A Peer Activity Study in eDonkey & Kad

Thomas Locher[1], David Mysicka[1], Stefan Schmid[2], Roger Wattenhofer[1]
[1] Computer Engineering and Networks Laboratory (TIK), ETH Zurich, CH-8092 Zurich, Switzerland.
{lochert@tik.ee.,dmysicka@, wattenhofer@tik.ee.}ethz.ch
[2] Chair for Theory of Distributed Systems, University of Paderborn, D-33102 Paderborn, Germany.
schmiste@mail.upb.de

***Abstract*— Although several fully decentralized peer-to-peer systems have been proposed in the literature, most existing systems still employ a centralized architecture. In order to compare these two paradigms, as a case study, we conduct measurements in the eDonkey and the Kad network—two of the most popular peer-to-peer systems in use today. We re-engineered the eDonkey server software and integrated two modified servers into the eDonkey network in order to monitor traffic. Additionally, we implemented a Kad client exploiting a design weakness to spy on the traffic at arbitrary locations in the ID space. The goal of this study is to provide insight into the spacial and temporal distributions of the peers' activities and also examine the searched contents. Finally, we discuss problems related to the collection of such data sets and investigate techniques to verify the representativeness of the measured data.**

I. Introduction

Today's peer-to-peer (p2p) networks come in different flavors. On the one hand, there are completely decentralized systems such as the *Kad* network which is based on a *distributed hash table* (DHT) [5], [13] where both the task of indexing the content and the content itself is distributed among the peers.[1] Other systems still rely on centralized entities, e.g., a cluster of servers takes care of the data indexing in the *eDonkey* network and so-called trackers handle the peers in *BitTorrent* swarms. A server-based solution has the advantage that it is easier to realize and that it works reliably as long as the servers function correctly. Clearly, the downside of this approach is that the servers can only sustain a certain number of peers, implying that the scalability is limited and that an overload of concurrent requests can easily cause a system failure. Purely decentralized systems do not depend on the availability of any particular entity; however, such systems often demand larger contributions from all participants.

This paper examines popular representatives of the two network types: the server-based eDonkey and the decentralized Kad network. eDonkey is one of the largest p2p networks in use today; millions of users around the planet use it to share various types of multimedia contents. While there are other clients to gain access to the eDonkey network, by far the most popular client is *eMule*[2]. Additionally, eMule allows its users to connect to the *Kad* network. This network, which is based on *Kademlia* [4], is currently the most popular distributed hash table.

In order to investigate various properties of eDonkey and Kad, we collected large amounts of data from both networks. For this purpose, we reverse-engineered the eDonkey server software and published two own servers which successfully attracted a considerable amount of traffic despite the fact that our servers never returned any real content. For our Kad tests, we implemented a client that is capable of spying on the traffic at any desired position in the ID space. Section II reports on the setup of our measurement infrastructure.

In Section III, several measurement results are presented. We were particularly interested in the user behavior in both networks. In this paper, in contrast to other literature, we monitor the actual user requests and ignore automated requests which occur without any user intervention. Our measurements show that the temporal request distributions of the two networks are very similar, exhibiting a high activity in the early evening with high loads at the eDonkey servers or at the peers hosting popular files in Kad. We also found that both networks are predominantly used in European countries, but there are also many active users from Israel, China, Brazil, and the U.S. Section III also investigates the content shared in the two systems. For example, we find that popular content in the eDonkey world is often also popular in Kad and that eDonkey follows the popularity trends of the real world. In general, our results indicate that peer activity results in eDonkey directly carry over to the Kad network and vice versa.[3] Finally, we raise the question of the representativeness of the collected data. In the Kad network, accurate data on the activity of a specific file can be obtained, but due to the distributed nature of the DHT, it is inherently difficult to compute global aggregates such as the most active file in the network. On the other hand, in the eDonkey network, a server receives queries for virtually all keywords, but it has to compete against other servers for the requests. If only a minor fraction of the traffic arrived at our servers or if the servers to be queried were selected with respect to specific properties such as latency, the data could become biased. We will provide evidence that there is no critical bias in our measurements.

