Actively Monitoring Peers in KAD

Moritz Steiner and Ernst W. Biersack and Taoufik Ennajjary
Institut Eurécom, Sophia–Antipolis, France
{steiner, erbi, ennajjar}@eurecom.fr

Abstract

In recent years, a large number of DHTs have been proposed. However, very few of them have been deployed in real-life large scale systems. An exception is KAD, a DHT based on Kademlia that is part of the widely used eMule peer-to-peer system, which has more than 1.5 million simultaneous users. We have developed a very fast crawler and explored KAD for more than a month by crawling part of the KAD ID space, which takes only a few seconds. We find that there are *two classes* of peers, long-lived peers that participate in KAD for weeks and short-lived peers that remain in KAD no more than few days before they permanently leave. Most of the peers that join KAD for the first time are short-lived peers. Since inter-session times can be as large as one week, one needs to crawl KAD for more than a week to be able to identify and characterize short-lived peers.

1 Introduction

KAD is a Kademlia-based [5] peer-to-peer DHT routing protocol implemented by several peer-to-peer applications such as Overnet [8], eMule [3], and aMule [1]. The two open–source projects eMule and aMule do have the largest number of simultaneously connected users since these clients connect to the Edonkey network, which is a very popular peer-to-peer system for file sharing. Recent versions of these clients implement the KAD protocol. In the rest of the paper when we use the term KAD we refer to the aMule and eMule implementation.

Similar to other DHTs like Chord [11], Can [9], or Pastry [10], each KAD node has a global identifier, referred to as KAD ID, which is 128 bit long and is randomly generated using a cryptographic hash function. The KAD ID is generated when the client application is started for the first time and is then permanently stored. The KAD ID stays unchanged on subsequent join and leaves of the peer, until the user deletes the application or its preferences file. Therefore, using the KAD ID a particular peer can be tracked even after a change of its IP address.

Routing in KAD is based on prefix matching: Node a forwards a query destined to a node b to the node in his routing table that has the smallest XOR-distance. The XOR-distance $d(a, b)$ between nodes a and b is $d(a, b) = a \oplus b$. It is calculated bitwise on the KAD IDs of the two nodes, e.g. the distance between $a = 1011$ and $b = 0111$ is $d(a, b) = 1011 \oplus 0111 = 1100$. The fact that this distance metric is symmetric is an advantage compared to CAN or Chord, since in KAD if a is close to b, then b is also close to a. Thus, both nodes can add each other in their respective routing tables. Without such a symmetry, peers would not learn useful routing information from the queries they receive.

The entries in the routing tables are called *contacts* and are organized as an unbalanced *routing tree*. Each contact consists of the node's KAD ID, IP address, TCP and UDP port. The *left* side contains contacts that have no common prefix with the node a that owns the routing tree (XOR on the first bit returns 1). The *right* side contains contacts that have at least one prefix bit in common. This tree is highly unbalanced and the right side of each tree node is (recursively) further divided in two parts, containing on the left side the contacts having no further prefix bit in common, and on the right side the contacts having at least one more prefix bit in common. A *bucket* of contacts is attached to each leaf of the routing tree. Each bucket can contain up to ten contacts in order to cope with peer churn without the need to periodically check if the contacts are still online. In summary, a node a node stores only a few contacts to peers that are far away in the overlay and increasingly more contacts to peers as we get closer a. For details of the implementation see [12].

For routing, a message is simply forwarded to one of the peers from the bucket with the longest common prefix to the target. Routing to a specific KAD ID is done in an iterative way, which means that each peer on the way to the destination returns the next hop to the sending node. While iterative routing experiences a slightly higher delay than recursive routing, it offers increased robustness against message loss and it greatly simplifies crawling the KAD network.

2 Related Work

Overnet was the first widely deployed DHT. Its implementation is proprietary. The operation of Overnet was discontinued in September 2006. The Overnet peer-to-peer network has been crawled by [2, 4, 7] and most 265,000 concurrent users have been seen online. The study most relevant to our work is the one by Bhagwan et al. [2]. A set of 2,400 peers was contacted every 20 minutes during two weeks. This study pointed out the *host aliasing problem* that is due to the fact that many peers periodically change their IP address. So, in order to properly compute session times and other peer-specific metrics one needs to use the KAD ID. This study also indicated that for systems where peers leave permanently, the values for the mean peer availability decrease as the observation period considered increases.

