Abstract—Crawling a DHT allows researchers to monitor the behaviour of peers, determine their geographic location, etc. However, it has always been an error-prone process as it is not feasible to capture a full snapshot of the Mainline DHT in a timely manner. Therefore, researchers have developed approximation methods which can run on subsets of the DHT and extrapolate those in order to reason on the size and structure of the complete DHT.

However, in this paper we introduce a new method of collecting information on peers connected to a DHT. We exploit the caches present at bootstrap servers to collect information on peers which are currently connected. Originally added to the bootstrap servers in order to be able to withstand more than 20,000 requests for peers each second, we now use the same mechanism as peers bootstrapping into the DHT to discover more than 20 Million peers in less than 2 hours. Using the bootstrap servers, we discover more than twice as many peers as BitMon, which crawls a subset of the DHT and then extrapolates.

Moreover, in contrast to related work which often require highly customized/tuned BitTorrent clients, our script consists of 90 lines of Python code and runs on a simple laptop.

I. INTRODUCTION

While BitTorrent initially relied solely on trackers in order to find peers downloading the same file, developers added support for a distributed hash table (DHT) in 2008. The DHT, allows peers to locate others downloading the same file, in order to join the swarm, and to download the metadata associated to a swarm.

Currently two incompatible DHT implementations are used in BitTorrent. Both are based on Kademlia [10], the Mainline DHT is mostly used to located peers, while Vuze [6] has a separate DHT which is additionally to store information. In this paper we will focus on the Mainline DHT, as this DHT is more popular as its used by numerous BitTorrent clients.

As most BitTorrent users connect to the DHT, it has become a popular source to monitor their behaviour. Assumptions made while modeling users with respect to average session time, geographical distribution, etc. constantly change and therefore need to be updated. Crawling the DHT allows researchers to do so, and hence methods have been developed to crawl the DHTs [11], [12], at a faster pace [8], and more accurately based on smaller subsets [13]. Collecting a subset is often preferred to collecting a full snapshot of the DHT because of the time it takes to create a full snapshot. Due to the dynamic nature of a DHT and P2P systems in general, inconsistencies will occur if a crawl takes a longer time, hence the general rule of thumb is to crawl as fast as possible.

Crawling a DHT, or part of a DHT, usually follows a similar approach. Initially, one or more instrumented clients connect to the DHT, and fill their routing tables. Next, a region/zone of the DHT is selected and all instrumented clients send lookup messages to all peers in their routing table in order to discover peers in the selected region. Newly discovered peers are additionally sent lookup messages, until the crawl ends due to time constraints or due to not discovering more peers.

However, besides crawling the DHT we can use another source to collect information on which peers are currently connected to the DHT, the bootstrap servers. Overlooked until now, bootstrap servers help BitTorrent peers connect to the DHT if they start their client. Without bootstrap servers, a BitTorrent client cannot connect to the DHT, as it cannot know which peers are online. Whenever, a peer connects to a bootstrap server, it is added to a ping-queue. After being in the ping-queue for 15 minutes, the bootstrap server sends a ping message to the peer. If it replies with a pong it gets added to a cache of good DHT peers, and is going to be used by the bootstrap server while replying to others.

After connecting to the DHT once, peers should be able to reconnect using the peers they connected to in their previous session. Yet, router.utorrent.com, the sole bootstrap server used in µTorrent [4] still receives roughly 20,000 requests per second [3]. Hence, the server is heavily optimized to be able to handle a large number of requests, and has a cache containing 10,000,000 peers [3].

By repeatedly sending a find_node message to a bootstrap server, we can collect the identifier, ip-address, and port of 16 peers closest to the identifier specified in the request each time. Allowing us to retrieve the complete cache of good DHT nodes, which as stated by [3], have replied to a ping message at most 66 minutes ago. In this paper we will present results obtained from such a collection process, relying solely on the bootstrap servers.

II. BACKGROUND

The Kademlia DHT [10] can be visualized as a ring. Each peer joining the DHT generates an identifier (ID) which determines its location in the ring. The identifier is generally computed by hashing the ip-address of a peer in order to generate a 160-bit number. Hence duplicate identifiers can occur, but we assume that the hash function does distribute the peers over the ring uniformly. After determining its location, a peer will perform a lookup to find its neighbors etc. Neighbors are peers which have a similar identifier, determined by computing the XOR distance between their identifiers.

