Explore the "Small World Phenomena" in Pure P2P Information Sharing System

Yi Ren Chaofeng Sha Weining Qian Aoying Zhou
Department of Computer Science and Engineering
Fudan University
{cfsha, wnqian,ayzhou}@fudan.edu.cn

Beng Chin Ooi Kian-Lee Tan
Department of Computer Science
National University of Singapore
{ooibc,tankl}@comp.nus.edu.sg

Abstract

Pure Peer-to-peer architecture is becoming an important model for information sharing among dynamic groups of users with its low cost of entry and its natural model for resource scaling with the community size. Recent studies on several pure P2P information-sharing systems have posed new questions and challenges in this area. By identifying two key factors in such an environment, we propose a new heuristic search algorithm to make better use of the "small world phenomena" among the peers in order to find the "six degrees of separation" more efficiently. We show by experiment that our heuristic algorithm out-performs the traditional BFS algorithm with an over 10% performance-increase when querying related information, and a 20% increase when a shift of interest takes place. The heuristic algorithm also has a better control over the number of node-to-visit using our Node-Count feature than the existing TTL mechanism.

1. Introduction

Pure Peer-to-peer architecture is becoming an important model for information sharing among dynamic groups of users with its low cost of entry and its natural model for resource scaling with the community size. We motivate our work in the field of P2P information sharing for two reasons. First, we believe a P2P information-sharing system is indeed needed and possible to implement; secondly, we believe that with a finer granularity of sharing and searching, a new heuristic search algorithm can be designed to explore the "small world phenomena" better among the peers, so that we can find the "six degrees of separation" more efficiently.

Compared with P2P information-sharing systems, current P2P file-sharing systems have two major limitations: they only support searches by an identifier or by keyword matching. In "structured" P2P file-sharing systems like Chord [14], CAN [12], Tapestry [16] and Pastry [5], files or their location information are stored according to their hashed identifiers and retrieved by these identifiers using different DHT routing algorithms. In "unstructured" P2P file-sharing systems, either a global index [11] or a global schema [6] is held to facilitate searching. Hence, in both groups of systems, no rich query based on information content is supported. Secondly, P2P file-sharing systems only have limited varieties of topics and data types. Like in Napster, Gnutella and Kazaa, most files being shared are music and movie files, which can be roughly categorized into related topics like Pop, Rock, Sci-Fi, Drama etc. While in an information sharing systems, the topics of such data can span from music to politics, from history to automobile etc., like in a WWW environment and most of the files are text-based documents. Hence, only a small number of people are using P2P file-sharing systems now comparing to the vast majority of people who use search engines like Google, Toema etc., where a large amount of topics are shared.

A P2P information sharing system differs from a centralized search engine as well. A centralize search engine exhibits a similar model as a hybrid P2P system like Napster, where queries are submitted to the centralized server and data is retrieved from other computers. Thus, a centralized search engine also suffers from a single-point-of-failure problem and can easily become a performance bottleneck. On the other hand, a P2P information-sharing system can provide other features that are hard, if not impossible to achieve in a centralized search engine:

- In a P2P information-sharing system, it is possible to cluster peers based on their interests, e.g. by adding peers with similar interests as one's neighbor. Thus, query results based on different clustering can be return to provide a personalized search experience.
- In a P2P information-sharing system, it is also possible to build an overlay subsystem to create virtual P2P groups, so that people can share and search data with whom they would like to share with, which is impossible to achieve in a centralized search engine.

Systems like PlanetP [3] and PeerIS [8] have explored the possibility of such an information-sharing system.

Hence, we believe a P2P information-sharing system is indeed needed and is possible to implement.

P2P information-sharing systems also provide new opportunities for better algorithms to explore the "small world phenomena" among the peers.

The first research on the problem of the "small world phenomena" can be dated back into pre-Internet context: a famous study done by Stanley Milgram. Milgram was seeking to determine whether most pairs of people in society were linked by short chains of acquaintances, and for this purpose, he recruited individuals to try forwarding a letter to a designated "target" through people they knew on a first-name basis [7]. Milgram concluded his research by showing that most pairs of people are joined by a median number of six steps, a so-called "six degrees of separation" principle.

Some of the current techniques in existing pure P2P systems do consider the phenomena. In systems like Gnutella, by selecting neighbor that returned the greatest number of results, they assume a high similarity among the neighbors so that a query of their common interest can be returned more efficiently. This resembles in human society that people are more willing to have friends with common interests. Another research by [1] indicates that by selecting neighbor with highest degree, searches in a network with nodes of a power-law distribution can be more effective. This again resembles in human society that people usually direct their needs to someone with more social-connections. Nevertheless, we believe the information available in existing P2P file-sharing systems, where only searches by identifier [5, 12, 14, 16] or by simple keyword matching [6,11] are supported, are not rich enough to make a sound judgment about where the query shall be directed. Hence, in a P2P information-sharing system, where a finer granularity of sharing and searching is available, more information such as similarity-based rankings may help to improve the search heuristic to make a better use of the "small world phenomena".

