Long Term Study of Peer Behavior in the KAD DHT

Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack
Eurecom, Sophia–Antipolis, France
{steiner,ennajjar,erbi}@eurecom.fr

***Abstract*—Distributed hash tables (DHTs) have been actively studied in literature and many different proposals have been made on how to organize peers in a DHT. However, very few DHTs have been implemented in real systems and deployed on a large scale. One exception is KAD, a DHT based on Kademlia, which is part of eDonkey, a peer-to-peer file sharing system with several million simultaneous users. We have been crawling a representative subset of KAD every five minutes for six months and obtained information about geographical distribution of peers, session times, daily usage, and peer lifetime. We have found that session times are Weibull distributed and we show how this information can be exploited to make the publishing mechanism much more efficient.**

Peers are identified by the so-called KAD ID, which up to now was assumed to be persistent. However, we observed that a fraction of peers changes their KAD ID as frequently as once a session. This change of KAD IDs makes it difficult to characterize *end-user* behavior. For this reason we have been crawling the entire KAD network once a day for more than a year to track end-users with static IP addresses, which allows us to estimate end-user lifetime and the fraction of end-users changing their KAD ID.

I. Introduction

PEER-TO-PEER systems have seen a tremendous growth in the last few years and peer-to-peer traffic makes a major fraction of the total traffic seen in the Internet. The dominating application for peer-to-peer is file sharing. Some of the most popular peer-to-peer systems for file sharing have been Napster, FastTrack, BitTorrent, and eDonkey, each one counting a million or more users. Since these systems are mainly used by home-users and since the content shared is typically copyright-protected, the users of these systems often stay connected only as long as it takes them to download the content they are interested in. As a result, the user population of these peer-to-peer systems is highly dynamic with peers joining and leaving all the time.

In this paper, we focus on a single peer-to-peer system, namely KAD, which is the publishing and search network of eDonkey. Our aim is to characterize KAD in terms of metrics such as arrival/departure process of peers, session and inter-session lengths, availability, and lifetime.

To obtain the relevant raw data we decided to "crawl" KAD. Each crawl gives a snapshot of the peers active at that instant. The three major challenges in crawling are

- Time necessary to carry out a single crawl, which should be as short as possible to get a consistent view of the system.
- Frequency of the crawls, i.e. the time elapsed between two consecutive crawls should be short (no more than a few minutes) in order to achieve a high resolution for metrics such as session length.
- Duration of the crawl, which should be in the order of many months, to be able to correctly capture the tail of the session and inter-session length distributions.

We have built our own crawler, which will be described in Section IV, that meets all three goals.

While peer-to-peer systems have been explored previously using a crawler, the duration of these crawls was limited to a few days at best. We were able to crawl a subset of KAD for six months at a frequency of one crawl every five minutes, which makes a total of 51,552 snapshots. We obtained a number of original results such as:

- Session lengths are heavy–tailed, with sessions lasting as long as 78 days and are best characterized by a Weibull distribution, with shape parameter $k < 1$. One property of Weibull distributed session lengths is that a peer that has so far been up for t units of time will – we expect – remain up for a duration that is in the order of $O(t^{1-k})$. We can exploit this fact to use the past uptime in order to predict the remaining uptime.
- For many peers, the amount of time a peer is connected per day, called daily availability, varies a lot from one day to the next. This makes it difficult to predict daily availability.
- Contrary to what was known up to now, KAD IDs are not persistent and can change as frequently as once per session. By using a subset of peers with static IP addresses we can also show that the end-user lifetime is significantly longer than the KAD ID lifetime with 50% of the peers participating in KAD for six months or more.
- When classifying peers according to their geographic origin, the peers from China make about 25% of all peers seen at any point of time, while Europe is the continent where KAD is most popular. We also see a big difference between peers in China and Europe with respect to some of the key metrics such as session length or duration of daily usage.

The remainder of the paper is organized as follows. Section II presents related work followed by a section describing KAD. Section IV presents the measurement methodology followed by two sections that contain the results. We then discuss in Section VII how some of our findings can be used to improve the performance of KAD and summarizes the major findings in Section VIII.

II. Related Work

STUDIES measuring peer-to-peer networks may have different goals, such as analyzing the traffic patterns [27], [37]; learning about the content shared in the network [11], [26], [7]; or learning about the peers, their geographical distribution [11], their latency to the measurement site or their bandwidth [14], [26], and the user behavior expressed e.g. in session times or peer availability [2], [14], [35], [26], [36], [7]. There are also different ways to measure peer-to-peer networks, which can be either passive or active. Passive techniques consist of (i) instrumenting a client that captures all the traffic [37] sent and received, (ii) analyzing the central log file such as the track log of BitTorrent [13], or capturing the traffic of a whole network, e.g. at the POP or an ISP [27], [21].

Active techniques consist in crawling the peer-to-peer system. In some systems, such as Napster or eDonkey, it is sufficient to contact the server(s), instead of every single peer [7], [26], [23]. If one wants to know all peers in a DHT, such as Overnet or KAD, it is necessary to contact every single peer and to query it for contacts in its routing tables. Here crawling is equivalent to a graph exploration. Finding all peers amounts to building the transitive closure of the graph. Examples of DHTs studies that employ crawling are [11], [23], [2], [14], [35].

Overnet was the first widely deployed peer-to-peer application that used a DHT, namely Kademlia. The implementation of Overnet is proprietary and its operation was discontinued in September 2006 after legal action from the media industry. Overnet has been the subject of several studies [2], [14] and up to 265,000 concurrent users have been seen online. In our study we use the active measurement approach and want to learn about peer behavior. One study relevant to our work is by Bhagwan et al. [2] where a set of 2,400 peers in Overnet was contacted every 20 minutes over two weeks. This study discusses the *IP aliasing problem* which results from the fact that many peers periodically change their IP address. Therefore, in order to properly compute session times and other peer-specific metrics, one needs to use a globally unique identifier for each peer.

KAD is the first widely deployed open-source peer-to-peer system relying on a DHT. Two other studies on KAD that are very relevant to our work have been published by Stutzbach and Rejaie. The first one explains in detail the implementation of Kademlia in eMule [34] and the second one [35] compares the behavior of peers in three different peer-to-peer systems, namely BitTorrent, Gnutella and KAD. The results obtained for KAD are based on crawling a subset of the KAD ID space. We call a continuous subset of the total KAD ID space that contains all KAD peers whose KAD IDs agree in the high order k bits a **k-bit zone**. Stutzbach and Rejaie have implemented a custom crawler that allowes them to crawl a 10-bit zone in 3-4 minutes and a 12-bit zone in approximately 1 minute [35]. A total of 4 different zones were crawled, each one being crawled for 2 days. The short duration of the crawls implies that the maximum values for some metrics such as session times or inter-session times that can be observed are naturally limited to 2 days. The paper by Stutzbach and Rejaie [35] is the most relevant to our work and we will refer to the results reported on several occasions. As we will see, our work significantly extends the findings of Stutzbach and Rejaie.

Le Fessant at al. [11] crawled eDonkey for one week and connected 55,000 out of 230,000 peers. The geographical distribution of these peers is very similar to the one we have observed (cf. Section V-B), except for the large number of Chinese peers that we see.

In 2002, Saroiu et al. [26] presented one of the first measurement studies for Gnutella and Napster. They developed their own crawler that connects, in the case of Napster, to each of the 160 servers and asks for the connected clients. The Gnutella crawler explores the graph of neighbor relations. The Gnutella crawl spans 8 days and the Napster crawl spans 4 days. For both Gnutella and Napster, Saroiu et al. report median session times of about 1 hour, which is half the time compared to the peers in KAD (cf. Section VI-A).

Chu et al. [7] repeated the measurements of Saroiu et al. and extended them to a duration of six weeks measuring session lengths and content popularity.

Qiao and Bustamante [23] compared the performance of structured and unstructured Overlay networks for the case of Overnet and Gnutella. For their study they performed session measurements for 7 days and reported median session times of 71 minutes for Gnutella and of 135 minutes for Overnet, which is very close to our results for KAD (cf. Section VI-A). In [5] the same authors present a new peer-to-peer system that makes use of the expected session times of the peers that follows a heavy-tailed Pareto distribution in order to improve resilience to churn.

Tian and Dai [36] analyzed the logs of Maze, a Chinese peer-to-peer network with about 20,000 concurrent users. All users are connected via the high speed Chinese research network. Although it is a peer-to-peer network, all peers connect every 5 minutes to a central server that writes a global log file. This enables an analysis to be made.

III. KAD Background

KAD is a Kademlia-based [16] peer-to-peer DHT routing protocol that is implemented by several peer-to-peer applications such as Overnet [20], eMule [10], and aMule [1]. The two open–source projects eMule and aMule 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.