All discovered peers are stored in a routing table. This routing table is divided in buckets of peers which share a
common prefix in their identifiers. Only peers which are “good” are stored in the routing table, which indicates that they have replied to a ping in the last 15 minutes. Using the buckets, a peer can efficiently route queries to their destination, requiring only \( \log N \) hops. When routing a query, a peer will query its routing table to find the peer which is closest to the destination.

III. Setup

In order to collect the peers currently in the caches of the bootstrap servers, we created a small script sending `find_node` messages repeatedly using 20 sockets to three different bootstrap servers. We generate a random peer-id and target-peer for each message, and a random transaction-id. The full specification of the `find_node` message can be found in [9].

We target `router.bittorrent.com`, `router.utorrent.com`, and `dht.transmissionbt.com` as these bootstrap servers are most popular, and run the script on a laptop, connected to the Internet over a 100Mbit connection. The complete script consists of only 90 lines of Python code.

Each response of a bootstrap server contains the ID, ip-address, and port of 16 peers, and during the collection process, we flush all received peers to disk in 100,000 peer increments. After receiving more than 100,000,000 peers we stopped the collection, which took 2.5 hours.

In the post-processing phase, we use MaxMind GeoIP2 City\(^1\) dataset to retrieve a longitude/latitude location for each ip-address collected.

IV. Results

After parsing all 100,000,000 returned peers, we found that we discovered 22,591,915 peer identifiers. Some peers share the same an ip-address, and hence we discovered 18,502,106 ip-addresses. As expected the discovery of new peers slows down substantially, as many already discovered peers are being returned by the bootstrap servers. It took 1.5 hours to discover 20,000,000 peer identifiers, and less than 2 hours to discover 21,000,000.

In contrast, BitMon [8] approximates that there are between 6 and 9 Million peers connected in the DHT\(^2\), substantially less than the 22 Million we discovered. BitMon bases their estimate on a “extrapolation of a representatively chosen sample of peers”, the likely cause of this error.

In Figure 1 we show the distribution of peer identifiers. As expected the peer identifiers are uniformly distributed over the whole ID range. Comparing this distribution, to the one collected by Steiner et al. [11] in their paper on KAD, we can see much less outliers. Steiner et al. attributed their outliers to modified clients which were trying to free-ride the system. In our dataset we cannot find such behaviour.

Figure 2, visualizes the geographic locations of the peers in our dataset. We can observe the following, the peers seem to primarily be located in Europe, the US, Brazil, Japan, and India. In Europe, we can see that there are a lot of peers located in the UK, the Netherlands, and Belgium. Moreover, we can identify some bigger cities in the map, e.g. Madrid, Lisbon, Rome, Athens, Istanbul, Bucharest, Kiev. A similar pattern can be found across other continents, big cities are visible and less populated are empty. Please note that whenever the geolocation dataset could not locate a city, it returned the center of a country as an estimate, clearly visible in Africa were most countries have a single dot in the center.

Another observation is that almost no peers are found to be connecting from within China. We believe that this is caused by BitTorrent users in China using a local BitTorrent client called Thunder, or Xunlei. It was shown by Dughel et al. [7] that Xunlei integrated their own DHT. Moreover, although they did connect to the Mainline DHT in the past, they might be doing this using a different bootstrap server and therefore those peers are not present in our dataset.

Tables Ia and Ib give a summary of the geographic locations of peers in our dataset. Most peers originate from Russia, and Europe in general. This is consistent with previous studies, however the US has dropped to the third place, with only 5.7% of peers. BitMON [8], which continuously monitors the DHT, states that almost 10% of peers originate from the US\(^3\).

Figure 3 shows the ports used by peers, the default port used by BitTorrent (6881) is clearly visible.

---

\(^1\)http://www.maxmind.com/en/city
\(^2\)https://dsn.tm.kit.edu/english/2936.php
\(^3\)https://dsn.tm.kit.edu/english/2936.php
Fig. 2: Plotting the geographic locations of all peers collected.