Our main contributions in this paper are:

- We identify two important factors in a P2P information-sharing environment, which haven't been considered by previous works. Namely, the dramatic increase of available topics comparing with the relatively limited number of people's interests, and the shifting rates of their interests.
- We design a heuristic search algorithm to take advantage of the similarity-based rankings available in a P2P information-sharing environment, so that a sound judgment can be made to direct queries more efficiently. Thus, explore the "small world phenomena" better.
- We add a new node-to-visit feature to our heuristic algorithm. Instead of using time-to-live (*TTL*), we set an upper bound *NodeCount* to the total number of node-to-visit. We show by experiments that the new feature has a better control than the existing *TTL* mechanism, which generates an exponential growth of the number of node-to-visit.

The rest of this paper is organized as follows. In Section 2, we give the description of the problem by identifying two important factors in a P2P information-sharing environment. We propose the heuristic search algorithm and the new node-to-visit control feature in Section 3. Experiments are discussed in Section 4. An overview of related works is presented in Section 5 and we conclude our research and propose future work in the last section.

2. Problem Description

Two important factors in a P2P information-sharing environment have put up new requirements for a search algorithm. Namely, the dramatic increase of available topics comparing with the relatively limited number of people's interests, and the shifting rates of their interests.

2.1. Increase of Topics

One of the major differences between the traditional P2P file-sharing system and a P2P information-sharing system is the dramatic increase of the amount of topics available, whereas people's interests tend to be limited.

In a traditional P2P file sharing system like Gnutella and Kazaa, most files being shared are music and movie files, which can be roughly categorized into related topics like Rock, Pop, Sci-Fi and Drama etc. Several factors limit the diversity of topics in such a P2P file-sharing system. In a traditional P2P file-sharing system, the granularity of sharing is coarse-grained. The diversity of topics of music and movie files are usually quite limited and these files can be well identified by a global schema, like author, title etc. On the other hand, more topics reside in a form of documents, which are hard to distinguish one from the other by a simple schema. Thus, information that cannot be well represented by a schema is left out of the scenario. In addition, at the time of emergence of the P2P file-sharing systems, either effective search techniques haven't been developed like in the field of image retrieval, or successful algorithms in a centralized environment haven't been ported and tested in the P2P environment such as text-based content retrieval. Nonetheless, some works have been done targeting both problems. For the latter problem, systems like PlanetP [3] and PeerIS [8] have made successful explorations. Hence, we have full confidence that a P2P information sharing system can be done and will have a dramatic increase of the diversity of the topics in its shared data.

Despite the dramatic increase of the topics, we argue that people's interests are still limited. We study the inter-

est-span of the people in the Google newsgroups and find out that people in the newsgroups have an average of five interests comparing to over 800 newsgroups. This is only a count of the first level newsgroups. There can be even more if we count the sub groups, e.g. alt.* has over 2000 sub groups. Although these topics might overlap each other, with the vast amount of groups, it is clear that its increase will out-weight the people's interests tremendously in a P2P information-sharing system.

2.2. Shift of People's Interest

We argue that the interest shift rate is also a vital factor, especially in today's dynamic information era. This may not be a significant problem in a P2P file-sharing system and people seldom shift their interest, since the topics in such a system are quite limited and inter-related somehow. For example, people won't issue queries like "Find me a paper about P2P architecture" in systems like Gnutella, because these kind of information are simply not shared and can't be searched, whereas it is possible to issue all kinds of queries in an information-sharing system like search engines such as Google. Hence, in a P2P information-sharing system, with the dramatic increase of available topics, people will become more easily to change their interests and to discover new knowledge like in a WWW environment.

Two types of the interest shift can be identified:

- **Long-term interest shift:** A case that people decide to move into a new topic, and to keep exploring information for a long period. Thus, information about this topic may have a tremendous influence in its shared data and its later queries.
- **Short-term interest shift:** A case that people need only a quick skim about the new topic and information about the topic may not be explored later.

These two types of interest shift pose two requirements for an adaptive algorithm. The former requires an algorithm that can pull the peer into a cluster of its new interest; the latter only requires the search to be efficient once while leaving its original clustering intact. Because a long-term interest shift will have an effect on the overall topology of a P2P system, and the behavior of such an interest shift is still unclear, we will focus on the short-term interest shift in this paper.