KAD is the first widely deployed open-source peer-to-peer system relying on a DHT. Two very interesting studies on KAD have recently been published by Stutzbach. The first explains the implementation of Kademlia in eMule [12] and the other [13] compares the behavior of peers in three different peer-to-peer systems, one being KAD. The results obtained for KAD are, as in our case, based on crawling a subset of the KAD ID space. We refer to a *k-bit zone* as a continuous sub-set of the total KAD ID space that contains the all KAD peers whose KAD IDs agree in the first k bits. Stutzbach [13] crawled a 10-bit zone in 3-4 minutes and a 12-bit zone in approximately one minute. A total of 4 crawls were carried out. Each crawl lasted for only 2 days, which implies that the maximum values for some metrics such as session up-times or inter-session times that can be observed are naturally limited.

3 Measurement Methodology

3.1 Introduction

The independent arrival and departure of peers is called *churn* [13]. Churn is a major challenge for peer-to-peer systems as it affects the stability of the whole system and the availability of peers and shared objects. Churn also makes crawling a peer-to-peer system difficult: to get a consistent snapshot of the system at a given point in time, one needs to execute a given crawl in as little time as possible. For this reason, we have built a very fast custom crawler, called *Blizzard*, which is able to crawl a zone with an 8-bit prefix, i.e. approximately one 256-th of the entire peer set of KAD in less than 2.5 seconds. Given the speed of Blizzard, we were also able to crawl the entire KAD ID space.

Stutzbach [13] describes very well the various pitfalls when crawling a peer-to-peer system such as incomplete data due to crawler crashes, loss of network connectivity, or random failures due to temporary network instability. To address these problems, we run simultaneously two instances of Blizzard, one at the University of Mannheim, Germany connected to the German research network, and a second one at Institut Eurécom, France connected to the French academic network. This redundancy in crawling turned out to be useful: at some point due to network problems, one instance of the crawler was seeing fewer clients than the other one. We take the raw data of the two parallel crawls and merge them in such a way that, for a given round of the crawl, a peer is considered up when he has been seen by at least one crawler.

3.2 Crawling

Previous crawlers, such as the one by Stutzbach [13] are distributed and run simultaneously on multiple machines. We noticed that in a distributed crawl a lot of CPU time is used up for the synchronization between the machines. The main idea of Blizzard is to use *only one machine*, and keep all relevant information in main memory. After the crawl is completed, the results are written to disk. The implementation of Blizzard is straightforward: Blizzard runs on a local machine knowing several hundred contacts to start with. It uses a simple breadth first search and iterative queries: It first queries peers among its contacts in order to get to know more peers and so on.

4 Results for a Full Crawl

Since crawling the entire KAD ID space is not necessary for most of the metrics we are interested in, we stopped doing so after two weeks. However, the full crawl enabled us to validate that the results computed from the partial crawl are statistically significant. Also, the full crawl allowed us to make certain observations that would not have been possible with a partial crawl.

During a full crawl, we see between between 1.5 and 2 million peers. This number varies according to a diurnal as well as a weekly pattern and reaches its peak during the weekend, where the population is about 10% higher as compared to a week day.

4.1 Distribution over the hash space

A good hash function should assure a uniform distribution of the KAD IDs over the entire KAD ID space. We see that the peers are indeed uniformly distributed over the hash space, except for some outliers (Figure 1). An 8-bit zone contains the peers whose KAD ID agrees in the first 8 bits, thus one zone can theoretically contain 2^{120} hash values, while we actually observe between 12,000 and 20,000 peers per zone. For some zones we see a much higher number of peers, which are due to modified KAD clients. The modified KAD clients with the same KAD ID are always limited to a single country (Korea, Spain, Israel, China, Argentina). The outlier in zone `0xe1` is a modified client used in Israel, for which we counted more than 10,000 instances. Using modified KAD clients amounts for some sort of free-riding, since the load for publishing and forwarding that a single regular KAD peer carries will be distributed among all the modified peers with the same KAD ID. On the other hand, modified clients can use the peer-to-peer network like regular one. If there were too many modified KAD clients, the application performance would decrease, since the peers using the same KAD ID could not see one another and therefore could not download one from another.