TABLE I: Distribution of peers in collected data

<table>
<thead>
<tr>
<th>Country</th>
<th>Peers</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Russia</td>
<td>2,972,247</td>
<td>16.3%</td>
</tr>
<tr>
<td>India</td>
<td>1,390,344</td>
<td>7.6%</td>
</tr>
<tr>
<td>United States</td>
<td>1,046,746</td>
<td>5.7%</td>
</tr>
<tr>
<td>Ukraine</td>
<td>947,929</td>
<td>5.2%</td>
</tr>
<tr>
<td>Brazil</td>
<td>910,437</td>
<td>5.0%</td>
</tr>
<tr>
<td>France</td>
<td>614,171</td>
<td>3.4%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Continent</th>
<th>Peers</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Europe</td>
<td>9,534,675</td>
<td>52.4%</td>
</tr>
<tr>
<td>Asia</td>
<td>4,495,192</td>
<td>24.7%</td>
</tr>
<tr>
<td>North-America</td>
<td>1,684,784</td>
<td>9.3%</td>
</tr>
<tr>
<td>South-America</td>
<td>1,487,575</td>
<td>8.2%</td>
</tr>
<tr>
<td>Africa</td>
<td>629,723</td>
<td>3.5%</td>
</tr>
<tr>
<td>Oceania</td>
<td>374,087</td>
<td>2.1%</td>
</tr>
</tbody>
</table>

Port 51413 is the default port of Transmission [5], another popular BitTorrent client, 45682 the default port of µTorrent [4], and finally 55555 is the default port of BitComet [1]. The other ports, we could not trace to a specific client.

V. Remarks

The dataset as presented here, gives an overview of the state of the caches of three different DHT bootstrap servers. A single snapshot was collected at 4th of April 2014, and hence no trends can be observed. As future work, we aim to collect data more often in order to determine the average session times of peers, determine changes in overall size of the DHT etc. Moreover, as this snapshot was collected in 2.5 hours, it will not capture the daily pattern of users accurately, and we expect users in Asia to be underrepresented as for them it was the middle of the night.

However, if we compare our dataset to those previously collected, then we feel the results are consistent. And hence, using bootstrap servers to collect information on peers currently connected to the DHT has shown to be viable alternative.

Finally, we were quite surprised that the bootstrap server did not implement any kind of rate limiting. During the collection process, a single laptop was sending over 500 requests per second to each of the bootstrap servers, resulting in receiving roughly information on 10,000 peers per second. We expected the bootstrap servers to throttle, or even block us at some point, but this did not happen. We thought about using our supercomputer to run multiple scripts in parallel over a 40Gbit connection, in order to reduce the time it takes to collect the complete cache of the bootstrap servers, but opted against this as we did not want to crash the bootstrap servers.

VI. Conclusion

In this paper we have shown that collecting the identifiers, ip-addresses, and ports of peers currently connected in the DHT can be performed in a trivial manner by exploiting the bootstrap servers. In less than 2 hours, we collected information on more than 20 Million peers, without the need for special infrastructure, or expensive equipment.

Moreover, current state-of-the-art estimates based on crawling a subset of the DHT discover substantially less peers
(almost a factor of 3), and therefore their usefulness can be
called into question.

Finally, we have shown that in 2014 the use of BitTorrent is
still widespread, with more than with roughly 10 Million peers
being concurrently connected from within Europe alone. The
upcoming economies (BRIC), attribute to more than 25% of all
BitTorrent peers in the DHT, while the popularity of BitTorrent
in the US seems to be subciding compared to previous studies.

REFERENCES

assisted download acceleration on a massive scale. In N. Taft and
F. Ricciato, editors, Passive and Active Measurement, volume 7192 of
Lecture Notes in Computer Science, pages 231–241. Springer Berlin
A tool for automated monitoring of the bittorrent dht. In Peer-to-Peer
Computing (P2P), 2010 IEEE Tenth International Conference on, pages
1–2, Aug 2010.
org/beps/bep_0005.html.
System Based on the XOR Metric. In P. Druschel, F. Kaashoek, and
A. Rowstron, editors, Peer-to-Peer Systems, volume 2429 of Lecture
Notes in Computer Science, pages 53–65. Springer Berlin Heidelberg,
2002.
In Proceedings of the 7th ACM SIGCOMM Conference on Internet
ACM.
Internet Measurement, IMC ’06, pages 189–202, New York, NY, USA,
2006. ACM.
case of bittorrent mainline dht. In Peer-to-Peer Computing (P2P), 2013