Capturing Accurate Snapshots of the Gnutella Network

Daniel Stutzbach, Reza Rejaie
Department of Computer and Information Science
University of Oregon
{*agthorr,reza*}*@cs.uoregon.edu*

***Abstract*— A common approach for measurement-based characterization of peer-to-peer (P2P) systems is to capture overlay snapshots using a crawler. The accuracy of captured snapshots by P2P crawlers directly depends on both the crawling speed and the fraction of unreachable peers. This in turn affects the accuracy of the conducted characterization based on these captured snapshots. Prior studies frequently rely on crawling the network over an hour or more, during which time the overlay may change substantially. Moreover, none of the previous measurement-based studies on P2P systems have examined the accuracy of their captured snapshots or the impact on conducted characterization.**

In this paper, we present a fast P2P crawler called *Cruiser*, discuss aspects of its design, and evaluate its accuracy. Cruiser is able to crawl the million-node Gnutella network in around 7 minutes. Having these accurate snapshots enables us to quantify *(i)* the "relative error" in captured snapshot as a function of crawling speed, *(ii)* the tradeoff between the completeness and accuracy of captured snapshots, and *(iii)* the effect of snapshot accuracy on the correctness of characterizations of Gnutella.

I. Introduction

During recent years, the increasing popularity of peer-to-peer (P2P) networks has led to growing interest in measurement-based characterization of popular P2P systems (*e.g.*, [1, 2, 3]). These characterizations provide deeper insight into the behavior of P2P systems, essential for proper design and effective evaluation. These characterizations become more important as P2P systems rapidly increase in use over the Internet [4, 5]. A common technique to characterize a P2P overlay is to capture snapshots of the overlay topology with a crawler, which queries peers for a list of their neighbors, much the way a web spider retrieves web pages for a list of pointers to other web pages. These snapshots capture the overlay topology as a graph, with peers as vertices and connections as edges. Examining individual snapshots reveal various properties of the overlay (*e.g.*, the size and diameter of the overlay and node degree distribution), whereas the comparison of consecutive snapshots identifies dynamics of various properties as a function of time (*e.g.*, dynamics of peer participation and overlay structure). The accuracy of the conducted analysis based on the above methodology directly depends on the accuracy of the captured snapshots.

A perfect snapshot of a P2P overlay is captured if a crawl is complete and instantaneous. However, in practice neither of these conditions are met for the following reasons:

Rapidly Changing Topology: The goal of a crawler is to capture a perfect instantaneous snapshot of the overlay topology at an instant in time. However, crawlers contact participating peers in a progressive fashion. Therefore, capturing a snapshot may take a long time depending on the speed of the crawler and its available resources (*i.e.*, access link bandwidth and processing power). Given the moving nature of P2P systems, as the duration of the crawl increases, the captured snapshot becomes more *distorted* compared to the instantaneous ideal because more nodes arrive or depart during the crawl [6]. Moreover, in light of reported uptimes of just a few minutes [7, 8], crawls taking 30 to 120 minutes [9, 10] are likely to be highly distorted. Furthermore, the duration of the crawl determines the time between back-to-back snapshots and thus determines the granularity of captured dynamics. In other words, we cannot explore how the topology has changed over a period of 10 minutes if it takes an hour to capture a snapshot. Precisely quantifying the distortion in a captured snapshot is difficult because instantaneous reference snapshots are not available for comparison.

Unreachable Peers: Captured snapshots are often incomplete because a non-negligible portion of discovered peers are not directly reachable by the crawler. Previous studies often assume that unreachable peers have departed the system and simply exclude them from the captured snapshots. However, as we show in this paper, a majority of unreachable peers are either located behind a firewall (*i.e.*, NATed) or, more interestingly, are receiving too many SYN packets (*i.e.*, overloaded). Therefore, ignoring these peers could introduce a non-negligible error in the captured snapshot. Moreover, there is a fundamental tradeoff between the *completeness* and the *accuracy* of a captured snapshot by a crawler. A more persistent crawler can use longer timeouts when contacting nodes or retry failed attempts, increasing the number of successful contacts and the amount of data collected, but at the expense of increasing the crawl duration and thus the distortion in the capture compared to the ideal instantaneous snapshot.