After reviewing related work in Section V, the paper concludes in Section VI.

[1] Unstructured decentralized systems such as *Gnutella* are not considered in this study.

[2] See http://www.emule-project.net/.

[3] This observation is not self-evident, given that we analyze only user-generated events.

II. Measurement Framework

The eMule client provides access to the classic, server-based eDonkey network and the decentralized Kad network, which is, as mentioned before, an implementation of the distributed hash table Kademlia [4]. The different nature of the two networks requires different measurement techniques. In the following, we will first present our approach to collect data in the eDonkey network. Subsequently, we will report on the functionality of our Kad client which allows us to monitor traffic at arbitrary spots in the ID space.

A. eDonkey Network

When a user issues a query using the eMule client, the keywords of the query are sent to a subset of servers, which subsequently respond to the client with information about where to obtain the requested file. We found that the peers iterate over the list of servers contained in their server file, querying one server after the other as long as less than 300 results have been returned. The order of servers in this list reflects the history of when peers learned about these servers, i.e., old servers are at the top of the list while new servers are appended at the end of the list.

Today, there is a large number of eDonkey servers all over the world, most of which are based on the *lugdunum*[4] software. This software is not open-source as the developers try to prevent the creation of fake servers or any other undesirable modification that could endanger the correct functioning of the lugdunum servers. In order to collect data in the eDonkey network, we reverse-engineered the server software and set up two servers ourselves which operate as follows. Initially, our server imports all known eDonkey servers from a file and announces itself to every server on that list, one after the other. For each server on the list, a *server list request* is sent, followed by a *server status request* and a *server description request*. In return, our server receives a list of servers that are alive, and the current status and description of the corresponding server. As a side effect of these queries, our server is added to the other server's list. This is vital as peers keep their server lists up to date by periodically asking the servers they are connected to for their lists of currently known servers; i.e., once our server appears in these server lists, all peers will quickly learn about the existence of our servers. In order to remain a member of these lists, our servers correctly answers the status requests of other servers. However, due to legal concerns, we neither store nor return any real data. Moreover, we pretend having a high number of users and shared files, but we deny any login requests and reply with a message indicating that our server is full.

Due to the iterative lookup procedure described before, our servers are contacted perpetually, regardless of which servers the peers are connected to. As a result, we can collect a large amount of data about many different kinds of requests, making it possible to compute global aggregates such as the most popular keyword in the network, or the most active peer's IP address. Naturally, this data is only representative if we receive a substantial fraction of all requests in the network. This issue is discussed in more detail in Section III.

[4]http://lugdunum2k.free.fr/kiten.html

B. Kad Network

In the Kad network, information about the location of specific files is stored at the participating peers themselves, which all have so-called *overlay IDs*. In order to find a file for a given keyword k, a peer computes a hash function $h(k)$ of k and routes, in a multi-hop manner, the request to the peer P having the overlay ID closest to $h(k)$. This peer P store the hash codes of all the files associated with this keyword. The matching filenames and the corresponding hash codes of these files are then returned. Given a hash code $h(f)$ of a file f, it is then possible to get a list of all the peers possessing a copy of f by again routing to the peer whose ID is closest to $h(f)$ as this peer is responsible for the sources of f.

Note that in Kad, information is replicated several (i.e., ten) times in a zone where peers agree in the first 8 bits with the published key. Usually, this so-called *tolerance zone* contains several thousand peers. While most of the peers are very close to the key, this is not always the case, e.g., due to churn and also for keys that are very popular and published by many different peers.

