Real Datasets for File-Sharing Peer-to-Peer Systems

Shen Tat Goh[1], Panos Kalnis[1], Spiridon Bakiras[2], and Kian-Lee Tan[1]

[1] Department of Computer Science,
National University of Singapore,
3 Science Drive 2, Singapore
{gohst, kalnis, tanlk}@comp.nus.edu.sg
[2] Department of Computer Science,
Hong Kong University of Science and Technology,
Clear Water Bay, Hong Kong
sbakiras@cs.ust.hk

Abstract. The fundamental drawback of unstructured peer-to-peer (P2P) networks is the flooding-based query processing protocol that seriously limits their scalability. As a result, a significant amount of research work has focused on designing efficient search protocols that reduce the overall communication cost. What is lacking, however, is the availability of real data, regarding the exact content of users' libraries and the queries that these users ask. Using trace-driven simulations will clearly generate more meaningful results and further illustrate the efficiency of a generic query processing protocol under a real-life scenario.

Motivated by this fact, we developed a Gnutella-style probe and collected detailed data over a period of two months. They involve around 4,500 users and contain the exact files shared by each user, together with any available metadata (e.g., artist for songs) and information about the nodes (e.g., connection speed). We also collected the queries initiated by these users. After filtering, the data were organized in XML format and are available to researchers. Here, we analyze this dataset and present its statistical characteristics. Additionally, as a case study, we employ it to evaluate two recently proposed P2P searching techniques.

1 Introduction

Distributed peer-to-peer (P2P) systems provide an alternative architecture to the traditional client/server model and their initial success has captured the attention of the research community during the past few years. P2P nodes are both clients and servers and do not depend on centralized infrastructure.Participation is ad-hoc and dynamic, since nodes may independently join or leave the network.

P2P networks are classified into two main categories: unstructured (e.g., Gnutella [1]) and structured (e.g., CAN [9] and Chord [13]). Unstructured broadcast-based P2P networks are the most widely used systems today for information exchange among end-users, and provide the basis on which many popular

L. Zhou, B.C. Ooi, and X. Meng (Eds.): DASFAA 2005, LNCS 3453, pp. 201–213, 2005.
© Springer-Verlag Berlin Heidelberg 2005

file-sharing applications are built. Their popularity emerges primarily from their inherent simplicity; nodes that wish to exchange information, join randomly the overlay topology and are only responsible for their own data. The result is an inexpensive, easy-to-use system, which does not require any form of central administration. One major drawback, though, is the query processing protocol; whenever a node receives a query message, it broadcasts it to all of its neighbors. This is done recursively until a maximum number of hops is reached. This algorithm does not scale well to a large population size, since the whole network is overwhelmed with query messages.

As a result, research has focused on designing efficient search protocols that reduce the overall communication cost. Most of the reported results, however, are based on ad-hoc synthetic data. Clearly, the availability of real data regarding the content of users' libraries and the exact queries that these users ask, would generate more meaningful and realistic results. Motivated by this fact, we developed a Gnutella-based probe and gathered detailed data from a large and diverse user population.

In this paper, we present the data that we collected from around 4,500 Gnutella users over an extended time period. Our dataset contains information about each node (e.g., its connection speed and the software it uses) together with the index of the entire users' libraries, which is around 850,000 files in total. Additionally, we capture the exact queries initiated by each node. These data were filtered, organized in XML format and are now available to the public [3]. Moreover, since music sharing is very common in P2P networks, we processed separately a subset of the data consisting only of music files. There are around 2,000 nodes sharing almost 200,000 songs which we further organized based on the title, artist and genre (e.g., pop, rock, etc). We analyzed these data and present here some useful statistics and distribution graphs. Finally, as a case study, we investigate the performance of two recently proposed P2P searching techniques, namely Dynamic Reconfiguration [4] and Interest-based Locality [12], using the collected workload. To the best of our knowledge, our work is the first one to deal with the exact contents of the users' libraries and correlate them with the observed query patterns.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 describes the data collection methodology and presents an analysis of the dataset. Section 4 gives a brief overview of two case studies on which the generated workload was applied, followed by the detailed results of the trace-driven simulations. Finally, Section 5 concludes our work.

