

IR-Wire: A Research Tool for P2P Information Retrieval

Shefali Sharma, Linh Thai Nguyen, Dongmei Jia

Illinois Institute of Technology

Chicago, IL 60616, USA

{sharshe, nguylin, jiadong}@iit.edu

ABSTRACT

We introduce a system for information retrieval research in the peer-to-peer file-sharing domain. Our system, IR-Wire, is based on the popular Gnutella protocol, giving us access to a large user base and a large data set. As a search tool, IR-Wire maintains many statistics and implements a number of information retrieval ranking functions. As a research tool, our main focus, IR-Wire contains a data logger and analyzer. The data logger logs both incoming and outgoing queries and query results and provides a way to create a snapshot of the entire data set shared by the users. The data analyzer provides a simple user interface for data analysis. We briefly discuss an analysis conducted on a million incoming queries that were collected in our log files.

Categories and Subject Descriptors

H.3.5 [Information Storage and Retrieval]: Online Information Services – *Web-based services*.

General Terms

Algorithms, Design, Performance.

Keywords

IR-Wire, information retrieval, open source, query log analysis.

1. INTRODUCTION

Peer-to-peer (P2P) technology [6] is very popular, with file sharing being a leading P2P application. With the demise of Napster in July 2001 both semi-centralized systems like FastTrack and completely decentralized P2P file sharing systems like Gnutella gained popularity. Gnutella is the third-most-popular file sharing network in the Internet. It is thought to host on average approximately 2.2 million users daily, although around 400,000 to 500,000 are online at any given moment [8]. Because of the immense popularity of these networks, it is imperative to find ways to improve their performance. Research tools for easy experimentation are needed that will help study both the data flowing through these networks and the behavior and preferences of their users. Although such tools exist for centralized systems (Section 2), we know of no readily available P2P counterparts.

We develop a system built on top of LimeWire's Gnutella system [9]. LimeWire is an open source program written in Java. It allows users to share any type of file and runs on Windows, Macintosh, Linux, Sun, and other computing platforms. Specifically, to the LimeWire client we have added the following functionalities:

- Statistics maintenance – Each client collects statistics about shared files.
- Information retrieval-style ranking functions – Each client implements a number of ranking functions.
- Data Logger – Each client can log incoming data, and outgoing data with results.
- Data Analyzer – An independent component that analyzes logged data.

2. RELATED WORK

Much research on peer-to-peer systems focus on how to model the peer-to-peer network [16-19]. Most of them deal with modeling the lower levels of peer-to-peer network, such as modeling peer-to-peer protocols, modeling the way files are replicated over the network, or modeling the network topologies. Our work does not focus on the lower level of the network, but the application level of a peer-to-peer information retrieval system.

There are a number of peer-to-peer information retrieval projects that have been developed, such as Peerware [12], Anthill [20], Alvis [21], and Nutch [22], to name a few. Most of these projects focus on query routing [12], resource discovery [20], or the design structure of a peer-to-peer information retrieval system [21]. Most of them implement search capabilities, but have less of a focus on data logging and analysis.

Most of current works on query log analysis are done for Web search engines. Silverstein et al. [1] analyzes a very large collection of queries logged by AltaVista for a period of six weeks, it contains almost one billion queries. Beitzel et al. [2] analyzed the changes of queries in terms of query popularity and uniqueness over time.

Other works are similar, in that they also reported statistics about query length, query distribution, the most popular queries and the most popular terms [3-5]. Some works also reported user behavior like the length of the query entered or how often the initial queries are modified.

To our knowledge, little work has been done on query logging as well as query log analysis for peer-to-peer systems even though researchers in the field have agreed that one is required [23]. The reason may be that it is assumed that queries in peer-to-peer systems are similar to queries in Web search engines, thus observations from Web search engines query logs can also be used to model peer-to-peer systems.

