Towards a Basic DHT Service: Analyzing Network Characteristics of a Widely Deployed DHT

Konrad Jünemann, Philipp Andelfinger, and Hannes Hartenstein
Steinbuch Centre for Computing (SCC)
Karlsruhe Institute of Technology (KIT), Germany
{juenemann, hartenstein}@kit.edu, philipp.andelfinger@kit.edu

***Abstract*—Distributed Hash Tables (DHTs) prove valuable for distributed architectures by providing distributed information lookup, routing, and data storage while promising good scalability and robustness at the same time. From this point of view, a DHT could be seen as a basic service that can be used to build distributed applications. Whereas today no widely deployed and publicly accessible basic DHT service exists and thus DHT-based applications have to deploy their very own DHT networks, DHTs consisting of millions of peers are formed by file sharing clients. Although the interfaces of typical DHTs used for file sharing are too narrow, a basic DHT service could probably be created by bundling a suitable client implementation with file sharing software. In this paper, we evaluate whether a basic DHT service could suit the needs of DHT-based applications in terms of stability, number of participating peers, the peers' session lengths, geographical distribution and peer connectivity when deployed similar to DHTs driven by file sharing. As these metrics mostly depend on end user behavior rather than on the DHT protocol, we report on measurement results gathered from monitoring of the BitTorrent Mainline client's DHT over six months. We analyze which metrics would fit a basic DHT service and which could prove problematic. Furthermore, we discuss resulting technical requirements for an appropriate DHT protocol.**

I. Introduction

A Distributed Hash Table (DHT) is a Peer-to-Peer (P2P) network that provides distributed lookup and storage of key/value pairs. Extensive research has led to well analyzed DHT protocols such as Kademlia [15], Chord [23] or Pastry [18]. A general overview about various protocols is given in [14].

Mostly driven by academic research projects, a diverse set of DHT-based applications emerged during the last couple of years. Those applications serve a wide range of use cases including distributed file systems [19], [5], content distribution [9], distributed DNS [1], and traffic information services [20], to name a few. Although some of these applications use DHTs specially tailored to their respective use cases, many have similar requirements for the underlying DHT. Those could be built on top of a single shared, general purpose DHT.

The idea of installing a public DHT service that would be shared by a multitude of different DHT-based applications is not new but led to the introduction of OpenDHT [16][1]. OpenDHT was designed as a public DHT service that ran on top of the PlanetLab [2] testbed. Whereas OpenDHT proved valuable as an easy way to develop and debug new DHT-based applications without having to worry about DHT deployment, running OpenDHT nodes on PlanetLab entailed two disadvantages: first, the DHT did not scale with growing numbers of clients as the DHT consisted of a fixed number of (centrally administered) PlanetLab nodes only and clients did not contribute to the DHT. Second, the OpenDHT nodes had to be maintained.

Simultaneously, the file-sharing community started leveraging DHTs to replace centralized components within their file sharing networks. Therefore, DHT clients were bundled with file sharing clients, resulting in widely deployed DHTs introduced by clients like Azureus [31], eDonkey or the BitTorrent Mainline client [13]. These DHTs do not directly suit the needs of most DHT-based applications as for instance one can typically store only minimal information within the DHT or lookups provide poor performance. However, a more generalized, OpenDHT-like implementation could overcome those issues. By bundling a suitable implementation with file sharing clients, a basic DHT service consisting of several millions of peers could be created. While such a DHT service could still be used for file sharing purposes, it would also constitute a foundation for other DHT-based applications without requiring any dedicated infrastructure. In particular, the applications would benefit from having access to an established, highly active and well populated DHT from the very beginning.

When considering a global DHT service formed by millions of independent, unreliable peers rather than few properly maintained machines, the DHT's overall behavior is far more opaque. Moreover, certain key characteristics of the DHT mainly depend on the end users' behavior respectively the DHT's primary use case rather than on the chosen DHT protocol. We identify three fundamental requirements we assume to be shared by most conceivable DHT-based applications: reliability of the distributed service, persistence of data inserted into the DHT and short lookup duration. Characteristics of the DHT relevant with regard to these requirements include:

DHT size: The number of concurrently participating peers affects scalability, overall performance as well as the amount of resources such as computing power or storage space available within the DHT. Fluctuations might lead to varying overall performance.

Locality: The distribution of the peers' regions of origin might allow or forbid optimizations that leverage locality to speed up lookups or data delivery. Moreover, if most peers

[1]OpenDHT was taken down in 2009.

978-1-4577-0638-7 /11/$26.00 ©2011 IEEE

originated from a specific region the DHT might depend on the technical infrastructure of these countries being available.

