行政院國家科學委員會補助專題研究計畫 期中進度報告

對等式內容網路之搜尋與傳遞演算法及安全議題研究

計畫類別：■ 整合型計畫
計畫編號：NSC 92－2213-E－002－087－
執行期間： 92 年 8 月 1 日至 93 年 7 月 31 日

計畫主持人：林宗男
共同主持人：
計畫參與人員：王欣平, 沈彥男, 張永煌, 林柏江

成果報告類型(依經費核定清單規定繳交)：■精簡報告

執行單位：國立臺灣大學電信工程學研究所

中 華 民 國 93 年 5 月 7 日

九十二年度行政院國家科學委員會專題研究計畫

對等式內容網路之搜尋與傳遞演算法及安全議題研究

92-2213-E-002-087

期中報告

2004.5.7

本計畫執行至今，在對等式網路之搜尋演算法及效能分析上，已有顯著的成果。其研究成果已有二篇著名國際會議的發表。第一篇發表於今年四月於美國 Chicago 所舉行的「**The 4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2004)**」(accepting rate 33%)，論文題目為「Search Performance Analysis and Robust Search Algorithm in Unstructured Peer-to-Peer Networks」。第二篇發表於今年六月於法國 Saint-Malo 所舉行的「**The 18th Annual ACM International Conference on Supercomputing (ICS04)**」(accepting rate 20%)，論文題目為「Dynamic Search and Performance Analysis in Unstructured Peer-to-Peer Networks」。茲附上於 ICS04 所發表之論文全文作為期中研究成果之依據。

Dynamic Search and Performance Analysis in Unstructured Peer-to-Peer Networks

Hsinping Wang ,Tsungnan Lin, Chia Hung Chen, Yennan Shen
Graduate Institute of Communication Engineering
National Taiwan University, Taipei, Taiwan
tsungnan@ntu.edu.tw

ABSTRACT
Recently Peer-to-Peer networks (P2P) have gained great attention and popularity. One key challenge aspect in a P2P resource sharing environment is an efficient searching algorithm. This is especially important for Gnutella-like decentralized and unstructured networks since they have power-law degree distributions. In this paper, we propose a dynamic search algorithm that decides the number of running walkers dynamically with respect to peers' topological information and search time state. The dynamic search is able to control the extent of messages generating temporally by the simulated annealing mechanism, thus being a scalable search. Furthermore, we present a unified quantitative search performance measurement, *Search Efficiency*, to objectively capture dynamic behavior of various search algorithms in terms of scalability, reliability and responsiveness. We quantitatively characterize, through simulations in dynamic P2P environments, the performance of various existing searching algorithms. The proposed dynamic search outperforms other search algorithms in terms of *Search Efficiency* in both the local and global search spaces.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Search Process.*

General Terms
Algorithms, Measurement, Performance, Design, Reliability

Keywords
P2P, Search algorithm, Gnutella, Modeling

1. INTRODUCTION
Recently, a newly innovated architecture of Peer-to-Peer (P2P) [5, 6, 7, 8] networks has caught people's eyes and becomes popular for that peer-to-peer networks treat all client nodes functionally equivalent. Unlike the centralized client-server model, the nodes (servents) in the P2P network behave as clients and servers at the same time. P2P networks emphasize the ability to research out, discover, and connect with others regardless of whether a prior relationship exists. In addition to the ability to pool together large amounts of resources, the advantages of P2P systems include self-organization, load balancing, adaptation, and fault-tolerance. The rapid growth of Gnutella [1] is an example for the appealing architecture. Recent measurement data suggests that P2P applications have a significant impact on the Internet traffic [3, 4].

ICS'04, June 26–July 1, 2004, Saint-Malo, France.

In this paper, we focus on Gnutella-like decentralized unstructured P2P environments since these systems are actively used by a large community of Internet users today [13, 14, 27]. These networks, while not centrally planned in structure, grow according to a simple self-organizing process. Recent measurements [1, 13, 15] show that they have power-law degree distributions. In order to function well, the highly unstructured networks need efficient search algorithms.

Current search algorithms in Gnutella-like networks tend to be inefficient, either generating too much load on the system [2, 16], or not meeting users' requirements [11]. Hierarchical structure like KaZaA [27] can reduce network traffic by using superpeers. However, the query operations among superpeers are the same as Gnutella networks. Current Gnutella flooding search algorithm sends query messages to every possible peer. Doing so may waste overwhelming bandwidth unnecessarily and break down the systems eventually. Another approach is to keep the number of query messages constant. This may result in the slow responsiveness. A good search algorithm should be able to find a good balance from different perspectives with contradictory goals.

In this paper, we propose a quantitative measure criterion of searching performance: *Search Efficiency* (SE). Search efficiency gives an objective performance measure from both users' and networks' perspectives to address the performance issues of scalability, reliability, and responsiveness. The factors taken into account by SE include the number of results found, success probability, search speed, and the total number of messages generated.

Current Gnutella searching algorithm, flooding, is known to have the scalability problem [2, 13]. It suffers from the exponentially growing number of search messages. On the other hand, it has been suggested random walker search algorithm [10, 12] can improve the scalability problem. We find that, in our simulations, the search efficiency of random walker algorithm almost remains the same regardless of the search time because the number of the query messages (walkers) remains constant regardless of the network topology. However, random walker search algorithm suffers from poor search efficiency in the short term although it does have higher search efficiency compared to that of flooding search in the long term. It is also difficult to determine the optimal number of walkers in a dynamic environment in advance.

We therefore propose a dynamic search algorithm trying to solve the problem of scalability and how to determine the optimal number of running walkers. The proposed mechanism can decide

the number of walkers dynamically with respect to the peer's linking status and the search time, making it possible to decide the optimal number of query messages. Moreover, by the dynamic decision mechanism, a search can fully explore the local search space to get responsive results and control the impact on the network in the global space by a "simulated annealing" mechanism, resulting in a scalable search. Simulation results demonstrate the proposed mechanism outperforms existing search algorithms in terms of high search efficiency in both long term and short term.