Zeinalipour-Yazdi and Folias logged query messages and analyzed them for Gnutella network in 2002 [14]. Their work is most similar to our work. To collect user queries, they have deployed

their Peerware on 17 workstations to collect all pass-through-messages. In a period of 5 hours, they have collected about 15 million query messages. The limitation of their approach is that, not all query messages are forwarded to their peers. Most likely, the query messages they have received are similar to what they were sharing. Thus their query collection may not be representative of all queries flowing through the network. In developing IR-Wire we have modified our routing tables to capture all queries flowing through the network. Therefore, we receive a complete set of all user queries in a given time period. In addition, Zeinalipour-Yazti and Foliás collected peer-to-peer queries in June 2002. Therefore, their query collection may be out-dated, since what the users were looking for 4 years ago may be totally different from what they are looking for today.

IR-Wire helps build an unbiased query collection. It focuses on IR research in a large-scale Gnutella-based peer-to-peer, file-sharing application allowing users to collect more up-to-date queries and providing an easy interface for doing analysis on the data set.

3. BRIEF DESCRIPTION OF GNUTELLA

We choose to base our work on the Gnutella protocol because it is not just popular but also well studied. In the Gnutella network, a user searches for a file by issuing a keyword query. This query is sent to all the clients it is actively connected to (this number is usually small, on the order of 10). The client further forwards this query to all of its neighbors in the Gnutella network. This process repeats until the query's Time-To-Live (TTL) expires or the packet has reached a client that is a predetermined number of "hops" away from the sender [7].

Gnutella specifies an unstructured and highly distributed network where each node is fully autonomous, independently controlling its local repository of shared files. Incoming queries are compared against the files descriptors (since these shared files are binary files, they need external descriptors). File descriptors are generally implemented via file names. Matching results are returned to the peer that issued the query.

4. SYSTEM DESCRIPTION

The goals of IR-Wire are twofold: a search tool and a research tool. The overall architecture of our system is now described along with an explanation of the functionality of each component.

4.1 IR-Wire System Architecture

The architecture of the system is depicted in Figure 1. IR-Wire's logging functionalities can be turned off if basic LimeWire functionalities are desired. The Data Analyzer is separate from the "main" LimeWire system. Loading the logged data into a MySQL database is an optional, separate batch process.

4.2 System Components

Our system consists of the following modules: *IR+* for improving search; the LimeWire system with few modifications; Data Logger, Loader and Analyzer modules.

The modifications made to the core LimeWire system is minimal, which should simplify the incorporation of our enhancements into future versions of LimeWire.

4.2.1 *IR+*

This module uses the meta-data associated with the file and utilizes IR techniques to rank results, disambiguating them,

thereby improving the quality of the results for a particular search. It implements the following additional ranking functions [10]:

- **Term Frequency:** The result whose descriptor contains the most query terms is ranked highest.
- **Fraction:** The result whose descriptor covers the highest fraction of query terms is ranked highest.
- **Cosine Similarity:** A result is ranked based on the cosine similarity score between the query and its descriptor.
- **TF/IDF:** This is similar to cosine similarity, but with each term weighted based on document frequency. This ranking function requires an estimation of the global document frequency of each term computed using the local shared repository.
- **Group size:** Results that refer to the same file are grouped and the rank score is the size of the group. This ranking function is implemented in most P2P file-sharing systems, including LimeWire.

This list of ranking functions is not exclusive. The *IR+* component is designed such that the implementation of additional ranking functions is straightforward.

4.2.2 *LimeWire System-Query Logging*

The original Gnutella protocol floods queries in the network, limiting its scalability [7][25]. To deal with the scalability concern, *ultrapeers* [19] are used. An ultrapeer is a peer that is judged to be highly reliable and capable of handling more Gnutella workload. The set of ultrapeers makes possible a two-tier system, with ultrapeers and *leaf* (regular) nodes. Ultrapeers flood queries to each other, and leaves connect only to ultrapeers (not to other leaves). Ultrapeers can be seen as "proxies" for leaf nodes in the Gnutella network.