We created our own Kad client in order to collect data on the peer activity in the Kad network. Our client exploits the fact that Kad uses randomly chosen overlay IDs, which enables us to place our peers at any desired place in the ID space. On the one hand, performing measurements in the Kad network is simpler than in the eDonkey network. This is due to the fact that a small set of peers close to the hash of a file f will be contacted by all peers interested in obtaining this file f. Thus, as there is a unique location where peers obtain information about f, data of good quality can be collected by occupying the corresponding area around this ID and spying on the traffic. On the other hand, the distributed nature of the Kad network renders it more difficult to measure global quantities such as the most popular file in the network. Answering such a query would require to occupy a large portion of the entire ID space. Hence, we confine ourself to acquiring small samples of the entire traffic and try to juxtapose these samples and the data acquired in the eDonkey network in a reasonable manner.

III. Measurements

This section summarizes our measurement results. We investigated the distribution of the user base across countries of both eDonkey and Kad and also the temporal and spacial distribution of the users' requests. In addition, the concrete content that users search in the system is examined.

A. Request Distributions

Within a few days after announcing our servers, they attracted much traffic. Figure 1 shows the activity of our servers during 4 days. We see that the request pattern remains fairly stable across all days. On average, during a measurement period of 2 weeks, our servers received roughly 1,550 login requests, 448 keyword requests and 150,228 source requests

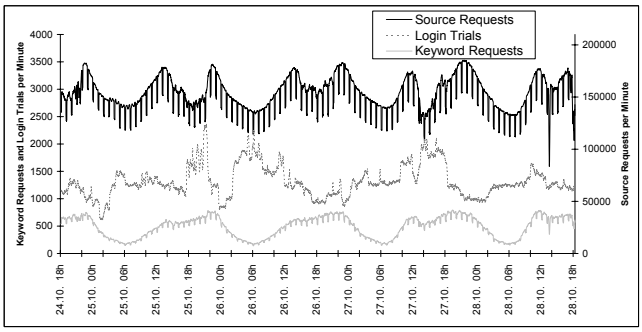

Fig. 1: Different server requests over time. The y-axis for the source requests is shown on the right, for the login trials and the keyword requests it is shown on the left.

per minute. The average bandwidth required to run each server is approximately 300 KB/s. Note that a correct server requires substantially more bandwidth as it has to reply to all keyword and source requests. Due to the additional traffic caused by re-announcing our servers at other servers once per hour, our servers are overloaded for a short time resulting in regular drops of handled requests, which is most apparent in the curve of the recorded source requests.

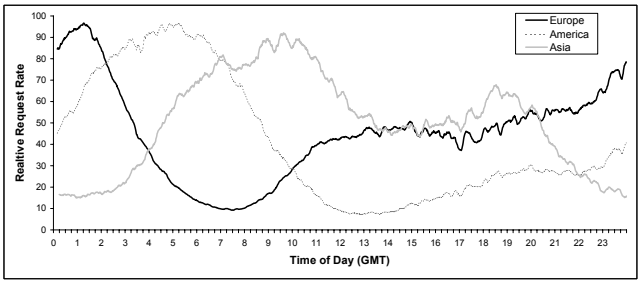

Fig. 2: Temporal distribution of keyword search requests on an average day on eDonkey, grouped by continents. The time on the x-axis is based on the Greenwich Mean Time.

The keyword searches are particularly interesting to study, as they are entered by users directly and are hardly automated. Consequently, the amount of search requests varies over the day. Figure 2 shows this distribution for different continents. The figure reveals that in Europe and America the minimum number of requests is reached in the early morning and this number continuously increases until midday, where it stays on a more or less constant level during the whole afternoon. Then it increases again after the working hours until the maximum is reached at around midnight. The curve for Asia looks slightly different; the maximum is also reached at midnight, but there is not such a sharp decline during the night, and the number of requests even increases again reaching a second local maximum in the early morning. Note that the maximum number of requests is set to 100% for each continent in order to show this diurnal pattern. The total number of requests per day in Europe, America, Asia, and Africa plus Middle East are 397,060, 156,322, 42,287, and 48,850, respectively, which necessitates this normalization and also demonstrates the predominance of Europe in the eDonkey network.

