Una aproximación epidémica para el problema de direccionamiento de consultas semánticas en redes p2p estructuradas

Colmenares G. Luis E. y Solís L. Eder

An epidemic approach for the semantic query routing problem in the structured p2p networks

***Abstract*— In this paper, we propose an epidemic model, which is used to search for data in a structured *p2p* network. Structured *p2p* networks, carry out deterministic searches without associating the interests of each node. Their efficiency is demonstrated by performing a comparison between the proposed epidemic algorithm, and the algorithm that uses *DHT* Bamboo. The requests are based on *Zipf's* law, like they were *World Wide Web*. By utilizing epidemic algorithms, it is possible to replicate data, and therefore, add up semantics to the information on the network, and as a result, reduce the number of rounds or hops performed to accomplish such a search, in a distributed system such as a structured *p2p* networks.**

***Keywords*— DHT, fanout, epidemic model, structured p2p.**

***Resumen*— En el presente trabajo se propone un modelo epidémico, que se utiliza para realizar búsquedas de datos en una red p2p estructurada. Las redes p2p estructuradas realizan búsquedas deterministas sin relacionar los intereses de cada nodo. La eficiencia se demuestra realizando una comparación de la propuesta del algoritmo epidémico con el algoritmo DHT que utiliza *Bamboo*. Las peticiones realizadas se basan en la ley de Zipf como si fuera el World Wide Web. Con la utilización de un algoritmo epidémico es posible replicar datos, y así agregar semántica a la información en la red y como resultado se reduce el número de rondas o saltos para las búsquedas en un sistema distribuido como son las redes p2p estructuradas.**

***Palabras clave*— DHT, fanout, epidemic model, structured p2p.**

I. INTRODUCCIÓN

Los Algoritmos Epidémicos tienen gran popularidad en la diseminación de información en sistemas distribuidos de gran escala, particularmente en sistemas punto a punto (*peer-to-peer, p2p*) que funcionan a través de Internet o en redes *adhoc*. Una cuestión importante en el uso de los algoritmos epidémicos y los sistemas *p2p* ha sido la importancia de los intereses de cada nodo a la hora de recibir o compartir una determinada información.

La técnica de diseminación de información se basa en imitar la forma de expansión de las epidemias. Estos algoritmos reproducen la manera en que se difunden las enfermedades contagiosas o los rumores en un entorno social, de manera que un individuo infectado de una enfermedad pasa los gérmenes de la misma a otro individuo con el cual mantiene contacto directo o indirecto. Estableciendo la analogía con un sistema distribuido, una nueva información recibida es distribuida de manera aleatoria a nuevos nodos o procesos, y éstos reproducen nuevamente el mismo ciclo, evitando que haya un único servidor o *cluster* a cargo del proceso de transmitir dicha información [1].

En ciencias de la computación, la utilización de estos algoritmos epidémicos ha sido analizada en aplicaciones como la detección de fallos, la agregación de datos, el monitoreo y descubrimiento de recursos y la replicación de bases de datos [2].

El problema que se aborda en este trabajo es el direccionamiento de consultas semánticas en redes *p2p* [3] mediante la realización de copias de documentos en nodos de una red *p2p* estructurada, es decir, la replicación de datos para simular una *CDN* (*Content Delivery Network*) con la implementación de un algoritmo epidémico.

El problema de direccionamiento de consultas semánticas (*Semantic Query Routing Problem*, *SQRP*) consiste en tener una palabra clave en la consulta y la decisión de un peer y sus peer vecinos de reenviar la consulta. Lo que distingue a éste tipo de búsquedas de otras existentes (tales como: búsquedas de imágenes, texto libre, entre otras.) es que en éstas los datos de búsqueda necesitan tener una descripción semántica asociada con ellos [4].

Se realiza una comparación de la eficiencia del algoritmo con el modelo epidémico infecta por siempre, con el algoritmo de la herramienta *Bamboo*[5], que utiliza una *DHT (Distributed Hash Table).* El escenario es una red *p2p* y se utilizan un número de rondas para llegar a un nodo de la red. Cada nodo de la red representa un servidor de la *CDN* [6].

II. TRABAJO RELACIONADO

Manuscrito recibido el 21 de Mayo de 2012. Este trabajo fue respaldado por la Facultad de Ciencias de la Computación y Vicerrectoría de Investigación de Posgrado de la Benemérita Universidad Autónoma de Puebla.