Connectivity and session length: Limited peer connectivity due to firewalls or NAT gateways might significantly slow down lookups. The peers' average session lengths might affect to what amount stored data needs to be replicated or refreshed.

In this paper we analyze these characteristics by presenting results gained from continuous monitoring of the BitTorrent Mainline client's DHT for more than 6 months. The Kademlia-based [15] BitTorrent Mainline DHT was chosen as it is currently one of the most widely deployed DHTs. However, we do not argue that a public DHT service should be based on Kademlia or any other specific protocol. We rather assume that file sharing would remain the DHT's primary use case for some time if the DHT were deployed in a way similar to any of the widely used DHTs. Thus it would show similar characteristics. Our main findings are summarized as follows:

(i) With the exception of diurnal fluctuations, no unforeseen peaks or breakdowns of the DHT have been witnessed. A public DHT could thus potentially constitute a stable foundation for higher level applications.

(ii) The DHT size shows a strong diurnal fluctuation, increasing by about 60% during the day. DHT-based applications have to be able to cope with this variation.

(iii) The peers are fairly well distributed over most industrialized countries. DHT-based applications could thus leverage peer locality in most parts of the world.

(iv) As 50% of peers are located behind a NAT gateway or firewall, the DHT should include countermeasures against limited connectivity peers to prevent decreased performance.

The remainder of this paper is organized as follows. In Section II we present related work before giving a brief overview about the BitTorrent Mainline DHT in section III. Section IV presents our measurement results and illustrates the used measurement methodology. In section V we discuss the implications of our results and their validity in the future. Section VI concludes the paper.

II. Related Work

A wide range of measurement studies analyzing different DHT protocols and networks exists. However, while most other studies focus on analyzing a *single* aspect such as churn [24], [28], peer connectivity [10], [27], or lookup performance [7], [12], [25], we aim at giving a consistent view on *multiple* aspects relevant to the use case of a basic DHT service. As all of our evaluations are based on the same data, all shown results are comparable to each other. The data set was gained by continuously measuring the DHT for more than 6 months. In that time, we found 15.9 millions of unique peer IDs, each with the potential of being found several hundreds of times during the last 6 months, as almost 50 crawls were performed a day. To our knowledge, this makes this study the longest and most extensive DHT measurement study performed till now. Furthermore, most other studies do not analyze the evolution of the examined metrics. In our study, we analyze fluctuations over time when they occur or state otherwise. Moreover, up to date results are continuously published on our website by our automated measurement tool.

Other studies exist that evaluate DHT properties related to our study. Among others, the session length of peers within the KAD network is analyzed in [21], [22]. In line with our findings for the BitTorrent Mainline client's DHT, the largest number of KAD peers is found to originate from Europe.

In [10], the connectivity properties of peers within the BitTorrent Mainline DHT are analyzed. Our connectivity analysis is similar to the 24 hour measurement presented by the authors. We found more fully accessible peers but less unresponsive peers. We will compare our findings to related studies in greater detail in the appropriate sections.

III. The BitTorrent Mainline DHT

The BitTorrent Mainline DHT is one of multiple Kademlia-based DHTs that enable access to and distributed storage of key/value pairs. Within the DHT, peers are identified by a *node ID* chosen at random from a 160 bit ID space. Keys are mapped to the same ID space by a hash function. A peer is responsible for storing a specific key/value pair if the peer is one of the 8 closest peers to the key's hash. Proximity between IDs is determined by the XOR metric, which is defined as the bitwise exclusive or (XOR) of two IDs, interpreted as an integer. The Kademlia protocol relies on four types of remote procedure calls (*RPCs*): PING, to check a peer's availability, STORE, to save a key/value pair on the addressed peer, FIND_NODE, to locate peers close to a given node ID, and FIND_VALUE, to retrieve a value from a peer by the corresponding key.

For retrieving the value for a specific key, a peer has to perform a *lookup* to find the 8 peers closest to the key's hash (*target ID*). During a lookup, a peer sends FIND_NODE RPCs to those of the peers known to him that are closest to the target ID. Additional FIND_NODE queries are sent to the peers returned by previous responses to RPCs. This way, the target ID is approached iteratively. The procedure terminates when the 8 closest known peers return no previously unknown peers that are even closer to the target ID.

IV. Measuring DHT Characteristics

In this section we illustrate key characteristics of the Mainline BitTorrent DHT as well as their evolution. Our analysis is based on results collected by continuous measurement from August 2010 to February 2011.