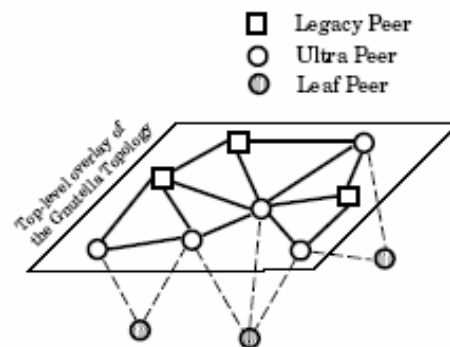


Figure 1. Two-Tier Topology of Modern Gnutella

Leaf nodes interact with ultrapeers to control the amount of incoming queries by the Query Routing Protocol (QRP) [11]. The QRP specifies that nodes create query routing tables by hashing keywords of all the file descriptors that they share and storing these hash values in a bit-vector (a form of the Bloom Filter [11]). By exchanging these routing tables with neighbors, nodes know what queries their neighbors can match and make routing decisions appropriately. Leaf nodes create these routing tables, and send them to their ultrapeers. Ultrapeers, thus, route only a subset of the incoming queries, the queries that are likely being answered, to their leaves.

Specifically, queries are conjunctive – a query matches a descriptor if all query terms are contained in the descriptor. A bit-vector (i.e., routing table) with more bits set, therefore, receives more queries. In practice, bits are set as files are added to the shared local repository. Clients with few shared files, therefore, have very sparse bit-vectors.

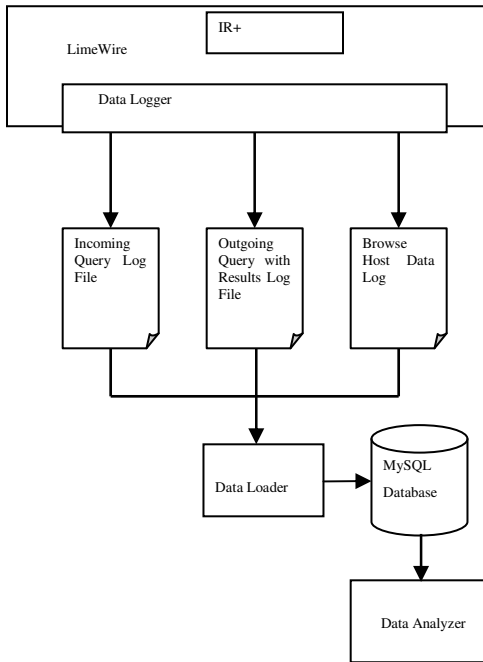


Figure 2. Architectural Design of IR-Wire

Our goal is to log all queries that are being issued in the Gnutella network to get an accurate picture of what users of peer-to-peer file sharing systems are searching for. One way to do this is to share as many files as possible. This option is not realistic, given the size of the bit-vector (64KB [11]). Another option is to make our LimeWire node an ultra-peer. This option also, is not practical, as it requires either maintaining a high bandwidth Gnutella node for a long period of time, thereby being elected as an ultrapeer, or modifying the LimeWire software to allow a client to masquerade as an ultrapeer. The former is difficult because it requires long-term dedicated resources, without any guarantees that the client will become an ultrapeer. The latter, masquerading as an ultrapeer, is possible, but introduces instability into the network if our ultrapeer frequently joins and leaves.

To get all incoming queries for our analysis, we choose an easier option. We set all the bits in the bit-vector to falsely claim that we are sharing files that contain all the keywords the users are searching for. This causes the ultrapeer’s bit-vector to be set with all ones as well, causing all the network traffic to be routed to that ultrapeer, and then, of course, to us. This may cause some

flooding in the network but is necessary in yielding an unbiased data set.

4.2.3 Data Logger and Analysis Component

This component is responsible for logging and analysis of all the different types of queries with the results obtained from the search. It contains the following sub-components:

- The log files - IR-Wire maintains three types of log files. Other than storing the incoming queries in the incoming query log file, IR-Wire also allows the user to log outgoing queries as well as the query’s results in another log file. Work is also in progress to facilitate storing the data retrieved from browsing of peers’ shared repositories. LimeWire allows users to view the contents of a peer through the “browse host” facility. IR-Wire will use this facility to gather data on a host, and then use this data to perform other searches and other browse host actions. It will thus create a snapshot of the data shared by all the hosts.
- Data Logger - The Data Logger reads from and writes to the log files. Data Logger functions are called from within the LimeWire code to log data to the appropriate log files or view the log through the LimeWire interface.
- Data Loader – The Data Loader reads the log files and stores the data in the corresponding MySQL tables. It has no interaction with the LimeWire code. Loading is done as a batch process if the user wants to use the tables.
- Data Analyzer – The Data Analyzer allows the user to perform various analyses through a simple user interface where selections can be made. The interface is under construction. The user will be able to either write SQL queries to the database and save the results retrieved or select from the analysis already implemented. The analyses that have already been implemented are for the incoming query messages¹. These include retrieving average query length, query length distribution, queries popularity distribution, query categorization by the type of file (audio/video/program) desired by the user, and correlation analysis (e.g., those performed in [1]). We have discussed some results of these analyses in Section 5. This module also does not interact with the LimeWire code. Analyses on the data are performed on the log files or querying is done on the tables depending on the user preference.

4.3 Limitations to our Logging Capabilities

Our current version of IR-Wire is limited in its data logging capabilities due to the specifics of the Gnutella protocol. These specifics do not allow us to collect specific “session data” on users, which prevents session analysis (e.g., how the user modifies his/her queries to improve the results received),

Gnutella query routing is done on a hop-to-hop basis. Queries are routed from a peer to its neighbors, who are not given information on where the peer got the query from in the first place. Consequently, query replies (which do contain the IP address of the replying node) must follow a reverse path to arrive at the query originator.

¹ These are the incoming query requests or simply query messages. The standard Gnutella protocol contains five types of messages: ping (discover hosts on network), pong (reply to ping), query (search for file), query hit (reply to query), push (download requested for firewalled servents).

Reverse path routing is likely a security feature [24], increasing the anonymity of users. We are currently seeking holes in this security mechanism that may allow us to track users.

Another limitation is that we were not able to get the time the query was sent by the user. This and the fact that we cannot identify the originators of queries prevent us from doing session analysis which would help us to study user behavior.

5. QUERY LOG ANALYSIS

We analyze the attributes of the incoming query log and report our findings. The analysis here is cursory, and meant only to demonstrate what kind of analyses are possible with the data collected by IR-Wire.

5.1 LimeWire Search Engine

LimeWire allows users to share files of any type, like .mp3, .avis, .jpgs, .tiffs etc. It is capable of multiple simultaneous searches, available in several different languages. Also, it allows users to make many different types of searches. Other than the normal keyword based queries, users can also search for the shared data on a given IP address (the “browse host” facility). Even for the keyword based queries the user can query using just the query string or he/she can specify other attributes like genre, album, or artist. This type of query is called a “rich query.” In the keyword based search the user can have either an un-constrained search which searches for files of any type or a constrained search for a specific type of file like audio, video, program, etc.

In our analysis, we have restricted ourselves to keyword based queries that are not rich queries. We are also restricting ourselves to only English queries (LimeWire allows queries to be made in many different languages). As we will show, these restrictions should not significantly hurt the generality of our results, as most queries are keyword-based, English queries.

5.2 IR-Wire Incoming Query Log

We kept our IR-Wire running the whole day to collect incoming queries on Saturday, May 13th, 2006. Over the weekend, there is less business traffic and more recreational traffic on the network, compared with the case of weekdays. Since Gnutella is arguably used primarily for recreational purposes, we should be able to yield a reasonably representative set of queries on the weekend. We collected the data in 20 log files of 100MB each. Each line in the log file is a query request. The following attributes are stored for each query: The original query string, the time when the query was received, values to indicate if the search was constrained or un-constrained and type of file the user is searching for (e.g., audio, video, program, document, image).

We preprocess all the query strings by ignoring any case differences, removing stop words, replacing any punctuation with white space, and compressing white space to single spaces. Furthermore, we observed that some queries include unrecognizable non-ASCII strings. We believe that most of them are non-English queries, which are removed from the analysis. This constituted about 25% of the query log.

After preprocessing, totally there are 775,605 queries and 65% (498,130) of them are distinct queries without considering replicas². Note that we ignore the capital letter in our analyses.

² The collected query logs are available upon request.