Traditional adaptive and search algorithms only address the clustering problem, while as we indicate later in our experiments it doesn't perform as well as our heuristic algorithm when the interest shift takes place. Hence, the interest shift rate should be a key factor in a pure P2P information sharing system.

3. Algorithm

In present well-known P2P systems, two search techniques can be found [15]:

- **Breadth-first traversal (BFS)**: A system-wide maximum *TTL* value is set in terms of hops and search by flooding all the neighbors. Systems like Gnutella adopt this mechanism.
- **Depth-first traversal (DFS)**: Each peer forwards the query to only one of its neighbors within a depth limit of *D*, like in Freenet.

In this paper, we present a BFS-DFS Combined heuristic algorithm based on the similarity rankings available in an information-sharing system.

3.1. Heuristic Algorithm

Table 1. Symbols used in our heuristic algorithm

Symbol	Meaning
T_{low}	Lower threshold of result rankings
T_{high}	Higher threshold of result rankings
N_{result}	Number of results to be considered
$M_{neighbor}$	Number of neighbors to send query
Query	Current Query
Resultlist	Linked List of query results
Peer	Peer being queried
Similarity	Value based on the Resultlist

The major idea behind our heuristic algorithm is that by taking the similarity rankings that reflect the similarity between the query and the shared information, a more efficient search can be done. When the rankings are low, there is a low probability that the searched peer may share a common interest with the query, so may its neighbors. Thus, the query should be sent to fewer neighbors. On the other hand, when the rankings are high, the probability of the peer and its neighbors being able to answer the query may be high. Hence, more neighbors should be consulted. The pseudo code in Figure 1 summaries our heuristic algorithm and Table 1 describes the symbols we used in the algorithm.

```
Procedure 1 Peer.Main()
1:  Query=Peer.GetQuery()
2:  Resultlist=Peer.ProcessQuery(Query)
3:  Resultlist.Sort()
4:  Similarity=Peer.EvalSimilarity(Resultlist)
5:  if Similarity> T_high then
6:        M_neighbor=Peer.GetNumOfNeighbors()
7:  else if Similarity<T_low then
8:           M_neighor=1
9:        else
10:          M_neighbor=Peer.GetNumOfNeighbors()*
                  (Similarity- T_low)/(T_high- T_low)
11:       end if
12: end if
13: if M_neighbor=0 then
14:       M_neighbor=1
15: end if
16: Send Query to M_neighbor neighbors
17: return
```

```
Procedure 2 Peer.EvalSimularity()
1:  count=0
2:  totalrank=0
3:  while not Resultlist.EOF() do
4:          result=Resultlist.GetCurrentResult()
5:          if count>N_result then
6:                  break
7:          else
8:                  count=count+1
9:                  totalrank=totalrank+result.GetRank()
10:                 Resultlist.NextResult()
11:         end if
12: return totalrank/N_result
```

Procedure 3 *Peer*.GetNumOfNeighbors()
Return the number of the peer's neighbors

Procedure 4 *Peer*.GetQuery()
Get the query the peer just received

Procedure 5 *Peer*.ProcessQuery(*Query*)
Process the query and return a linked list of results with rankings

Figure 1. Pseudo code of our heuristic algorithm

3.2. Node-to-Visit Control Feature

The fundamental problem of existing time-to-live (*TTL*) mechanism is that the number of node-to-visit is an exponential of *TTL*. Usually, applications limit *TTL* to less or equal to 7 like Gnutella, which means only 7 levels of control is possible. Because it may not be able to know the number of neighbors for all the nodes beforehand, the total number of node-to-visit eventually maybe enormous and out of control.

Instead of using *TTL*, we use an exact *NodeCount* as the upper bound for the total number of node-to-visit. To implement the new node-to-visit feature takes 4 steps as follows.

For visited nodes:

1) If this node has been visited before, send the *NodeCount* back to the host node and algorithm terminates. Otherwise, proceed to 2.

2) Decrement *NodeCount* by 1. If *NodeCount* reaches zero, proceed to 3. Otherwise, divide the *NodeCount* into $M_{neighbor}$ parts and forward the query with the new *NodeCount* to $M_{neighbor}$ neighbors. Algorithm terminates.

3) Register node ID to the host node for future *NodeCount* update. Algorithm terminates.

For host node:

4) If there are both extra feedback *NodeCount* and registered node IDs, distribute the *NodeCount* to registered nodes and start step 2 for these nodes. Keep periodic pulling until current query terminates.