Similar to other DHTs like Chord [32], CAN [24], or Pastry [25], each KAD node has a global identifier, referred to as a KAD ID, which is a 128-bit long random number 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 joining and daparture of the peer, until the user deletes the application or its preferences file[1]. Therefore, using the KAD ID, a particular peer can be tracked even after a change of its IP address. This is important since many ISPs reassign IP addresses to their customers as often as once a day.

A. Routing

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 to b. 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 $d(a, b) = 1011 \oplus 0111 = 1100$, and the distance between $a = 1011$ and $c = 1100$ is 0111. Thus a is closer to c than to b, since $d(a, b) = 1100 > d(a, c) = 0111$. The fact that this distance metric is symmetric is an advantage compared to other systems, e.g. Chord, since in KAD if a is close to b, then b is also close to a. Therefore, a node a that announces its existence to a node b might be added by node b to its routing table.

The entries in the routing tables are called *contacts* and are organized as an unbalanced *routing tree*: A peer P stores only a few contacts to peers that are far away in the KAD ID space and increasingly more contacts to peers closer in the KAD ID space. For details of the implementation see [34]. For a given distance, P knows not only one peer but a *bucket* of peers. 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. Each contact consists of the node's KAD ID, IP address, TCP and UDP port. The *left* side of the routing tree 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 of the routing tree 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 into 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.

To route a message toward its destination the next hop is chosen 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 the crawling of the KAD network.

B. Publishing

A **key** in a peer-to-peer system is an identifier used to retrieve information. KAD distinguishes between two different keys:

[1]As we will see later, not all peers in KAD behave this way.

- A **source key** that identifies the content of a file and is computed by hashing the *content* of a file.
- A **keyword key** that classifies the content of a file and is computed by hashing the tokens of the *name* of a file.

In KAD, keys are not published just on a single peer that is numerically closest to that key, but on 10 different peers whose KAD ID agrees at least in the first 8 bits with the key. This 8-bit zone around a key is called the tolerance zone. All peers inside a tolerance zone around a given keyword are qualified to accept a publication for this keyword.

Keys are periodically republished: **source keys** every 5 hours and **keyword keys** every 24 hours by default. Analogously, a peer on which a source key or keyword key was published will delete the information after 5 and 24 hours, respectively. This way re-publishing is done in exactly the same way as publishing.

The peer that accepts the publish message for a keyword returns the load factor to the publishing peer. The load factor takes values between 0 and 100 and is computed as a function of the number of publications for the specific keyword and the total number of publications the peer received and stored. If the load factor is below 20, the default republishing delay of 24 hours is kept; otherwise it is adjusted as follows: *republishing delay* $= \frac{load\ factor}{100} * 7 * 24$. The maximum republishing delay can thus be as long as 7 days.

The four most important message types for the route, publish, and search process are:

- `hello`: to check if the other peer is still alive and to inform the other peer about one's existence and the IP address and KAD ID.
- `route request/response(kid)`: to find peers that are closer to the KAD ID `kid`.
- `publish request/response`: to publish information.
- `search request/response(key)`: to search for information whose hash is `key`.

IV. Measurement Methodology

A. Crawling KAD

We have developed *Blizzard*, our own crawler for KAD, with the aim to crawl KAD very frequently and over a duration of several months. Our crawler logs for each peer P the time of the crawl, the IP address of P, the KAD ID of P, and whether or not P has responded to the crawler.

In a large peer-to-peer system such as KAD, peers are constantly joining and leaving, which makes it difficult to get a consistent view of the system. Therefore, the overall duration of a single crawl should be as short as possible. To speed up a single crawl, previous crawlers (such as [35]) were often distributed and ran simultaneously on multiple machines. However, we noticed that in a distributed crawl a lot of CPU time is used up for the synchronization between the different machines. To make our crawler run very fast, we decided to run Blizzard on a *single* machine and to keep all relevant information in main memory. The implementation of Blizzard is straightforward: It starts by contacting a seed peer run by us. Then it asks the seed peer for a set of peers to start with and uses a simple breadth first search and iterative queries. It queries the peers it already knows to discover new peers. For every peer returned, the crawler checks if this peer has already been discovered during this crawl. We use a hash table of already discovered peers that fits in main memory, which makes this test very efficient. After one crawl is completed, the results are written to disk.

At the beginning of each crawl, the number of new peers discovered grows exponentially before it approaches asymptotically the total number of peers in KAD. At some point the crawl needs to be stopped, otherwise the crawl accuracy decreases, since new peers are joining the system all the time [33]. We choose to stop querying new peers when 99% of the peers discovered have been queried. We then wait for 30 more seconds for late replies before terminating the crawl.

Not all the peers discovered can be contacted directly by the crawler. Approximately half of the peers queried do not respond to the crawler. There are two main reasons why a peer does not respond to our queries: either the peer has left the system, or the peer is behind a NAT that blocks our query. For the crawler it is not possible to distinguish between these two cases.

The crawler is implemented as two asynchronous threads: One thread to send the `route requests(kid)` (Alg. 1) and the other one to receive and parse the `route responses` (Alg. 2). A list that contains all the peers discovered so far is used and maintained by both threads. The receiving thread adds the peers extracted from the `route responses(kid)` to the list, whereas the sending thread iterates over the list and sends 16 `route requests(kid)` to every peer in the list. The value of the KAD ID `kid` is different in each of the 16 `route requests`. Care is taken to assure that each value of `kid` falls in a different bucket of the peer's routing tree, which allows us to minimize the overlap between the sets of peers returned in the response.

There are 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 independent instances of our crawler, one at the Universität Mannheim, Germany, connected to the German research network, and a second one at Eurécom, France, connected to the French academic network. Running two crawls in parallel turned out to be very useful: at some point, due to network problems, one instance of the crawler was seeing fewer peers than the other one. Also, occasionally one of the two crawlers crashed.

B. Full Crawl

The speed of Blizzard allows us to crawl the entire KAD ID space, which was never done before. Such a **full crawl** of KAD takes about 8 minutes. The first million peers are identified in about 10 seconds, the second million in 50 seconds; thereafter the speed of discovery decreases drastically since most peers returned in the `route response` messages have already been seen during the same crawl. A full crawl of KAD produces about 3 GBytes of inbound and 3 GBytes of outbound traffic.

Algorithm 1: send thread (is executed once per crawl)

```
Data: peer: struct{IP address, port number, kid}
Data: shared list Peers = list of peer elements
/* the list of peers filled by the
   receive thread and worked on by the
   send thread                          */
Data: int position = 0
/* the position in the list up to which
   the peers have already been queried
   */
Data: list ids = list of 16 properly chosen kid elements
1 Peers.add(seed); /* initialize the list with
     the seed peer                      */
2 while position < size(Peers) do
3     for i=1 to 16 do
4         destkid = Peers[position].kid ⊕ ids[i];
              /* normalize the bucket to the
              peers position            */
5         send route requests(destkid) to
          Peers[position];
6     position++;
```

Algorithm 2: receive thread (waits for the `route response` messages)

```
Data: message mess = route response message
Data: peer: struct{IP address, port number, kid}
Data: shared list Peers = list of peer elements
/* the list shared with the send thread
   */
1 while true do
2     wait for (mess = route response) message;
      foreach peer ∈ mess do
3         if peer ∉ Peers then
4             Peers.add(peer);
```

A full crawl was done once a day from March 20, 2007 to May 25, 2008.

C. Zone Crawl

A full crawl generates an extremely high amount of trace data and network traffic (with peak data rates close to 100 Mbit/sec). Also carrying out just one crawl per day is not really sufficient to capture the dynamics of KAD peers over short timescales, which is needed to measure e.g. session times. For this reason, we decided to carry out a **zone crawl** on an 8-bit zone, where we try to find all active peers whose KAD ID has the same 8 high-order bits. Such a zone crawl that explores one 256-th of the entire KAD ID space, takes less than 2.5 seconds. The high resolution and long duration zone crawl from September 23, 2006 to March 20, 2007 allowed us to collect 51,552 snapshots (one every 5 minutes) of a subset of all KAD peers. In section V, we will use the results of a full crawl to validate that the subset of KAD peers captured via a zone crawl is indeed a representative sample of the peers in KAD.

D. Data Cleaning

Crawling a zone happens periodically with a new crawl every five minutes and the two crawlers at the Universität Mannheim and Eurécom being synchronized. A peer that replied to at least one of the two crawlers during round i is considered to be up at round i [2]. The snapshots obtained by both crawlers are not always identical. The difference in the number of peers discovered is sometimes in the order of 10%, whereas most of the time the difference is less than 1%. Analyzing these differences reveals that the peers seen by one crawler but not by the other one are well distributed over all countries and the entire IP address space.