Colmenares G. Luis E. hasta la fecha se ha de desempeñado como Profesor de Tiempo Completo, se encuentra actualmente realizando investigación y docencia en el área de Sistemas de Tiempo Real, Sistemas Distribuidos y Cómputo Ubicuo en la Facultad de Ciencias de la Computación de la Benemérita Universidad Autónoma de Puebla BUAP, Apdo. Postal J-32, Ciudad Universitaria, Puebla, México. (Teléfono: (52) 222-2 29-55-00 Ext. 7214, e-mail: lecolme@cs.buap.mx).

Solís L. Eder es estudiante de la Facultad de Ciencias de la Computación y actualmente está realizando su trabajo de Tesis.

A. Redes p2p

Las redes *p2p* tienen dos funciones principales: la búsqueda y la compartición de ficheros. Estas redes, se han asociado a la búsqueda de ficheros. En las primeras arquitecturas, se utilizaba el mecanismo del índice central, donde todos los usuarios se registraban en un servidor central que servía para encontrar los contenidos. Las búsquedas se hacían en el servidor central, y las transferencias de datos entre los clientes afectados. Éste tenía el problema de escalabilidad: el servidor central se convertía en un cuello de botella al verse saturado de peticiones de múltiples usuarios.

Para solucionar este problema, la siguiente generación de sistemas *p2p* apareció con la red *Gnutella* [7]. El esquema utilizado es inundar con mensajes hasta que encuentre el contenido solicitado, a estas redes se les llama no estructuradas y consiste en una red de nodos conectados anárquicamente entre sí, los cuales no dependen de ningún servidor centralizado. No obstante, el problema de la localización de los recursos es un problema no determinista. Para encontrar un determinado recurso, un nodo envía un mensaje de búsqueda a cada uno de los nodos a los que está conectado. Estos, a su vez, realizan la misma operación, de forma que la búsqueda se expande por una cantidad de nodos exponencial desde el nodo origen, "inundando" la red de mensajes.

Los retos clave de estos sistemas son:

1) Evitar los cuellos de botella que se pueden producir en determinados nodos y por tanto, se distribuyen las responsabilidades de igual forma entre los nodos existentes.
2) Adaptarse a las continuas entradas y salidas (y también caídas) de nodos. Para ello, hay que dar responsabilidades a los nodos que entran y redistribuyen las responsabilidades de los nodos que salen de la red.

Por último, está la generación de redes *p2p* estructurada, que pueden utilizar las clásicas tablas de *hash*, pero en este caso, los *buckets* de hash son los nodos físicos de la red. Este tipo de servicio *p2p* se denomina comúnmente como *DHT*. Las *DHT* proporcionan mecanismos para añadir, borrar o localizar claves de hash. Se construyen por encima de las redes sobrepuestas (*overlay*) *p2p* y suelen ser eficientes, resistentes a fallos, y autoorganizativas. Encima de estas redes se pueden construir, además, otros servicios interesantes, como enrutamiento y localización de objetos descentralizados, servicios de *multicast* y *anycast* escalables. En particular, la abstracción *DHT* almacena pares clave-valor. El valor siempre se guarda en el nodo de la red *overlay*, al cual le pertenece el hash de la clave. Su estructura interna es un grafo (anillo, árbol, o lugares geográficos), lo cual satisface la condición que el número de saltos necesarios para encontrar un determinado valor en la *DHT* es típicamente O(*log n*), donde *n* es el número total de nodos en el sistema. En las redes *p2p* estructuradas como *Bamboo*, un nodo tiene dos tipos de vecinos lejanos y cercanos. Los vecinos lejanos (*table routing*) y los vecinos cercanos (*leaf set*) ayudan a que el límite máximo sea el O(log *n*) [8].

El uso de estándares abiertos para la creación de redes *p2p* estructuradas basadas en *DHT* como *Bamboo* es uno de los más usados debido a que considera el *churn-rate*[8] . El *churn-rate* es el proceso de conexión y desconexión de los *peers*. Dentro de la *DHT*, el mecanismo de búsqueda, proporciona un modelo con

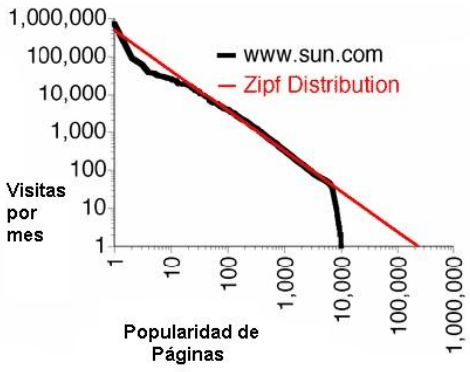

Fig. 1 Ley de Zipf.

búsquedas deterministas, que ocultan al usuario: petición de *routing*, costos de *churn-rate*, balanceo de carga y disponibilidad.

