norway ▪ export coffee europe roasted ▪ group growth month states ▪ into yen markets industry ▪ management improvement international monetary
5	▪ financial year net export growth ▪ bank economists business sector investors ▪ commercial international market buy price ▪ company restructuring program sale business ▪ organisation petroleum exporting countries company
6	▪ arbitration commission nationally to all workers ▪ coffee export revenue market world price ▪ news conference opposition tax increases program

	▪ policy enterprise management reform economic reduce ▪ national statistics institute inflation annual rate
7	▪ eurodollar and us treasury bond contracts spokesman ▪ lower export registration price reintegro coffee export ▪ federal capital accords development technical projects investment ▪ profits schilling lending profit interest federal budget ▪ open export international coffee organization yesterday market
8	▪ agricultural producers to which australia and thailand belong ▪ program sector products sector demand growth economy bussines ▪ cocoa pact stock manager purchasing international coffee organization ▪ world bank network countries emergency repair program return ▪ materials state firms free market prices fixed quota
9	▪ following the community's new dynamic of integration he says ▪ american caribbean sugar exporting countries years continuous low prices ▪ financial assets physical production facilities textile plants buyers demands ▪ commercial banks registered government securities dealers securities accounts bills ▪ increase purchases parts car makers contracts commerce suppliers firms
10	▪ united states and japan will soon settle their trade dispute ▪ economic policies government public sector administration inflation budget growth forecasts ▪ market participation crop could exports production tonnes similar down similar ▪ policy exchange rates fluctuation ranges possible accompanied least coordinated interest ▪ andriessen proposed freeze most prices coupled reductions other support mechanisms

ANEXO B

Tabla de tiempos experimentales para resolver consultas incrementando el número de nodos y la cantidad de términos

Términos / Nodos	1	2	3	4	5	6	7	8	9	10
Centralizado (1)	6	2.5	1.875	1.5	2.5	2.5	2.5	3.125	2.5	3.5
3	23.5	0	0	8	8	9.6	6.4	16	19	19
4	23	6	3.75	3.75	7.5	12	6	11.25	15	15
5	24	3.2	4	8	11	14.8	8.8	15	14.4	18
6	19.5	6.4	4	8	15.5	15.8	9.2	15.25	15.2	18.6
7	23.25	9.2	11.25	15.75	15.25	15.2	15.6667	15.5	15.2	21.2
8	23.5	12.2	11.75	15.5	16	18.6	12.8	19.5	16	15.6
9	23.25	15.6	15.5	11.75	15.25	15.2	12.2	15.25	15.2	21.4
10	23.25	15.4	11.5	11.75	15.75	15.6	12.4	15.75	15.4	15.8
11	73.25	15	11.75	16	15	18.6	15	19	15.8	15.8
12	27	15.6	19.5	15.25	16	15.6	19	23.5	15.6	25.2
13	31	18.6	15.5	15.5	15.5	15.4	15.5	16	24.8	22
14	35	21.6	19.25	15.5	15.75	15.4	15.5	15.75	18.8	25
15	35	18.8	15.5	19.25	23.5	25	18.6	23.25	24.6	24.8
16	38.75	18.6	15.25	23.75	23.5	28	18.8	23.5	24.8	25.4
17	54.5	25.2	15.25	27.25	31.5	24.8	28.2	23.5	34.4	28.2
18	38.75	25	11.75	27.25	31	27.8	24.8	30.75	24.6	24.6
19	42.75	31.4	19.25	31.5	31.75	24.6	18.2	23.75	22.4	24.8
20	35	31	20	27.25	31	25	18.6	23	28	25

ANEXO C

Tabla con los documentos recuperados relevantes a cada consulta

Consulta	Documentos Esperados (Usando el Sistema Centralizado)	Documentos encontrados (Usando APSE)
algeria 1987	1 19285 on saudi-arabia.xml 0.9999999999999999 2 3455 on saudi-arabia.xml 0.9999999999999998 3 5635 on algeria.xml 0.98058067569092 4 176 on algeria.xml 0.98058067569092 5 16601 on algeria.xml 0.98058067569092 6 14512 on algeria.xml 0.98058067569092 7 17167 on algeria.xml 0.9486832980505138 8 11645 on algeria.xml 0.9486832980505138 9 10444 on algeria.xml 0.9486832980505138 10 2522 on venezuela.xml 0.8574929257125441	[Rank 0.9999999999999999] DocID 19285 on saudi-arabia.xml [Rank 0.9999999999999998] DocID 3455 on saudi-arabia.xml [Rank 0.98058067569092] DocID 5635 on algeria.xml [Rank 0.98058067569092] DocID 176 on algeria.xml [Rank 0.98058067569092] DocID 16601 on algeria.xml [Rank 0.98058067569092] DocID 14512 on algeria.xml [Rank 0.9486832980505138] DocID 17167 on algeria.xml [Rank 0.9486832980505138] DocID 11645 on algeria.xml [Rank 0.9486832980505138] DocID 10444 on algeria.xml [Rank 0.8574929257125441] DocID 2522 on venezuela.xml
usda general sales	1 13097 on algeria.xml 1.0000000000000002 2 13094 on sri-lanka.xml 1.0000000000000002 3 11645 on algeria.xml 1.0000000000000002 4 3942 on iraq.xml 0.9428090415820635 5 12002 on ussr.xml 0.9428090415820635 6 12943 on turkey.xml 0.9271726499455306	[Rank 1.0000000000000002] DocID 13097 on algeria.xml [Rank 1.0000000000000002] DocID 13094 on sri-lanka.xml [Rank 1.0000000000000002] DocID 11645 on algeria.xml [Rank 0.9428090415820635] DocID 3942 on iraq.xml [Rank 0.9428090415820635] DocID 12002 on ussr.xml [Rank 0.9271726499455306] DocID 12943 on turkey.xml
switzerland austria sweden norway	1 12909 on west-germany.xml 0.8333333333333334	[Rank 0.8333333333333334] DocID 12909 on west-germany.xml
financial year net export growth	1 7589 on venezuela.xml 0.8451542547285166 2 227 on australia.xml 0.8164965809277259	[Rank 0.8451542547285166] DocID 7589 on venezuela.xml [Rank 0.8164965809277259] DocID 227 on australia.xml
eurodollar and us treasury bond contracts spokesman	1 16785 on japan.xml 0.8099238707340584 2 16756 on japan.xml 0.8099238707340584	[Rank 0.8099238707340584] DocID 16785 on japan.xml [Rank 0.8099238707340584] DocID 16756 on japan.xml
agricultural producers to which australia and thailand belong	1 21391 on thailand.xml 0.760638829255665	[Rank 0.760638829255665] DocID 21391 on thailand.xml
following the community's new dynamic of integration he says	1 12909 on west-germany.xml 0.5033781631867288	[Rank 0.5033781631867288] DocID 12909 on west-germany.xml
united states and japan will soon settle their trade dispute	1 10781 on japan.xml 0.8795932074125605 2 10665 on japan.xml 0.8795932074125605	[Rank 0.8795932074125605] DocID 10781 on japan.xml [Rank 0.8795932074125605] DocID 10665 on japan.xml