As one might expect, the distribution of the search requests in the Kad network is similar. Figure 3 depicts the temporal distribution of requests again for the three continents in the Kad network. Again, the curve for Asia is quite different from the others. As opposed to the other continents, the maximum number of requests in Asia is reached in the morning and not late in the evening. We occupied 14 randomly chosen IDs and logged all requests on these peers and used the average number of requests in this figure.

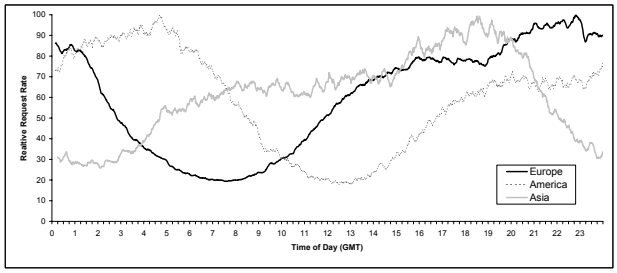

Fig. 3: Temporal distribution of keyword search requests on an average day on Kad, grouped by continents. 14 monitoring peers in Kad are used to compute these numbers. The time on the x-axis is again based on the Greenwich Mean Time.

We can look at the origins of the requests in more detail and observe that European countries play an important role in eDonkey, the only country among the five most active countries outside of Europe is Brazil. Figure 4 depicts the percentage of all requests originating from each of the 20 most active countries per month, both for the eDonkey and the Kad network in descending order of activity in the eDonkey network. A first observation that can be made is that the spacial distribution is more concentrated in Kad than in eDonkey. Moreover, it can be seen that the same countries are the most active ones in both networks. Note that, although eMule grants access to both networks, users have to enter *manually* where they want to search and thus this result is not self-evident. Furthermore, the Kad network seems to be significantly more used in Europe, especially in Italy and France, than elsewhere. The question whether this is due to a less strict legislation remains open.

It is difficult to assess the popularity of these networks by comparing the absolute number of requests, as there are countries with a much larger population or a higher Internet penetration rate. For this reason, we have normalized the request rates received from each country by the number of Internet users in that country.[5] As can be seen in Figure 5, the picture looks different in the normalized case. There are three quite active countries, Morocco, Algeria, and Israel, while all other countries have a comparably small number of requests per Internet user per month. The reason for this exceedingly high number of request originating from Morocco

[5]Data obtained from http://www.internetworldstats.com.

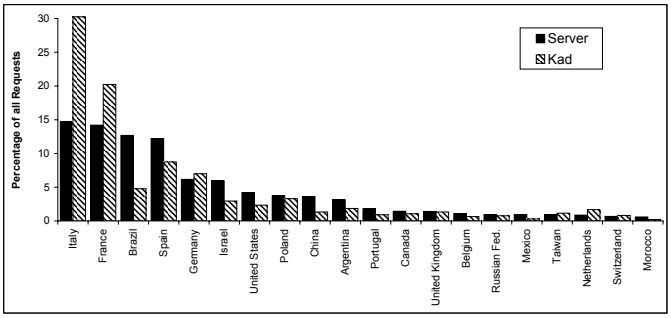

Fig. 4: Origins of keyword search requests on our servers and in the Kad network.

and Algeria might be simply due to the small number of Internet users in these countries. Another possible reason is that relay servers are positioned in these countries in order to obfuscate network traffic. The observation that a large number of requests originate from a small number of IP addresses supports this claim. As there are many different IP addresses active in Israel and given that it is generally one of the most active countries, it seems that these networks are simply highly popular in Israel, even more so than in Europe. As far as the other countries are concerned, the graph shows that there is not a significant difference between the popularity of eDonkey and Kad among them. What is more, the distribution for both networks has a long tail; as many as 21 countries exhibit a normalized search activity of at least 20% of the search activity of Spain, implying that both networks are popular in many countries. We further found that both networks are indeed much more popular in Europe than in the United States, the activity of the United States normalized by the number of Internet users is about 30 times smaller than the activity of Spain, making it the country with almost the smallest activity overall. Clearly, this is partly due to the large number of Internet users in the United States. Overall, only six countries contribute more keyword searches than the United States, which indicates that also in the United States both networks have a large user base. Finally, however, note that the data in Figure 5 could also be slightly biased, as the Internet penetration data might not be perfectly accurate.