Simulation experiments are performed in a dynamic P2P networking environment in order to collect convincing results for algorithms evaluation. The factors considered include the network topology, link distribution, peer's joining and leaving, and querying behavior as well as the activity of file sharing [9, 10, 17, 37]. Our dynamic network model is constructed based on these factors that strongly reflect the real measurement studies [1, 4, 10, 28, 36].

The rest of this paper is organized as follows. In Section 2, we briefly discuss various existing search algorithms. In Section 3, we describe in details the proposed dynamic search algorithm. The dynamic P2P environment modeling for our simulations is described in Section 4. Before presenting our simulation results, we introduce the unified evaluation criterion, Search Efficiency, in Section 5. Performance evaluations of various search algorithms is presented and discussed in Section 6. Finally, Section 7 concludes our work.

2. RELATED WORK

P2P search algorithms can be classified as blind search algorithms or learning-based algorithms. Flooding [18], random walk [10, 12, 17], modified-BFS [32], and expanding ring [10, 12] are examples of blind search algorithms, in which query messages are generated statically and relayed blindly to the neighbor peers.

When a node has the mechanism to build the knowledge, it can relay the query messages more intelligently. These algorithms include routing indices [34], local indices [11], intelligent search [32], and APS [33], etc. Routing indices [34] classifies each document into some thematic categories and can forward the query messages more intelligently based on its category. The operation of local indices [11] is similar to the super-peer networks that each node collects the file indices of the peers within its radius of r. While the search request is out of the node's knowledge, it performs a flooding search. Intelligent search [32] uses a function to compute the similarity between the search query and the recently answered requests. Nodes relay the query messages based on the similarity results. APS [33] builds the knowledge with respect to each file based on the past experience.

3. DYNAMIC SEARCH

The design goal of our dynamic search algorithm is to optimize the search performance: returning query hits responsively, as well as reducing numbers of redundant messages without reducing the success probability. We hypothesize that the intrinsic nature of search optimization methods in an unstructured P2P network is similar to many search problems of the combinatorial optimization methods. Therefore, we take an approach similar to the mechanism called simulated annealing [30], a popular optimization method applied in circuit design, artificial intelligence, and many other fields, to construct our dynamic search mechanism. The term simulated annealing derives from the roughly analogous physical process of heating and then slowly cooling a substance to obtain a strong crystalline structure. The procedure of simulated annealing allows the system to move consistently towards lower energy states, yet still "jump" out of local minima due to the probabilistic acceptance of some upward moves during the first few iterations.

The ideas used in simulated annealing are applicable for a robust unstructured P2P search. A robust search [38] has to search the local space globally (introducing upward moves during the first few iterations) to obtain responsive search results, and search the global space locally (cooling the substance) to control the unnecessary messages, leading to an efficient search (strong crystalline structure).

Before describing our dynamic search, we first introduce the idea of forwarding probability model, which mathematically specifies the forwarding behavior of each peer to deliver queries to its neighboring peers. The probability model will make it clearer and easier to formally illustrate our dynamic search mechanism.

3.1 Probability Model

The mechanism of forwarding a query message to the ith neighboring peer can be described by a probability function $p_i(t,o_i)$, where t denotes the current searching time and o_i represents the statistic information of the ith peer. When $p_i(t,o_i) = 1$, the query message is forwarded to the ith peer. If $p_i(t,o_i) = 0$, the ith peer will not receive the query message. When $0<p_i<1$, the message is forwarded to the ith peer with probability p_i. Assuming the number of links (neighbors) of certain peer is l, we obtain the number of messages the peer will forward at some search time t as $\sum_{i=1}^{l} p_i(t) = k$, where k is the number of forwarding walkers of the peer on average. For pure flooding search algorithm, the probability model, which every node adopts, can be described as

$$\forall i,\ p_i(t,o_i) = 1. \tag{1}$$

If the termination parameter TTL is used, then the probability function should be modified as

$$(1 - u(t - TTL)) \cdot p_i(t,o_i) \tag{2}$$

where $u(t)$ is the unit step function ($u(t) = 1$, when $t \geqq 0$, $u(t) = 0$, otherwise). In the case of random walk search algorithm, the function $p_i(t)$ does not depend on the knowledge of its peers since the algorithm randomly selects its peers to relay the query. Moreover, within each node, there is one additional constrain, $\sum_i p_i(t) = 1$, to let each node only forwards the received message to one neighbor.

3.2 Dynamic Search Algorithm

Dynamic Search is a multi-stage search with a probability model annealed with respect with search time. The mechanism for dynamic search attempts to explore the local search space more aggressively and limits the exploration gradually when search time elapses. Therefore, each node will determine the optimal number of query messages according to the information, for example, of search time and the number of neighbors. While the search is out of the local space (beyond n hops), dynamic search will change to a limited search (one example is to ask only one of its neighbor). The multi-stage probability model can satisfy the goals to search aggressively in the local, control the network impact in the global, and choose the number of optimal query messages in a dynamic fashion. We specify our dynamic forwarding mechanism by the following probability model and an illustrative example will be given latter.

Assume each node doesn't collect the statistics information of its peers with respect to the target ($p_i(t,o_i) = p_i(t)$). For each node applying the dynamic search algorithm, the probability function to forward to its ith neighbor at search time t is

$$p_i(t) = \underbrace{p^0 \cdot u(t) - p^1 \cdot u(t-1) - \cdots - p^n \cdot u(t-n)}_{Annealed\_Flooding} + p^u \cdot u(t-n) \quad (3)$$

where n is the limit to aggressively explore the search space. The probability p^0 controls the extent the node to query its neighbors and the remaining probabilities $p^0, p^1,\ldots,p^n$ (except p^u) "anneal" the querying extent (the upper indices are indices about time, not power orders). Since $p_i(t) \geq 0$ and the probability for "annealed flooding" must be non-negative, $p^0 \geq p^1 + p^2 + \cdots + p^n$ and $p^u \geq 0$, according to (3). To guarantee that the number of query messages will not grow exponentially, p^u can be set to be $1/l$, where l is the link degree of the relayed node. In this case, only one walker will be generated per peer.