In addition, the algorithm can be altered to distribute the feedback and registering tasks onto the nodes along the query path. Thus, the host node can be alleviated from being a possible performance bottleneck. More *NodeCount* can also be assigned later to the registered IDs if the user wants to contact more nodes.

It is obvious that this new feature provides a much richer control over the number of node-to-visit than the 7-level-control of the regular *TTL* mechanism. Also, without knowing the exact network topology and the number of neighbors for each node, it is still possible to control the number of node-to-visit and the number will never exceed *NodeCount*. As we later indicate in our experiments, this new mechanism does perform better than the existing *TTL* mechanism, especially when only 10%-30% of related documents are expected from each query.

4. Experiments

4.1. Models

Topology Model. A number of large distributed systems, ranging from social to communication to biological networks display a power-law distribution in their node degree [1]. This distribution reflects the existence of a few nodes with very high degree and many with low degree, a feature not found in uniformly random graphs.

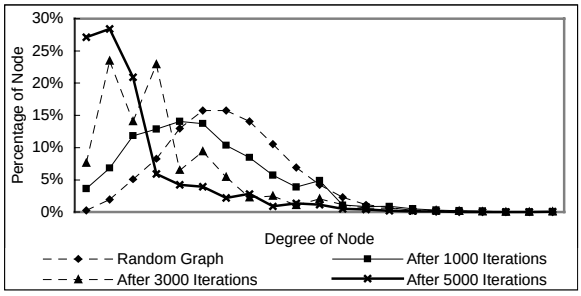

Figure 2. Node Degree Distribution

As shown in [1,4], pure P2P file-sharing systems do exhibit such topology. Thus, in this paper, we assume a network topology of a power-law distribution with power-law index around –2 (-2.3 in Gnutella[4]). To obtain such a power-law topology, topology generators [2] can be used. However, we take a different approach. Since our work focuses on the adaptive-ness of the algorithms and their effectiveness in a new information-sharing environment, we want to see how effective and adaptive such algorithms are to help cluster peers with similar interests. Hence, we use a random graph initially and by running simulated queries pertaining to the peer's interests, we are able to obtain such a power-law distribution as shown in Figure 2.

This result coincides with the beliefs in [1], which thinks it may not be coincidental that several large networks are structured in such a power-law topology. Rather, they find it likely that these networks could have evolved to facilitate search and information distribution.

Networks where locating and distributing information, without perfect global information tend to be power-law with exponents favorable to local search.

File Distribution Model. In a measurement study of Gnutella [13], we see that as high as 25% of the Gnutella clients do not share any files. Furthermore, about 75% of the clients share 100 files or less, whereas only 7% of the clients share more than 1000 files. A simple calculation reveals that these 7% of users together offer more files than the rest of the users. In another measurement [3], 500 users are sharing more than 6TB of data, where 9% of the users are responsible for providing the majority of the files in the community. A Weibull Distribution with parameters α=0.7 and β=46 is used to model the file distribution.

Although other distributions are available such as even distribution and may worth studying, it is necessary to base our research on documented measurements and we believe it is such a heterogeneous distribution that leads to the power-law distribution of the network topology. Hence, a Weibull file distribution is adopted in our study.

4.2. Experiment Setup

We conduct our experiment using a simulator we developed similar to the idea behind Anthill [10], which demonstrates that a complex adaptive systems (CAS) can be the basis of a programming paradigm for P2P applications.

A *TotTopics* of up to 500 and an *AvgInterests* of 5 are chosen according to our study of the Google newsgroups, which have roughly 800 major newsgroups and an average interest of 5 for people who have posts. We set an *AvgNeighbor* of 6 (average degree is roughly 3 in Gnutella [15]) to reflect the need for more neighbors with the growth of *TotTopics* and *AvgInterests* but still in respect to the hardware limitations of the nodes. *HopCount* of 7 is taken from the default value in Gnutella and *NodeCount* is chosen as indicated in Table 2 so that a comparable result can be achieved in the experiments. In [4] and [13], approximately 8000-10000 nodes can be crawled in a Gnutella network. Hence, a maximum scale of 16000 is chosen to reflect the size of real P2P networks.

We only consider *Max Doc Returned* as the neighbor selection rule in our experiments. Because only *Max Doc Returned* and *Max Degree* are more related to the "small world phenomena" as described in Section 1, while others are about network latency, message queue length etc.[15], which are not our focus in this paper. On the other hand, although *Max Degree* can also result in a power-law distribution as *Max Doc Returned* does, one or two of the nodes will be connected by almost all the rest of the nodes as we discover in our experiments. This might be a desirable factor for a search algorithm; however, it is undesirable for real application users. Thus, only *Max Doc Returned* is used.