However, we realized that a peer that is up may occasionally be declared by both crawlers as not responding, i.e. considered as being down. One reason can be that the peer is overloaded and does not reply to our query. To validate this hypothesis we ran a KAD peer on an ADSL line: when neither the machine nor the peer application was loaded it always responded to the crawlers. When the machine was loaded with heavy calculations the peer still responded. However when the KAD peer was loaded with a large number of simultaneous downloads it frequently did not respond to the route requests of our crawlers.

Another reason can be that the path between the two crawlers and the peer is disrupted somewhere close to the peer. In both cases, the crawlers will not receive a response from the peer even when it is up and running. While it is not possible to tell exactly why a peer is not answering, we implemented the following data cleaning rule that we consider "reasonable": When a peer P that has been reported up at round $i-1$ does not reply to either of the two crawlers during the next round i, and then replies again during round $i+1$, then peer P will also be considered up at round i.

We refer to this filtering mechanism as eliminating 1 hole. One can of course generalize this approach to eliminating i holes, which means considering a peer that responded during crawl k, then did not respond for up to i consecutive crawls before responding again, as being continuously up from crawl k to crawl $k+i+1$.

Since we have no answer as to what data cleaning technique is the most appropriate, we ran different experiments where we eliminated i holes, with $i \in \{0,1,2,3,5\}$. The resulting Cumulative Distribution Functions (CDFs) of the session times are shown in Fig. 1 and the first two moments of the session times in Table I. Of course, the bigger the holes we eliminate the larger the mean session times. However, it is important that independently of the number of holes eliminated, the session times could always be a perfect fit using a Weibull distribution, which is described by two parameters, referred to as scale and shape. We will come back to this in section VI-F.

[2] As we can see in Algorithm 1 the crawler sends 16 route requests to each peer. A peer is considered alive if at least one route response is received by the crawler.

For the rest of the paper we will use the data with 1 hole eliminated.

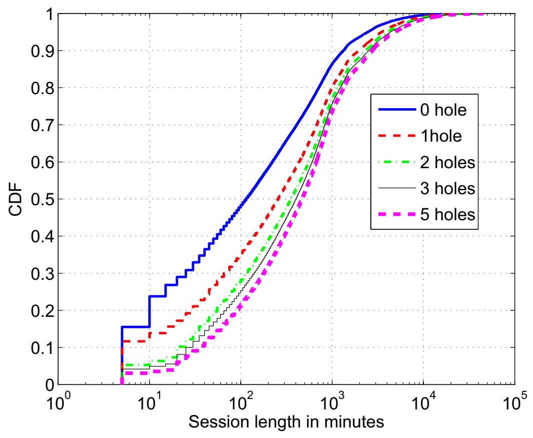

Fig. 1. CDF of the session times (1,2,3, and 5 hole(s): after data cleaning, 0 hole: raw data without data cleaning).

TABLE I
SESSION CHARACTERISTICS BEFORE AND AFTER DATA CLEANING. ELIMINATING i HOLES MEANS THAT WE CONSIDER A PEER AS *connected* EVEN IF IT DOES NOT RESPOND DURING i CONSECUTIVE CRAWLS.

	Weibull		Session times	
Eliminating	Scale	shape	mean	std dev.
Raw data	55.91	0.52	113.73	297.41
1 hole	97.62	0.56	169.21	405.26
2 holes	129.53	0.61	199.62	455.9
3 holes	144.28	0.63	215.8	486.2
5 holes	165.4	0.65	238.38	525.25

The following two sections present the results of our crawl. Section V will provide general information such as the number of KAD users and their geographic distribution and will also discuss KAD ID aliasing and its implications. Section VI will focus on statistics related to session times that are very relevant for the optimization of certain design parameters of KAD such as the republishing interval and metrics that characterize the daily usage behavior of KAD clients.

V. Global View of KAD

In this section, we will present results obtained via a full crawl of KAD, such as the total number of users, the geographic distribution of the users, and the distribution of the KAD IDs over the hash space. We will and compare these results where appropriate to the ones obtained via the zone crawl. Moreover we will characterize the fact of IP address aliasing and KAD ID aliasing.

A. Full Crawl

During each full crawl, we discovered between 3 and 4.3 million different peers. Between 1.5 to 2 million of these reply to our queries, and can thus be *directly contacted* by our crawler. The other peers either have left the system or are located behind NATs or firewalls. In the rest of the paper we will only report statistics on the peers that our crawler could contact *directly*.

In Fig. 2, we plot the distribution of the percentage of peers seen per country, using the Maxmind database [15] to resolve IP addresses to countries and ISPs. The continent with the highest percentage of peers is Europe (Spain, France, Italy and Germany), while the country with the largest number of peers is China. Less than 15% of all peers are located in America (US, Canada, and South America). We can also see that the geographical distribution of the peers obtained with the two zone crawls of an 8-bit zone each is very close to the result obtained with the full crawl, which is to be expected since the KAD IDs are chosen at random.

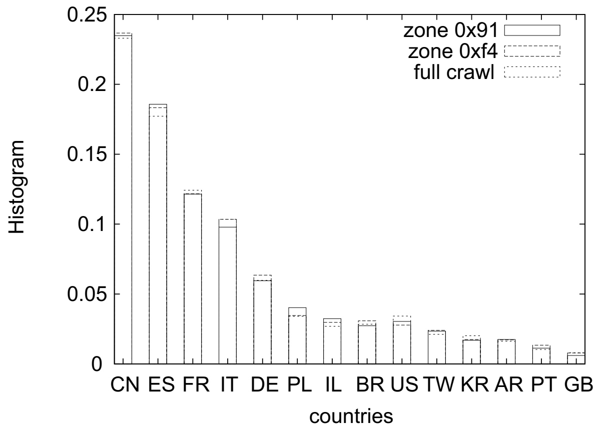

Fig. 2. Histogram of geographic distribution of peers seen on 2006/08/30.

In fact, we see from the results of the full crawl that the peers are uniformly distributed over the hash space, except for some outliers (Fig. 3). All the outliers are due to modified KAD clients that use the same KAD ID and are always limited to one country (Korea, Spain, Israel, China, Argentina). The outlier in zone `0xe1` is a modified client used in Israel, for which we counted 25069 instances [28]. We also observed 250,000 instances of KAD, with KAD IDs that systematically cover the entire KAD ID space. All the IP addresses belong to the domain `mediadefender.com`. By carrying out a so-called Sybil attack [9], [29], the company Media Defender is able to closely monitor all publish and search activities of all peers in KAD. We have filtered out these "anomalies" from our trace data since we are interested in characterizing the behavior of ordinary KAD peers.

Given that KAD IDs are uniformly distributed, we can estimate the total number of peers in KAD by simply counting the number of peers in a zone and multiplying this value by the number of zones (256 zones). Using Chernoff Bounds (see [19] Chapter 4) we tightly bound the estimation error.

Let $N(t)_{part}$ be the number of peers counted during a zone 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 the 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

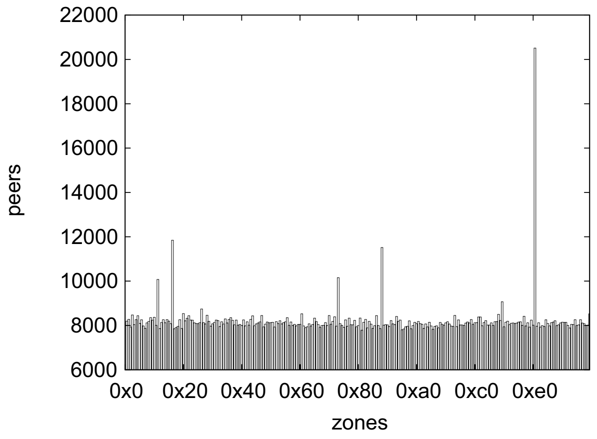

Fig. 3. The distribution of the peers over the hash space. The 256 8-bit zones on the x-axis go from `0x00` to `0xff`.

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.

B. Zone Crawl

All the results in the following subsection were obtained using the zone crawl of the 8-bit zone `0x5b` that lasted for 179 days.

In Fig. 4, we plot the number of peers seen that originate from China and some European countries. The number of peers in each country follows a diurnal pattern, with a peak around 9 PM local time. The eight-hour time shift between Europe and China is clearly visible.