A. Measurement Platform

For our measurements, we used a measurement platform called BitMON [11] implemented in our research group. *BitMON* is an extensible tool for automatic monitoring of the Mainline BitTorrent DHT. BitMON combines conducting measurements with the subsequent analysis of measurement results. Results of selected measurements are presented in real-time on our website[2]. For accessing the BitTorrent DHT, BitMON utilizes *JKad*. JKad is a framework for implementing

[2]http://dsn.tm.kit.edu/english/2924.php

clients for DHTs based on Kademlia we developed for our research. JKad is especially well suited for academic purposes. It currently supports the BitTorrent Mainline DHT and can be easily extended to additional DHTs. The Java source code for both JKad and BitMON is available on request.

B. DHT Sampling

For making realistic assumptions about a widely deployed P2P network such as the Mainline BitTorrent DHT, one has the option of either crawling the whole P2P network or extrapolating from a smaller, representative subset of all peers.

A full crawl of the BitTorrent DHT with up to 10 millions of concurrently participating users not only takes longer than a partial crawl but also has a bigger memory footprint. Results gathered from a full crawl are thus less accurate on the timescale and inconvenient to save for later analysis.

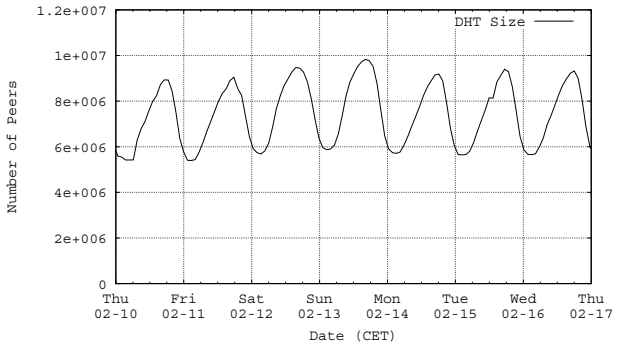
Thus, we decided to crawl smaller partitions of the DHT to collect representative subsets of the complete peer population. Most of the measurement results presented in this paper are based on these subsets, unless stated differently. In the following, we will describe the crawler we implemented for our measurement platform BitMON. It was run on a dual core machine using Ubuntu 9.10, directly connected to the Internet by a symmetric 1 GBit/s connection.

For retrieving a representative data sample, our crawler tries to find all peers within a small interval of Kademlia's ID space only, which can be examined in a shorter time and with less usage of bandwidth. The interval spans over 2^{152} Kademlia IDs, representing the 256^{th} part of the whole ID space. Different from typical crawlers, our algorithm is based on DHT lookups only, which return the 8 closest peers to a specified target ID. Using these lookups the ID interval is repeatedly scanned in multiple cycles. While the *scan rate* – the number of DHT lookups sent per second – stays the same, the *accuracy* of the scan – the number of lookups per cycle – doubles from cycle to cycle. Thus, each cycle takes twice the time of the one before. The scanning rate was set to 10 lookups per second.

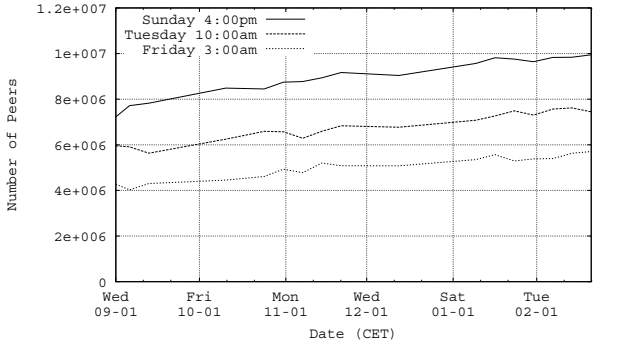
When extrapolating, one has to make sure that the sample can be seen as representative. In our case, the samples taken during our measurements can be expected to be representative under the assumption of peers being uniformly distributed over the ID space. While every peer *should* pick a (uniformly distributed) random ID according to the Kademlia protocol, one could question if this is the case in praxis. A study of the also Kademlia-based DHT *KAD* [22] indicates that peers mainly pick random IDs with the exception of very few clients that use fixed, hard coded IDs, resulting in a small number of overused IDs. However, peers using the same ID as others will not actively contribute to the DHT in any way and are thus not seen as participating in the DHT. Our crawler detects at most one peer per ID due to the use of ordinary DHT lookups.

C. DHT Size

For accurately measuring the DHT size one has to consider the *churn rate*. The *churn rate* denotes the number of peers joining and leaving the DHT at any point in time. As new peers will join the DHT at any time there will always be new peers to find. Thus, the number of peers found per second will not converge against zero but against the churn rate. Therefore it is critical to stop crawling when the number of peers found per second converges against the churn rate.



(a) Weekly DHT fluctuation.



(b) Weekly maximum, minimum and average number of participating peers.

Fig. 1. Evolution of the DHT size.