In order to simulate an information-sharing environment, after we distribute the files for each node with a Weibull distribution, we divide the files into different *TopicGroups* according to *AvgInterests*. Each *TopicGroup* represent a topic closely related to an interest. Next, we divide each *TopicGroup* into *SubGroups*. By doing this:

- A high similarity ranking is returned if there is a hit in both *TopicGroup* and *SubGroup*.
- A medium ranking is returned if there is only a hit in the *TopicGroup*.
- A low ranking is returned if there is no hit at all.

We compare our heuristic algorithm with the traditinoal BFS algorithm, which is used in Gnutella. We collect the data by issuing queries on random peers with 7 sets of different *TTL* and *NodeCount*. These queries are divided into two groups: *Related Queries* and *Non-Related Queries* to reflect people's interest shifts. Each data is collected by running the simulation 2000 times and taking the average. The symbols we use in our simulation can be found in Table 2.

Table 2. Symbols used in the simulation

Symbol	Meaning	Value
TotTopics	Total topics available	50 250 500
AvgInterests	Average interests	5
AvgNeighbor	Average neighbors per peer	6
HopCount	Query's TTL in BFS	1..7
NodeCount	Query's TTL in Heuristic Algorithm	$\frac{AvgNeighbor^{HopCount+1}-1}{AvgNeighbor-1}$
Scale	Scale of simulation	4000 8000 16000
TopicGroup	Files in a same topic	
SubGroup	SubGroup in a TopicGroup	
Max Doc Returned	Add the node with the most documents returned as a neighbor	
Max Degree	Add the node with the highest degree as a neighbor	
Related Query	A random query about a *subgroup* of existing *topicGroups* on the peer (no shift of interests)	
Non-Related Query	A random query about a *subgroup* of non-existing *topicGroups* on the peer (shift of interests)	

4.3. Metrics

Performance-Increase. This metric is used to compare the effectiveness of the heuristic algorithm in retrieving relevant documents with the BFS algorithm when visiting the same number of nodes either with or without the interest shifts.

Definition 4.3.1*: N* being the number of nodes visited, $D_{BFS}(N)$ is the percentage of relevant documents retrieved for BFS algorithm when issuing *Related Query*; $D_H(N)$ is the percentage for the heuristic algorithm. $D_{BFS\text{-}S}(N)$ and $D_{H\text{-}S}(N)$ are the measurements when issuing *Non-Related Query* for the two algorithms, respectively (S stands for shift of interests).

Definition 4.3.2*:* For *N* number of nodes being visited, the performance-increase when issuing *Related Query* (*PI*) and *Non-Related Query* (PI_s) are as follows.

$$PI(N) = \frac{D_H(N) - D_{BFS}(N)}{D_{BFS}(N)} \quad (1)$$

$$PI_S(N) = \frac{D_{H-S}(N) - D_{BFS-S}(N)}{D_{BFS-S}(N)} \quad (2)$$

Control Effectiveness. This metric is used to compare the effectiveness of the control between traditional *TTL* and our new *NodeCount* feature.

Definition 4.3.3*: D* being the percentage of the relevant documents retrieved over the total number of relevant documents, $T_{actual\text{-}TTL}(D)$ is the actual number of nodes that need to be visited in order to retrieve *D* percent of relevant documents when issuing *Related Query*. $T_{actual\text{-}NodeCount}(D)$ is for our *NodeCount* feature. $T_{actual\text{-}TTL\text{-}S}(D)$ and $T_{actual\text{-}NodeCount\text{-}S}(D)$ are for *Non-Related Query*.

Definition 4.3.4*: D* being the percentage of the relevant documents retrieved over the total number of relevant documents, $T_{max\text{-}TTL}(D)$ is the maximum number of nodes the BFS algorithm can visit for the minimum value of *TTL* that makes $T_{actual\text{-}TTL}(D)$ actual visits of the nodes. $T_{max\text{-}NodeCount}(D)$ is the minimum value needs to be specified in *NodeCount* in order to make $T_{actual\text{-}TTL}(D)$ actual visits. $T_{max\text{-}TTL\text{-}S}(D)$ and $T_{max\text{-}NodeCount\text{-}S}(D)$ are for *Non-Related Query*.