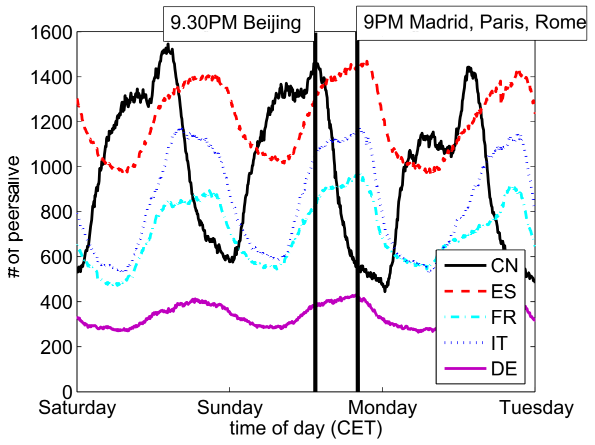

Fig. 4. Peers online according to country of origin.

Table II summarizes the basic findings on the zone crawl. The peers seen came from 168 different countries and 2384 providers. For the KAD IDs seen on the first day of our zone crawl, we observe that about one third of the peers come from Europe and about one fourth from China. If we compare the **lifetime** of the peers, which is defined as the difference between the time a given KAD ID was seen the last time and the time this KAD ID was seen the first time, we notice that the lifetime of peers in China is much smaller than that of

TABLE II
KEY FACTS ABOUT THE PEERS SEEN DURING THE ZONE CRAWL SPANNING 179 DAYS ORGANIZED BY COUNTRY OF ORIGIN (LT=LIFETIME).

	Total	China	Europe	Rest
Different KAD IDs	400,278	231,924	59,520	108,834
Different IP addresses	3,228,890	875,241	1,060,848	1,292,801
KAD IDs seen for a single session	174,318	131,469	11,644	31,205
KAD IDs with LT $\leq$ 1 day	242,487	183,838	15,514	43,135
KAD IDs seen for the first time on				
- 1st crawl	5,670	455	2,879	2,336
- 1st day	18,549	4,535	6,686	7,328
- 60th day	1,893	1,083	259	551
KAD IDs seen for the first time on 1st day				
- with LT $\leq$ 1 day	2,407	1,568	286	553
- 1 day $<$ LT $\leq$ 1 week	1,368	497	393	478
- 1 week $<$ LT $\leq$ 1 month	2,735	791	944	1,000
- LT $>$ 1 month	12,039	1,679	5,063	5,297
- LT $>$ 3 months	8,423	936	3,679	3,808
avg. of the median session time per peer (minutes)	165	103	326	210
avg. of the median-inter session time per peer (minutes)	1,341	586	2,825	2,136

TABLE III
KEY FACTS ABOUT THE PEERS SEEN DURING THE ZONE CRAWL SPANNING 179 DAYS ORGANIZED BY ISP OF ORIGIN (LT=LIFETIME).

	Total	Europe				China		
		Proxad	Orange	Dt. Tel.	Telefonica	DDV	CNCgroup	ChinaNet
Different KAD IDs	400,278	5,565	4,834	3,129	8,294	55,668	75,300	79,057
Different IP addresses	3,228,890	5,446	4,668	3,099	7,930	42,163	55,449	65,964
KAD IDs seen for a single session	174,318	941	677	502	1,356	34,863	40,745	40,306
KAD IDs with LT $\leq$ 1 day	242,487	1,209	899	649	1,822	48,195	57,730	56,518
avg. of the median session time per peer (minutes)	165	408	324	376	359	107	87	112
avg. of the median inter-session time per peer (minutes)	1,341	2,880	2,908	2,601	2,527	375	641	788

peers in the other countries. More than half of the peers in China were seen for the duration of only *one* session. We will come back to this point in the next subsection V-C.

Table III presents the relevant statistics for some of the ISPs in Europe and China where most of the peers originate. For both Europe and China, the key metrics of the peers do not vary much across the different ISPs.

Arrivals and Departures: Since we crawl the same zone in KAD once every 5 minutes, we can determine the number of peers that join and leave between two consecutive crawls. Knowing the arrival rate of peers is useful since it allows us to model the load in KAD due to newly joining peers. Each time a peer joins, it first contacts other peers for information to populate its routing table, before it publishes the keywords and source keys for all the files it will share.

In Fig. 5 we depict the CDF of the number of peers that arrive and that depart between two consecutive zone crawls. We see that the distributions for arrivals and departures are the same. This is to be expected, since we observe the system in "steady state": in this case, the system should behave like $G/G/\infty$, for which, according to Little's Law, the arrival rate is equal to the departure rate [18]. The arrival process is very well described by a Negative Binomial distribution (see Fig. 5b of [31]).

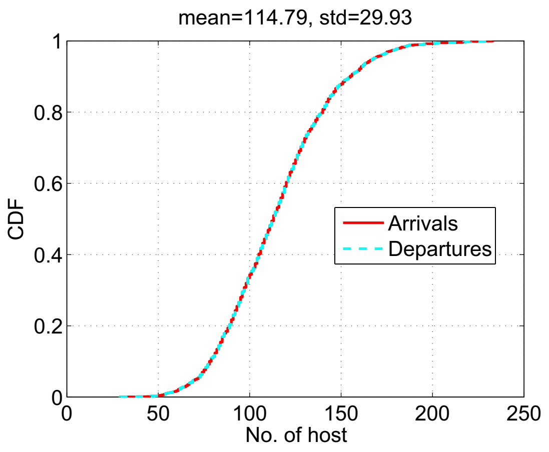

Fig. 5. CDF of the number of arrivals and departures.

C. Aliasing

IP Address Aliasing: It has been known for several years [2], [14] that many peers frequently get assigned new IP addresses, which is referred to as **IP address aliasing**. For instance, we know that some ISPs in France change the IP address of their ADSL customers approximately every 24 hours, while others assign static IP addresses to their clients. We observed a total of 400,278 distinct KAD IDs and 3,228,890 different IP addresses (see Table II). In Europe, a peer has on average about 18 IP addresses, whereas in China the number is 4 IP addresses per peer. About 80% of the peers in China have only one IP address since their lifetime is much shorter than the lifetime of peers in other parts of the world. We saw that the number of different IP addresses per peer is strongly correlated with the peer lifetime (see Fig. 6 of [31]).

KAD ID Aliasing: Up to now it was assumed that KAD IDs are persistent, i.e. the same end-user of KAD *permanently* keeps the same KAD ID across all its sessions. As it turns out, this is not true. We refer to the fact that KAD IDs are non-persistent as **KAD ID aliasing**.

We see in our zone crawl approx. 2,000 new KAD IDs a day, which means that for the entire KAD system the number of new KAD IDs per day is around 500,000. If we extrapolate, this makes about 180 Million KAD IDs a year. It is hard to believe that there exist such a large number of different end-users of KAD.

Figure 6 reports the number of *new* KAD IDs per day. i.e. KAD IDs seen for the first time, according to country of origin. More than 50% of the new KAD IDs are from peers in China, which is more than one order of magnitude greater than the number of new KAD IDs seen for any other country such as Spain, France, or Germany.

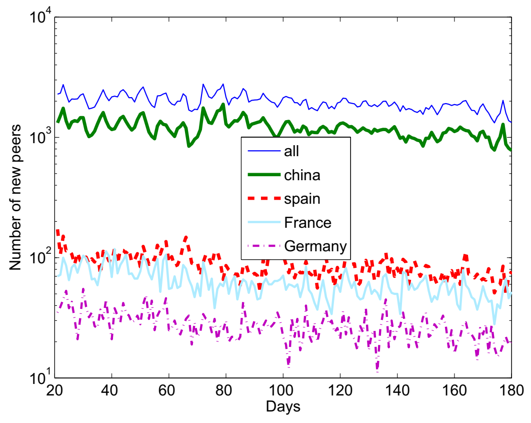

Fig. 6. New KAD IDs according to country of origin.

We were curious to find out whether it is plausible that many end-users really stop using KAD after one session, or whether the same users come back with a different KAD ID. To investigate KAD ID aliasing, we need to look for peers with *static IP addresses*, which we can track for *non-persistent* KAD IDs. We know that, for instance in France, one of the ADSL providers (Proxad) assigns static IP addresses to customers who are located in areas where the service offer is completely “un-bundled”.

Our hypothesis is that a peer that keeps the same IP address and port number for 10 days is assigned a static IP address. Therefore, we take the logs of the two full crawls (cf. Section V-A) of March 20, 2007 and March 30, 2007 and extract the 140,834 peers that have the same IP address, port number and KAD ID in both crawl logs. IP addresses running more than one KAD ID are filtered out. This way we exclude all users having dynamically assigned IP addresses, moreover we exclude all users with static IP addresses who were not online on March 20 and March 30. We call this set of peers a **pivot set**.

Since this heuristic is very strict, the number of users with static IP addresses is underestimated. However the pivot set still contains enough peers to make statistically meaningful statements. 32% of the peers in the pivot set come from Spain, 18% from France, 5% from Poland and Italy, 4% from the US and Taiwan, and 3% from Israel and Argentina.