En la actualidad existe software como Emule [9] y Bittorrent [10] que tienen como objetivo el intercambio de ficheros. En el trabajo de Lavastida [11] se hace especial énfasis en la categorización de cada nodo asignándole un peso y se realiza la distribución de la información en nodos específicos.

B. Problema del Enrutamiento de Consultas Semánticas

El problema del enrutamiento de consultas semánticas, consiste en una red representada por el grafo de conexiones (G), un conjunto de palabras-clave (R) distribuida en sus nodos y un conjunto de consultas semánticas (*C*) emitidas dinámicamente por los nodos como sigue:

Una consulta semántica puede ser originada desde cualquier nodo en el tiempo T_0, asumiendo un tiempo de reloj con unidades fijas. Un nodo que origina la consulta o recibe la consulta de otro nodo en el tiempo (T_0+i) puede procesar la consulta localmente y/o reenviar una réplica de la consulta a un conjunto de vecinos inmediatos en el tiempo T_0+i+1. El procesamiento de la consulta termina cuando todas las palabras-clave esperadas son encontradas.

Se busca maximizar la cantidad de recursos encontrados y minimizar la cantidad de saltos realizados por una entidad de búsqueda para encontrar dichos recursos [4, 11].

C. Ley de Zipf

El lingüista norteamericano *George Zipf*, formuló la ley de *Zipf* para las frecuencias de las palabras en los textos en inglés. La palabra más común, *the*, aparece el doble de veces que la segunda, *of*, el triple que la tercera, el cuádruple que la cuarta y así sucesivamente. Pero es aplicable a muchos otros fenómenos. El último que han descubierto es el ajedrez: la jugada más usada ocurre el doble de veces que la segunda más usada, el triple que la tercera; en resumen, la ley de *Zipf* se representa mediante la ecuación (1*)*.

$$P_n \approx 1/n^a \qquad (1)$$

donde P_n representa la frecuencia de una palabra ordenada *n-ésima* y el exponente *a* es próximo a *1*. Esto significa que el segundo elemento se repetirá aproximadamente con una

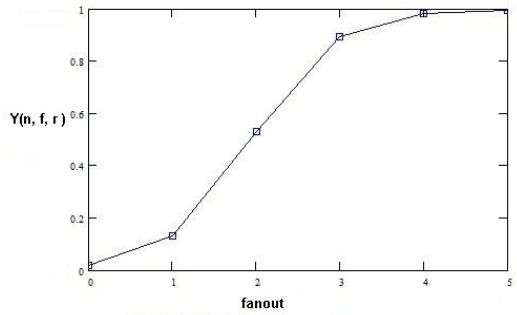

Fig. 2 Modelo Infecta por siempre.

frecuencia de *1/2* de la del primero, y el tercer elemento con una frecuencia de *1/3* y así sucesivamente: En la Figura 1, se puede visualizar la ley de *Zipf*, en el eje *x*, representa la popularidad de páginas con respecto al eje *y*, las visitas por mes que se realizan en el *WWW* a páginas con alta popularidad.

III. Modelos Matemáticos Epidémicos

A. Modelos de Población finita

Incluye la incorporación de una población de tamaño *N*, donde X_r es el número de individuos infectados en la *r-ésima* ronda de propagación epidémica [1, 2].

En cada ronda cada individuo infeccioso con probabilidad P_k intentará contaminar a *k* miembros de la población total. Estos *k* miembros se elegirán aleatoriamente desde la población total.

Los casos son:

1) Infectar y morir: modelo en el que un individuo intenta contaminar a otros por una sola ronda, y luego se detiene.
2) Infectar siempre: modelo en el que los individuos infectados siguen siendo infecciosos en todo momento.

B. Descripción del modelo infecta por siempre

En el modelo infectar por siempre, los individuos infecciosos intentan contaminar a *f* miembros en cada ronda, la fórmula (2), representa el número esperado de miembros infectados en *r* rondas.

$$Y_r \approx \frac{1}{1+ne^{-fr}} \qquad (2)$$

donde:

Y_r : Probabilidad de infección después de un número de rondas *r*.
n : Número de población.
f : Infectados por ronda (*fanout*).
r : Número de rondas.