5.3 Initial Findings

As shown in Table 1, the average number of terms per query is 2.94, which validates that queries tend to be short in general. Similar average query length is reported in [5], which indicates that a short query trend exists in both peer-to-peer file-sharing systems and centralized Web search engines.

Table 1. Statistics on Query Length and Query Frequency

	Query Length	Query Frequency
Max	12	623
Min	1	1
Average	2.94	1.56
Std Dev	1.27	2.21

Besides that, we report the query length distribution in Figure 4. Only unique queries are considered in this case. Most of the queries contain 1 to 6 query terms. There is only one query that has 10, 11 or 12 terms and 3 queries which have length 9. 75% of queries contain 2 to 4 keywords.

In Figure 5, we compare the frequencies of unique queries with their frequency ranks. The query ranked 1st is the most popular (frequent) query and next one is the 2nd most popular query and so on. Queries that have exactly the same query terms are treated as replicas of one query, even if clients type in the query terms in a different order. We observe that query popularity follows a Zipf-like distribution. In contrast to the fact that the frequency of the most popular query is 623, about 75% of all the queries occur only once in the whole day, which indicates that the interests of Gnutella users are quite diverse or users tend to describe their needs differently. Later we show what exactly those top ranked queries are.

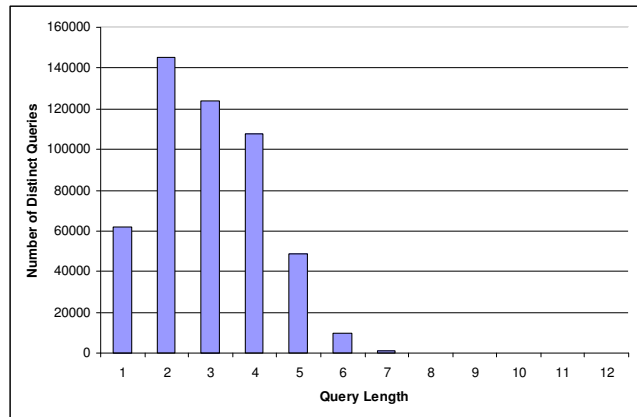


Figure 4. Query Length Distribution

We categorized all the logged queries based on the desired types included in a query request message. As shown in Figure 6, a large portion of queries does not have constraints on the desired

file types, which is represented by the legend “all”. In the cases that one or multiple types are defined by clients along with keywords, more than 70% of the queries are for audio files, 20% are for video and the other 10% queries are for images, programs and documents. This is the same as we expected, because most clients in a P2P file-sharing system tend to search for songs or movies on the purpose of entertainment, other than documents or programs.

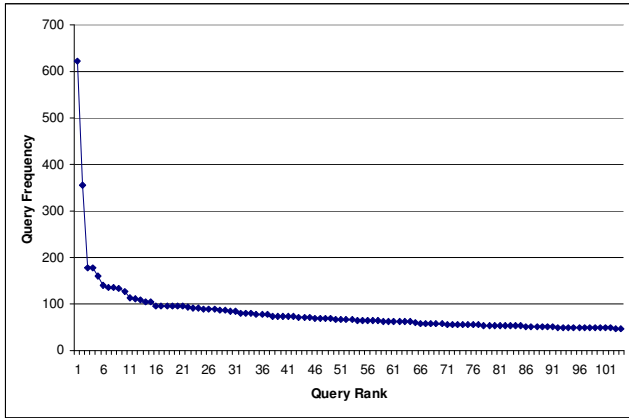


Figure 5. Query Popularity Distribution

In Table 2, we report exactly the 25 most popular queries with their frequencies and the top 25 keywords with the number of queries in which each keyword is contained. Surprisingly, the query “neonode” is the most popular query. As far as we know, Neonode is a Swedish manufacturer of mobile phones [15]. Other mysterious queries occur as well, such as “pdmckaziejdnbt.” Because a search for “neonode” using IR-Wire returns 0 byte files, we are guessing that neonode may be a “control query,” used for tracking network connectivity.