One can easily discover that flooding and random walk algorithms are two static instances of the proposed model. If we set $p^0 = 1$ and other probability parameters to zeros, it is the flooding search model, as stated in (1). If we set $p^0 = 1/l$ and the remaining to zeros to make $\sum_{i=1}^{l} p_i(t) = 1$, this model is the random walk search. Furthermore, if choosing the probability for $t = 0$ as $0 < p^0 < 1$, the probability model becomes modified-BFS [32] where p^0 is the fraction parameter of modified-BFS. Apparently, these existing search algorithms only adopt the forwarding mechanism specified by p^0, and thus their mechanisms are time-invariant and don't adapt dynamically over search time. Our proposed algorithm is designed to be more flexible and adaptive in the dynamic network environments for producing more efficiency compared to the existing algorithms.

Besides, in the pure random walk search, it is difficult to determine the optimal number of walkers in advance. With this dynamic forwarding mechanism, peers can generate probabilistically the dynamic number of query messages according to their linking status and the time state of the search. Therefore, users have the flexibility of how aggressively they want to search the network and how far they want to extend the search coverage. If a user wants to search results aggressively or covers the search in a wider range, he or she can set a lager number of n. The number of query messages is determined topologically and probabilistically based on local connections of the searching node within the radius of n.

4. DYNAMIC P2P MODELING

In order for strong evaluation environment, we have built a simulator, with search algorithms in question embedded, to model possible aspects in file-sharing P2P systems: network topology [1], peer cycle [37], peer querying [9], and object replication [10]. The significance of this simulator comes from those modeled aspects that strongly reflect the real measurement studies [1, 10, 28, 36, 9], and the dynamic modeling of peer cycle, thus producing virtually realistic results. We discuss the construction of the dynamic P2P environment, in which the simulator is built, in two tracks—network charactering and peer behaving—in the following two sub-sections, respectively.

4.1 Network Modeling

For the network topology modeling, we choose to model our P2P network as one of the popular P2P systems, Gnutella, to provide a network context in which peers can perform their intended activities. The measurements in [1, 10] have suggested that the topology of Gnutella network has the property of two-segment power-law link distribution. Thus, we construct a P2P network of 100,000 peers in our simulator, in which the link distribution follows the reported the two-segment power law. The resulted statistics of the topology embedded in our simulator are that the maximum link degree is 632, mean is 11.73, and standard deviation is 17.09. Once the node (peer) degrees are chosen, we connect these peers randomly and reassure every peer is connected (each peer has at least one link).

For the object distribution of the network, we assume there are 100 distinct objects with replication ratio of R = 1%; totally there are 100,000 objects in the network. The distribution of the 100,000 objects over the network follows the measurement characteristics reported in [28]. In addition, due to the dynamic environment—peers join and leave dynamically—described in the following sub-section, the total number of objects available in the network will fluctuate according to the network size (number of on-line peers) but the replication ratio will roughly remain constant.

4.2 Dynamic Peer Behavior Modeling

Our dynamic peer behavior modeling largely follows the proposed idea of peer cycle [37], which includes joining, querying, idling, leaving, and joining again to form a cycle. The joining and leaving operations of peers (include idling) are inferred and then modeled by the uptime and session duration distributions measured in [28, 36]. These measurement studies show similar results in the peer uptime distribution, where half of the peers have uptime percentage less than 10% and the best 20% of peers have 45% uptime or more. We use the log-quadratic distribution suggested in [36] to re-build the uptime distribution, which is plotted in Figure 1. But for the session duration distribution, those two studies lead to different results. The median of session time in [36] is about 15 minutes while it is 60 minutes in [28]. In our modeling, we choose the median session duration time to be 20 minutes. The re-built session duration distribution is drawn in Figure 2.

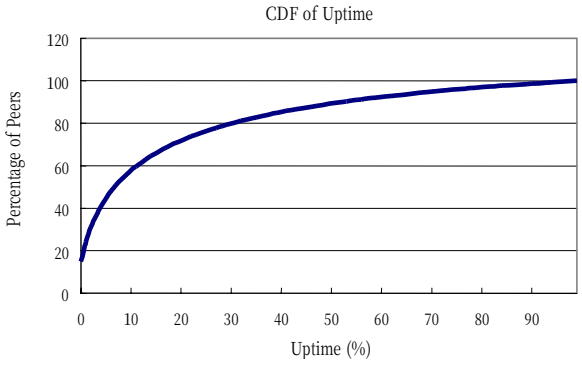

Figure 1. CDF of uptime

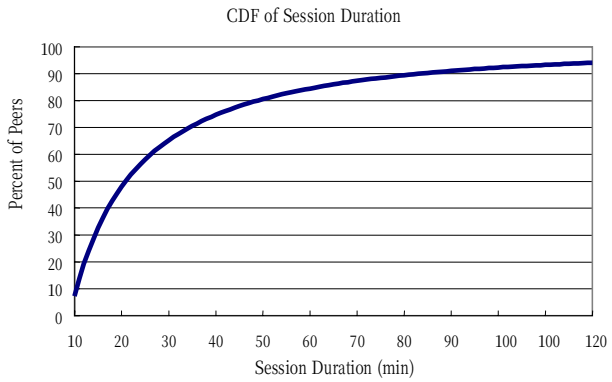

Figure 2. CDF of session duration

By these two rebuilt distributions, we can generate a probability model to decide when a peer should join or leave the network and how long it should continually be online. The basic rule to assign peers' attributes is that peers with higher link degrees are assigned to higher uptime percentages and longer session durations, and vice versa. With these conditions, we map a *two-hour* long dynamic join/leave pattern for peers. On average, there are 10 peers joining or leaving simultaneously. Since the mean value of uptime distribution is about 18%, the resulting average number of online peers is 18,152. Moreover, the maximum number of online nodes is 24,218 while the minimum number is 4,886.