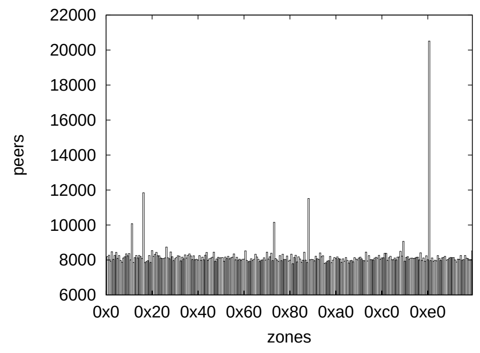

Figure 1: The distribution of the peers over the hash space. The 256 8-bit zones on the x-axis go from `0x00` to `0xff`.

4.2 Aliasing and Artifacts

In many cases, ISPs attribute a new IP address to their DSL subscribers approximately every 24 hours. Since the KAD ID does not change, it is possible to track individual peers even when they change their IP address. Over a period of two weeks, we saw thousands of peers with the same KAD ID but twenty or more different IP addresses.

During our crawls, we saw some anomalies such as a large number (more than one hundred) of peers with the same KAD ID and/or the same IP address. (i) Peers with the same KAD ID and the same IP address: a contiguous range of port numbers was used, which in our opinion hints to a single organization running many instances of a KAD peer to monitor the KAD overlay. (ii) Peers with the same IP address and different but carefully chosen KAD IDs: The KAD IDs have some kind of pattern indicating that they have been chosen "by hand". This also hints to a single organization running many instances of a KAD peer. (iii) Peers with the same IP address but different randomly chosen KAD IDs and ports: These can be different clients behind a NAT sharing the same public IP address.

4.3 Geographic Distribution and Total Population

In Figure 2, we plot the distribution of the percentage of peers seen per country. Almost 25% of the peers are located in China and the geographic region with the highest percentage of peers is Europe (Spain, France, Italy and Germany). Less than 15% of all peers are located in America (US, Canada, and South America). In total, peers from about 200 countries have been reported. We can also see that the results obtained with two partial crawls of 8-bit zones are very close to the ones obtained with the full crawl.

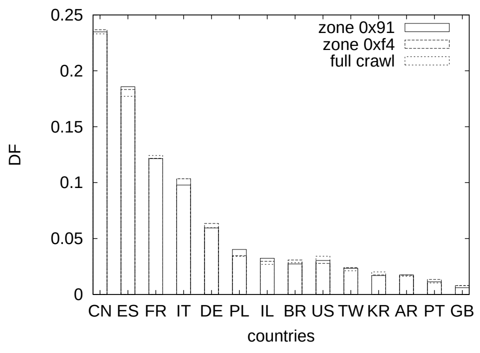

Figure 2: The geographic distribution of peers on 2006/08/30 of the entire KAD ID space compared to those of different 8-bit zones.

We have previously shown that the number of peers is uniformly distributed over the hash space. This property will allow us to estimate the total number of peers in the system by simply counting the number of peers in a zone. If we do a partial crawl of one 256-th of the entire KAD ID space we can use Chernoff Bounds (see [6] Chapter 4) to estimate the total population size and tightly bound the estimation error.

Let $N(t)_{part}$ be the number of peers counted during a partial crawl of an 8–bit zone at time t and $\hat{N}(t) := 256 * N(t)_{part}$ the estimate for the total number of peers in the KAD system. The true value $N(t)$ for total number of peers at time t is very close to the estimate $\hat{N}(t)$, with high probability. More precisely:
Prob$(|N(t) - \hat{N}(t)| < 45000) \geq 0.99$, which means that our estimate $\hat{N}(t)$ has most likely an error of less than 3% for a total population of at least 1.5 million peers.