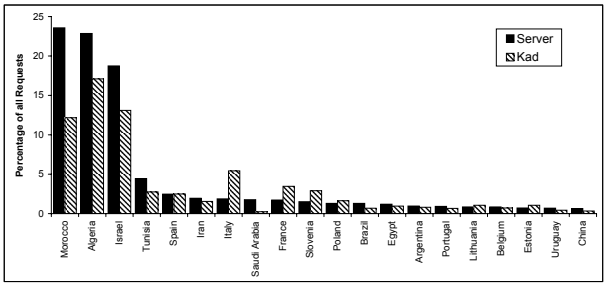

Fig. 5: Keyword search requests normalized by the number of Internet users of the 20 most active countries on our servers and in the Kad network.

B. Search Contents

The main objective of both the eDonkey and the Kad system is to provide users with a mechanism to find and download files. Information about the searched content can be an interesting source for research, for example, such data might give insights into the potentially different preferences of users in different countries.

For this purpose, a record indicating the popularity of each data item in each country would be required. Unfortunately, the compilation of such a record is quite difficult—not only in Kad, but also in the eDonkey network. One reason is that there is no automatic one-to-one correspondence between keywords and files. There might be different spellings of the same keywords, files containing the same content are typically available in different languages, and the corresponding filenames often contain typing errors. Moreover, the popularity of the files we monitor in Kad can change quickly, particularly when versions of the same content, e.g., a video file, of increased quality appear. Figure 6 plots different versions found when quering for a specific exemplary keyword during a period of 50 days. Version v_1 is the worst quality, v_2 is the same content in better quality, and v_3 has the best quality. As expected, the number of occurrences of v_1 decreases over time, first at the expense of v_2, and after v_3 becomes more and more popular, the number of occurrences of v_2 start decreasing as well.

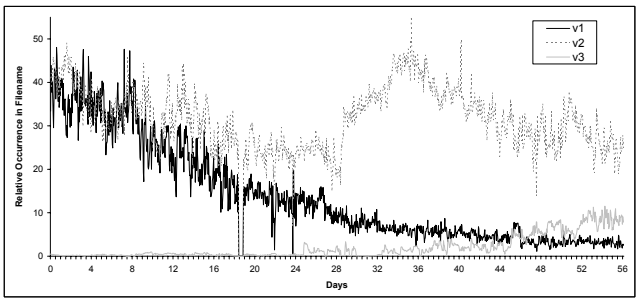

Fig. 6: Different quality versions, distinguished by specific keywords in the filename, in percentages of all files.

In another experiment, we tried to evaluate to what extent the popularity of certain content in eDonkey and Kad corresponds to the popularity of the same content in the real world. To this end, we observed the popularity of newly released movies in eDonkey and Kad. We find that there is indeed a strong correlation, i.e., movies that are currently playing in movie theaters are popular both in eDonkey and Kad. Figure 7 shows this correlation for a specific movie. In this figure, the total gross[6] in the U.S. is depicted for each day and also the number of requests for this movie on our servers. The movie opened on October 5, but it did not attract many moviegoers until the next weekend. Since then, the daily gross is declining again with smaller peaks at the weekends as usual. In this graph, we see that the popularity in eDonkey roughly follows these trends. Observe that the request pattern in the

[6]Data obtained from www.boxofficemojo.com.

network is delayed for about a week, reaching its maximum about a week after the movie reached its peak. Experiments using other content yielded more or less the same graph, also with a certain delay. In order to take the Kad network into account, we further compared how often keywords are looked up in eDonkey and in Kad and found that basically the same keywords are looked up more often than others in both networks.