We model the dynamic querying model as Poisson distribution with idle time λ= 50 minutes; that is, each peer will initiate a search every 50 minutes on average. Since there is no direct measurement about the idle time, we just use an experiential value. The choice of this parameter is insensitive to our search performance evaluation. With the idle time 50 minutes, there are thus about 6 queries or searches processing concurrently in the network on average. Totally, in this 2-hour simulation, we generate 43,632 search queries. Furthermore, for the query distribution of search objects, we model it as zipf distribution with parameter a = 0.82, similar to the ones used in [10, 33]. Finally, our simulator's central clock is triggered per second, which measures a hop for messaging passing and serves as a basic time unit for all peer operations.

5. A UNIFIED ANALYSIS CRITERION: SEARCH EFFICIENCY

A good search algorithm must be scalable, reliable, and responsive. A unified performance indicator, which takes all factors into consideration, is important and helpful. In this section, we will propose a unified search analysis criterion and demonstrate its power by analytically formulizing performance of flooding and random walk in a structured graph—full binary tree.

5.1 Search Efficiency

We argue that an efficient search tends to be a scalable one because it doesn't generate a huge number of redundant messages in an uncontrolled fashion or overwhelmingly waste the network bandwidth unnecessarily. In addition, an efficient algorithm means that the query messages generated during the search process should have a high hit rate (finding the target objects). To design an objective search performance, we first consider "query efficiency".

In the most efficient case, one query message will result in one query hit. If a search generates 100 messages and gets 20 query hits (20/100 = 0.2), it should be considered better or more efficient than a search that incurs 50 messages to obtain 1 query hit (1/50 = 0.02) although the latter generates fewer messages. On the other hand, if a search gets 100 hits at the cost of 10,000 messages, it should be worse or less efficient than a search that obtains 50 hits at the cost of 100 messages although the former acquires more query hits. In a word, the key point is the ratio of query hits gained to messages incurred, or the efficiency a search transforms the search messages into search hits. With this reason, we may simply define the query efficiency as the percentage of the number of query hits to the number of messages generated. Furthermore, we consider a normalization factor: replication ratio R. Replication ratio is defined as the ratio of the average number of available files in the network to the number of nodes, i.e., the network size. If the replication ratio is high, a search tends to find targets easily; otherwise, it is hard to get a hit. However, the efficiency of search algorithms should not be affected by network conditions. Hence, we must cancel out this factor in the query efficiency evaluation. Therefore, "*Query Efficiency* (QE)" to measure the transformation efficiency from query messages to query hits can be defined as:

$$QueryEfficiency = \frac{QueryHits(\cdot)}{QueryMsg(\cdot)} \times \frac{100\%}{ReplicaitonRatio} \quad (4)$$

If the replication ratio is 1%, it means a search covering 100 nodes (un-repeatedly) will get a query hit on the average. In this case, when a search generates 100 messages to obtain a hit, *Query Efficiency* is calculated as 100%. From a statistic point of view, it indicates these 100 messages efficiently visit 100 distinct nodes, and no redundant messages occur.

However, it is possible to have a *Query Efficiency* over 100%. It doesn't mean the query messages traverse distinct nodes more than the number of messages or the number of redundant messages is negative. In fact, it implies that the search is apt to find target object over the average basis. This is true especially when a node has the training mechanism to build the knowledge of the target [11, 32, 33, 34].

Another important factor in the search performance is "search responsiveness". Responsiveness is the ability to respond quickly to meet the needs of a user. In other words, a responsive algorithm is the one with a fast lookup mechanism. A direct measurement of responsiveness is the search response time; the responsiveness is inverse proportional to response time. To incorporate search responsiveness into *Query Efficiency*, we define *Hits Responsiveness* assessing the search speed and quality as:

$$HitsResponsiveness = \sum_{h=1}^{TTL} \frac{QueryHits(h)}{h} \quad (5)$$

where h is the response time, measured in hops, $QueryHits(h)$ is the number of query hits between search time $h-1$ and h, and TTL is the termination condition of search algorithms.

It is intuitive that a search getting 10 hits during t = 2 performs better than one getting the same hits during t = 3 if the number of messages incurred is out of considerations. Therefore, in this evaluation metric, we put more weight on the query hits found in short response time while less weight on hits in longer time.

Finally, we introduce the factor of *Success Rate*, the ability to find the target successfully, to evaluate the reliability of a search

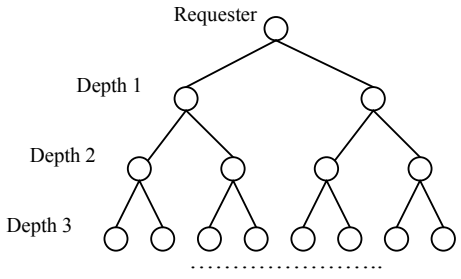

Fig. 3 A strictly binary tree with the requester at the root node.

algorithm. Hence the unified performance criteria "*Search Efficiency(SE)"* can be defined as:

$$SearchEfficiency = \frac{\sum_{t=1}^{TTL} QueryHits(t)/t}{QueryMsg} \times \frac{SuccessRate}{R} \quad (6)$$

We don't explicitly evaluate the performance in terms of coverage and success rate as [33] and [35], respectively. First of all, large search coverage doesn't directly imply a successive search or many query hits, or even a responsive search. Only query hits directly indicate the success and search result quality. Second, large coverage also means a large number of messages, which is probably negative for the search performance. Due to the unclear and indirect indication, we don't consider those factors solely to evaluate the search performance.