We developed a fast and efficient Gnutella crawler, called *Cruiser*, that can capture the Gnutella network with one million peers in around 7 minutes using six off-the-shelf 1 GHz GNU/Linux boxes in our lab. Cruiser's crawling speed is about 140k peers/minute which is orders of magnitude faster than previously reported crawlers (*i.e.*, 2 hours for 30K peers (250/minute) in [10], and 2 minutes for 5K peer (2.5k/minute) in [2]).

Cruiser achieves this orders of magnitude increase in crawl speed as follows: *(i)* it leverages several features of modern Gnutella including its two-tier topology, efficient new handshake mechanism, and high degree of node connectivity among top-level peers, *(ii)* it substantially increases the degree of concurrency during the crawling process by deploying a master-slave architecture and allowing each slave crawler to contact hundreds of peers simultaneously. We present the high level architecture and various systems issues and performance bottlenecks for Cruiser.

We focus on the Gnutella network as a representative P2P system for several reasons. First, several indicators show that Gnutella has a large and growing population of active users generating a considerable fraction of Internet traffic. Over the past year, the number of active Gnutella users has more than tripled and is currently at around 1.3 million simultaneous peers [11], making it one of the largest P2P systems [4, 12]. Additionally, Gnutella lends itself more readily to study as it has several mature, open-source implementations and fully open protocol specifications. In future work, we plan to adapt Cruiser to other P2P systems. Finally, we believe that most of the raised issues and findings are relevant to other P2P systems, as the two-tier structure is used in several of the most popular systems (such as FastTrack and eDonkey).

Compared with previous crawlers, Cruiser can capture significantly more accurate snapshots of the Gnutella network. Having more accurate snapshots enables us to quantify *(i)* the "relative error" in captured snapshot as a function of crawling speed and the ratio of unreachable peers, *(ii)* the tradeoff between completeness and accuracy of captured snapshots, and *(iii)* the effect of snapshot accuracy on the correctness of characterizations of Gnutella.

II. Modern Gnutella

In this section, we briefly describe a few key features of modern Gnutella [13] that are used by Cruiser. The original Gnutella protocol has limited scalability due to its flat overlay. To address this limitation, modern Gnutella clients implement a two-tiered network structure by dividing peers into two groups: *ultrapeers* (or super-peers) and *leaf* peers. As shown in Figure 1, each ultrapeer neighbors with several other ultrapeers within a top-level overlay. The majority of the peers are leaves that are connected to the overlay through a few ultrapeers. Furthermore, modern Gnutella clients implement a mechanism that allows high-bandwidth, un-firewalled leaf peers to become ultrapeers in order to maintain a proper ultrapeer-to-leaf ratio in the overlay. Those peers that do not implement the ultrapeer feature can only reside in the top-level overlay and do not accept any leaves. We refer to these peers as *legacy* peers. We also refer to the legacy peers and ultrapeers collectively as the *top-level* peers. Our recent measurements [11] reveal that the degree of connectivity among top-level peers is much higher than that in the flat original Gnutella.

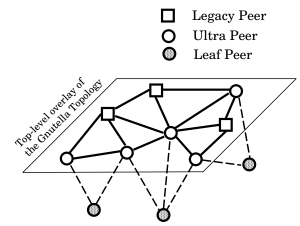

Fig. 1. Two-Tier Topology of Modern Gnutella

III. The Gnutella Cruiser

Our primary goal in the design of Cruiser is to minimize distortion in captured snapshots by maximizing the speed at which Cruiser explores the overlay topology. We deploy several techniques and features to achieve this design goal, as described below.

a) Handshaking: Modern Gnutella clients implement a special handshaking feature [14] designed to facilitate crawling; it allows a quick query to a peer for connectivity information. This allows Cruiser to learn the addresses of the peer's neighbors and, for an ultrapeer, the addresses of its leaves. Previous crawlers relied on another feature of the Gnutella protocol, namely Ping-Pong messages, to retrieve neighbor information. However, this technique was less efficient (requiring more round-trip times) and has no way to report information about the two-tier aspect of the topology.

b) Two-Tier Networks: Cruiser leverages the two-tier structure of the modern Gnutella network (Figure 1) by only crawling the top-level peers (*i.e.*, ultrapeers and legacy peers). Since each leaf must be connected to an ultrapeer, this approach enables us to capture all the nodes and links of the overlay by contacting a relatively small fraction of all peers. Furthermore, the high degree of peer connectivity within the top level overlay substantially increases the rate of discovery for new ultrapeers. Overall, this strategy leads to a major reduction (around 85%) in the duration of a crawl without any loss of information.