2 Related Work

Research in the P2P area was triggered by the apparent success of systems like Napster [2] and Gnutella [1]. Napster is a hybrid system, since it maintains a centralized index which is used for searching. Gnutella, on the other hand, is a pure P2P system and performs searching by Breadth-First-Traversal (*BFT*). Each peer that receives a query propagates it to all of its neighbors up to a

maximum of d hops. The advantage of BFT is that by exploring a significant part of the network, it increases the probability of satisfying the query. The disadvantage is the overloading of the network with unnecessary messages. Yang and Garcia-Molina [14] observed that the Gnutella protocol could be modified in order to reduce the number of nodes that receive a query, without compromising the quality of the results. They proposed three techniques: *Iterative Deeping*, *Directed BFT*, and *Local Indices*. A technique similar to Local Indices is used in Ref. [15], the only difference being that indices are kept only in a subset of powerful nodes called *super-peers*.

Several studies have performed measurements in a wide range of P2P systems. Saroiu et al. [10] studied the characteristics of peer-to-peer users in the Gnutella and Napster file-sharing systems. In particular, the authors measured several parameters, including bandwidth, delay, availability (i.e., the fraction of time a user is active), and sharing patterns. Sen and Wang [11] measured flow-level information at multiple border routers of a large ISP network. They collected data from three popular P2P systems over a period of three months. The reported results illustrate a large skew in the distribution of traffic across the network, at different levels of spatial aggregation.

Contrary to the above studies that focus on P2P traffic characterization, the work by Gummadi et al. [7] provides some useful insight regarding the nature of file-sharing workloads. The authors analyzed a 200-day trace from the Kazaa network, and showed that P2P workloads are substantially different from their Web counterparts. Specifically, object popularity changes over time (with new objects being more popular), and the aggregate popularity distribution does not follow a Zipf curve. In addition, the authors observed a considerable locality in the P2P workload, which may be exploited by object caching.

In contrast to our work, none of the above papers provides the exact contents of the users' libraries together with the actual user queries.

3 Data Analysis

We implemented our probe by modifying a Limewire client [8], which connects to Gnutella networks. Limewire is implemented in Java and the source code is publicly available and well-documented. We forced our client to be promoted to an ultra-peer. In this way, we were able to observe all the queries submitted by leaf nodes connected directly to the probe. For each query, we captured the IP address[1] and port number of the initiating leaf node, a time-stamp and the query string (i.e., a set of keywords). We used the *Browse Host* operation to retrieve the contents of the leaf peers' libraries. Notice that the peers respond to this operation since our probe is an ultra-peer. The information of each peer includes its address, the type of the connection as reported by the client (e.g., Modem, Cable, etc.) and the index of its library. Index entries are composed by the filename, the filetype and the size of the file in bytes. The resulting dataset

[1] To preserve anonymity, we replaced the IP by a randomly generated unique key.

Table 1. Statistics for the generic and the music files dataset

	Generic Dataset	Music files Dataset
Number of Users	4,447	2,000
Number of queries	11,075	5,462
Total number of files	845,454	195,023
Number of distinct files	505,818	58,848
Number of artists	n.a.	15,499
Number of Genres	n.a.	245

relates the library of each user with the queries he asked. Except from requests originating from leaf nodes connected to the probe, many queries arrive through other ultra-peers. In such case, we cannot always retrieve the peer's index, since some users restrict the *Browse Host* operation for remote peers.

Peers may enter and leave the network frequently. Ideally, we want to record all the queries issued by a specific user, irrespectively of how often he reconnects. Unfortunately, due to dynamic IP, it is not easy to distinguish the identity of a peer. To minimize the problem, we do not rely on the IP address but we compare the contents of the libraries. If at least 90% of the contents of a peer (also considering the order) are the same as the contents of another, we assume that the two peers are identical. We allow a 10% difference, since a user might add or delete some files while he is off-line. Nevertheless, we cannot guarantee that all the queries are captured; a peer may reconnect to a different ultra-peer and its subsequent queries are not necessarily routed through our probe.