Usually a series of M search experiments will be performed. SE at the hop count H can be calculated as:

$$\mathrm{SE_{h=H}} = \frac{\sum_{i=1}^{M}\left(\sum_{h=1}^{H} QueryHits_i(h)/h\right)}{\sum_{i=1}^{M}\sum_{h=1}^{H} QueryMessages_i(h)} \times \frac{SuccessRate}{R} \times 100\% \quad (7)$$

where i is the index for the M different searches, R is the replication ratio, and $QueryMessages_i(h)$ means the number of messages generated for the ith search at hop = h (between hop $h-1$ to h). With this hop-based SE_h, we can observe clearly the characteristics of various search algorithms at different hops. Most importantly, SE gives an overall view on the scalability, reliability and responsiveness of a search algorithm.

5.2 Analytic Evaluation of Search Efficiency

To see how SE evaluates the overall performance, we derives the following analytic analysis in the simple network topology of strictly binary tree for BFS and Random walk search algorithm.

Assume an N-vertex strictly binary tree with depth of $\log_2 N$ and that the requester is at the root node. Moreover, we assume that the objects are uniformly distributed in the tree. Before proceeding to analyze the efficiency of the specific algorithms, we first prepare two critical factors that are on a common basis of the search coverage C whose actual value depends on the specific algorithm and depth a search arrives at. First, since the objects are assumed to be uniformly distributed, the number of objects searched out (*QueryHits*) is proportional to the search coverage C. Thus, we have

$$QueryHits(t) = R \times C \,. \quad (8)$$

Second, the success rate of a search, although not so straightforward to derive, is also relevant to the search coverage. To begin with, we know that each node owns the targeted object with a probability of R; that is, each node doesn't have it with a probability of $1-R$. Suppose a search covers C vertices and thus the probability that those C nodes share no targeted object is $(1-R)^C$. Inversely, the probability for those C nodes to share one and more than one objects, or equivalently *SuccessRate*, is determined by

$$SuccessRate = 1-(1-R)^C \quad (9)$$

Breadth first search performs as broadcasting the received queries to all neighbors except where the received query came from. Therefore, by the regular structure of binary tree, the search coverage terminated at depth *TTL* is given by

$$Coverage\ (C) = \sum_{t=1}^{TTL} 2^t \quad (10)$$

Furthermore, it is interesting to see that the number of messages required to traverse the tree is just the same as the quantity of its search coverage thanks to the very nature of BFS. Thus, $QueryMsg = C = \sum_t 2^t$. According to Equation (4), (8), and (10), we can get

$$QueryEfficieny\big|_{BFS} = \frac{R \cdot \sum_{t=1}^{TTL} 2^t}{\sum_{t=1}^{TTL} 2^t} \times \frac{100\%}{R} = 100\% \quad (11)$$

Surprisingly, the formula of QE_{BFS} yields a constant, 1 or 100%, which is invariant regardless of the replication ratio R or the termination depth *TTL*. By the definition of *QE*, it means that BFS is a perfectly query-efficient search in the context of a binary tree; that is, BFS generates *no* redundant messages while traversing a binary tree.

Then, *SE* defined in Equation (6) for BFS in a binary tree becomes

$$\begin{aligned} SE\big|_{BFS} &= \frac{\sum_{t=1}^{TTL} R \cdot 2^t/t}{\sum_{t=1}^{TTL} 2^t} \times \frac{1-(1-R)^{\sum_{t=1}^{TTL} 2^t}}{R} \\ &= \frac{\sum_{t=1}^{TTL} 2^t/t}{\sum_{t=1}^{TTL} 2^t} \times \left[1-(1-R)^{\sum_{t=1}^{TTL} 2^t}\right] \end{aligned} \quad (12)$$

The derived SE_{BFS} is a little complex for one to gain a clear picture of its properties. To deliver a clearer one, we assume the replication ratio $R \ll 1$, which is true in real networks. Thus

$$SE_{BFS} \cong \frac{\sum_{t=1}^{TTL} 2^t/t}{\sum_{t=1}^{TTL} 2^t} \times \left[1-\left(1-R\cdot\sum_{t=1}^{TTL} 2^t\right)\right] = R \cdot \sum_{t=1}^{TTL} 2^t/t \,. \quad (13)$$

For instance, $R = 0.001$ and $SE_{TTL=1} = 0.2\%$, $SE_{TTL=2} = 0.4\%$, and $SE_{TTL=3} = 0.67\%$ according to Equation (13). Note *SE* is strictly increasing with respect to *search time t*. $SE_{TTL=2}$ is exactly twice of $SE_{TTL=1}$ and $SE_{TTL=3}$ is more than three times of $SE_{TTL=1}$. The reasons are two-fold. First, as formula (11) shows, BFS in a binary tree is perfectly query-efficient, which means every query effectively contributes to the search coverage and in turn produces promising increase in

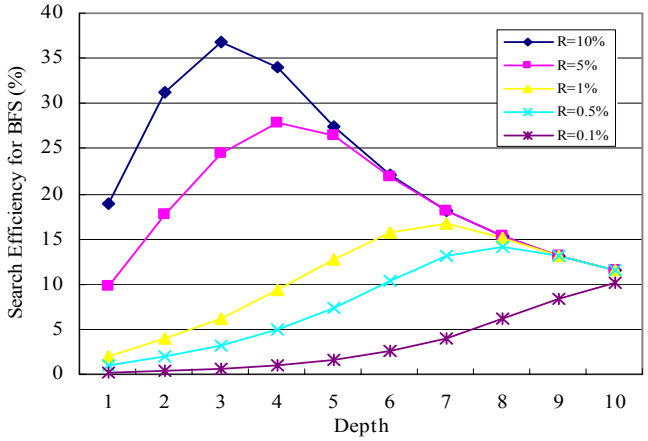

Figure 4. *Search Efficiency* of BFS in a full binary tree by various *R*s and *TTL*s