c) Distributed Architecture: Cruiser employs a master-slave architecture in order to achieve a high degree of concurrency and to effectively utilize available resources on multiple desktop PCs[1]. A master process coordinates among multiple slave processes that act as virtually

[1]Using a master-slave architecture also allows us to deploy Cruiser in a distributed fashion if Cruiser's access link becomes a bottleneck.

independent crawlers and crawl disjoint portions of the network in parallel. The slaves communicate with the master using loose synchronization. Each slave has an independent 2000-element queue of addresses to contact, which the master fills. The slaves report back with the data they've gathered. The master culls addresses from the data and uses this to fill the queues. The crawl terminates when all the queues are empty.

d) Asynchronous Communications: Each slave process crawls hundreds of peers in parallel using asynchronous communications. Cruiser implements an adaptive load management mechanism to ensure that slave processes remain busy but do not become overwhelmed. This is important for the steady progress of the crawl especially when different slave nodes have heterogeneous processing capabilities. Toward this end, each slave monitors its CPU load and adjusts its maximum number of parallels connections using an AIMD algorithm similar to TCP's congestion control mechanism. In practice, each PC typically runs with close to 1,000 parallel connections, contributing an additional speed-up of nearly three orders of magnitude.

Each slave maintains a parameter, call max_concur, that limits the maximum number of open connections, similar to a TCP congestion window. No new connections will be attempted if it would exceed that threshold. Because there is a high delay between opening connections and the increase in CPU load[2], max_concur should not be adjusted too frequently. To measure CPU load, each slave sets up a timer to fire every half-second. If this timer is more than 50% late, this is interpreted as a signal of high CPU load: max_concur is multiplicatively decreased to 90% of its value and is not changed until the timer is on time. When the timer is on time, max_concur is linearly increased by one. Similar to TCP Slow Start, max_concur is multiplicatively increased by 1.2 during the initial phase of crawling to quickly reach an appropriate value. These parameters were set empirically through experimentation with our own PCs. A similar adaptation mechanism should be incorporated to adjust the total number of parallel connections in order to avoid congestion on the access link of the crawler, though thus far this has not been a bottleneck for our systems.

We have experienced other system issues in the development of Cruiser that are worth mentioning. In particular, we needed to increase the limit on the number of open file descriptors on each Linux box. Otherwise, many connection attempts return immediately with an automatic "Connection Refused" error. In a similar vein, we increased the number of connections that our lab firewall could track to prevent the firewall from dropping packets due to this constraint.

[2]This delay is the time from when the crawler sends the first TCP SYN packet until the connection is established and data is returned by the peer, *i.e.*, at least two round-trip times.

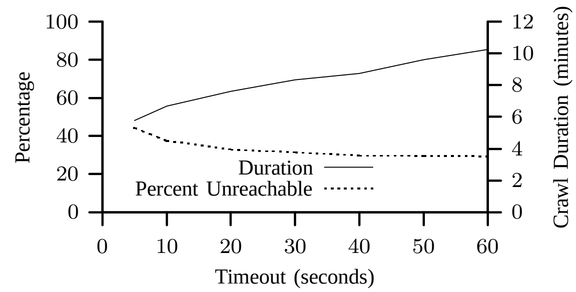

Fig. 2. Effects of the timeout length on crawl duration and snapshot completeness

e) Appropriate Timeouts: When peers are unresponsive, waiting for TCP to timeout and give up attempting to connect takes a long time. On our systems, a full TCP timeout to an unresponsive address takes more than 3 minutes. While this is suitable for many interactive and automated applications, we conducted an evaluation of the cost versus benefit of different timeout values for crawling. As a function of the timeout length, Figure 2 shows the duration of the crawl and the percentage of peers that were unreachable. We see that while very low timeouts (less than 10 seconds) result in a dramatic increase in the number of unreachable peers, there are diminishing returns for using longer timeout values, while the crawl length (and thus distortion) continues to increase. In other words, if a peer has not responded after 10 seconds, it is unlikely to ever respond. Therefore, we use a timeout of 10 seconds, providing an additional speedup of more than a factor of two.