En el modelo infecta por siempre, cada nodo infectado intenta contaminar a *f* nodos por cada ronda hasta que toda la población esté infectada, al proceso de nodos infectados por ronda se le conoce como *fanout.* Además un nodo ya contagiado nunca muere y sigue infectando a sus vecinos a pesar que ya están contagiados, es decir, nunca termina de contagiar.

Así, como se puede apreciar en la Figura 2, la proporción del número de individuos infectados con el número de los no infectados aumenta exponencialmente rápido en promedio, por un factor de e^f en cada ronda. En este caso el *fanout =2* y realiza 5 rondas para infectar un total de 48 nodos.

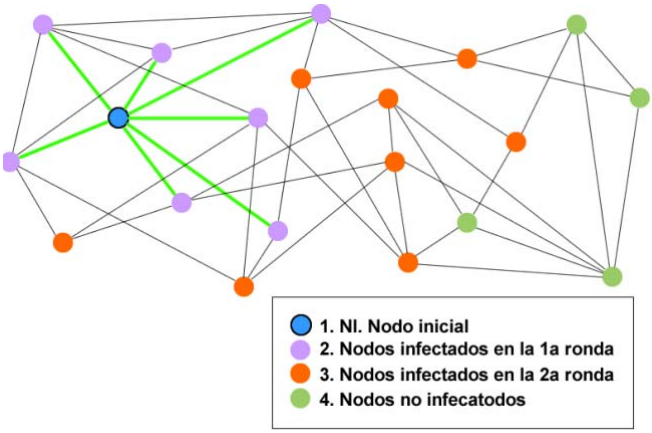

Fig. 3 Comportamiento del modelo infecta y muere.

C. Descripción del modelo infecta y muere

En la figura 3, se muestra el comportamiento del modelo epidémico infecta y muere en una red. Se toma un nodo inicio, y en la 1ª ronda de la infección se contaminan a todos sus vecinos del nodo inicial. En la *2ª* ronda se contaminan a los vecinos de cada uno de los nodos infectados en la *1ª* ronda. Continua la infección hasta *n* rondas necesarios para contaminar a todos los nodos de la red. Después de infectar a los nodos vecinos, cada nodo debe morir, esto es, una vez que un nodo infecta a sus nodos vecinos, este no puede ser infectado nuevamente en una ronda posterior.

En la fórmula (3) que es una ecuación de punto fijo del modelo infecta y muere, la variable π simboliza la proporción acumulativa de nodos eventualmente infectados en cada ronda y *f* representa el *fanout*, que son los nodos infectados en cada ronda. La utilización de este algoritmo no se considera porque los nodos infectados resultantes fueron menos significativos para rondas y número de nodos pequeños.

$$\prod \approx 1 - e^{\pi f} \qquad (3)$$

IV. Descripción del Algoritmo

El algoritmo epidémico propuesto realiza la propagación de un mensaje en la red de nodos. El algoritmo inicia a partir de cualquier nodo aleatorio y las peticiones siguen la ley de *Zipf* que infectan a un número fijo de nodos, el cual está definido por el *fanout*. Este proceso se repite tantas veces el número de rondas. En la realización de este algoritmo se ocuparon diversos lenguajes de programación como lo son java, *bash*, *awk* y *perl*. Este algoritmo consta de tres fases para llevar a cabo la infección en la población: Historial, Propagación y el Gestor.

A. Historial

Esta etapa del proceso ayuda a conocer los nodos que se han infectado en una ronda. Este proceso es conocido como el

TABLA I. RESULTADOS CON *FANOUT* = 4

Estado	*% epidémico*
1	8.33
2	35.41
3	91.66
4	97.916
5	100

historial de un nodo, los nodos infectados se vuelven propagadores de la infección. El historial de un nodo, es la información que se almacena en cada nodo y así se puede conocer los antecedentes de las peticiones. Para la elaboración de esta parte se realizó el código script en *perl*.

Con el historial se toman los nodos infectados y se vuelven propagadores de la infección.

Los ficheros generados con la herramienta *Bamboo* son trazas que se realizan para conocer el funcionamiento de cada nodo durante la búsqueda de información. Para poder usar el archivo *check-obj-ptrs* se necesitan los ficheros *experiments* generados por *Bamboo*. Por esta razón se diseñó un programa en *bash*, que tiene como función localizar y asignar como parámetros de entrada los *experiments* al script *check-obj-ptrs* que se han modificado.

Una vez obtenido las trazas que corresponden al número de saltos que un nodo recorre de un nodo a otro nodo en la consulta semántica, se puede realizar la obtención del recorrido total, solamente sumando todos los recorridos de los nodos involucrados y así obtener el número de saltos totales.