5 Results from Partial Crawls

5.1 Introduction

When Blizzard crawls an 8-bit zone it contacts between 12,000 and 20,000 peers and receives replies from approximately 5,000 to 8,000 peers. The ones that did not reply are either peers that have left (for which the routing table entries are stale) or peers behind a NAT. The total network traffic generated during a single round of the crawl is 19 – 26 MByte inbound and 15 – 19 MByte outbound.

We have been crawling the 8-bit zone `0x5b` every 5 minutes for over one month. During the very first crawl, we saw 5795 peers. During the remaining 287 crawls of the first day we saw another 12297 different peers, which add up to a total of 18092 different clients for the first day. Every day new peers arrive that we never saw before. In Figure 3 the number peers that have never be seen before by the crawler is plotted. We can clearly identify the first week-end (days 3 and 4) where more new peers arrive. Since these peers will then be already known on the following weekends, the number of new peers on the consecutive week-ends is less important.

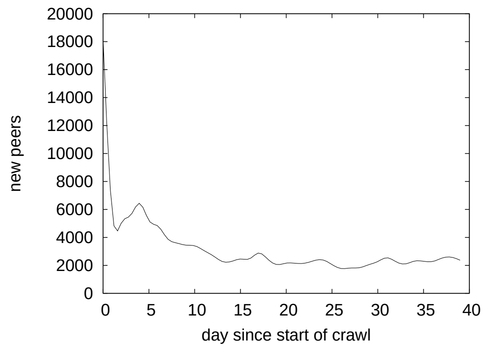

Figure 3: The number of KAD peers never seen before in zone `0x5b` per day.

In the following, with "peers that arrived on the n-th day" we mean "peers that have been seen *for the first time* during a crawl on the n-th day". Please note that the total number of peers seen during the crawls on the n-th ($n > 1$) day consists of the peers that have already been seen on previous days plus the ones that have been seen for the first time on day n.

We will also use the following terminology: *up from second crawl* means all the peers sampled during crawls 2–288 on the first day (the very first crawl is excluded), while *n-th day* means the peers seen the first time during one of the crawls on day n. To compute the metrics of interest, we will use data obtained by crawling over a period of either 23 or 38 days.

5.2 Session Length

Most of the peers will not be online, i.e. connected to KAD, all the time. By crawling KAD we can determine for each peer k the instances $t_1^j(k), ..., t_n^j(k)$ when k joined and the instances $t_1^l(k), ..., t_m^l(k)$, with $m = n - 1$ or $m = n$, when k has left KAD.

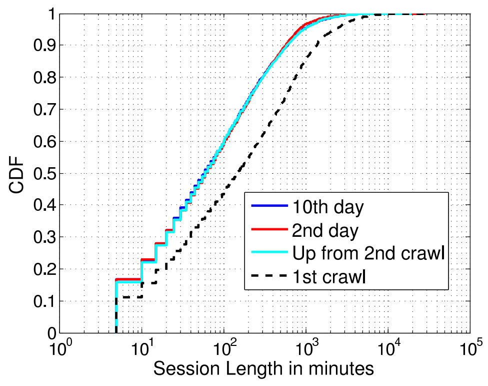

Figure 4: CDF of the Session length of peers.

In Figure 4 we plot the distribution of the **session length**, which is defined as the time a peer was present in the system without any interruption, i.e. $t_i^l(k) - t_i^j(k)$ for $i \in \{1, ..., m\}$. We can notice that the session length of the peers seen in the first crawl follows a distribution that is different from the distribution of the peers seen during later crawls. Also, the mean session time and its standard deviation are more than twice as large for the peers seen during the first crawl than for the peers seen during crawls 2-288. In fact, when we sample KAD for the first time, we have a much higher chance to see peers that are connected "most of the time" than peers that are connected from time to time and only for short periods. This means that a single snapshot of the system cannot give a representative picture of the characteristics of the peers: to improve the quality of the measurements, we need to sample the system many times.

We did a distribution fitting for the session times and found that the Weibull distribution provides a very good fit .