We then take the logs of the full crawls starting March 31st, 2007 to look for peers in the pivot set that have *changed their* KAD ID.

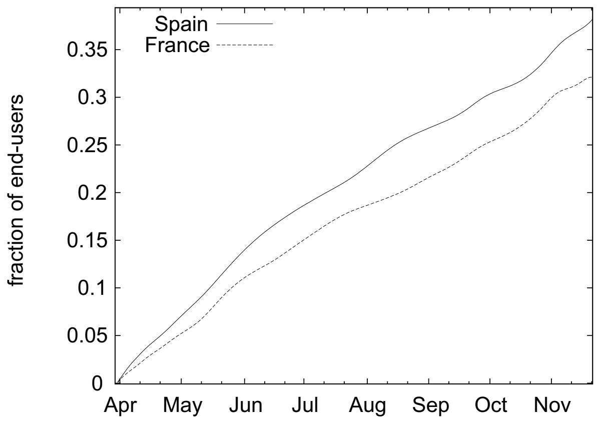

Fig. 7. The fraction of peers in the pivot set that changed their KAD ID at least once.

In Fig. 7, we plot the fraction of peers from the pivot set that change their KAD ID at least once. Since we perform a full crawl only once a day, we are not able to estimate the rate of change of the KAD IDs. Instead, we can only detect which peers have changed their KAD ID. We see that a significant fraction of end-users in different countries change their KAD ID over time. After seven months, more than one third of the end-users in Spain and France changed their KAD ID at least once.

A very recent study confirmed that KAD ID aliasing is quite common. Pietrzyk et al. [21] monitored a population of about 20,000 ADSL clients in France for ten days. About 20% of the peers change their KAD ID for every new session and some peers change it even during a session. In comparison to clients that do not change their KAD ID, these peers have longer session times, whereas the amount of files they share is significantly smaller. It seems that peers who frequently change their KAD ID do so in order to improve their anonymity.

Implications of KAD ID Aliasing: The fact that the KAD ID assigned may be non-persistent obliges us to distinguish between a peer and an end-user:

- A **peer** is an instance of KAD identified by a fixed KAD ID.
- An **end-user** is a physical person who launches a peer to participate in KAD. The same end-user can, at different times, participate in KAD via different peers.

When KAD ID aliasing occurs, it is not really possible to characterize the lifetime of *end-users* tracking a KAD ID, as compared to the lifetime of peers. We will see in section VI-F how we can use the peers in the pivot set to estimate the lifetime of end-users.

VI. Peer View

IN this section, we will present metrics that describe the behavior of individual peers, such as lifetime, session and inter-session time, residual uptime, and daily availability using the observations made with our 179-day zone crawl. To estimate the end-user lifetime we also make use of the data obtained via the full crawl.

Using these metrics we will compare the peer behavior of different countries. Knowing the session statistics allows us (i) to validate implementation choices of KAD and (ii) to make suggestions on how to improve the efficiency of KAD (cf. Section VII).

A. Session Statistics

Most of the peers will not be online, i.e. connected to KAD, all the time. By crawling KAD every five minutes, we can determine precisely for each peer k the instances $t_1^j(k), ..., t_n^j(k)$ when k joined and the instances $t_1^l(k), ..., t_n^l(k)$ when k has left KAD. We define the **session length** as the time a peer was present in the system without any interruption, i.e. $t_i^l(k) - t_i^j(k)$ for $i \in \{1, ..., m\}$. For the peers that were online on the first crawl, we did not consider the first session, since we can not know when it began. Analogously, we did not consider the sessions that were still ongoing during our last crawl. For the European countries the distributions of the session times are very similar. The Chinese peers, however, have significantly fewer long session times above 2 hours.

The session length of the peers seen in the *first* crawl is about twice that of the peers seen for the first time during later crawls of day 1 (Fig. 8). When we crawl KAD for the first time, we have a much higher chance of seeing 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 crawl of the system cannot give a representative picture of the characteristics of the peers: Instead, we need to sample the system many times.

For the peers seen in the first crawl, we observe session times (in minutes) with a mean = 670, standard deviation = 1741 and median = 155. For the peers seen during the remaining crawls on the first day these values are only about half as large with mean = 266, standard deviation = 671 and median = 75. In both cases, the coefficient of variation, which is defined as the ratio between standard deviation and mean, which characterizes the “variability” of a distribution, is between 2 and 3.

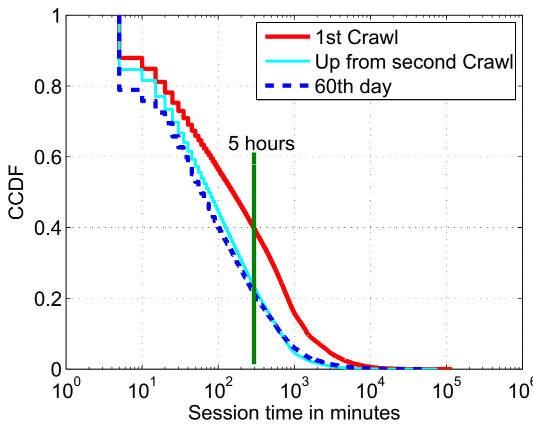

Fig. 8. CCDF of the session lengths per KAD ID.

Weibull fit of the session time distribution: The empirical distribution of the session length exhibits a considerable tail. At least 0.1% of the sessions are longer than 1 week and the longest session observed is 78 days. We did a distribution fitting for the session times and found that the Weibull distribution provides a very good fit (See Table I for Weibull parameters).

The Weibull distribution has two parameters $k > 0$ (*shape*) and $\lambda > 0$ (*scale*). The Weibull distribution with $k < 1$ is part of the class of the so-called sub-exponential distributions, for which the tail decreases more slowly than any exponential tail [12]. Sub-exponential distributions are a subclass of the class of heavy-tailed distributions [4]. This implies that knowing the past (uptime) of a peer allows us to predict the future (residual uptime). More formally, if S denotes the session length then the expected residual uptime is $\mathbb{E}[S - t|S > t] \sim O(t^{1-k})$, i.e. it grows *sub-linearly*. For comparison: if S where Pareto distributed, the growth of its residual uptime would be linear, i.e. $O(t)$.

B. Remaining Uptime

Figure 9 shows the expected residual uptime for the scale and shape values that describe the session length of peers seen in the first crawl. There is a nice fit between the empirical values and the interpolation using a function whose growth is $O(t^{1-k})$. We see that for small observed uptime values the remaining expected uptime values are considerable: A peer that has been up for 1,000 minutes will have a remaining expected uptime of 1,500 minutes. Based on their data set, which did not contain any sessions longer than 1 day, Stutzbach and Rejaie [35] concluded that KAD sessions times could be fit either by a log-normal or a Weibull distribution. Our crawl, which allowed us to observe sessions that lasted several weeks, confirms this point.

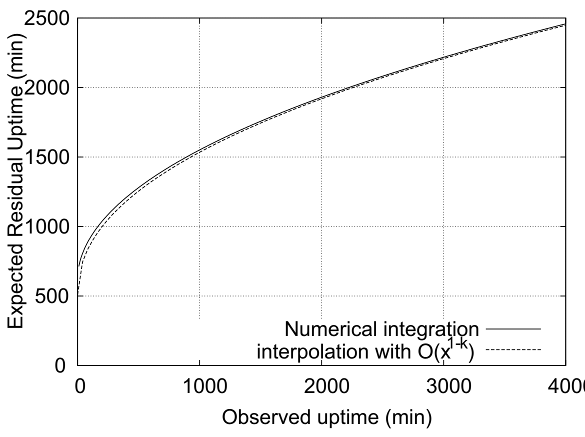

Fig. 9. Expected residual uptime for $k = 0.54, \lambda = 357$ (for peers seen in the first crawl).

The eMule and aMule implementations of KAD only publish on peers that have been up for *at least 2 hours*. Source keys will expire after 5 hours and keyword keys after 24 hours. We may ask whether selecting a peer that has been up for at least 2 hours will increase the chances that this peer will be up for another 5 or 24 hours.

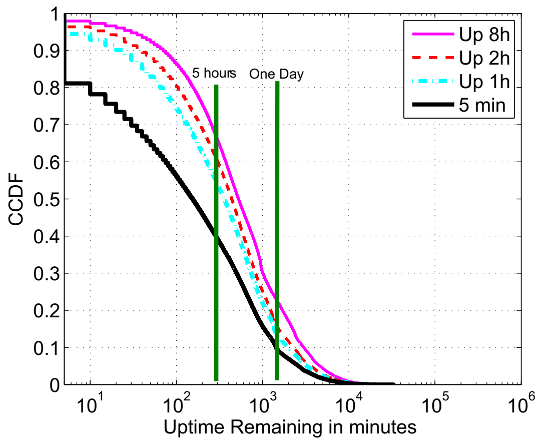

Fig. 10. CCDF of the remaining uptime of peers, given the uptime so far, for peers seen in the first crawl.