A. Unreachable Peers

A non-negligible portion of discovered peers in any arbitrary crawl (30%-38%) are not reachable by a crawler. More specifically, TCP connections to these peers either timeout (15%-24%), are dropped (6%-10%), or are refused with a TCP reset (5%-7%) by the contacted peers. Since firewalls can exhibit timeout or refusal behaviors, there is no reliable test to distinguish between departed and firewalled peers. We also discovered that some of the unreachable peers are actually overwhelmed ultrapeers that sporadically accept TCP connections and can be contacted after several attempts. This transport-layer refusal means that the application is not able to call *accept()* sufficiently fast which leads to a TCP listen buffer overflow. We also noticed that connections to most of these overwhelmed ultrapeers exhibit long RTT ($>$ 1sec) and little to no loss. This indicates that their CPU is the bottleneck. Despite this finding, we did not incorporate a multiple attempt strategy into the crawler for two reasons: *(i)* it only marginally increases the number of reachable peers at the cost of significantly increasing the duration of each crawl which in turn increases distortion in captured snapshot, and *(ii)* it is intrusive and may exacerbate the existing problem.

Unreachable ultrapeers can introduce the following errors in a captured snapshot: *(i)* including unreachable peers that were departed, *(ii)* missing links between unreachable ultrapeers and their leaves, and *(iii)* missing links between two unreachable ultrapeers. To quantify these errors, it is important to determine what portion of unreachable peers were departed versus firewalled or overloaded. Previous studies assumed that these unreachable peers have departed or located behind a firewalls, and excluded them from their snapshots.

We have conducted further investigation to determine the status of these unreachable peers. First, we devised the following simple technique to identify the ratio of departed peers in each snapshot. We performed back-to-back crawls to capture two snapshots. Then, the unreachable peers in the first snapshot that were missing from the second snapshot are considered "departed peers" during the first snapshot. This approach revealed that departed peers constitute only 2–3% of unreachable peers in each snapshot. Second, we examined those unreachable peers that Cruiser considers to have timed out. Since overwhelmed ultrapeers refuse connections, we hypothesized that this group of peers is firewalled.

To verify this hypothesis, we randomly selected 1000 (about 3% of) peers that were unreachable due to time out, and re-contacted them every 5 minutes[3]. Interestingly, more than 92% of these peers were never reachable at all. This implies that timeout is a good indicator for firewalled peers. In summary, our investigation revealed that in each crawl, 2%–3% of unreachable peers are departed, and most of 15%–24% peers which timeout are firewalled. The remaining unreachable peers are either firewalled or overwhelmed ultrapeers.

IV. Performance Evaluation

We have been running Cruiser on six 1Ghz Pentium III PCs in our lab during the past year and have captured more than 20,000 snapshots of the Gnutella network. In this section, we present results to evaluate the ability of Cruiser to capture accurate snapshots and examine a few key tradeoffs.

a) Impact of Crawling Speed: To examine the impact of crawling speed on the accuracy of captured snapshots, we adjusted the crawling speed (and thus crawl duration) of Cruiser by changing the number of parallel connections that each slave process can open. Note that slow crawlers can effectively emulate the behavior of previously reported crawlers which have a lower degree of concurrency. We performed two back-to-back crawls for each crawling speed. We define δ_+ and δ_- as the number of new and missing top-level peers in the second snapshot compared to the first one, respectively (normalized by the total number of peers in the first crawl). Figure 3 depicts $\delta = \frac{\delta_+ + \delta_-}{2}$ as a function of crawl duration. The first snapshot was captured with the maximum speed and serves as a reference whereas the speed (and thus duration) of the second snapshot has changed. The duration of the second snapshot is shown as the x value. This figure clearly demonstrates that accuracy of snapshots decreases with the duration of the crawl, because the increased δ reflects changes in the topology that occur *while the crawler is running*. Additionally, Figure 3 shows the fraction of edges that were created or torn down during the crawl, *i.e.*, reported by only one of two contacted peers.

b) Completeness of Snapshots: To examine the completeness of snapshots captured by Cruiser, we kept track of the following variables during each crawl: the number of discovered top-level peers, the number of leaves, the number of links between ultrapeers, and the number of links to leaves. Figure 4 presents variations of these four variables as a function of the number of contacted peers in a sample crawl. Note that the number of discovered top-level peers as well as leaves curve off which is evidence that Cruiser has captured nearly all the participating peers. Links between top-level peers somewhat curves off. Finally, links to leaves is linearly increasing with the number of top-level peers because each top-level peers provide a unique set of links between itself and its leaves.