For measuring the churn rate, we used a modified version of our crawler that continuously scans an ID-interval using a fixed accuracy of 2^{14} lookups per cycle and scan rate of 160 lookups per seconds. For every cycle, the crawler saves the number of peers found in this cycle that were previously unknown. After the first couple of cycles, the crawler knows virtually all participating peers within the interval. Thus, the only peers found in later cycles are peers that recently joined the DHT and hence constitute the churn. During our measurements, we found a average churn rate of 288.23 peers per second.

In the following we present our findings on the number of concurrently participating peers (DHT size) as well as its short and long term fluctuations. All presented numbers are estimations gained from extrapolating from collected samples by multiplying the number of peers found within the 256^{th} part of the DHT's ID space by 256. Considering possible measurement inaccuracies the absolute number of participating peers might slightly deviate in reality. However, the relation between single data points should be unaffected.

Figure 1(a) depicts how the DHT size changes during one week. A strong diurnal cycle can clearly be seen, comparable to the findings of Steiner et al. within the KAD network [22].

We observed a daily maximum at around 6 pm CET and a daily minimum at 4 am. From minimum to maximum the DHT size increases by about 60%. Furthermore, the DHT size shows a weekly pattern, increasing during the weekend and reaching its weekly maximum on Sunday.

Figure 1(b) depicts the long term evolution of the DHT size by plotting the DHT size at the weekly minimum, average and maximum from August 2010 to February 2011. We found the DHT size to be increasing rapidly at a rate of up to 6% a week, growing from a weekly maximum of about 7.2 million to 9.8 million peers in 6 months.

Thus, both the DHT service and any DHT-based application on top of it have to be able to cope with strong diurnal fluctuations. Despite that, we found the DHT size to be quite stable: we did not notice any unforeseen peaks or break downs during the observed period of time. A DHT service induced by file sharing clients could thus form a stable foundation for higher level applications. However, as the size of the BitTorrent Mainline DHT is still growing rapidly, it is not yet clear how many peers can be expected to be present in the long run. Up-to-date results of our automated measurement are continuously being published at our website.

D. Session Length

We define the *session length* as the duration a peer joins a DHT for. Session time is thus defined as $t_s = t_l - t_j$ where t_j and t_l denote the points in time at which the peer joined respectively left the DHT. Short session lengths are the main reason for high churn rates, which can have a strong impact on the DHT's lookup performance and stability, depending on the chosen DHT protocol.

For measuring session lengths, one has to determine both t_j and t_l. Our measurement is based on BitMON's sampling process. To estimate t_j, BitMON compares a just collected sample with the next older one. Peers that are found within the younger sample but not in the older one are assumed of having joined during one of both crawls. During our measurements this was the case for an average number of 22348 peers per crawl. For each of those peers, t_j is set to the starting time of the crawl that produced the younger sample. All other peers are dropped. The inaccuracy of our estimation of t_j is within $\pm\,[0, d)$, with d as the duration of a single crawl. During our measurements, d was 36 min. For determining t_l, BitMON continuously sent Kademlia PING requests to the remaining peers. Per second, 10 peers were contacted. If a peer did not answer on 3 consecutive requests, t_l was set to the timestamp of the last received response. For an average number of 22348 peers the maximum delay between two PING requests sent to the same peer was $22348/10s^{-1} \approx 37$ min. The inaccuracy of our estimation of t_l is thus $[0, 37)$. However, for long-living peers this number is much smaller (for instance halved for peers living for 2.5 hours), as fewer peers needs to be contacted at later stages. The overall inaccuracy of t_s for a single peer is thus below $[0, 37) \pm [0, d)$. Hence, our estimation of the *average* session length is exceeding the real value by at most 37 min. Deviation is even smaller for longer session lengths.

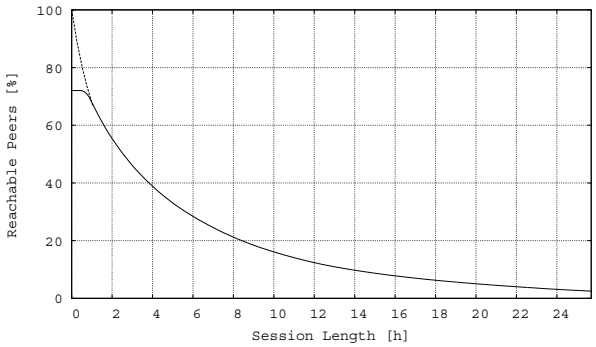


Fig. 2. Complementary cumulative distribution function (CCDF) of the session lengths of peers.

Figure 2 depicts the complementary cumulative distribution function (CCDF) of the peers' average session length. The shown plot is based on 173 runs taken between January and March 2011. As BitMON does not start with sending PING requests until the crawl necessary for peer detection finishes (that is $d = 36$ min after t_j), shorter session lengths are falsely detected as 0 min. To avoid confusion, we hinted the most likely run of the shown curve for the missing data by adding a dashed line.