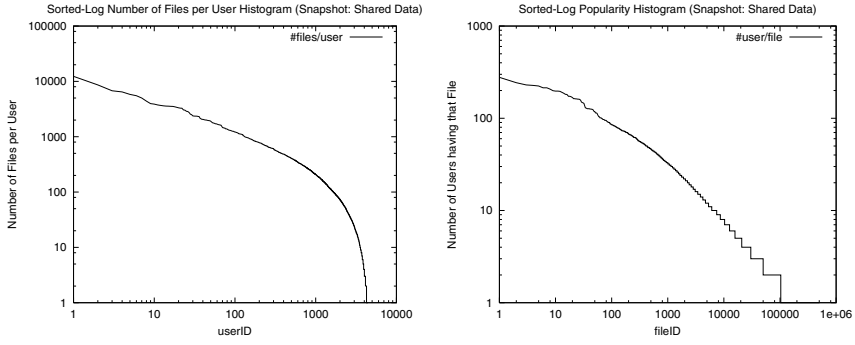
Data were collected over a two months' period. We employed two probes, one in Singapore and the other in Hong Kong[2], hoping to capture a geographically diverse population. Additionally, during the collection period the probes were disconnected and reconnected to the network several times, ensuring that our data are independent of any specific network configuration.

3.1 Generic Dataset

Here we analyze our generic dataset consisting of a set of almost 4,500 users, the indexes of their libraries (around 850,000 files of various types) and they queries they asked. Table 1 presents summarized statistics of the data. The dataset is available online [3] in XML format.

Figure 1 shows the relation between users and files; observe that both axis are logarithmic. In Figure 1(a) we show the number of files per user, after sorting the users in descending order according to the number of files they have. It is clear that most of the users have a significant number of files, although there exist some users with many files and others with only a few; this is similar to the results of Saroiu et al. [10]. In Figure 1(b) we present the popularity of each file (i.e., the number of users that have a particular file). As expected, the graph resembles a Zipf distribution.

[2] The domains of the captured clients where not restricted to these areas.



(a) Files per user. (b) Users per file.

Fig. 1. Distribution of files

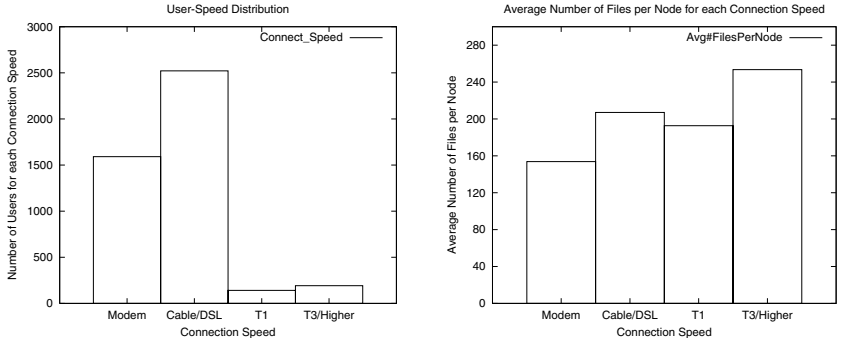
In Figure 2 we present statistics related to the connection speed of the peers. Figure 2(a) shows the number of peers for each connection speed category. It is obvious that the slow connections dominate the network. Notice that these are the speeds reported by the peers themselves. Many users, however, deliberately report low bandwidth to discourage other peers from downloading files [10]. In the next graph (Figure 2(b)) we draw the average number of files shared by nodes belonging to a specific connection speed category. Although we observe some variations, it seems that the size of a user's library does not depend significantly on the connection speed.

In Figure 3(a) we present the number of queries initiated by each user. Both axis in this graph are logarithmic. The graph resembles a Zipf-like distribution, indicating that some users ask a lot of queries, while most others ask only a few. We also investigate the relationship between the number of queries asked by user and their connection speed. In contrast to our intuition, the connection speed seems to be irrelevant.

Finally, Figure 3(b) combines the queries with the contents of the users' libraries. It shows, for an average user, the cumulative value of the answers returned by other users, as a percentage of the total answers. For example, during the process of answering a specific query, if a node contacts 50 other peers it can retrieve around 62% of the available answers in the entire network. From the graph it is obvious that for any query a node needs to contact at most 120 out of the 4,500 peers, in order to find all the qualifying answers in the network. This fact indicates that it is possible to develop algorithms which answer queries efficiently in large P2P systems.