In Fig. 10, we plot the remaining uptime of peers given that they have already been up 5 minutes, 1h, 2h, or 8h. For this analysis we choose the set of peers seen in the first crawl since this is the view a joining peer has of the system. We see that a higher uptime translates into a higher remaining uptime. This means that the minimum age-based peer selection as implemented in eMule and aMule is a sensible policy.

Only about 20% of the peers with an uptime of 2 hours will remain up for at least another 24 hours. Therefore, the only way to ensure that keywords remain available for 24 hours is to publish information about a keyword on *more than one peer*, as is done by eMule and aMule.

Trying to increase the lifetime of published content only makes sense for those peers that are themselves highly available. Therefore, the publication strategy needs to choose the content lifetime as function of both, the past availability of the publishing peer and the availability of the peers where the data is going to be published.

C. Next Session Time

One may ask the question whether consecutive sessions are correlated in length. If there is a strong positive correlation, one could use information about past session lengths as a predictor of the length of future sessions: If a publishing peer could predict its session time it could then choose the optimal value for the expiration time of the information it publishes in KAD.

If we take all session length samples and compute the coefficient of correlation over consecutive session lengths we obtain a value of 0.15, which indicates that there is almost no correlation. However, if we only consider session lengths up to 1 day, there is a considerable positive correlation of 0.85, which was also observed by Stutzbach and Rejaie (Figure 10(b) of [35]).

D. Inter-Session Time

The **inter-session time** is defined as the time a peer k is continuously absent from the system, i.e. $t_{i+1}^{j}(k) - t_{i}^{l}(k)$ for $i \in \{1, ..., n\}$.

The peers seen in the first crawl have, on average, not only longer session times but also smaller inter-session times than peers seen the first time in later crawls (cf. Table II). The average inter-session time is 1110 minutes for peers seen in the first crawl compared to 1349 minutes for peers seen first during crawls 2 up to 288. Peers in China have much shorter inter-session times than peers in Europe (cf. Table II). The longest inter-session time observed is 177 days. For the inter-session times we could not find a distribution that makes a good match with our observed data.

E. Lifetime of Peers

For a given KAD ID k, let $t^{j}(k)$ be the time this KAD ID is seen joining KAD for the first time, and let $t^{l}(k)$ be the time this KAD ID is seen for the *last* time. The lifetime of KAD ID k is defined as $t^{l}(k) - t^{j}(k)$. Since our crawl is of a finite duration, we can never be sure if a peer with KAD ID k will not come back after we stopped crawling. To make such an event very unlikely, we have decided to compute the lifetime only for peers with KAD IDs seen for the last time 60 days or more before the end of our crawl. We set the cut-off at 60 days, since the inter-session times seen are very rarely longer than 60 days.

Among the peers seen on the first day, about $2/3$ of the peers have a lifetime longer than one month and close to 45% have a lifetime longer than three months (Table II).

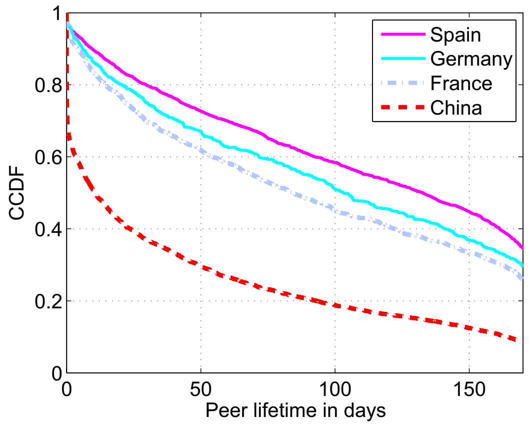

Fig. 11. CCDF of the lifetime of the *peers* seen on the first day according to their country of origin.

Figure 11 depicts the complementary cumulative distribution (CCDF) of the peers seen on the first day. There is a big difference in the lifetime of peers from China as compared to Europe: more than a third of the Chinese peers disappear after only one day and only 10% have a lifetime of more than 150 days, while close to 40% of the peers in Europe have a lifetime of more than 150 days.

In fact the lifetime of KAD IDs strongly depends on the number of times a peer reconnects to the system. Figure 12 shows the CCDF of the number of sessions for peers coming from China and Europe. About 30% of the Chinese peers use the same KAD ID for only one session compared to 5% for the

European peers, which explains the difference in peer lifetime seen in Fig. 11.

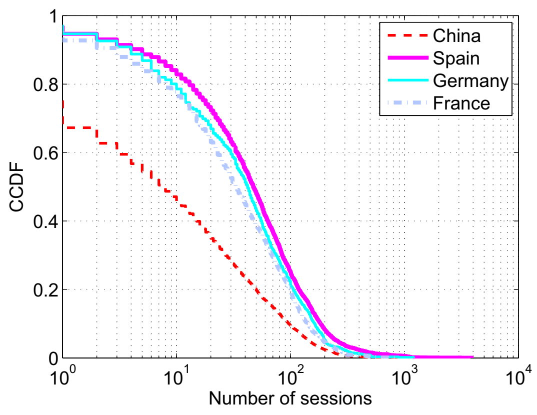

Fig. 12. CCDF of the number of sessions per KAD ID.

F. Lifetime of End-users

The lifetime of end-users expresses in some way the satisfaction of end-users. If users come back again and again it means that they rely heavily on peer-to-peer file sharing for accessing and exchanging content with other users.

Due to KAD ID aliasing (cf. Section V-C) we can not estimate the lifetime of end-users by measuring the lifetime of KAD IDs (cf. Section VI-E). The only way to measure the lifetime of end-users is to analyze peers from the pivot set that we already used in section V-C to measure aliasing of KAD IDs. Since the pivot set contains over one hundred thousand peers from numerous countries we expect that these peers are representative of all peers in KAD. In Fig. 13, the CCDF of the lifetime of end-users is plotted. End-user lifetimes are significantly larger than peer lifetimes (see Fig. 11). 50% of the end-users have been using KAD for 6 months and more.

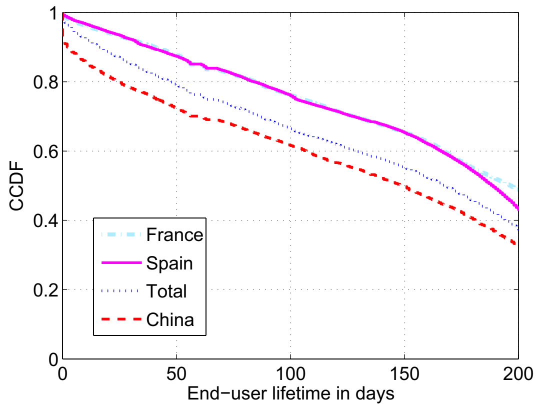

Fig. 13. CCDF of the lifetime of the *end-users* having static IP addresses and fixed port numbers.

G. Daily Availability

Characterizing availability is important for building efficient distributed applications such as overlay multicast or distributed file systems. For instance, availability guided file placement can help reduce the cost of object maintenance [17], which may potentially be prohibitive as was pointed out by Blake [3].

Availability in the case of KAD measures the *usage behavior*, i.e. how many hours a day users are connected and how they use KAD over longer time periods such as weeks.

Daily availability measures the fraction of time a peer is connected per day. Daily availability expresses the "intensity" of participation of users in the exchange of files. For a given peer P, we define **daily availability** of P as the percentage of time P was seen online that day. For a peer that was first seen at day i and last seen at day j, we will get a time series of daily availability values that has $j-i+1$ elements. We define the **mean daily availability** as the average of those $j-i+1$ values.

Peers in China spend much less time per day connected than peers in Europe (Fig. 14). The "online times" for peers in Europe are quite impressive, with 40% of the peers being connected more than 5 hours per day and 20% even more than 10 hours per day.

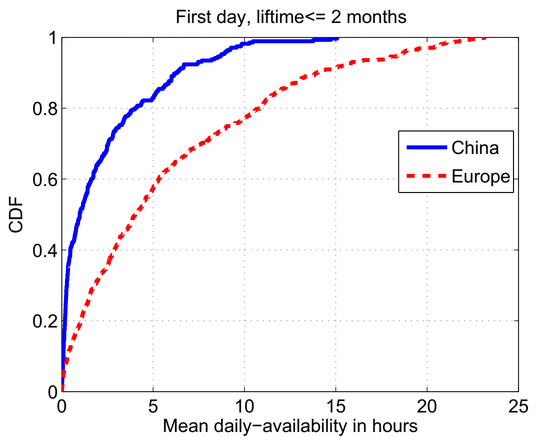

Fig. 14. CDF of the mean daily availability of peers seen the first day.