B. Propagación

En esta segunda etapa, se obtiene la información de los nodos contagiados en una ronda, mediante el uso del historial para que así se pueda expandir la infección a toda la red distribuida. El programa que obtiene al conjunto de nodos está elaborado en el lenguaje *AWK*, por ser un lenguaje especializado en el procesamiento de textos para tomar el archivo que nos genera el historial y procesarlo de una manera sencilla.

En esta etapa se optó por utilizar este lenguaje de programación porque utiliza expresiones regulares que se asociaron con los ficheros de salida que se obtienen de la herramienta de *Bamboo*. Una vez obtenido los ficheros salida se utiliza *Gnuplot* para la graficación de los resultados.

C. Gestor

Es un script que se encarga de manipular las dos etapas anteriores, es decir, de la coordinación para efectuar la propagación de la infección. Esta coordinación se realiza a través de la petición de un número que describe las rondas que se deben de realizar para expandir la infección en una red y un *fanout* que está representado en *Bamboo* como las réplicas de un documento. Las replicas en los documentos se pueden hacer de dos a tres replicas en toda la red debido a que exponencialmente representa una media de replicas promedio para una red de 48 nodos.

D. Propuesta solución

El algoritmo se inicializa en un nodo aleatorio que pertenece a la red, y se asigna un número de rondas, este nodo se convierte en el origen de la infección y lo propaga a todos los demás nodos. Después de la primera ronda se genera el historial, que proporcionan los nodos infectados, para que estos nodos se conviertan en el nuevo origen de la infección en la red, hasta que toda la red quede totalmente propagada se termina, o si no sucede esto, se termina hasta que se cumpla el número de rondas que se establecieron desde un principio, ver el pseudocódigo 1.

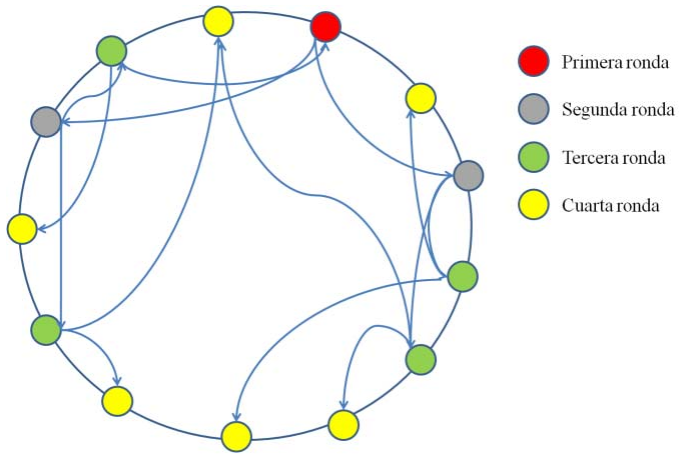

Fig. 4 Algoritmo epidémico en cuatro rondas.

Pseudocódigo 1: **Algoritmo epidémico**

1. Inicio
2. Inicializa la infección en un nodo perteneciente a la red.
3. Crea historial de infectados
4. Se verifica que existan nodos que no estén infectados.
   - 4.1 Para 1 hasta número de rondas-1
     - 4.1.1 Lee archivo historial.
     - 4.1.2 Inicia la infección con el nuevo conjunto de infectados.
     - 4.1.3 Actualizar el historial de infectados.
     - 4.1.4 Se verifica todos los nodos estén infectados.
       - 4.1.4.1 Si es así, detiene la infección.
   - 4.2 Fin Para
5. Fin

La tabla I, permite visualizar el funcionamiento del algoritmo epidémico cuando el *fanout* es igual a 4.

En la Figura 4, se muestra una imagen representativa del algoritmo epidémico, en donde se pueden ver los nodos infectados por ronda en una herramienta *DHT* de una topología circular como *Bamboo*.

V. EVALUACIÓN DEL ALGORITMO EPIDÉMICO EN UNA RED *P2P* ESTRUCTURADA

El escenario que se utiliza es el siguiente, 48 (2^6) nodos en una red *p2p* estructurada, el número de saltos para llegar a un nodo en una *DHT* es de 6 saltos máximo por que el número de

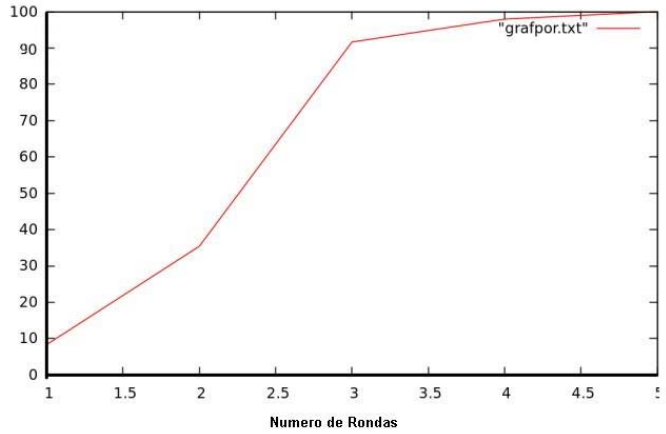

Fig. 5 Red de 48 nodos.