3.2 A Special Case: Music Files

A substantial percentage of the traffic in P2P systems is due to the exchange of music files among the users. To facilitate experimentation in this domain, we extracted from the original data a subset consisting only of music files. There were



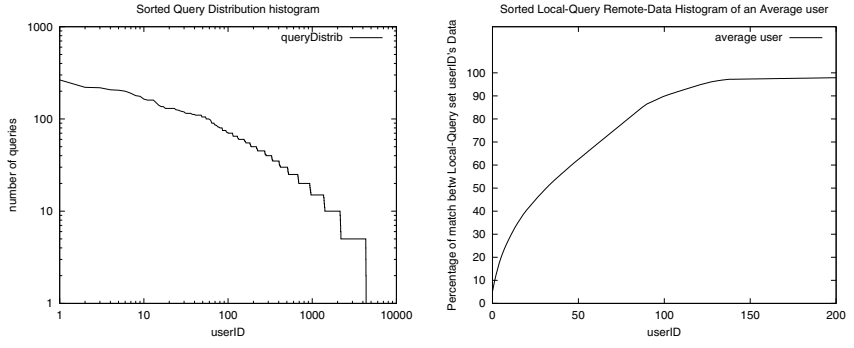
(a) Number of users vs. connection (b) Number of files vs. connection

Fig. 2. Group by connection speed

2,000 nodes containing at least one music file, while we captured approximately 200,000 such files in total; detailed statistics are presented in Table 1. Due to the restricted domain, we were able to capture additional attributes for each file. From the filename itself, we extracted the song title and the artist. Then, by consulting song databases available in the Web, we categorized each song by its genre (e.g., pop, rock, etc.) In total, 245 different genres were identified. The music file dataset is also available online [3] in XML format.

In general, we observed that the distribution of songs among users is similar to the distribution of general files presented in Figure 1. Moreover, the song popularity within a genre also follows a Zipf distribution. Due to lack of space, we do not present the corresponding graphs. The interested user should refer to the long version of this paper [3].

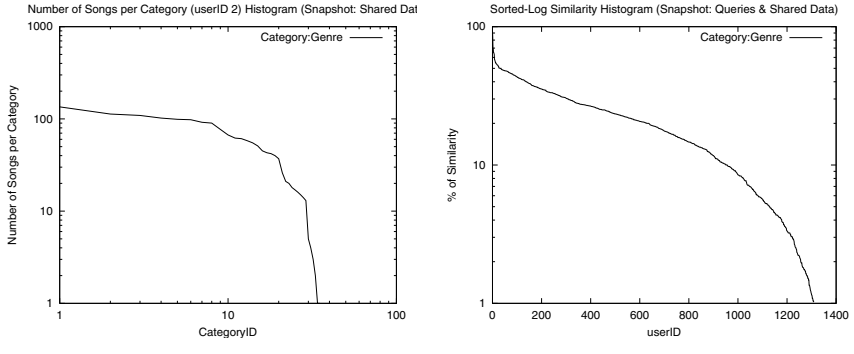
Figure 4(a) shows the number of songs per category. Interestingly, here the distribution does not follow Zipf's law, since many categories have a lot of songs while many others have only a few. In the next graph (Figure 4(b)) we investigate whether the queries asked by users are similar to the contents of their libraries. For instance, we want to know whether a user who owns mostly rock songs is likely to search for another rock song. To verify this, we first generated a histogram for each user's library based on the songs' genre. Then, we evaluated all the queries of each user against our entire song dataset and generated a histogram based on the genre that included all the qualifying songs. Finally, for each user, we calculated the overlap between the histogram of his library and the histogram of his queries. The graph shows that for many users their queries exhibit substantial similarity with their libraries. This fact could be exploited by an algorithm to generate an enhanced network topology based on the users' interests as reflected by their shared libraries. Other groupings are are possible (e.g., a query about a rock ballad is compatible with pop songs). Such an in-depth analysis is outside the scope of this paper.



(a) Number of queries per user

(b) Cumulative % of answers found in other users for an Average User

Fig. 3. Distribution of queries and answers



(a) Songs per category (genre)

(b) Similarity between queries and library contents

Fig. 4. Statistics for the music files

4 Case Study

As a case study, in this section we evaluate two recently proposed methods, namely the Dynamic Reconfiguration [4] and the Interest-based locality [12]. Both attempt to minimize the network traffic by identifying nodes which are beneficial in terms of content.

The intuition behind *Dynamic Reconfiguration* [4] is that there are groups of users in the network that share common interests. The method attempts to identify groups of compatible nodes and dynamically reconfigure the network to bring such nodes close to each other; thus, consequent queries will be answered with fewer hops.