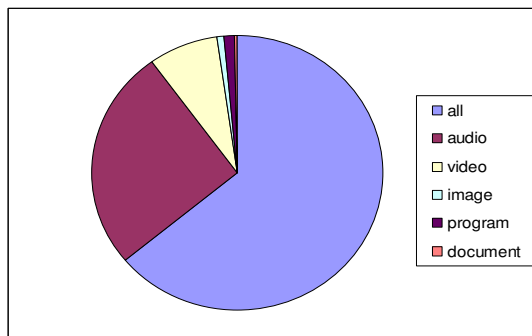


Figure 6. Query Categorization by Types of Desired Files

One interesting observation is that a couple of the top 20 queries, such as “poseidon,” “mission impossible,” “da vinci code” and so on, are newly released movies. So they are in high demand and have been searched frequently by Gnutella clients during the time period of our testing. It indicates that query popularity in P2P file-sharing systems is closely related to the temporal aspect. Hence, it is not necessary that a popular query in the current time period is

popular consistently over time. This can be examined in greater detail by measuring the query frequency distribution and recording the most popular queries periodically at various weeks or even months, which is left as our future work.

We also notice that some file extensions, avi, mp3 and dvd, are among the most popular query terms. The reason we believe is that for an original query containing a keyword such as song.mp3, it becomes two terms song and mp3 after replacing all the punctuations by white space which is used to detect and separate two adjacent keywords in the preprocessing phase. Due to the large amount of queries for music or movies, these terms occurs much more times and are counted as popular query terms.

Table 2. Top 25 Most Popular Queries and Terms (After Preprocessing)

Top 25 queries	Freq	Top 25 terms	Num Queries Contained
Neonode	623	love	6235
Love	355	dj	5172
Pdmckaziejdnbt	178	los	4926
Poseidon	160	live	4790
Pthc	139	feat	3822
Time	135	avi	3737
smallville vessel	135	ft	3668
bibcam	134	big	3627
ptsc	126	remix	3486
lsbar 001a kdquality	114	girl	3289
divx ita men dvd rip	112	im	3255
mission impossible	108	sex	3253
99bb	105	mp3	3135
lco	96	star	3090
istock	95	pthc	3088
civilization 4 iv	95	john	2921
kokeshi	95	te	2873
afford www buylegalmp3	95	time	2860
istock l2	95	en	2821
mihimaru gt	93	xvid	2802
eureka	92	day	2700
da vinci code tom hamks	91	full	2661
fm2006	90	dvd	2652
lsh	89	man	2618
template zip	88	life	2611

6. CONCLUSION AND FUTURE PLANS

In this paper we introduced a P2P file-sharing system called IR-Wire built on top of Limewire’s Gnutella system to monitor and

collect user queries, in order to reveal the real-world of queries in Gnutella network. This work is meant to address a need for research tools and data for P2P IR, expressed in [23]. We also presented our observations and analysis of almost 1 million incoming queries collected in the log. The statistical results show that queries tend to be short. 75% of queries contain 2 to 4 keywords. Query popularity follows a Zipf-like distribution and the majority of queries are for music and movies. A lot of query terms are closely related to temporal aspect, so their popularity may shift dramatically over time.

Various other analyses will be implemented for the incoming queries, new analyses will be done on outgoing queries and the results returned. Among other things in the pipeline, there is a logging and analysis of the “browse host” queries which will provide a good insight into the data users are sharing.

For the incoming queries, we plan to use a data mining technique (association rule mining) to reveal the correlation among query terms, the correlation among attribute-values of the rich queries, as well as the correlation between a query term and an attribute-value. This information, if available, is very helpful, since it can be used to optimize the query routing protocol of peer-to-peer systems.

We also plan to collect the results returned by the system for each user query. One approach is to use the logged incoming queries as our node’s outgoing queries and record all returned results. The use of automatically generated queries is also possible. The set of returned results for each query may give us some knowledge about how efficient the search in peer-to-peer network is.

We also want to investigate whether there is a correlation between the shared collection of a user and the set of queries s/he issued. Our conjecture is that, users are normally looking for files that are similar to those they are sharing.

The code and the data collected will be available for download from the IIT IR website (<http://www.ir.iit.edu>), or by request.

7. ACKNOWLEDGMENTS

We would like to thank the reviewers of this paper for their very helpful comments.