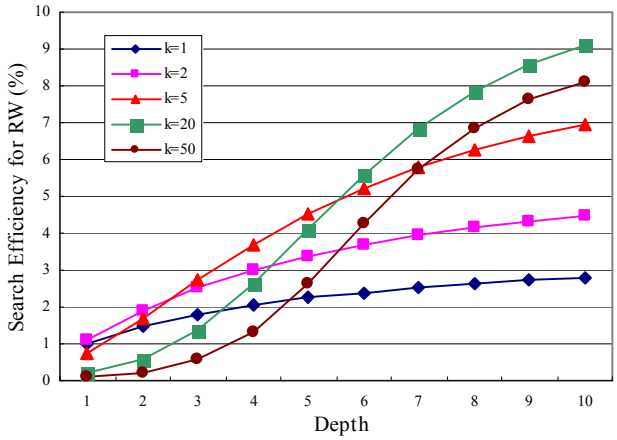

Figure 5. *Search Efficiency* of RW in a full binary tree by various number of walkers and *TTL*s with *R* = 1%

SE. Second, the speed query hits are returned is faster than the decaying factor of response time *t*. Figure 4 plots the SE_{BFS} for different *R*s and *search depth*.

When it comes to RW search, we use multiple "walkers" to traverse the network. Each walker independently searches the network and randomly chooses one of the next-hop neighbors to continue its journey until the limit of *TTL* hops.

The chance that each vertex at depth *t* is visited by a random walker is with equal probability of $1/2^t$. Thus, the probability that all of the *k* walkers don't visit certain node is $(1-1/2^t)^k$. As a result, at depth *t*, the average number of nodes visited (*Coverage per Depth*) by *k* random walkers is given by the expectation value

$$E(X)_t = 2^t\left[1-(1-1/2^t)^k\right]. \quad (14)$$

By (8), $QueryHits(t) = R \cdot E(X)_t$. Moreover, the query messages of random walk are generated per hop for each walker till terminated by *TTL* limit, hence $QueryMsg = k \cdot TTL$. As a result, *QE* of *k*-random walks is

$$QE\big|_{RW=k} = \frac{\sum_{t=1}^{TTL} R \cdot E(X)_t}{k \cdot TTL \cdot R} = \frac{\sum_{t=1}^{TTL} E(X)_t}{k \cdot TTL}. \quad (15)$$

Furthermore, by (9), we can get

$$SuccessRate = 1-(1-R)^C = 1-(1-R)^{\sum_{t=1}^{TTL} E(X)_t}.$$

Therefore, *Search Efficiency* for *k*-random walks is

$$SE\big|_{RW=k} = \frac{\sum_{t=1}^{TTL} E(X)_t / t}{k \times TTL} \times \left[1-(1-R)^{\sum_{t=1}^{TTL} E(X)_t}\right] \quad (16)$$

where $E(X)_t$ is determined by (14).

Assume a network with R = 1%, we can have a series of performance results of *SE* in terms of various numbers of walkers in Figure 5. We observe that all *SE*s consistently increase with respect to the depth or search time. Nevertheless, they all are smaller than that of BFS due to the too slow covering. As for the number of walkers *k*, too large (e.g. 50) or too small (e.g. 2) *k*s give degraded performance, thus resulting in a strong sensitivity in the design of *k*.

6. SIMULATION EVALUATION

We compare four search algorithms (flooding, modified-BFS, random walk, and dynamic search) to evaluate Search Efficiency performance. For modified-BFS, we perform an experiment of fraction parameter of 0.2. The number of walkers for random walk is chosen as 10 [33]. For dynamic search, parameters are set as n = 2, $p^0 = 1.0$, $p^1 = 0.7$, $p^2 = 0.3$, and $p^u = 1/l$, where l is the link degree of the peer applying this model. With these specifications, the dynamic search will operate as flooding at hop count h = 1, as modified-BFS with probability of 0.3 at hop count h = 2, and as random walk during h≧3.

Dynamic network environments are as stated in Section 4. We perform a 2-hour simulation, totally initiating 43,632 searches for each algorithm (M = 43,632). The environment data and the query distribution are all saved in files, and all the simulations for each search algorithm use the same files specifying the same network environment and peer dynamic behavior; so that, we can make sure the fair performance evaluation for each search algorithm. The simulation performance data are collected in Table 1 and the simulations results for SE defined in (7) with hop depths h = 1 to 7 are plotted in Figure 4.

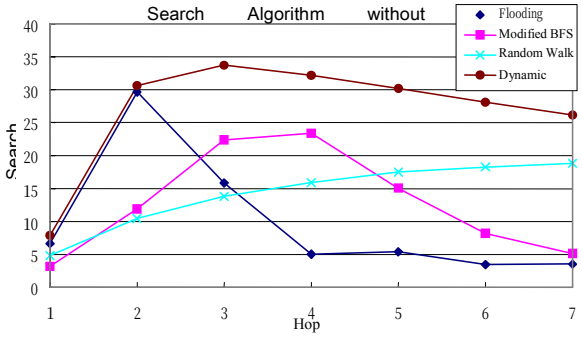

Figure 6. Search Efficiency comparison

We can observe in Figure 6 that the efficiency of flooding is high compared with random walk in the local space (h = 2) but decrease dramatically in the global region. Modified-BFS sustains the high efficiency from h = 3 to h = 4 but performs poorly in the long-term space. The performance degradation of these two algorithms is due to the huge number of redundant messages, as discussed in [10, 35]. Random walk, on the other hand, needs some "warm up" time to explore the search space and the performance increase is delayed to the long-term search space due to the limited number of random walkers. In the long-term search space, its search efficiency is rather consistently high compared with flooding and modified-BFS. As for the dynamic search, it inherits the high performance of flooding in the local and the consistent performance of random walk in the global, thus performing outstanding search efficiency within all hop depths. Although the performance of dynamic search decays when hop depths increase (there are still some redundant messages), it still outperforms other search algorithms over all search time. Moreover, in Section 3, we claim that our dynamic algorithm decide the number of walkers each node forwards to the next-hop peers in a "simulated annealing" style. We therefore show the analysis of the number of messages (walkers) on a hop-depth basis in Figure 7 based on the data in Table 2.