Definition 4.3.5*: D* being the percentage of the relevant documents retrieved over the total number of relevant documents, the control effectiveness (CE) when issuing *Related Query* and *Non-Related Query* for both TTL and *NodeCount* are as follows.

$$CE_{TTL}(D) = \frac{T_{actual-TTL}(D)}{T_{max-TTL}(D)} \quad (3)$$

$$CE_{TTL\text{-}S}(D) = \frac{T_{actual-TTL\text{-}S}(D)}{T_{max-TTL\text{-}S}(D)} \quad (4)$$

$$CE_{NodeCount}(D) = \frac{T_{actual-NodeCount}(D)}{T_{max\text{-}NodeCount}(D)} \quad (5)$$

$$CE_{NodeCount\text{-}S}(D) = \frac{T_{actual-NodeCount\text{-}S}(D)}{T_{max\text{-}NodeCount\text{-}S}(D)} \quad (6)$$

4.4. Experiment Results and Analysis

Impact of the increase of total topics available. As we state in Section 2, the increase of the total topics in a P2P information-sharing system will have a great impact on the performance of the search and adjusting algorithms.

Table 3. Increase according to the percentage of the retrieved relevant documents for different *TotTopics*

	10%-30%	30%-50%	10%-50%
4000 5-50	15.86%	10.79%	13.04%
4000 5-250	14.77%	12.78%	14.17%
4000 5-500	14.89%	9.13%	10.76%
Average	**15.17%**	**10.90%**	**12.66%**
4000 5-50 S	15.06%	9.96%	12.22%
4000 5-250 S	22.77%	17.55%	19.87%
4000 5-500 S	29.67%	22.87%	26.27%
Average S	**22.50%**	**16.79%**	**19.45%**

In Figure 3 and Figure 4, 3 sets of *totTopics* (50 250 500) are simulated over 4000 peers with and without interest shift. It is obvious that our heuristic algorithm outperforms the traditional BFS algorithm for the first 1500 nodes visited, with 10% increase for *Related Query* and 17% increase for *Non-Related Query*.

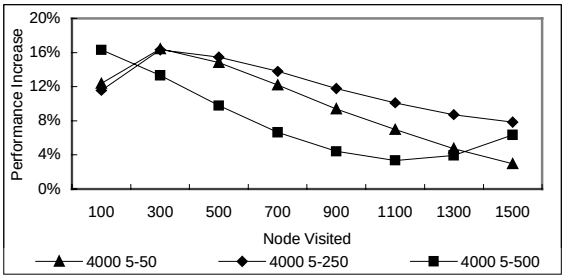

Figure 3 Performance Increase with NO interest shift. The amount of total topics ranges from 50 to 500

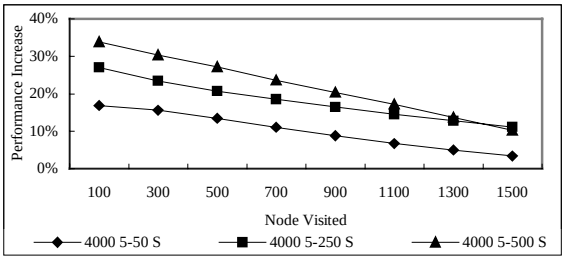

Figure 4. Performance Increase with interest shift. The amount of total topics ranges from 50 to 500.

However, we believe users are more interested in the quality of the results instead of the number of nodes visited. Thus, we average the performance-increase according to the percentage of relevant documents retrieved by the heuristic algorithm over the total number of relevant documents and the results are shown in Table 3. We argue that most users in an information-sharing system only require a small portion of the total relevant documents available, usually 10%-30% is sufficient. Thus, our heuristic algorithm has an even high performance-increase

(15% for *Related Query* and 23% for *Non-Related Query*) in terms of the percentage of the returned documents.

Table 4. Performance-Increase according to retrieved relevant document percentage for different *Scale*

	10%-30%	30%-50%	10%-50%
4000 5-500	14.89%	9.13%	10.76%
8000 5-500	12.48%	9.68%	10.83%
16000 5-500	12.36%	8.37%	10.21%
Average	**13.24%**	**9.06%**	**10.60%**
4000 5-500S	29.67%	22.87%	26.27%
8000 5-500S	21.07%	18.63%	19.97%
16000 5-500S	10.09%	6.43%	8.13%
Average S	**20.28%**	**15.97%**	**18.12%**

Hence, by making a better use of the "small world phenomena", our heuristic algorithm will perform well with *Related Query*, and perform even better with *Non-Related Query* over the traditional BFS algorithm, which queries only by flooding. In addition, as shown in Figure 4 and Table 3, with the shift of people's interests, the heuristic algorithm performances better as more topics are available. This exactly justifies our concerns for the two key factors in a future P2P information-sharing system, namely the increase of topics and people's interest shift.