5.3 Inter-Session Time

The **inter-session time** is defined as the time a peer k is continuously absent from the system, i.e. $t^j_{i+1}(k) - t^l_i(k)$ for $i \in \{1, ..., n\}$. Figure 5 depicts the **complementary cumulative distribution function** (CCDF) of the inter-session times. As was already the case for the session times, the distribution of the inter-session times of the peers seen during the first crawl is different from the peers seen during later crawls. Also, the mean inter-session time for the peers seen during the first crawl is slightly smaller (697 min. vs. 877 min.). For the inter-session times we could not find a distribution that was matching well our observed data. In particular, the Weibull distribution did not fit.

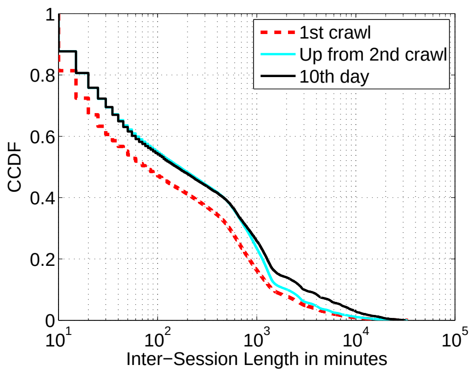

Figure 5: CCDF of the Inter-session time of peers.

A closer look at Figure 5 reveals that only very rarely (less than 1%) the inter-session time is larger than 10,000 minutes, which corresponds to 7 days. We will therefore use a *threshold of 7 days* to determine which peers have newly joined KAD and which ones have permanently left: a peer that (i) has been seen for the first time during a crawl at day 8 or later has **newly joined** KAD and a peer that (ii) has left KAD and has not re-joined for at least 7 days has **permanently departed**.

5.4 Permanent Departure

In Figure 6 we plot the fraction of peers that have permanently left the system T days after they were first seen. This figure shows that there are two classes of peers: One class of **long-lived** peers that stay in KAD over weeks and whose rate of permanent departure is very low (most of the peers first seen on the first and second day of the crawl are long-lived). A second class of **short-lived** peers that join KAD but soon leave permanently. Among the peers that newly joined KAD 70% of them had permanently left within 5 days and less then 20% stay for 15 days or more. More than 40% of these peers connected to KAD only once. If we extrapolate from these observations onto the entire KAD system we arrive at impressive numbers: About 750,000 new peers join KAD every day for the first time and over 500,000 of them will not stay for more than a few days.

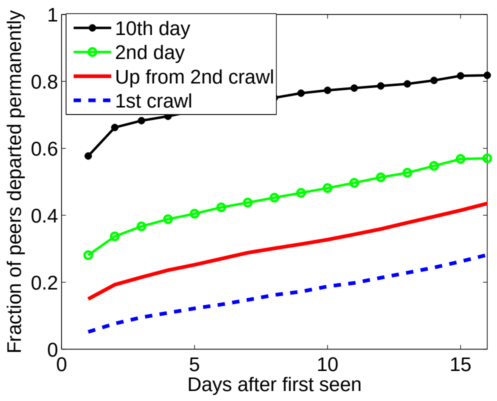

Figure 6: Fraction of peers that permanently left KAD.

In Figure 7 we plot the evolution of the number of peers connected (referred to as **peers up**). The peers that arrived for the first time at day 10 are mostly short-lived and will soon leave permanently, which entails a dramatic decrease in the number of peers within the first few days of their first appearance. After about 5 days the number of peers up decreases by very little. In fact, the peers that still participate in KAD are in fact long-lived peers that will stay for weeks. Every day a small percentage of the peers that join for the first time are long-lived peers, which allows to slowly "renew" the set of long-lived peers and maintain its overall size at a constant level.