When a node initiates a query, multiple peers may reply and statistics are gathered for all of them. All search results are not equally beneficial. A user will prefer to download a song from a node with high bandwidth. Moreover, the larger the results list, the lesser its significance for the reconfiguration process, since it cannot differentiate the compatible from the incompatible peers.

Based on these observations, the network reconfiguration process is implemented as follows. (i) Each obtained result accounts for a benefit of c/TRN, where c is the bandwidth of the answering link and TRN is the total number of results. Notice that the Gnutella Ping-Pong protocol, which performs exploration, specifies that information concerning bandwidth capacity is propagated together with the query reply. (ii) Periodically, each node checks the cumulative benefit of all nodes for which it keeps statistics, and includes in the new neighborhood the most beneficial ones. (iii) When a new node needs to be added, an invitation message is sent. (iv) The invited node always accepts an invitation evicting the least beneficial neighbor if necessary. (v) Neighbor log-offs trigger the update process. Note that in order to avoid frequent reconfigurations, when a node is evicted it does not attempt to replace the evicting neighbor immediately. Such a node will obtain a new neighbor if: (a) it receives an invitation from another node or, (b) reaches a reorganization threshold. In Ref. [4] the Dynamic Reconfiguration method is shown to be around 50% better than Gnutella in terms of message overhead, for a synthetic dataset.

Interest-based locality [12] is trying to improve the scalability of Gnutella-type search protocols, by introducing the concept of interest-based shortcuts. Shortcut lists are maintained at each node inside the network, which contain information (e.g., IP addresses) about other nodes that have answered a query in the past. Assuming that P2P users exhibit interest similarities, these nodes might be able to answer subsequent queries from the same user. The basic idea is to create a new overlay structure on top of the existing P2P network (based on these lists), and perform the content search in two steps. The nodes in the shortcut list are queried first (one by one, starting from the most beneficial node) until the requested file is found. If the search is not successful, the underlying P2P network is utilized by employing the standard flooding-based protocol.

In the basic algorithm, shortcuts are discovered through the Gnutella-type flooding protocol. Anytime a query is not resolved via the shortcut list, new candidate nodes are discovered following the flooding process. Then, a new node is selected and added to the shortcut list, possibly replacing a less beneficial shortcut. The size of the list is limited to ten entries, while its content is continuously updated due to the dynamic nature of the network (i.e., nodes entering or leaving the network). The importance of each shortcut (which also reflects its position in the sorted list) is determined by its success rate, i.e., the percentage of requests that it was able to answer successfully. Several enhancements to the basic algorithm were evaluated in Ref. [12] but the performance gain was relatively small compared to the increased complexity.

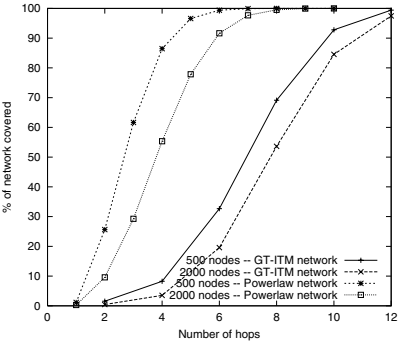


Fig. 5. Number of hops to cover the network

4.1 Experimental Setup

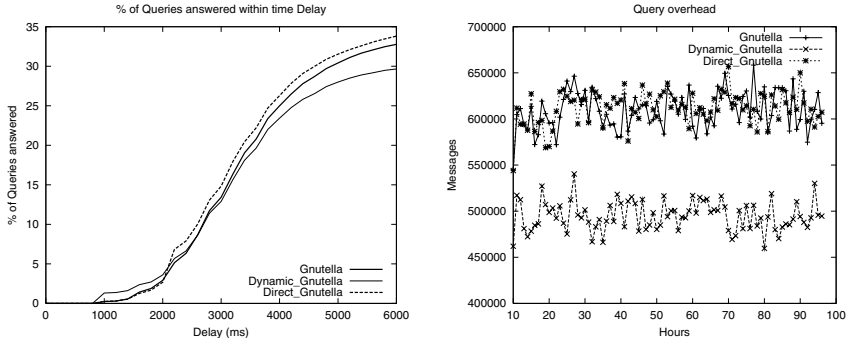
We developed a discrete event simulator in order to measure the query response time in addition to the number of messages which are exchanged. We split the users into three categories, according to their connection bandwidth; each user may be connected through a 56K modem, a cable modem or a LAN. The mean value of the one-way delay between two users is governed by the slowest user, and is equal to 300ms, 150ms and 70ms, respectively. The standard deviation is set to 20ms for all cases, and values are restricted in the interval $\mu \pm 3\sigma$. We experimented with various query rates. When the query rate is too high, the nodes are overloaded and all methods suffer. In the graphs we present here the query rate is slow enough to avoid this problem.