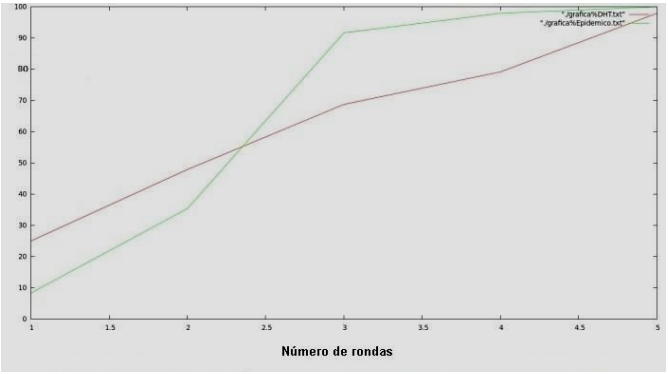

Fig. 6 Comparación de los algoritmos con fanout = 2.

TABLA II. RESULTADOS CON *FANOUT* = 2

Estado	%DHT	*% epidémico*
1	25	8.33
2	47.91	35.41
3	68.75	91.66
4	79.16	97.916
5	97.91	100

nodos está entre $2^5 < 48 < 2^6$ por el $O(\log n)$. Además se utilizan de 10 a 100 peticiones basadas en la ley de *Zipf*, y se realizan las peticiones de manera aleatoria desde cualquier nodo de la topología.

Se realizaron múltiples pruebas del algoritmo epidémico, desde 10 hasta 48 nodos, en las cuales se nota que el crecimiento de la replicación de datos se comporta de manera exponencial. Es decir:

$$N^k \tag{4}$$

Donde, en la fórmula (4):

N = *fanout*

k = número de rondas

En la Figura 5, se muestra una gráfica de una red con 48 nodos y un *fanout* de 4, el eje *Y* representa el porcentaje de nodos

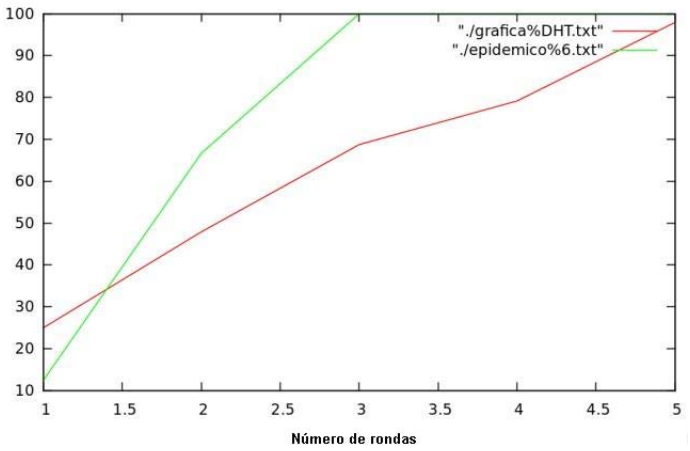

Fig. 7 Comparación de los algoritmos con fanout = 6.

TABLA III. RESULTADOS CON *FANOUT* = 6

Estado	%DHT	*% epidémico*
1	25	12.5
2	47.91	66.67
3	68.75	100
4	79.16	100
5	97.91	100

visitados en una ronda, y en el eje de las *X* es el número de rondas.

A. Comparación con el algoritmo DHT de Bamboo en una red p2p estructurada

Como parte de esta investigación se busca demostrar que el algoritmo epidémico actúa de manera más eficiente en la búsqueda de datos que el método *DHT* de *Bamboo*. Por esto, se utilizaron los dos algoritmos con 48 nodos para obtener la gráfica de la Figura 6 y los datos obtenidos se muestran en la Tabla II. Donde el eje de las *Y* representa el porcentaje de número de nodos recorridos o conocidos en un salto o ronda, y el eje *X* representa el número de rondas.