Our experiments show that 59.1% of peers have session lengths of higher than 100 min. 50% of peers have a session length of more than 153 min while 5% of peers are participating for more than 20 h. Session lengths hence prove to be heavy tailed. Within the KAD network, Steiner et al. found ratios between 49% and 79% for a session length of more than 100 min, and ratios between 5% and 9% for session lengths of more than 20 h, depending on the used method of data cleaning [21]. This similarity between different DHTs shows that the chosen metrics can indeed be seen as independent from the particular DHT protocol (compare also [26]).

In summary, we observed a high amount of peers with rather short session lengths and a small amount of peers with very long session lengths. This has two direct implications for a basic DHT service: First, information stored within the DHT either has to be replicated or transferred to ensure long-time accessibility. In [28] an analytical study on the impact of data replication is given. Second, the DHT protocol should try to identify and favor long-living peers for data storage to decrease overhead caused by data replication.

E. Peer Origin

In the following we present the composition of the DHT according to the peers' origin. We mapped the peers' IP addresses to its countries of origin using the Maxmind GeoIP database [29]. Table 3(a) lists what ratio of peers originated from specific continents during September 2010 to March 2011. With 38.5%, Europe is the most dominant continent within the DHT.

In Figure 3(b), we plot the diurnal fluctuation of the number of peers by their continent of origin. It can be seen that the peak time is about 7 pm CET for European peers, 7 pm EST for American peers and 9 pm CST for Asian peers. The

fluctuation is strongest for the European continent because of the high population density and the limited number of different time zones.

The results show that the peers of a DHT mostly driven by file sharing would be fairly well distributed over Asia, Europe and America. Thus, the DHT's operability would not depend on the technical infrastructure of a few countries alone. However, the number of peers per inhabitant of a specific region would by far be highest in Europe. Furthermore it can be seen that DHT applications that try to harness locality would have to be able to cope with very strong diurnal fluctuations, as for instance the number of European peers increases by almost 400% during the day. If e.g. information needs to stay available during the night, an application might have to move data regularly.

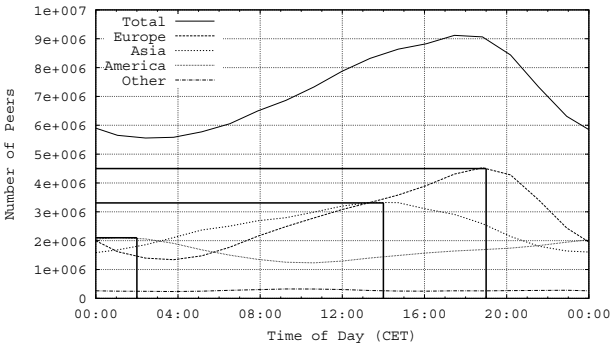
Most of our measurements were run from the Steinbuch Centre for Computing (SCC) in Karlsruhe, Germany. Because of the high ratio of found peers coming from Europe one could ask whether this observation might have differed if made from another country. To verify this, we reran our measurement from a PlanetLab [32] node located at the DePaul University, Chicago, USA. Figure 4 shows that the graphs match perfectly. Thus, we assume our results of not being spoiled in this aspect.

F. Guarded Hosts

The performance of DHT lookups is heavily influenced by the probability of RPCs to fail. One important reason for failing RPCs is the number of *guarded hosts*. Guarded hosts are defined as peers being located behind a NAT gateway or firewall [27]. Hence guarded hosts might be unable to receive packets from peers they did not sent a packet to before, depending on their gateway or firewall configuration. In this paper, we will differentiate between the NAT gateway variations *full cone*, *restricted cone*, *port restricted cone* and *symmetric* as defined in [17].

Continent	Ratio [%]
Europe	38.50
North and South America	23.36
Asia	34.55
Other	3.59

(a) Ratio of peers per continent found from November 09 till December 01.



(b) Peak DHT participation by continent.

Fig. 3. Analysis of peer origin by continent.

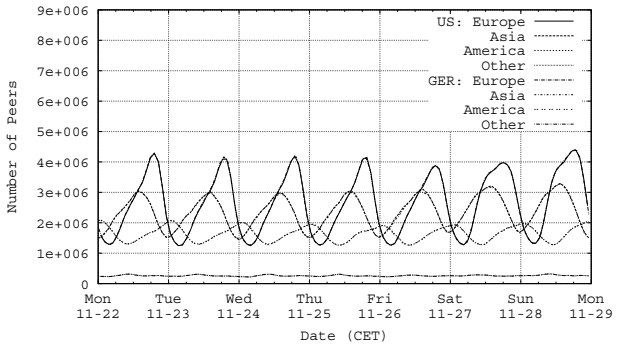


Fig. 4. DHT composition measured from Germany and the USA.