We used two network topologies for our experiments: (i) power-law [6] networks comprising of 500 and 2000 nodes, where the average number of neighbors per node was set to 3.2, and (ii) stub networks with 500 and 2000 nodes, produced with the GT-ITM [5] generator. In Figure 5 we show the percentage of the nodes that can be reached within 1 to 12 hops for each of the network topologies. Notice, that we did not keep the client population constant within the duration of each experiment. Instead, we properly set the arrival and departure rate of nodes in the system, in order to maintain the desired average population size.

In what follows, we compare the normal Gnutella protocol (denoted as *Gnutella* in the graphs) with Dynamic Reconfiguration (denoted as *Dynamic-Gnutella*) and the Interest-based Locality method (denoted as *Direct-Gnutella*).

4.2 Performance Evaluation

First, we consider a GT-ITM network with 2000 nodes. We measure the response time from the moment that a user submits a query, until the moment when the first result arrives. In the experiment of Figure 6(a) we allow the message to propagate for up to six hops and present the cumulative response time. The graph, for example, shows that after 6000 msec Dynamic-Gnutella was able to find answers for a little less than 30% of the submitted queries, while this percentage grows to almost 35% for Direct-Gnutella. The important observation



(a) % of queries answered vs. delay

(b) Query overhead

Fig. 6. GT-ITM network, 2000 nodes, 6 hops

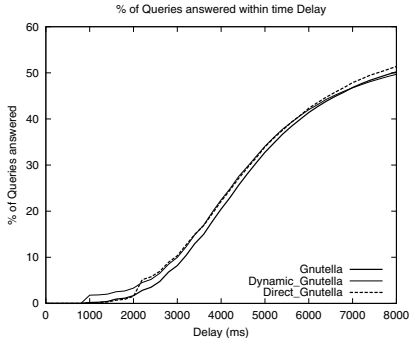
here is that the Dynamic method can be worse than plain Gnutella in terms of response time due to the reorganization overhead. Moreover, the performance improvements of Direct-Gnutella are not significant.

Figure 6(b) shows the number of messages transferred in the network per hour, for a simulated period of 100 hours. The Dynamic method needs less messages because a node does not propagate the query further as soon as a result is found. Because of the reconfiguration process, compatible nodes are gathered closer so query propagation stops earlier. The Direct method, however, needs to perform a Gnutella-style search if the results are not found by following the shortcuts. Since this is usually the case, it needs as many messages as the plain Gnutella protocol.

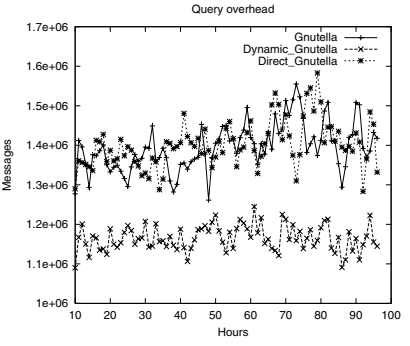
Figure 7(a) and Figure 7(b) present the respective results when the number of hops is increased to eight. Since all methods can reach more peers, the absolute number of the answered queries increases. However, the differences among the algorithms diminish, since the additional results are further away.

Figures 8(a) and 8(b) present similar results for a power-law network of 500 nodes. Here, we allow a smaller number of hops, because the connectivity of the network is much higher (see Figure 5). The results are similar to the previous ones. The only difference is that the number of transmitted messages is almost the same for all methods. This is due to the higher connectivity of the network: no matter how the network is reconfigured it is very likely that a query will reach a highly connected peer which will generate many messages for the Dynamic method as well as for Gnutella.