En la gráfica de la Figura 7, se puede apreciar que el algoritmo epidémico crece más rápido en las últimas rondas, aunque en las primeras rondas su crecimiento fue más lento que el algoritmo de *DHT* de *Bamboo*. Sin embargo, si el *fanout* es un número mayor, el algoritmo epidémico será mejor que la *DHT*.

Se muestra un ejemplo en la Figura 7, en una red de 48 nodos y un *fanout* de 6:

En la tabla III, el algoritmo de *DHT* no tiene variación como lo tiene el algoritmo epidémico, es debido a que el *fanout* afecta solo al algoritmo epidémico. Aunque el *fanout* para la *DHT* sería algo similar a los vecinos cercanos y vecinos lejanos que contiene la red *p2p* estructurada de *Bamboo.*

VI. CONCLUSIONES Y TRABAJO A FUTURO

En el desarrollo de esta investigación, se demostró que los algoritmos epidémicos son directamente proporcional al *fanout*, es decir, que tienden a ser exponencial entre más grande sea este

número. Aunque se realizaron pruebas con *fanout*= 2, 4 y 6, hacerlo con un número más grande resultaría inmediato la infección a todos los nodos. Además el *fanout* representa el interés que los nodos tienen entre sus vecinos es decir la semántica de la información. El algoritmo *DHT* de *Bamboo* realizó las búsquedas en menor número en las primeras *k* rondas comparado con el algoritmo epidémico, debido a que las DHT mantienen un $O(log\ n)$. El número de nodos de una red *p2p* estructurada es del orden 2^n el número de saltos máximo es *n*, el utilizar un número de nodos más grande representa que el algoritmo epidémico sea aún más rápido en la búsqueda de información que el algoritmo DHT, debido a que la DHT no contiene información semántica en sus nodos.

En este primer trabajo se utiliza un algoritmo epidémico para demostrar que es posible replicar información y así agregar semántica a la información de la red y como resultado el tiempo de las búsquedas sea menor en un sistema distribuido como son las redes *p2p* estructuradas.

El uso de los algoritmos epidémicos es importante porque se recorren los caminos de la redes con amplia difusión, lo que genera un tráfico en la red, sin embargo si se utilizan algoritmos epidémicos con búsquedas semánticas, se reduce el trafico de mensajes y se pueden emplear para eliminar el correo *spam* o reducir el número de peticiones en el *WWW*.

Como trabajo a futuro se propondría realizar escenarios, con otras topologías de redes, con tráfico en la red, con protocolos de transporte TCP o UDP, y así generar nuevas mediciones de los algoritmos epidémicos que han sido presentados en este trabajo.

REFERENCIAS

[1] Tim Daniel Hollerung, Peter Bleckmann,(2004), *Epidemic Algorithms Topic 9*, University Padeborn, August, 4th, 2004.

[2] P.T. Eugster, R. Guerraoui, A.-M. Kermarrec, L. Massoulie. (2004), *From Epidemics to Distributed Computing*. *IEEE Computer*, Vol. 37, No. 5, May 2004, pp. 60-67.

[3] Aguirre M. Tesis de Maestría, Algoritmos de Reconocimiento de Patrones que Guíen la Búsqueda de Información en Redes Complejas, noviembre del 2008 en Tamaulipas, México.

[4] Michlmayr E., *Ant Algorithms for Self-Organization in Social Networks*. Tesis Doctoral, Women's Postgraduate College for Internet Technologies (WIT), Institute of Software Technology and Interactive Systems, Universidad de Tecnología de Vienna, 2007.

[5] Sean Rhea, Brighten Godfrey, Brad Karp, John Kubiatowicz, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, Harlan Yu. *OpenDHT: A Public DHT Service and Its Uses*. Proceedings of ACM SIGCOMM 2005, August 2005, ["Online"]. Disponible, http://www.bamboo-dht.org y http://opendht.org.

[6] Balachander Krishnamurthy, Craig Wills, Yin Zhang. (2001), *On the Use and Performance of Content Distribution Networks*, ACM Sigcomm Internet Measurement workshop.

[7] M. Ripeanu. (2001) *Peer-to-Peer Architecture Case Study: Gnutella Network*. Technical Report– 2001-26, Department of Computer Science, University of Chicago, 24 de Julio del 2001.

[8] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz, (2004). *Handling Churn in a DHT*. *Proceedings of the USENIX Annual Technical Conference*, June 2004

[9] Y. Kulbak, D Bickson.(2005), *The eMule Protocol Specification,* Technical Report – 2005-03, Leibniz Center, School of Computer Science and Engineering, The Hebrew University, 01/2005.