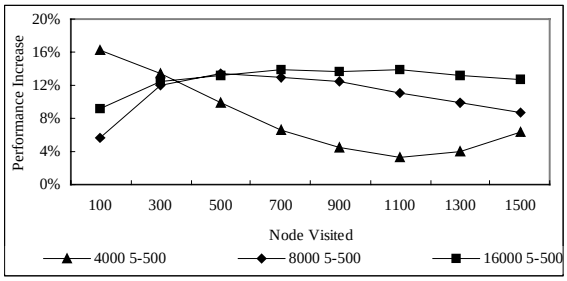

Figure 5. Performance-increase with NO interest shift. *Scale* ranges from 4000 peers to 16000 peers.

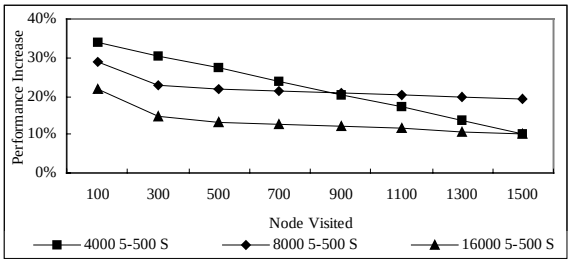

Figure 6. Performance-increase with interest shift. *Scale* ranges from 4000 peers to 16000 peers.

Scalability of the heuristic algorithm. Previous experiments indicate the effectiveness of our heuristic algorithm with the increase of the total topics available and people's interest shift. However, these experiments are run over 4000 peers only and we show that our heuristic algorithm can be scalable as well in this part.

In Figure 5 and Figure 6, 3 sets of *Scale* (4000 8000 16000) are simulated with *TotTopics* of 500 and *AvgInterests* of 5. For the first 1500 nodes, an average performance-increase of 10% is found with *Related Query*, and 18% with *Non-Related Query*.

In terms of percentage of relevant documents retrieved ranging from 10%-30%, 13% and 20% of performance-increase are achieved when issuing *Related Query* and *Non-Related Query*, respectively. Although the performance-increase drops as scale increases in Figure 6 and Table 4 with *Non-Related Query*, a 10% of performance-increase is still obtained over the traditional BFS algorithm for a scale of 16000 nodes. Therefore, our heuristic algorithm does scale well with the size of the sharing community.

Control Effectiveness. All the experiments shown in Figure 7, 8 and Table 5 are done using *TotTopics* of 500 and *AvgInterests* of 5. N stands for *NodeCount* and T for *TTL*.

As indicated in Figure 7 and Figure 8, the *TTL* mechanism has a much rigid shape than *NodeCount*, which is the result of *TTL*'s coarse-grained control of the node-to-visit. On the other hand, the control efficiency of *NodeCount* decreases as the percentage of relevant documents retrieved increases. However, as we indicate in previous sections, usually a 10%-30% will suffice the needs of the users. Thus, the *NodeCount* mechanism can successfully visit 70% of the nodes indicated in *NodeCount* for *Related Query* and 55% for *Non-Related Query*, comparing to 39% and 34% in the traditional *TTL* mechanism.

5. Related Work

Many studies have been conducted to improve the efficiency of search algorithms in a pure P2P environment. Recent studies of Gnutella [1,4,13] have shown that the degree of the nodes in Gnutella does exhibit a power-law distribution and a new decentralized search algorithm to seek high-degree nodes is proposed [1]. Yang and Garcia-Molina have studied the efficiency of various search and adaptive algorithms in a pure P2P file-sharing system [15]. They show that their techniques use up to 5 times fewer resources while maintain the same quality of results as current techniques.

Improvements to make the search more scalable in systems like Gnutella have also been conducted by [9]. In their paper, they take advantage of the heterogeneity of the peers' machine abilities to build a pure P2P system more scalable and efficient than the existing Gnutella.

Besides these structural improvements, new applications like information retrieval have also been explored. In [3], a text-based content search and retrieval system, PlanetP is developed. It indicates that P2P computing model is potentially powerful for information sharing between *ad hoc* communities of user, and an application such as text-based content search is possible and can be made effective by taking the ranking information available into account.

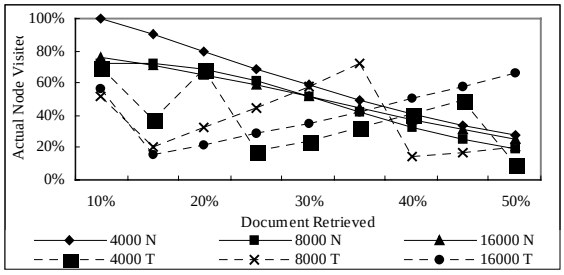

Figure 7. Control Effectiveness (CE) in terms of percentage of relevant documents retrieved with NO interest shift.