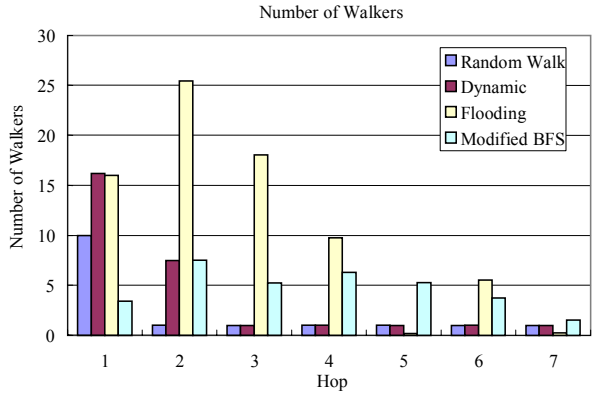

Figure 7. The average number of walkers each peer generates at various search time for search algorithms

The walker number analysis in Figure 7 further explains the performance characteristics shown in Figure 6 by the principles of simulated annealing. For flooding, the apparent large number of forwarding walkers "overheats" the search system (overwhelms it by messages) since it lacks the "cooling" process essential for the optimization. For random walk, the initial number of walkers is 10, but it imposes a hard limit on the walker number as 1 when h≧2. This hard constraint corresponds to the lack of chance to jump out the local minima in the annealing process, thus resulting in a too "cool" state of annealing and this is why it "warms up" slowly. But for our dynamic search, the number of forwarding walkers is 16.2 initially, annealed to 7.5, and finally kept a constant of 1. In this way, dynamic search aggressively explores the local search space, giving chance to jump out the local minima, to return responsive query hits and increase the success probability. In the global search space, it moderately controls the forwarding messages, applying the cooling process, thus leading to consistent search performance.

7. CONCLUSION

In this paper, we propose a scalable and dynamic search algorithm. With its dynamic probabilistically annealing forwarding mechanism, the proposed algorithm can decide the number of walkers dynamically with respect to the peer's topological information and the search time and then solves the problem of how to determine the optimal number of random walkers. In addition, through the "simulated annealing" forwarding mechanism, the messages generated can be controlled in a constant, thus avoiding the overwhelming messages incurred by the flooding search. Therefore, the dynamic search is inherently a scalable search.

Another focus of this paper is the dynamic P2P simulation environment. Based on the dynamic P2P environment designed according to the real measurement [28, 36], more convincing results are collected. We also propose a performance evaluation criterion, *Search Efficiency* to measure scalability, reliability, and responsiveness of a search algorithm on a unified basis and make it possible to get an objective and overall evaluation for various search algorithms.

From simulation results, our dynamic search outperforms other search algorithm in all hop-based analyses. It inherits the high performance of flooding in the local and keeps the consistent performance of random walk search in the long term.

ACKNOWLEDGEMENT

This work is supported in part by National Science Council (Taiwan) grant NSC92-2213-E-002-087.

8. REFERENCES

[1] Matei Ripeanu, Adriana Iamnitchi and Ian Foster. Mapping the Gnutella Network. IEEE Internet Computing, vol. 6, issue 1, Jan/Feb 2002, pp. 50-56.

[2] K.Sripanidkulchai, The popularity of Gnutella Queries and its Implications on Scalability, white paper, Carnegie Mellon Univ. Pittsburgh, Feb. 2001.

[3] D. Gallagher and R.Wilkerson. Network performance statistics for university of South Carolina. http://eddie.csd.sc.edu, Oct. 2001.

[4] D. Plonka. Uw-madion napster traffic measurement. http://net.doit.wisc.edu/data/Napster, Mar. 2000.

[5] FreeNet website. http://freenet.sourceforge.net.

[6] Gnutella website. http://gnutella.wego.com.

[7] Napster website. http://napster.com.

[8] Morpheus website. http://www.morpheus-os.com.

[9] L Kunwadee Sripanidkulchai. The popularity of gnutella queries and its implications on scalability. In *O'Reilly's www.openp2p.com*, February 2001.

[10] Q. Lv, P. Cao, E. Cohen, K. Li and S. Shenker. Search and replication in unstructured peer-to-peer networks. ICS, June 2002.

[11] Beverly Yang and Hector Garcia-Molina. Improving Search in Peer-to-Peer Networks. ICDCS, 2002.

[12] L. Adamic, R. Lukose, and B. Huberman. Local Search in Unstructured Networks. Handbook of Graphs and Networks:

From the Genome to the Internet, S. Bornholdt and H.G. Schuster (eds.), Wiley-VCH, Berlin, 2002.

[13] Clip2. Gnutella: To the bandwidth barrier and beyond. http://www.clip2.com/gnutella.html, 2000.

[14] Kelly Truelove. Gnutella: Alive, well, and changing fast. http://www.openp2p.com/pub/a/p2p/2001/01/25/truelove0101.html.

[15] Theodore Hong. Performance. In Andy Oram, editor, Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology, chap 14, p 203-241, O'Reilley, 2001.

[16] M. Jovanovic, F.Annexstein, and K. Berman. Scalability Issues in Large Peer-to-Peer Networks: A Case Study of Gnutella. Tech. Report. Univ. of Cincinnati, Lab. For Networks and Applied Graph Theory, 2001.

[17] L. Adamic, R. Lukose, A. Puniyani, and B. Huberman. Search in Power-Law Networks. Phys. Rev. E, Vol. 64, pages 46135-46143, 2001.