Next day availability: Observing availability over several days or weeks can give an indication about the stability of peer participation in KAD over time.

Figure 15 shows a scatter plot of the daily availability for each peer for two consecutive days. We see that for a given availability value on the first day, the availability of these peers on the second *varies widely*. If we compute the correlation of the availability on day i and day $i+1$ of each peer we get a low value of 0.52. Given that KAD is predominantly used to download copyright-protected content, the users probably stay connected the least possible time required to download the requested content. As the download time depends on various factors such as content size, content popularity, and available bandwidth, the daily availably will vary accordingly.

Figure 16 plots the daily availability (in hours) time series for a random set of peers over a duration of 100 days. While there are a few peers for which the daily availability changes little over time, most of the peers exhibit daily availability values that vary a lot.

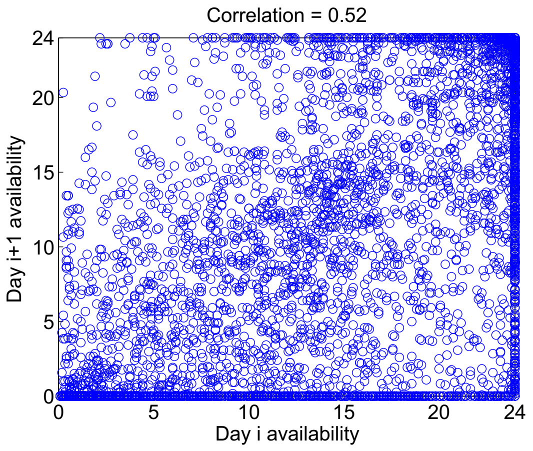

Fig. 15. Scatter plot of the availability on day i vs. the availability of day $i+1$ for peers seen in the first crawl.

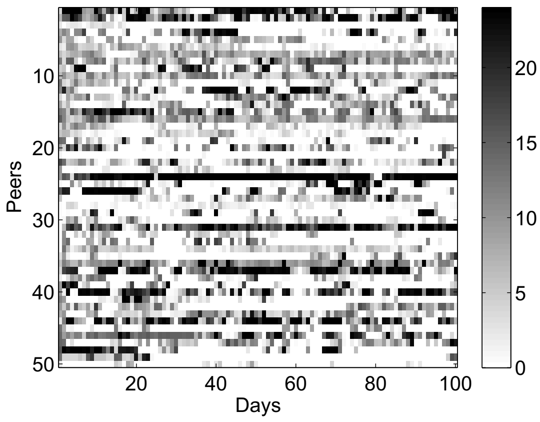

Fig. 16. Daily availability in hours of 50 randomly chosen peers seen in the first crawl.

Douceur [8] analyzed different traces of machine availability[3] from Microsoft, Internet, Gnutella, and Napster. Applying a Fourier transformation to the availability values he found cyclic behavior in the daily availability of the Microsoft machines, but did not find any diurnal patterns for the other traces. We did a Fourier and Wavelet transformation on the daily availability time series of our KAD peers and could not find any cyclic behavior or diurnal patterns.

To formally quantify the daily availability patterns of peers, we use a metric called **approximate entropy** $ApEn$, which is a “regularity statistic” that quantifies the unpredictability of fluctuations in a time series. For details on the approximate entropy see [22] and the Appendix. This metric was recently used by Mickens [17] to analyze several traces of machine availabilities such as the Microsoft trace and the Overnet trace. We calculate $ApEn$ of the daily availability of KAD peers seen on the first day. The smaller the value for $ApEn$, the more regular the daily availability values over time. Thus, small values of $ApEn$ imply that similar patterns of measurements will be followed by additional similar measurements. However, if the time series is highly irregular, the occurrence of similar availability patterns will be very unlikely for the following days, and $ApEn$ will be relatively large.

[3]The traces are available at
`http://www.cs.berkeley.edu/~pbg/availability/`

We were able to confirm the results of Mickens for the Microsoft trace, where 80% of the values of $ApEn(m)$,where $m = 2$, are close to zero, which indicates that daily availability varies little over time. On the other hand, the $ApEn(m)$ values for the KAD trace are much higher (see Fig. 17). About 50% of the peers have $ApEn(m)$ values above 0.5, which indicates that the daily availability values are quite irregular. Mickens made a similar observation for the Overnet trace.

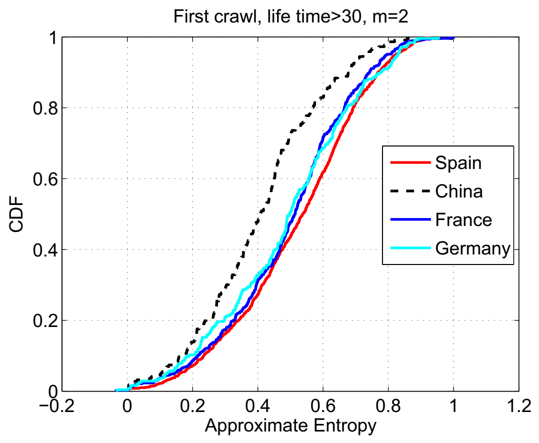

Fig. 17. CDF of the approximate entropy for peers seen the first day.

VII. Design Implications

THE results presented in this paper can be used to validate design choices made by the developers and improve the performance of the implementation of KAD in various ways: In today’s implementation of KAD, a source key that points to the peer that holds the content will expire 5 hours after it has been published. On the other hand, the median session length of peers is only 155 minutes and less than 40% of the sessions are longer than 5 hours (cf. figure 8). This means that in more than 60% of the cases, a peer that publishes a source key will leave KAD before the reference to that file expires. As a consequence, many references to sources will be *stale*, resulting in unsuccessful attempts to download that file. An improvement of the current implementation could be to first publish a source key with an expiration time much smaller than 5 hours. Each time the published source key expires, the peer that owns the file republishes the source key, progressively increasing its expiration time.

In [30] we measured that the total traffic in KAD due to *publishing* is about 100 times higher in volume (bytes) than the total *search traffic*. In a follow-up to this measurement study, we have shown how to exploit the fact that session times are Weibull distributed in order to reduce the publish traffic by one order of magnitude [6].

We have seen (cf. Section VI-A) that for peers with session times less than one day, the duration of the next session is highly correlated to the duration of the previous session. This means that a peer could set the initial expiration time of a source key using the value of its last session time.

Peers seen first during the very first crawl have much higher mean session durations and smaller inter-session times than peers seen later for the first time. As already suggested by Stutzbach and Rejaie [35], this fact can be exploited to find “more stable” peers without knowing anything about the history of the peers: One simply “crawls” KAD *once* and selects the peers that are online at that instant.

VIII. Conclusion

In this paper we have investigated the user behavior of KAD, which is currently the only DHT deployed on a large scale. Studying KAD poses a number of unique challenges to be addressed. To obtain the necessary data,

- We have implemented a very fast and highly efficient crawler of KAD. The speed of our crawler made it possible, for the first time ever, to carry out a full crawl of the entire KAD system. It also allowed us to track the behavior of a representative subset of KAD peers (with a precision of ± 5 minutes) over a period of almost six months. To the best of our knowledge this is the longest crawl of a peer-to-peer system ever carried out.
- We need to crawl for such a long duration to unambiguously identify the peers that joined KAD for the first time and to “capture the tail” of the session and inter-session time distributions, which is in the order of months.
- To be able to cope with transient network and machine failures, we ran two crawlers in parallel and we “post-processed” our measurements to account for missing replies of peers that are overloaded.

We have carried out a full crawl once a day in order to

- Validate that crawling a single zone will return a sample of the peers in KAD that is representative of the entire KAD network.
- Obtain a subset (pivot set) of peers with static IP addresses that can be used to estimate the rate of change of KAD IDs and the lifetime of *end-users* and not only the lifetime of KAD IDs.
- Detect various “anomalies” such as thousands of peers with the same KAD ID or a company “observing” the entire search and publish traffic via thousands of *sybil* peers.

Our high resolution zone crawl lead to a number of interesting findings:

- Session times are heavy tailed following a Weibull distribution.
- KAD IDs are not necessarily persistent as was assumed up to now. Nevertheless, the most important metrics such as session times and inter-session times are not affected by the non-persistent KAD IDs.
- The total number of peers online at any time can be precisely estimated.
- Peers in China differ significantly from peers in Europe with respect to key metrics such as session time, inter-session time, peer lifetime, and daily availability.
- The majority of clients use KAD every day for many hours.
- Since most of the content is copyright protected and the sharing of such content is illegal, users take measures to reduce the risk of being tracked by changing their KAD ID frequently or by staying connected only as long as necessary to download the desired content.