c) Accuracy-Completeness Tradeoff: To examine the accuracy-completeness tradeoff for captured snapshots, we modified Cruiser to stop the crawl after a specified period. Then, we performed two back-to-back crawls and repeated this process for different durations. Figure 5 clearly demonstrates the accuracy-completeness tradeoff. During short crawls (on the left side of the graph), δ is high because the captured snapshot is incomplete, and each crawl captures a different subset. As the duration of the crawl increases, δ decreases which indicates that the captured snapshot becomes more complete. Increasing the crawl length beyond four minutes does not decrease δ any further, and achieves only a marginal increase in number of discovered peers (*i.e.*, completeness). This figure reveals a few important points. First, there exists a

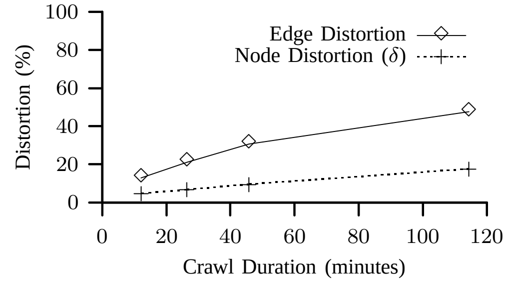

Fig. 3. Effect of crawl speed on the accuracy of captured snapshots.

[3]Note that each attempt translates into several attempts by TCP to establish a connection by sending SYN packets.

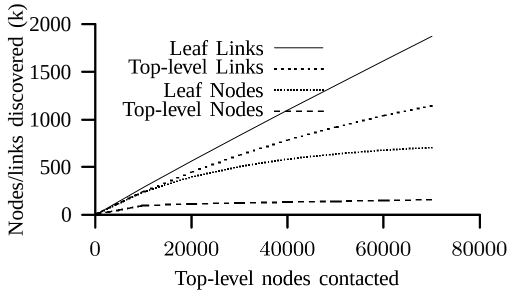

Fig. 4. Cumulative dicovered information about overlay nodes and links as a function of number of contacted peers

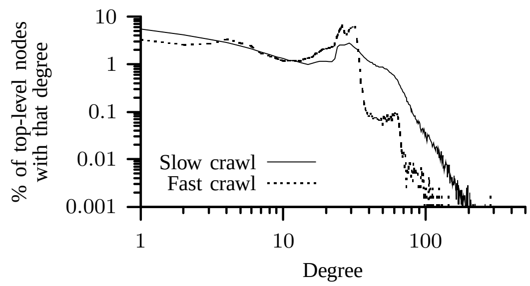

Fig. 6. Observed top-level degree distributions in a slow and a fast crawl

"sweet spot" for crawl duration beyond which crawling has diminishing returns if the goal is simply to capture the population. Second, for sufficiently long crawls, Cruiser can capture a relatively un-stretched snapshot. Third, the change of $\delta = 4\%$ is an upper-bound on the distortion due to the passage of time as Cruiser runs. The relatively flat delta on the right suggest that around 4% of the network is unstable and turns over quickly.

d) Impact of Snapshot Accuracy on P2P Characterization: To explore the impact of snapshot accuracy on analysis and conclusions, we plot two observed degree distributions in Figure 6: one captured running Cruiser at top speed, and one captured with Cruiser limited to 60 parallel connections, similar to the 50-connection crawler used in an earlier study [10]. The obtained degree distribution from the quick snapshot is relatively flat up until around 30, where there is a noticeable spike, then it drops off considerably with another small spike at degree 45. This corresponds with the default degrees that popular Gnutella clients attempt to maintain. There are negligible peers with higher degree. However, the degree distribution from the slower, more distorted snapshot exhibits a slight downward trend, followed by a power-law tail. This roughly matches the two-piece power-law degree distribution that was reported by Ripeanu et al. [10] with a similar slow crawler a few years ago. Note that the observed power-law degree distribution is a side-effect of unequal peer uptimes. More specifically, to a slow crawler, peers with long uptimes appear as high degree because many short-lived peers report them as neighbors. However, this is a mischaracterization since these short-lived peers are not all present *at the same time*. This comparison is an evidence that reported power-law degree distributions in unstructured P2P networks by previous studies may merely be an artifact of inaccurate measurement, and we demonstrate with certainty that Gnutella does not have a power-law degree distribution today.