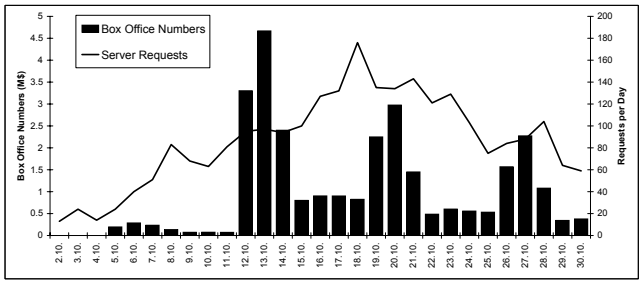

Fig. 7: Comparison of the box office gross and the requests on our servers for a specific movie.

Due to space constraints, we cannot describe these other experiments also concerned with the question of content popularity. Our findings all indicate that there is not only a strong correlation between eDonkey and Kad, but also between the two networks and the popularity of content in the real world.

IV. Representativeness

Conducting measurement studies of distributed systems is a difficult endeavor. Even if large amounts of data is collected, the statistical significance of the empirical results might be limited if the data is biased. In order to obtain solid claims, it is important that the underlying data be either complete, or a uniform and random subset thereof. In this section, we provide evidence that our data can be considered representative.

We consider the data collected by the servers first. As mentioned before, the servers receive requests for all possible keywords. However, since a peer does not send requests to all the servers in its server list, i.e., some servers might receive completely different requests, which could potentially bias the collected data. As the eMule clients typically send *source requests* to both networks, in order to estimate what fraction of all requests we receive, we compared the number of source requests at our eDonkey server with the number of source requests obtained in Kad. Our experiments showed that for a given file, we receive roughly 10 times more request in Kad than at the server. Since virtually all requests for a given file are received in Kad, this indicates that our server roughly receives 10% of all keyword requests in the network—a surprisingly large number. At the same time, the distribution of the origins of the requests does not differ between the two networks. This suggests that they are already contacted with a reasonably large probability, although our servers are relatively new, and also that they get a more or less random subset of the entire traffic.

In the Kad network, it is easy to obtain unbiased request data for a given file, since all requests for a particular file are routed to the same ID. However, making statements about the global distributions of the requests requires to collect data at all locations in the ID space, which is impossible. In this paper, we have taken a best-effort approach and aimed at getting data from a moderately large set of peers whose IDs are distributed uniformly at random. By averaging these measurements, we get similar distributions as those measured in eDonkey, which indicates that the obtained data is fairly representative. Although we believe that the quality of our results is quite good, it has to be taken into account that, similarly to our client, other peers can also choose their overlay IDs at will, which could bias such a random sampling approach. It is known that there are communities that select their Kad IDs from a small subset of the entire ID space [8].

V. Related Work

Measurement studies are an important means to gain deeper insights into the working of distributed systems. While theoretic models allow researchers to reason formally about a system's behavior and to prove its properties, such models are often simplifications and may not reflect reality well. For more complex systems, *in silico* experiments are conducted, desirably for as many points in the parameter space as possible. However, although such simulations—and also experiments on PlanetLab [3]—can provide additional confidence in a system's performance, it is not until the real deployment when the system properties become clear.

There exist many measurement results for various p2p systems today. Saroiu et al. [6] have analyzed several characteristics such as the bottleneck bandwidths of the peers participating in Gnutella and Napster. Adar et al. [1] have investigated the contributions of the Gnutella users. An important algorithmic challenge in p2p computing is understanding churn, and hence traces of membership changes in the systems deployed today [8] have been collected. There is also a community aiming at reverse-engineering closed-source projects such as Skype by studying the traffic patterns [2].