For measuring the number of guarded hosts we used a similar experiment design as described in [10]: our measurement engine had access to 3 different UDP sockets A_1, A_2 and B, where A_1 and A_2 were bound to the same IP address, but different ports. Socket B was bound to a second IP address. For becoming popular among other peers, we were continuously sending FIND_NODE queries to peers that were returned by previous requests. We limited our sending rate to a maximum of 1000 queries per second. Whenever a packet coming from a peer that had not been queried yet was received on socket A_1, 3 PING queries were sent from each of our 3 sockets (9 queries in total). Each packet was sent after a delay of 30 sec. We then categorized the peer depending on which of our sockets we received replies.

Table I summarizes the findings we made during our measurements running from September 2010 to March 2011. Each run took about 1 hour. During one run an average number of 2745 peers were categorized. The found amount of peers behind NAT gateways (IP and IP_AND_PORT category) matches very well the observations made in [10]. However, we found significantly more fully accessible peers (48.3% over 38.2%) but significantly less peers that did not respond (6.2% over 18.2%).

Our results show that around 50% of peers suffer from limited connectivity, compromising DHT lookup performance. Whereas this is not a significant problem for file sharing purposes, lookup performance is of much higher importance for most other use cases. A high percentage of guarded hosts might thus limit the applicability of many DHT-based applications. Hence, countermeasures should be taken: in some cases NAT gateways could be bypassed using hole punching techniques as described in [6], [8]. Additionally, the DHT protocol could optimize its performance by excluding guarded hosts, favoring fully accessible peers [3] or use a higher degree of parallel requests to speed up lookups [7]. In [10] some more recommendations for dealing with guarded hosts are given.

It should also be noted that although we did not observe any significant variation during our measurements, the ratio of guarded hosts might significantly change in the future, for instance with the advent of IPv6.

Category	Replied to	Ratio [%]	Explanation (see also [17])
FULLY_ACCESSIBLE	A_1, A_2, and B	48.3	Peer is directly connected to the internet or behind a *full cone* NAT gateway.
IP	A_1, and A_2	4.4	The peer is behind a *restricted cone* NAT gateway.
IP_AND_PORT	A_1	38.2	The peer is behind either a *symmetric* or *port restricted cone* NAT gateway.
FIREWALL	None.	6.2	The peer is behind a firewall.
OTHER	Any other combination.	2.8	May occur due to packet loss.

TABLE I
GUARDED HOST DISTRIBUTION.

Client name	Avg. lookup duration [s]	Packets sent per lookup	Observed lookups
Transmission 2.12	250.1	59	368
Mainline Client 5.2.2	78.5	6927	396

TABLE II
LOOKUP PERFORMANCE OF DIFFERENT DHT IMPLEMENTATIONS.

G. Lookup Performance

The most elementary service provided by a DHT is to provide lookups for key/value pairs. File sharing clients typically require only one lookup to access a file while file downloads typically take at least minutes. Thus, DHT lookups typically neither need to be fast nor efficient for file sharing purposes. However, DHT lookup performance is a key metric for most other DHT-based applications. For instance, lookups need to be as fast as possible considering a DHT-based name service as described in [1] or the traffic information system proposed in [20]. Thus, providing fast yet efficient lookups is an important requirement for an open DHT platform. In the following, we will briefly analyze the performance of DHT lookups within the BitTorrent Mainline DHT.

Contrary to all other metrics present in this paper, DHT performance is not only influenced by network characteristics but more significantly depends on the DHT protocol. As we will see, even different client implementations participating within the same DHT vastly differ in terms of performance. Among others, implementations vary in the duration they wait for answers to a sent request or the number of queries sent in parallel. Unfortunately, for most available DHT clients no details about the lookup implementation are known as the client's source code is not publicly available. Moreover, it is hard to measure the lookup's performance, as the only chance is reverse engineering the client's network traffic. Thus, we present the performance of two publicly available open source DHT implementations.

Table II summarized our results for Transmission 2.12 [33] and the BitTorrent Mainline Client V.5.2.2 [30]. For our measurement, we let each implementation lookup almost 400 random IDs. Our results show vast differences in both lookup duration and the number of packets sent per lookup: the Mainline Client performs 3 times faster at the cost of more than a 100 times more sent packets. Whereas the performance of neither of the two examined implementations would suffice the needs of a basic DHT service, it can also be seen that client optimization significantly affects lookup performance. Thus, a DHT implementation of a basic DHT service could improve its overall performance by optimizing its lookup algorithm.