It is worth clarifying that the impact of snapshot accuracy (or granularity) on the correctness of conducted characterizations is likely to vary for different types of characterization, *i.e.*, the required degree of accuracy for captured snapshots depends on the desired characterization. For example, a study on churn only requires information about participating peers and may not need to directly contact all peers. In contrast, to study the overlay topology, a captured snapshot should include all edges of the overlay which requires the crawler to directly contact every ultrapeer otherwise a connection between unvisited peers will be easily missed.

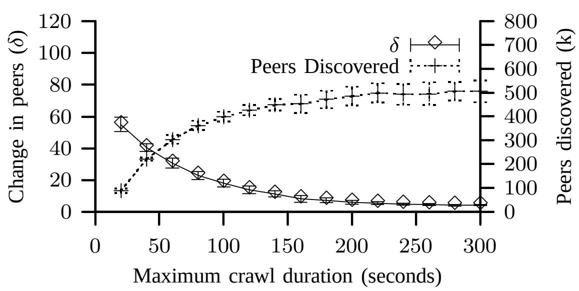

Fig. 5. Error as a function of maximum crawl duration, generated by running two crawls back-to-back for each x-value and computing the δ. Averaged over 8 runs with standard deviation shown.

V. Conclusions and Future Work

In this paper, we present Cruiser, a fast crawler for two-tier peer-to-peer systems such as Gnutella. We present the different techniques used in Cruiser to achieve its high speed, including leveraging the two-tier structure, a distributed architecture, asynchronous communications, and choosing appropriate timeout values. We also present techniques for quantifying the measurement inaccuracy introduced by crawl speed and present evidence that the error in Cruiser's snapshots is reasonably small.

In future work, we plan to use measurements gathered by Cruiser to characterize the Gnutella overlay topology, including its static and dynamic properties. We also plan to extend Cruiser to crawl other two-tier peer-to-peer systems for comparative analysis.

References

[1] Ranjita Bhagwan, Stefan Savage, and Geoffrey Voelker, "Understanding Availability," in *International Workshop on Peer-to-Peer Systems*, 2003.

[2] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble, "Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts," *Multimedia Systems Journal*, vol. 8, no. 5, Nov. 2002.

[3] Alexander Klemm, Christoph Lindemann, Mary Vernon, and Oliver P. Waldhorst, "Characterizing the Query Behavior in Peer-to-Peer File Sharing Systems," in *Internet Measurement Conference*, Taormina, Italy, Oct. 2004.

[4] Thomas Karagiannis, Andre Broido, Nevil Brownlee, Kimberly Claffy, and Michalis Faloutsos, "Is P2P dying or just hiding?," in *Globecom*, Dalls, TX, Nov. 2004.

[5] Thomas Karagiannis, Andre Broido, Michalis Faloutsos, and kc claffy, "Transport Layer Identification of P2P Traffic," in *International Measurement Conference*, Taormina, Italy, Oct. 2004.

[6] Daniel Stutzbach and Reza Rejaie, "Evaluating the Accuracy of Captured Snapshots by Peer-to-Peer Crawlers," in *Passive and Active Measurement Workshop*, Boston, MA, Mar. 2005, Extended Abstract.

[7] Subhabrata Sen and Jia Wang, "Analyzing Peer-To-Peer Traffic Across Large Networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, pp. 219–232, Apr. 2004.

[8] Krishna P. Gummadi, Richard J. Dunn, Stefan Saroiu, Steven D. Gribble, Henry M. Levy, and John Zahorjan, "Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload," in *SOSP*, 2003.

[9] clip2.com, "Gnutella: To the Bandwidth Barrier and Beyond," Nov. 2000.

[10] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi, "Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design," *IEEE Internet Computing Journal*, vol. 6, no. 1, 2002.

[11] Daniel Stutzbach and Reza Rejaie, "Characterizing Two-Tier Overlay Topologies in Modern P2P File-Sharing Systems," Tech. Rep. CIS-TR-2005-01, University of Oregon, Eugene, OR, Feb. 2005.

[12] "slyck.com," http://www.slyck.com, 2005.

[13] Anurag Singla and Christopher Rohrs, "Ultrapeers: Another Step Towards Gnutella Scalability," Gnutella Developer's Forum, Nov. 2002.

[14] Lime Wire LLC, "Crawler Compatability," Gnutella Developer's Forum, Jan. 2003.