[18] Gnutella protocol specification v0.6. http://rfc-gnutella.sourceforge.net/draft.txt

[19] K. Aberer, M. Punceva, M. Hauswirth, and R. Schmidt. Improving data access in P2P systems. IEEE Internet Computing, vol. 6, no. 1, Jan/Feb 2002, pp. 58-67.

[20] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. Chord: A scalable content-addressable network. Proceedings of SIGCOMM'2001, August 2001.

[21] A. Rowstron and P. Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. Proceedings of SOSP'01, 2001.

[22] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. Proceedings of SIGCOMM'2001, August 2001.

[23] Ben Y. Zhao, John Kubiatowicz, and Anthony Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, University of California at Berkeley, Computer Science Department, 2001.

[24] Megan Thomas and Ellen W. Zegura. Gt-itm: Georgia Tech. Internetwork Topology Models. In http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html, 1997.

[25] Edith Cohen and Scott Shenker. Replication strategies in unstructured peer-to-peer network. SIGCOMM2002.

[26] eDonkey website. http://www.edonkey2000.com.

[27] KaZaA website. http://www.kazaa.com.

[28] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. MMCN, San Jose, CA, USA, January 2002.

[29] Leslie Lamport, Robert Shostak and Marshall Pease, The Byzantine General Problem. ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, Pages 382-401, July 1982.

[30] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. Science, V. 220, No. 4598, page 671~680, 1983.

[31] A. Rowstron and P. Druschel, Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems, International Conference on Distributed Systems Platforms (Middle), Nov. 2001.

[32] V. Kalogeraki, D. Gunopulos and D. Zeinalipour-Yazti, A Local Search Mechanism for Peer-to-Peer Networks, CIKM, Nov. 2002.

[33] D. Tsoumakos and N. Roussopoulos. Adaptive Probabilistic Search (APS) for Peer-to-Peer Networks. Technical Report CS-TR-4451, Un. of Maryland, 2003.

[34] Arturo Crespo and Hector Garcia-Molina. Routing Indices for Peer-to-Peer Systems. Proceedings of the International Conference on Distributed Computing Systems (ICDCS). July 2002.

[35] S. Jiang, L. Guo and X. Zhang. LightFlood: an Efficient Flooding Scheme for File Search in Unstructured Peer-to-Peer Systems. ICPP, Oct. 2003.

[36] J. Chu, K. Labonte, and B. Levine. Availability and Locality Measurements of Peer-to-Peer File Systems. SPIE, 2002.

[37] Z. Ge, D. R. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley. Modeling Peer-Peer File Sharing Systems. INFOCOM, April 2003.

[38] T. Lin and H. Wang. Search Performance Analysis and Robust Search Algorithm in Unstructured Peer-to-Peer Networks. GP2PC, April 2004.

Table 1. Simulation data of 43,632 searches from different nodes for various search algorithms in the dynamic Gnutella network. "Coverage" means the number of nodes visited un-repeatedly. "Message" shows the total number of messages generated. "Rsp Time" indicates the average time a search spends when it succeeds. "Query Hits" means the number of the targets returned. "Success (%)" represents the percentage a search returns the target successfully. "Hit Rsp" is calculated according to Equation 5. "SE (%)" is calculated as Equation 7.

	Hop	1	2	3	4	5	6	7
Flooding	Coverage	11.53	265.55	3455.76	14179.83	18773.39	19103.41	19103.45
	Message	16.00	423.00	7774.00	79206.00	91404.50	158220.30	158268.90
	Rsp Time	1.95	2.82	2.33	2.20	2.13	2.18	2.23
	Query Hits	0.11	2.72	35.61	145.66	192.20	195.47	195.47
	Success (%)	9.71	76.63	99.57	99.99	99.99	99.99	99.99
	Hits Rsp	0.11	1.64	12.38	39.89	49.20	54.71	55.98
	SE(%)	6.67	29.63	15.85	5.036	5.38	3.44	3.54
Random Walk	Coverage	6.81	16.56	26.14	35.78	45.21	54.60	63.82
	Message	10.00	20.00	29.90	39.80	49.70	59.50	69.20
	Rsp Time	1.00	1.59	2.08	2.54	2.99	3.39	3.79
	Query Hits	0.07	0.18	0.29	0.40	0.51	0.62	0.73
	Success(%)	6.81	16.66	26.44	33.39	40.96	47.26	53.09
	Hits Rsp	0.07	0.13	0.16	0.19	0.21	0.23	0.25
	SE(%)	4.82	10.45	13.81	15.91	17.50	18.28	18.81
Dynamic	Coverage	11.69	116.39	229.43	341.44	448.51	553.48	654.11
	Message	16.20	137.40	258.50	379.60	499.50	619.40	738.00
	Rsp Time	1.00	1.82	2.08	2.25	2.40	2.51	2.61
	Query Hits	0.12	1.29	2.58	3.84	5.06	6.25	7.40
	Success(%)	10.65	59.89	76.77	84.33	89.05	92.01	93.92
	Hits Rsp	0.12	0.70	1.14	1.45	1.69	1.89	2.06
	SE(%)	7.84	30.64	33.73	32.18	30.18	28.09	26.17
Modified BFS	Coverage	4.18	27.50	148.75	860.51	3669.17	9396.35	13450.03
	Message	3.40	28.90	162.70	1002.50	5430.30	21957.10	46972.70
	Rsp Time	1.00	1.86	2.60	3.08	3.12	3.13	3.13
	Query Hits	0.03	0.27	1.52	8.86	37.81	96.69	138.32
	Success(%)	3.23	22.66	64.05	97.53	99.95	99.99	99.99
	Hits Rsp	0.03	0.15	0.57	2.40	8.19	18.01	23.95
	SE(%)	3.16	11.91	22.37	23.37	15.08	8.20	5.10