The conclusion from the above experiments is that the Dynamic and Direct variations of Gnutella can outperform the naïve protocol if the connectivity of the network is low and the allowed number of hops is limited. Then, the advanced methods can reach directly parts of the network which would take Gnutella several hops. On the other hand, if the network is well connected (e.g., power-law)

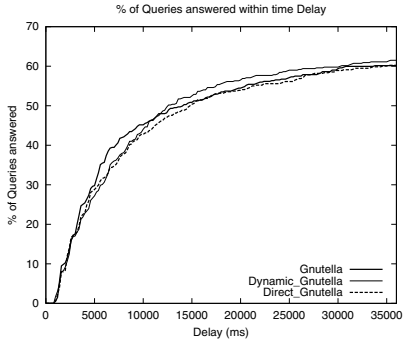


(a) % of queries answered vs. delay

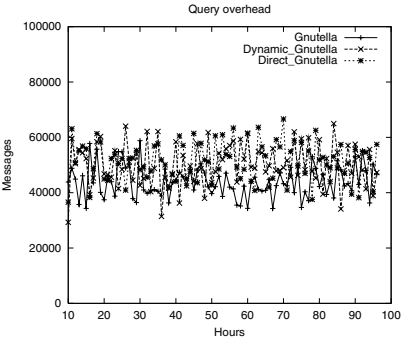


(b) Query overhead

Fig. 7. GT-ITM network, 2000 nodes, 8 hops



(a) % of queries answered vs. delay



(b) Query overhead

Fig. 8. Power-law network, 500 nodes, 4 hops

the performance difference diminishes since Gnutella can reach remote nodes easily.

The inherent drawbacks of the advanced methods are the assumptions that (i) during its online period each peer initiates enough queries to locate beneficial nodes, (ii) subsequent queries are relevant to the previous ones, and (iii) there is similarity among the contents of each peer. Our dataset reveals that in practice these conditions are unlikely to be met, therefore the performance of the advanced methods is not impressive. In particular, the first assumption seems to be the major factor behind these results. A peer that does not ask many queries will not able to discover many beneficial nodes. Furthermore, even when some beneficial nodes are discovered, there is no guarantee that they will stay on-line for a long period of time. Regarding assumptions (ii) and (iii), our dataset shows

some degree of similarity both among peer libraries and among the content of a peer's library and the queries that this peer asks. However, this behavior was limited to only a fraction of the total population.

5 Conclusions

In this paper we presented the characteristics of a large real dataset collected from the peers in the Gnutella network. We believe that this dataset will benefit all researchers in the P2P area because (i) it can become a standard benchmark to test various algorithms, and (ii) it provides realistic results since it is the only one to include not only queries but also the exact index of the peers' libraries. Initial analysis of the dataset revealed that real systems exhibit interesting characteristics that can be used to improve searching in P2P networks. For instance, we showed that in the music sharing domain, many users search for songs similar to their own libraries. Moreover, we used the dataset to evaluate existing P2P systems which attempt to identify beneficial peers. We found that in practice these systems may not perform as well as expected.

Acknowledgments

We would like to thank Yip Jun Kwan (Elton) for the implementation of the Gnutella probe and the collection of the data.

References

1. Gnutella home page. http://gnutella.wego.com.
2. Napster home page. http://www.napster.com.
3. Real dataset for file-sharing p2p systems. http://www.comp.nus.edu.sg/∼p2p.
4. S. Bakiras, P. Kalnis, T. Loukopoulos, and W. S. Ng. A general framework for searching in distributed data repositories. In *Proc. IEEE IPDPS*, pages 34–41, 2003.
5. K. Calvert, M. Doar, and E. W. Zegura. Modeling internet topology. *IEEE Communications Magazine*, 35:160–163, June 1997.
6. M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *Proc. ACM SIGCOMM*, pages 251–262, 1999.
7. K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proc. ACM SOSP*, pages 314–329, 2003.
8. Limewire Home Page. *http://www.limewire.com/*.
9. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proc. ACM SIGCOMM*, pages 161–172, 2001.
10. S. Saroiu, K. P. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proc. Multimedia Computing and Networking*, 2002.
11. S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. In *Proc. Internet Measurement Workshop (IMW)*, pages 137–150, 2002.

12. K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *Proc. IEEE INFOCOM*, pages 2166–2176, 2003.
13. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. ACM SIGCOMM*, pages 149–160, 2001.
14. B. Yang and H. Garcia-Molina. Efficient search in peer-to-peer networks. In *Proc. IEEE ICDCS*, pages 5–14, 2002.
15. B. Yang and H. Garcia-Molina. Designing a super-peer network. In *Proc. ICDE*, pages 49–60, 2003.