We have decided to study the eDonkey and the Kad networks as they are two of the largest overlay networks in use today, and as there does not exist much literature on these networks. Interesting results on the *Kad network* have been obtained by Biersack, Steiner, and others in [9], [10], [11], [12]. For instance, in [11], possible misuses of the protocol are discussed. Stutzbach et al. [15] describe implementation details of Kad in eMule, and [14] presents crawling results on the behavior of Kad peers. The paper closest to ours is by Steiner, Biersack and Ennajjary [8]. The authors have crawled the Kad network during several weeks and found, e.g., that different classes of participating peers exist inside the network. In contrast to their work which has studied the churn induced by the peers' joins and leaves, our focus is on the peer *activity* while the peers are online, which we measure by monitoring the lookups. As stated in [8], peer IDs can change frequently, even as often as once per download session while

other IDs remain in the network for several weeks. Due to these conditions and the fact that several peers might share the same IP address, it is hard to draw any conclusions about peer behavior when monitoring the peer IDs and the IP addresses in the network. Since keyword lookups are hardly automated, observing lookups is the best and presumably the only way to get insights into the activities of users in such networks. To the best of our knowledge, this is the first peer activity study by means of monitoring lookup requests in distributed networks. It is also the first study to take both server-based and decentralized systems into account.

VI. Conclusion

A large fraction of today's Internet traffic is due to peer-to-peer technology. Understanding the behavior of peers in such large networks might enable the development of new and more efficient distributed algorithms or even pave the way for novel applications in distributed systems.

In this paper, we have compared the peer activity in the server-based eDonkey network with the distributed hash table Kad, two of the largest peer-to-peer networks in use today. We have found that not only do most requests arrive roughly during the same time interval every day in both networks, the searched content is also quite similar. Moreover, by counting the number of source requests we found that our server receives roughly 10% of all eDonkey requests. Using this estimate, and given that we receive virtually all requests for certain keywords in Kad, we conclude that the eDonkey network is still more popular. In total, we estimate the total number of requests in eDonkey to be somewhere between 1.3 and 2 times larger than in Kad. It will be interesting to see how the situation develops in the near future. Furthermore, we conclude that data on the peer activity collected in either eDonkey or Kad can be used as a rough estimate of the behavior in the other network. We plan to keep collecting data for further experiments, and we also intend to publish some of the log files online as they might be useful for other studies as well.

References

[1] E. Adar and B. A. Huberman. Free Riding on Gnutella. *First Monday*, 5(10), 2000.

[2] S. Guha, N. Daswani, and R. Jain. An Experimental Study of the Skype Peer-to-Peer VoIP System. In *Proc. 5th International Workshop on Peer-to-Peer Systems*, 2006.

[3] A. Haeberlen, A. Mislove, A. Post, and P. Druschel. Fallacies in Evaluating Decentralized Systems. In *Proc. 5th International Workshop on Peer-to-Peer Systems*, 2006.

[4] P. Maymounkov and D. Mazières. A Peer-to-Peer Information System Based on the XOR Metric. In *Proc. 1st IPTPS*, 2002.

[5] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proc. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.

[6] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proc. of Multimedia Computing and Networking (MMCN)*, 2002.

[7] M. Steiner. Private Communication.

[8] M. Steiner, E. W. Biersack, and T. Ennajjary. Actively Monitoring Peers in the KAD. In *Proc. 6th IPTPS*, 2007.

[9] M. Steiner, D. Carra, and E. W. Biersack. Faster Content Access in KAD. In *Proc. 8th IEEE Conference on Peer-to-Peer Computing (P2P)*, 2008.

[10] D. Carra and E. W. Biersack. Building a Reliable P2P System out of Unreliable P2P Clients: The Case of KAD. In *Proc. ACM CoNEXT*, 2007.

[11] M. Steiner, T. En-Najjary, and E. W. Biersack. Exploiting KAD: Possible Uses and Misuses. In *Computer Communication Review 37(5)*, 2007.

[12] M. Steiner, T. En-Najjary, and E. W. Biersack. A Global View of KAD. In *Proc. of the 7th ACM IMC*, 2007.

[13] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proc. ACM SIGCOMM*, 2001.

[14] D. Stutzbach and R. Rejaie. Understanding Churn in Peer-to-Peer Networks. In *Proc. 6th IMC*, 2006.

[15] D. Stutzbach and R. Rejaie. Improving Lookup Performance over a Widely-Deployed DHT. In *Proc. 25th IEEE INFOCOM*, 2006.