8. REFERENCES

- [1] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. *Analysis of a Very Large Web Search Engine Query Log*. SIGIR Forum, 33(1):6--12, 1999.
- [2] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. *Hourly Analysis of a Very Large Topically Categorized Web Query Log*. SIGIR’04, 321-328, 2004.
- [3] B. J. Jansen, A. Spink, and J. Pedersen. *An Analysis of Multimedia Searching on AltaVista*. MIR’03, 186-192, 2003.
- [4] A. Broder. *A Taxonomy of Web Search*. SIGIR Forum, 36(2), Fall, 2002.
- [5] A. Spink, S. Ozmutlu, H. C. Ozmutlu, and B. J. Jansen. *U.S. Versus European Web Searching Trends*. SIGIR Forum 36(2), 32-38, 2002.
- [6] D. Milojevic, V. Kalogeraki, R. Lukose, K. Nagaraja1, J. Pruyne, B. Richard, S. Rollins, Z. Xu. *Peer-to-Peer Computing*. HP Laboratories Palo Alto, HPL-2002-57 (R.1), July 3rd, 2003
- [7] LimeWire Technical Document. http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf
- [8] <http://www.slyck.com/news.php?story=814>
- [9] LimeWire home page. <http://www.limewire.com>
- [10] W.G. Yee, O. Frieder. The Design of PIRS, a Peer-to-peer Information Retrieval System, DBISP2P 2004 Workshop
- [11] <http://rfc-gnutella.sourceforge.net/src/qrp.html>
- [12] *Peerware, A Real P2P Information Retrieval Testbed*. Web document, <http://www.cs.ucr.edu/~csyiazti/peerware.html>
- [13] LimeWire Technical Document. http://www.limewire.com/developer/Ultrapeers.html#_ftn1
- [14] D.Zeinlipour-Yazti, and T. Foliass. *A Quantitative Analysis of the Gnutella Network Traffic*. TR-CS-89, Dept. of Computer Science, Univ. of California, Riverside, 2002
- [15] <http://en.wikipedia.org/wiki/Neonode>
- [16] L. Zou, and M. H.Ammar. *A File-Centric Model for Peer-to-Peer File Sharing Systems*. Proc of the 11th IEEE Intl. Conf. on Network Protocols (ICNP’03), 2003.
- [17] S. Merugu, S. Srinivasan, E. Zegura. *P-sim: A Simulator for Peer-to-Peer Networks*. Proc. of the 11th IEEE Intl. Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS’03), 2003.
- [18] W. Yang, N. Abu-Ghazaleh. *GPS: A General Peer-to-Peer Simulator and its Use for Modeling BitTorrent*. Proc of the 13th IEEE Intl. Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, (MASCOTS’05), 2005.
- [19] Q. He, M. Ammar, G. Riley, H. Raj, and R. Fujimoto, *Mapping Peer Behavior to Packet-level Details: A Framework for Packet-level Simulation of Peer-to-Peer Systems*. Proc. of the 11th IEEE Intl. Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS’03), 2003.
- [20] O. Babaoglu, H. Meling, and A. Montresor. *Anthill: A Framework for the Development of Agent-based Peer-to-Peer Systems*. Proc. of the 22nd Intl. Conf. on Distributed Computing Systems (ICDCS’02), July 2002.
- [21] K. Aberer, F. Klemm, M. Rajman, and J. Wu. *An Architecture for Peer-to-Peer Information Retrieval*. Proc. of the 7th Annual Intl. ACM SIGIR Conf. Wrkshp on Peer-to-Peer Information Retrieval, July 2004.
- [22] Nutch project home page. <http://lucene.apache.org/nutch/>
- [23] H. Nottelmann, K. Aberer, J. Callan, and W. Nejdl, CIKM 2005 P2PIR Workshop Report, 2005, <http://p2pir.is.informatik.uni-duisburg.de/2005/report.pdf>
- [24] D. Bickson, D. Malkhi, A Study of Privacy in File Sharing Networks.
- [25] J. Ritter, Why Gnutella Can’t Scale. No, Really., Web Document, February, 2001, www.darkridge.com/~jpr5/doc/gnutella.html.