[10] Johan Pouwelse, Pawel Garbacki, Dick Epema, Henk Sips. (2005) *Peer-to-Peer Systems IV*, Volume 3640/2005 de Lecture Notes in Computer Science, The Bittorrent P2P File-Sharing System: Measurements and Analysis, págs. 205-216. Springer Berlin/Heildeberg.2005.

[11] Zilia Lavastida-López, Yudivián Almeida-Cruz, (2009) *Propuesta de un modelo para el intercambio automático de información en redes P2P*, VI Jornadas para el Desarrollo de Grandes Aplicaciones de Red (JDARE'09). Computación como Servicio, Desarrollo de Grandes Aplicaciones de Red. Alicante, España, octubre 15-16, 2009. ISBN 978-84-613-4894-7, GrupoM. Alicante. 2009. Páginas: 333-350.

Colmenares G. Luis E., nació en Tuxtla Gutiérrez Chiapas, México el 9 de Abril de 1969. Realizó sus estudios de la Licenciatura en Computación en la Benemérita Universidad Autónoma de Puebla en la Facultad de Ciencias Físico-matemáticas. Los estudios de Maestría en la Universidad de las Américas Puebla obteniendo el título de Maestro en Ciencias. El doctorado fue realizado en la Universidad Politécnica de Cataluña en Barcelona España, en la especialidad de Sistemas distribuidos en el Departamento de arquitectura de computadores.

Actualmente está como profesor investigador de tiempo completo en la Facultad de Ciencias de la Computación de la Benemérita Universidad autónoma de Puebla. Imparte asignaturas de las currículas de Licenciatura, Ingeniería y Postgrado. Pertenece al cuerpo académico de Sistemas de Información Promep, y colabora con la Dirección General de Innovación educativa. Esta en diferentes proyectos de Sistemas Distribuidos, Sistemas de Tiempo Real, Computo Pervasivo y Cómputo Ubicuo.

Luis Enrique Colmenares es miembro de SOMECYTA (Sociedad Mexicana en Ciencia y Tecnología Aeroespacial. Además es miembro de la red temática Conacyt "Tecnologías de la información y comunicaciones" de 2011-2015. Pertenece además al padrón de Investigadores de la Benemérita Universidad autónoma de puebla. Ganador del tercer lugar del equipo que fue mentor en la final nacional de diseño de software de Imagine Cup 2010 y En el 2011 recibió dos reconocimientos por la Benemérita Universidad Autónoma de Puebla Pue. México. En el 2012 participo como mentor en la final nacional de diseño de software en el Imagine Cup 2012.

Eder Solís López, nació en Puebla, Pue. México el 29 de junio de 1990. Realizo estudios de preparatoria en el Centro Escolar Niños Héroes de Chapultepec, preparatoria incorporada a la Benemérita Universidad Autónoma de Puebla. Ingreso a la Licenciatura en Ciencias de la Computación en la BUAP (2008). Ha asistido a diferentes congresos como: 50 años de la computación en México (UNAM 2008); Congreso de Computación, Informática, Biomédica y Electrónica (Universidad de Guadalajara 2009); 7° Congreso Nacional de Ciencias de la Computación (BUAP 2010); Congreso Nacional de Enlace Tecnológico (BUAP 2010); Eleventh Mexican International Conference on Computer Science (UAEM 2011); Twelfth Mexican International Conference on Computer Science (Universidad de Guanajuato 2012). Tiene conocimientos del área de programación en los lenguajes C, C#, Java, Visual Basic, PHP, mySQL y UML. A lo largo de su carrera ha tomado cursos extracurriculares de programación, redes e inglés.

A finales del año 2011 obtuvo la certificación en el lenguaje de programación Java "Oracle Certified Professional Java SE 6 Programmer". Para el año 2012 participo en la competencia tecnológica de Microsoft: "Imagine Cup 2012**"** con el equipo "DIGIMFILT" en la categoría "Diseño de Software"; el proyecto es un filtro de contenido pornográfico basado en el procesamiento digital de imágenes para el "Internet Explorer" donde se aplicaron tecnologías de Microsoft y el algoritmo RSOR de Pedro Ivan Tello Flores para la detección de desnudos en imágenes digitales; llegó a la final nacional en el mes de Abril en las instalaciones de Microsoft México en la ciudad de México.

En la actualidad se encuentra realizando su trabajo de Tesis para obtener el título de Licenciado en Ciencias de la Computación.