In academic research, some studies exist that aim at optimizing lookup performance in other Kademlia-based DHT networks: in [7] the authors achieve lookup durations of only 13 seconds within the Azureus network at moderate costs in terms of overhead while in [25] lookup durations of 2 seconds are achieved within the KAD network. However, as these measurements took place within different networks they are not directly comparable. Still it is shown that short lookup durations can be achieved in widely deployed DHTs.

V. DISCUSSION

Although our results suggest a public, fully decentralized DHT platform to be viable there clearly remain issues to be solved. In this section, we will briefly discuss some of the most important issues.

Whereas a fully decentralized DHT platform could to some extend be seen as a replacement of the OpenDHT project it entails the new challenge of unreliable peers: peers may not only fail at any time due to short session lengths, failure or connectivity issues but they might also be malicious, impairing the DHT's performance and stability or manipulating stored data. As trustworthiness, stability and performance are key for a public DHT platform, ensuring security would thus be a much more important requirement than it is today for DHT implementations primarily used for file sharing. A comprehensive analysis of current security problems in P2P networks and how to avoid them is given in [4].

Our study is based on the assumption that file sharing will stay the DHT platform's main use case for some time before more sophisticated applications become more dominant. When this happens (assuming those applications would ship with an own copy of the DHT client), end user behavior could significantly shift, depending on the respective application. For instance, session lengths might increase if the higher level application offers a service the user prefers to be always available. Applications that are only popular in certain parts of the world might change the geographic distribution of peers. Furthermore, effects induced by different applications would overlap, further influencing the DHT. In one possible asymptotic scenario every machine would include a "DHT stack" as they today include an IP stack. However, it is clear that the possible future of a global DHT platform would be hard to predict. Thus, continuous monitoring would be essential to timely adapt to potential changes.

VI. CONCLUSION

In this paper, we addressed the question whether a general purpose DHT could potentially serve as a stable foundation for

DHT-based applications if deployed similar to those widely deployed DHTs today used for file sharing purposes only. Therefore we analyzed key characteristics of the BitTorrent Mainline DHT such as the number of concurrently participating peers, the peers' origin, and connectivity properties. As these characteristics depend on end user behavior rather than on properties of the DHT protocol we assume them to apply as well to any other DHT deployed in a similar way. For our study we continuously monitored the BitTorrent Mainline DHT for a period of 6 months.

Overall, we found that a file sharing driven DHT could constitute a reliable foundation service: we did not observe any unforeseen breakdowns during our measurements, and the peers were rather well distributed over Asia (34.55%), America (23.36%) and Europe (38.50%). However, a public DHT platform would have to cope with strong diurnal fluctuations of the DHT size, which might be problematic for use cases that require data to be stored persistently over a longer time. This issue is intensified for locality aware applications, as for instance the number of European peers increases by almost 400% during the day. Also, with 50% of peers joining the DHT for less than 153 min, session lengths of most peers are rather short whereas only 5% of peers are participating for more than 20 h. It can thus be worthwhile to identify and favor long-living peers for tasks such as data storage. Furthermore, dealing with limited connectivity peers can be seen as essential, as around 50% of all peers are flawed in that regard.

One might argue that users may not be willing to participate in a generic DHT service used by arbitrary applications. However, users of today's file-sharing clients already participate in DHTs used for locating peers holding data unknown to the queried client's user. Nevertheless, questions of user control and trust have to be addressed in the future. As a next step towards a public DHT platform we are working on enabling DHT lookups to dynamically adapt to changing user behavior. We will also continue monitoring the DHT and present the results on our website.

REFERENCES

[1] I. Baumgart. P2PNS: A Secure Distributed Name Service for P2PSIP. In *Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications*, pages 480–485, 2008.

[2] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak. Operating System Support for Planetary-Scale Network Services. In *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1*, pages 19–19, 2004.

[3] A. Brampton, A. MacQuire, I. A. Rai, N. J. P. Race, and L. Mathy. Stealth Distributed Hash Table: a Robust and Flexible Super-Peered DHT. In *Proceedings of the 2006 ACM CoNEXT conference*, CoNEXT '06, pages 19:1–19:12, 2006.

[4] D. Chopra, H. Schulzrinne, E. Marocco, and E. Ivov. Peer-to-Peer Overlays for Real-Time Communication: Security Issues and Solutions. *Communications Surveys Tutorials, IEEE*, 11(1):4–12, 2009.

[5] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area Cooperative Storage with CFS. In *ACM SIGOPS Operating Systems Review*, volume 35, pages 202–215, 2001.

[6] J. Dinger and O. P. Waldhorst. Decentralized Bootstrapping of P2P Systems: A Practical View. In *NETWORKING 2009*, pages 703–715, 2009.