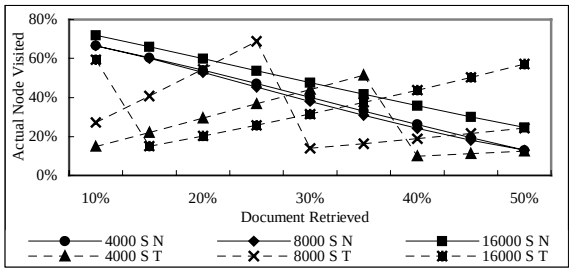

Figure 8. Control Effectiveness (CE) in terms of percentage of relevant documents retrieved with interest shift.

Table 5. Control Effectiveness (CE) in detail.

	16000	8000	4000	**Average**	16000 S	8000 S	4000 S	**Average S**
10%-30% N	64.68%	65.26%	79.32%	**69.75%**	59.78%	52.56%	53.62%	**55.32%**
10%-30% T	31.65%	41.15%	43.37%	**38.72%**	30.38%	40.99%	29.39%	**33.59%**
30%-50% N	38.04%	34.34%	42.02%	**38.13%**	35.90%	24.78%	26.36%	**29.01%**
30%-50% T	50.53%	36.35%	31.41%	**39.43%**	44.07%	18.95%	25.78%	**29.60%**
10%-50% N	51.34%	49.52%	60.89%	**53.92%**	47.87%	38.74%	39.98%	**42.20%**
10%-50% T	41.73%	36.63%	38.82%	**39.06%**	37.86%	31.76%	25.76%	**31.79%**

All the works above provide many useful insights in a pure P2P information sharing system. In this paper, we distinguish our study from theirs by focusing on two major factors in a P2P information-sharing environment and propose a new search heuristic which is more suitable for this environment based on the similarity rankings available to explore the "small world phenomena" better.

6. Conclusions

The flurry of research in P2P system has solved many problems as well as posed new challenges. Information sharing in P2P is just one of these new challenges. We believe a P2P information-sharing system is necessary and with its finer grained sharing, better algorithms can be designed to take advantage of the information available. In this paper, we first identify two key factors in a P2P information-sharing environment: the increase of total topics available and the shift of people's interests. Then, we describe our heuristic algorithm that decides the number of neighbors to contact based on the similarity of peer's data and the query it receives, and the new *NodeCount* feature to make better control over the number of node-to-visit instead of *TTL*. Finally, we show by experiments that our heuristic algorithm does make better use of the "small world phenomena" with an average increase of over 10% with *Related Query* and 20% when an interest shift takes place, and the *NodeCount* feature out performs the traditional *TTL* mechanism.

It remains to refine and evaluate our heuristic algorithm by running through real data collections. In the future, we will keep improving the control effectiveness of the *NodeCount* feature and find a mechanism to discover the diameter of peer clusters with similar interests so that even more efficient search algorithms can be designed.

References

[1] L. Adamic, R. Lukose, A. Puniyani, B. Huberman, "Search in Power-Law Networks," *Phys. Rev. E*, 64 46135 (2001)

[2] T. Bu and D. Towsley. On distinguishing between Internet power law topology generators. *INFOCOM*, 2002

[3] F. M. Cuenca-Acuna and T. D. Nguyen. "Text-Based Content Search and Retrieval in ad hoc P2P Communities". *The International Workshop on Peer-to-Peer Computing*, 2002

[4] Clip2. Gnutella measurement project, May 2001.

[5] P. Druschel and A. Rowstron, Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. *Middleware*, 2001

[6] Gnutella http://www.gnutella.com

[7] J. Kleinberg. Small-World Phenomena and the Dynamics of Information. *NIPS*, 2001

[8] B. Ling, Z. Lu, W. S. Ng, B. C. Ooi, K. L. Tan, A. Y. Zhou A Content-Based Resource Location Mechanism in PeerIS. *WISE*, 2002

[9] Q. Lv, S. Ratnasamy, S. Shenker Can Heterogeneity Make Gnutella Scalable? *IPTPS*, 2002

[10] A. Montresor, H. Meling, Ö. Babaoglu, Towards Adaptive, Resilient and Self-Organizing Peer-to-Peer Systems, *1st International Workshop on P2P Computing*, 2002

[11] Napster http://www.napster.com

[12] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, A Scalable Content-Addressable Network, *SIGGCOMM*, 2001

[13] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. *Multimedia Computing and Networking*, 2002.

[14] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM* 2001

[15] B. Yang, H. Garcia-Molina: Improving Search in Peer-to-Peer Networks. *ICDCS*, 2002

[16] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for faultresilient wide-area location and routing. *Technical Report U. C. Berkeley*, 2001.