5.5 Practical Implications

If we want to reliably store information on KAD, how should we select the peers to use? Figure 8 indicates that a policy selecting nodes based their **age**, i.e. how long ago they have joined KAD for the *first time*, gives much better performance than simply selecting peers at random. When we pick $k = 50$ peers that have joined KAD for the

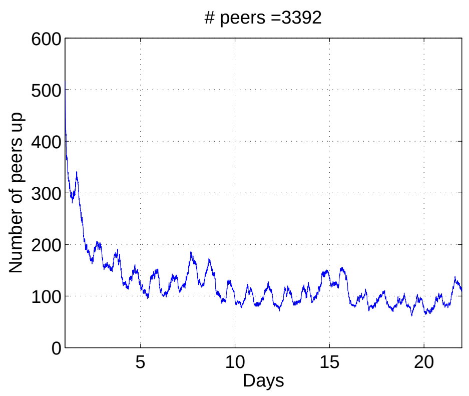

Figure 7: Evolution of the number of peers up that newly joined KAD at day 10.

first time at least 2 days ago (Exist $>$ 2 days), there is 90% chance that 22 days later at least 10 of these peers will be simultaneously up, while if we choose the peers purely at random, this chance will be less than 25%. We also tried selecting peers based on uptime, i.e. the time elapsed since the peer *last joined* KAD, but did not see much an improvement as compared to random selection.

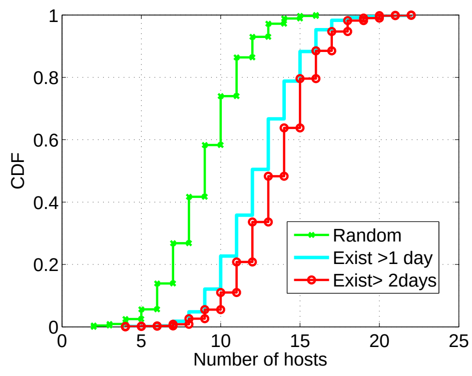

Figure 8: CDF of the minimum number of peers simultaneously available over 22 days among $k = 50$ chosen among the peers that newly joined KAD at day 10.

6 Conclusion

We have reported some of our findings obtained from crawling KAD, the largest currently deployed DHT. Results on the peer arrival and departure process, peer availability, remaining uptime, and correlation between consecutive session lengths could not be presented here for space reasons.

Our full crawl allowed us to identifiy some occurences of abnormal behavior, which have not been reported before and which are caused by modified KAD clients.

Using the observations of a partial crawl that spans more than a month, we were able to unambiguously identify the peers that joined KAD for the first time and describe their behavior. We have seen that most of the KAD peers will stay only for short time, while a small percentage of peers will remain in KAD for weeks. To precisely characterize these long-lived peers and obtain the full distribution of their lifetime we need to continue crawling KAD.

References

[1] A-Mule. `http://www.amule.org/`.

[2] R. Bhagwan, S. Savage, and G. Voelker. Understanding availability. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, pages 256–267, 2003.

[3] E-Mule. `http://www.emule-project.net/`.

[4] K. Kutzner and T. Fuhrmann. Measuring large overlay networks - the overnet example. In *Proceedings of the 14th KiVS*, Feb. 2005.

[5] P. Maymounkov and D. Mazieres. Kademlia: A Peer-to-peer informatic system based on the XOR metric. In *Proc. 1^{st} IPTPS*, pages 53–65, March 2002.

[6] M. Mitzenmacher and E. Upfal. *Pobability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge Press, 2005.

[7] N. Naoumov and K. Ross. Exploiting p2p systems for ddos attacks. In *International Workshop on Peer-to-Peer Information Management*, May 2006.

[8] Overnet. `http://www.overnet.org/`.

[9] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. ACM SIGCOMM*, 2001.

[10] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale Peer-to-peer systems. In *Proceedings of Middleware*, Heidelberg, Germany, November 2001.

[11] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-to-peer lookup service for Internet applications. In *Proceedings of SIGCOMM*, pages 149–160, San Diego, CA, USA, 2001. ACM Press.

[12] D. Stutzbach and R. Rejaie. Improving lookup performance over a widely-deployed DHT. In *Proc. Infocom 06*, Apr. 2006.

[13] D. Stutzbach and R. Rejaie. Understanding churn in peer-to-peer networks. In *Proc. Internet Measurement Conference (IMC)*, Oct. 2006.