[7] J. Falkner, M. Piatek, J. P. John, A. Krishnamurthy, and T. Anderson. Profiling a Million User DHT. In *Proc. of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, pages 129–134, 2007.

[8] B. Ford, P. Srisuresh, and D. Kegel. Peer-to-Peer Communication Across Network Address Translators. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, pages 13–13, 2005.

[9] M. J. Freedman, E. Freudenthal, and D. Mazières. Democratizing Content Publication with Coral. In *Proc. of the 1st Conf. on Symposium on Networked System Design and Impl. - Volume 1*, pages 18–18, 2004.

[10] R. Jimenez, F. Osmani, and B. Knutsson. Connectivity Properties of Mainline BitTorrent DHT nodes. In *IEEE Ninth International Conf. on Peer-to-Peer Computing, (P2P '09)*, pages 262–270, 2009.

[11] K. Jünemann, P. Andelfinger, J. Dinger, and H. Hartenstein. Demo: BitMON: A Tool for Automated Monitoring of the BitTorrent DHT. In *Proceedings of the IEEE International Conference on Peer-to-Peer Computing (IEEE P2P'10)*, 2010.

[12] J. Li, J. Stribling, T. M. Gil, R. Morris, and M. F. Kaashoek. Comparing the Performance of Distributed Hash Tables Under Churn. In *Lecture Notes in Computer Science*, 2005.

[13] A. Loewenstern. BitTorrent BEP5: DHT protocol. http://www.bittorrent.org/beps/bep_0005.html, 2008.

[14] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. *IEEE Communications Surveys and Tutorials*, pages 72–93, 2005.

[15] P. Maymounkov and D. Mazires. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *P2P Sys.*, pages 53–65, 2002.

[16] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. OpenDHT: a Public DHT Service and its Uses. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '05, pages 73–84, 2005.

[17] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. RFC3489 - STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). http://tools.ietf.org/html/rfc3489, March 2003.

[18] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proc. 18th IFIP/ACM Int. Conf. Distributed Systems Platforms (Middleware 2001)*, pages 329–350, 2001.

[19] A. Rowstron and P. Druschel. Storage Management and Caching in PAST, a Large-Scale, Persistent Peer-to-Peer Storage Utility. In *ACM SIGOPS Operating Systems Review*, volume 35, pages 188–201, 2001.

[20] J. Rybicki, B. Scheuermann, M. Koegel, and M. Mauve. PeerTIS: a Peer-to-Peer Traffic Information System. In *Proc. of the 6th ACM Intern. Work. on Vehicular InterNETworking*, VANET '09, pages 23–32, 2009.

[21] M. Steiner, T. En-Najjary, and E. Biersack. Long Term Study of Peer Behavior in the KAD DHT. *IEEE/ACM Transactions on Networking*, 17(5):1371–1384, 2009.

[22] M. Steiner, T. En-Najjary, and E. W. Biersack. A Global View of KAD. In *IMC '07: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, pages 117–122, 2007.

[23] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *SIGCOMM Comput. Commun. Rev.*, 31:149–160, 2001.

[24] D. Stutzbach and R. Rejaie. Evaluating the Accuracy of Captured Snapshots by Peer-to-Peer Crawlers. In *Passive and Active Network Measurement*, volume 3431, pages 353–357, 2005.

[25] D. Stutzbach and R. Rejaie. Improving Lookup Performance Over a Widely-Deployed DHT. In *Proc. of the 25th IEEE International Conf. on Computer Communications (INFOCOM 2006).*, pages 1–12, 2006.

[26] D. Stutzbach and R. Rejaie. Understanding Churn in Peer-to-Peer Networks. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC '06, pages 189–202, 2006.

[27] W. Wang, H. Chang, A. Zeitoun, and S. Jamin. Characterizing Guarded Hosts in Peer-to-Peer File Sharing Systems. In *Global Telecommunications Conference. GLOBECOM '04*, volume 3, pages 1539–1543, 2004.

[28] D. Wu, Y. Tian, and K.-W. Ng. Analytical Study on Improving DHT Lookup Performance under Churn. In *Sixth IEEE Intern. Conference on Peer-to-Peer Computing (P2P 2006)*, pages 249–258, 2006.

[29] GeoIP Website. http://www.maxmind.com/app/ip-location, 2011.

[30] Mainline Client. http://download.bittorrent.com/dl/archive/, 2011.

[31] Official Azureus Website. http://azureus.sourceforge.net/, 2011.

[32] Official PlanetLab Website. http://www.planet-lab.org/, 2011.

[33] Official Transmission Website. http://www.transmissionbt.com/, 2011.