The full dataset of the zone crawl and of the full crawl are available on the Web `http://www.eurecom.fr/~btroup/kadtraces`.

Acknowledgments

We would like to thank D. Carra for the feedback he gave us on the paper and the Computing Center of the Universität Mannheim for providing us with the resources necessary to carry out such a long crawl. We also thank the anonymous reviewers for their valuable and constructive suggestions.

Appendix

The approximate entropy ($ApEn$) is a “regularity statistic” that quantifies the unpredictability of fluctuations in time series.

The algorithm for computing $ApEn$ has been published in [22]. Here, we provide a brief summary of the algorithm. Given a sequence U_N, consisting of N measurements equally spaced in time $U(1), U(2), \cdots, U(N)$, we must choose values for two input parameters, m and r, to compute the approximate entropy, $ApEn(m, r)$, of the sequence U_N. The parameter m, specifies the pattern length, and r, defines the criterion of similarity. We denote a subsequence (or pattern) of m measurements, beginning at measurement i within U_N, by the vector $x(i)$. Two patterns, $x(i)$ and $x(j)$, are similar if the difference between any pair of corresponding measurements in the patterns is less than r, i.e., if

$$|U(i+k) - U(j+k)| < r \text{ for } 0 \leq k < m$$

Now consider the set of all patterns of length m [i.e., $x(1), x(2), \cdots, x(N-m+1)$], within U_N. We may now define

$$C_i^m(r) = \frac{n_i(r)}{N-m+1}$$

where $n_i(r)$ is the number of patterns that are similar to $x(i)$ (given the similarity criterion r). In this work, we used $r = 0.1 * std\ deviation$(daily availability) as suggested in [22]. The quantity $C_i^m(r)$ is the fraction of patterns of length m that resemble the pattern of the same length that begins at i. We can calculate $C_i^m(r)$ for each pattern of size m, and we define $\Phi^m(r)$ as the mean of these $C_i^m(r)$ values. The quantity $\Phi^m(r)$ expresses the prevalence of repetitive patterns of length m in U_N. Finally, the approximate entropy of U_N, for patterns of length m and similarity criterion r, is defined as

$$ApEn(m) = \ln\left[\frac{\Phi^m(r)}{\Phi^{m+1}(r)}\right]$$

i.e., as the logarithm of the ratio of the relative prevalence of repetitive patterns of length m and $m+1$.

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)*, pages 256–267, 2003.

[3] C. Blake and R. Rodrigues. High availability, scalable storage, dynamic peer networks: Pick two. In *HotOs*, 2003.

[4] O. Boxma and B. Zwart. Tails in scheduling. *Performance Evaluation Review*, 34(4), 2007.

[5] F. Bustamante and Y. Qiao. Designing Less-structured P2P Systems for the Expected High Churn. *IEEE/ACM Transactions on Networking (TON)*, 16(3), June 2008.

[6] D. Carra and E. Biersack. Building a Reliable P2P System Out of Unreliable P2P Clients: The Case of KAD. In *Proceedings of CoNEXT*, Dec. 2007.

[7] J. Chu, K. Labonte, and B. N. Levine. Availability and locality measurements of peer-to-peer file systems. In *Proceedings of SPIE*, July 2002.

[8] J. Douceur. Is remote host availabilty governed by a universal law. In *Sigmetrics Performance Evaluation Review*, 2003.

[9] J. R. Douceur. The Sybil Attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, pages 251–260, Mar. 2002.

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

[11] F. L. Fessant, S. B. Handurukande, A.-M. Kermarrec, and L. Massoulié. Clustering in peer-to-peer file sharing workloads. In *The 3rd International Workshop on Peer-to-Peer Systems (IPTPS)*, Oct. 2004.

[12] C. Goldie and C. Klueppelberg. Subexponential distributions. In R. Adler, R. Feldman, and M. Taqqu, editors, *A Practical Guide to Heavy Tails: Statistical Techniques for Analysing Heavy Tails*, 1997.

[13] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. Al Hamra, and L. Garces-Erice. Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In *Proceedings of the Passive and Active Measurement Conference*, Apr. 2004.

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

[15] Maxmind. http://www.maxmind.com/.

[16] P. Maymounkov and D. Mazieres. Kademlia: A Peer-to-peer information system based on the XOR metric. In *In Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, pages 53–65, Mar. 2002.

[17] J. Mickens and B. Noble. Exploiting availability prediction in distributed systems. In *In Proceedings of NSDI*, 2006.

[18] I. Mitrani. *Probabilistic Modelling*. Cambridge University Press, 1998.

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

[20] Overnet. http://www.overnet.org/.

[21] M. Pietrzyk, G. Urvoy-Keller, and J.-L. Costeux. Digging into KAD Users' Shared Folders. In *Poster of SIGCOMM*, 2008.

[22] S. M. Pincus. Approximate entropy as a mesure of system complexity. *Proceedings of the National Academy of Science*, 88:2297–2301, Dec. 1990.

[23] Y. Qiao and F. E. Bustamante. Structured and Unstructured Overlays Under the Microscope - A Measurement-based View of Two P2P Systems That People Use. In *Proceedings of the USENIX Annual Technical Conference*, 2006.

[24] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *In Proceedings of SIGCOMM*, 2001.

[25] 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, Nov. 2001.

[26] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking (MMCN)*, Jan. 2002.

[27] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. *ACM/IEEE Transactions on Networking (ToN)*, 2004.

[28] M. Steiner, E. W. Biersack, and T. En-Najjary. Actively Monitoring Peers in Kad. In *Proceedings of the 6th International Workshop on Peer-to-Peer Systems (IPTPS)*, 2007.

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

[30] M. Steiner, W. Effelsberg, T. En-Najjary, and E. W. Biersack. Load reduction in the kad peer-to-peer system. In *Proceedings of the Fifth International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P)*, 2007.

[31] M. Steiner, T. En-Najjary, and E. W. Biersack. A Global View of KAD. In *Proceedings of the Internet Measurement Conference (IMC)*, 2007.

[32] 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, 2001.

[33] D. Stutzbach and R. Rejaie. Evaluating the accuracy of captured snapshots by peer-to-peer crawlers. In *Proceedings of PAM*, Jan. 2005.

[34] D. Stutzbach and R. Rejaie. Improving lookup performance over a widely-deployed DHT. In *Proceedings of INFOCOM*, Apr. 2006.

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

[36] J. Tian and Y. Dai. Understanding the Dynamic of Peer-to-Peer Systems. In *Proceedings of the 6th International Workshop on Peer-to-Peer Systems (IPTPS)*, 2007.

[37] K. Tutschku. A measurement-based traffic profile of the edonkey filesharing service. In *5th Passive and Active Measurement Workshop (PAM)*, Apr. 2004.

PLACE PHOTO HERE

Moritz Steiner Moritz Steiner received his Dipl. Wirt. Inf. (M.S.) degree in Computer Science and Business Administration from the Universität Mannheim, Germany in March 2005. He spent the academic year 2002–2003 at the University of Nice, France. Since April 2005 he is working towards his Ph.D. His advisors are Prof. Dr. Ernst Biersack at Institut Eurecom and Prof. Dr. Wolfgang Effelsberg at the Universität Mannheim.

PLACE PHOTO HERE

Taoufik En-Najjary Taoufik En-Najjary received the B.S. degree in applied mathematics from Mohamed Ben Abdellah University (Fez, Morocco) in 2000, and the M.S. degree in statistics from University Pierre & Marie Curie (Paris VI) in 2001. In October 2001, Taoufik En-Najjary joined France Télécom R&D as Research Engineer working on voice conversion for speech synthesis systems. He obtained his PhD degree in Signal Processing and Telecommunication in April 2005 from University Rennes I (France).

Since January 2005, he is working as PostDoc at Institut Eurecom.

PLACE PHOTO HERE

Ernst W. Biersack Ernst Biersack studied Computer Science at the Technische Universität München and at the University of North Carolina at Chapel Hill. He received his Dipl. Inf. (M.S.) and Dr. rer. nat. (Ph.D.) degrees in Computer Science from the Technische Universität München, Munich, Germany, and his Habilitation à Diriger des Recherches from the University of Nice, France.

From March 1989 to February 1992 he was a Member of Technical Staff with the Computer Communications Research Group of Bell Communications Research, Morristown, US.

Since March 1992 he has been a Professor in Telecommunications at Institut Eurecom, in Sophia Antipolis, France. His current Research is on Peer-to-Peer Systems, Network Tomography of TCP Connections, and LAS Schedling in Edge Routers.