When KAD meets BitTorrent - Building a Stronger P2P Network

Juan Pablo Timpanaro, Thibault Cholez, Isabelle Chrisment, Olivier Festor

When KAD meets BitTorrent - Building a Stronger P2P Network

Juan Pablo Timpanaro, Thibault Cholez, Isabelle Chrisment*, Olivier Festor
INRIA Nancy-Grand Est, France
*LORIA - ESIAL, Henri Poincaré University, Nancy 1, France
Email: {juanpablo.timpanaro, thibault.cholez, olivier.festor}@inria.fr
Email: {isabelle.chrisment}@loria.fr

***Abstract*—The current wave of evolution that leads BitTorrent towards a fully decentralized architecture is both promising and risky. Related work demonstrates that BitTorrent's Mainline DHT is exposed to several identified security issues. In parallel, the KAD DHT has been the core of intense research and was improved over years. In this paper, we present a study that motivates the integration of both worlds. We provide a performance comparison of both DHTs in terms of publishing efficiency. We investigate the security threats and show that the current BitTorrent Mainline DHT is much more vulnerable to attacks than KAD. On the other hand, we demonstrate that the file download service provided by BitTorrent outperforms the one of KAD. Given the strengths and weaknesses of both DHTs, we propose a design in which the two P2P networks can be merged to form a fully distributed, efficient and safe P2P eco-system.**

***Index Terms*—KAD; BitTorrent; DHT; Performance; Security; P2P architecture**

I. Introduction

According to a recent study on Internet traffic [10], P2P applications still generate a large part of it. Among these P2P applications, BitTorrent is the biggest source being ranked first with over 5 million simultaneous users. Despite its success, BitTorrent has recently suffered from several legal issues and complaints from music and movie companies, including legal actions to successfully shut-down major trackers sites, such as The Pirate Bay or Mininova. This accelerated the evolution of the network towards a fully distributed approach offering the same level of service than a central tracker. That is precisely the purpose of the Mainline DHT embedded in the major BitTorrent clients since 2006. Alternatively, a second distributed implementation by the Azureus client was developed, however, it is only used by the Vuze client (previously Azureus). For this reason, we chose to study the Mainline DHT, which is widely implemented.

The same study [10] shows that eDonkey is the second most popular P2P network with around 2 million users. The main client of the eDonkey network, eMule, introduced in 2004 a new fully decentralized P2P network called KAD which was also designed to be compatible with eDonkey. Since the shutdown of the major eDonkey servers in the past years after lawsuits, KAD gained popularity. Thanks to its fully distributed architecture and the open-source nature of its clients (eMule and aMule), KAD has been widely studied and improved. The DHT is mature and evolved towards a robust DHT able to deal with both scale and security.

Given these facts, we argue that BitTorrent could easily benefit from the KAD DHT performance and features for indexing contents while KAD could benefit from the performance of the BitTorrent download protocol, both networks taking advantage of this collaboration. We provide the scientific evidence to motivate this shift. For this purpose, we make the following contributions:

- We compare the performance and properties of both BitTorrent Mainline DHT and KAD DHT.
- We investigate the security mechanisms in place in KAD and compare them with those in Mainline DHT.
- We compare the download protocols performance of both networks
- We propose a new architecture which merges the strengths of KAD at a DHT level (security and indexation scheme) and the BitTorrent download protocol.

To our knowledge, this is the first work that brings together a comparison between these two networks through real world experiments. It is also the first that offers an integration design for both worlds.

The document is organized as follows. In section II, we introduce both KAD and BitTorrent P2P architectures. In section III we summarize the work done on both performance evaluation and security for either BitTorrent and KAD. Our joint evaluations regarding performance and security are made in section IV and V respectively. Focus on the download protocols is made in section VI. The design of an integrated architecture merging both worlds is finally proposed in section VII. A discussion is presented in section VIII and finally a conclusion on the proposed integration and future works are given in IX.

II. Background on KAD and BitTorrent

While BitTorrent and KAD are both P2P networks providing a file sharing application, they use two different architectures to achieve this purpose. The file download service can be mainly divided in three steps: first, a user must retrieve through a search engine a list of possible files being given a set of keywords, second the client must retrieve the sources of the selected files, third it has to initiate connections towards those peers to download the file.

A. BitTorrent architecture

In the regular BitTorrent architecture, the first step is done by using the search engine of a website indexing and delivering torrent files. A torrent file is a collection of several data: the hash of the shared file and its pieces, the IP address of the tracker server in charge of listing the peers sharing that file. The second step is done by contacting the tracker to discover the peers composing the swarm, which is a file-centered entity composed of both seeders (peers that have already completed the download) and leechers (peers currently downloading) interested in a file. Finally, the client connects to these peers in order to download the file following the specific BitTorrent protocol for peers and pieces selection, including the tit-for-tat mechanism, which are actually the core of the BitTorrent protocol.

Retrieving the first peers to join the swarm is normally achieved through a central tracker but BitTorrent introduced DHTs so as to distribute this service. Currently, two distributed trackers can be found for BitTorrent, both based on a Kademlia distributed hash table. The Azureus DHT came first, and it is only used by this client. In second place, the Mainline client introduced its DHT a short time after and it was adopted by the other major clients. Both provide a simple indexation mechanism, in which, being given the ID of a torrent file[1] a peer can perform a DHT lookup to retrieve the list of peers already sharing the file. Like the central tracker, once the first peers have been retrieved, the new peer can join the swarm and share with the others.

B. KAD architecture

KAD is also based on the Kademlia protocol but achieve every step of the file sharing service in a fully distributed way thanks to its double-indexation mechanism. The first level associates keywords with files whilst the second associates files with sources (peers sharing the file). When a file is shared by a peer, the raw data and all the keywords composing its name are hashed separately with a MD4 function generating a KADID for each piece of information. Those KADIDs are then published into the DHT. The information concerning the file (fileID, filename...) are published towards the hash of each keyword (keywordID) to link them to the file. Second, the peer publishes its own information (KADID, IP address, port) towards the hash of the file to be referenced as a potential source. While the second level of indexation is similar to the service proposed by in BitTorrent's DHT, the first level indexing files with keywords is particular to KAD and allows to avoid any central component. Considering the search side, KAD DHT allows a peer to retrieve a list of files being given a set of keywords and a list of peers being given a fileID.

C. Kademlia DHT

As mentioned, the current widely deployed DHT are based on the Kademlia reference design [14] even if they differ between each-other in their implementation. Each node of Kademlia and each stored information has an identifier (commonly of 160 bits) setting its position in the address space of the DHT. All routing tasks are based on the XOR metric used to evaluate the distance between two peers, or between a peer and a key. Routing is done in an iterative way and with parallel lookups by using lookup requests to discover new peers close to a specific address. The routing table is composed of groups of contacts (called a K-Bucket) organized in a binary tree so that the closer an ID is to the current node, the more peers it knows for this part of the DHT. The peers able to index information are those that are close enough to the published hash. This distance is called the "tolerance zone" and is set to the first common 8 bits (most significant). After accepting a publication request for a given resource, a peer is in charge of indexing this specific content, and to answer the related search requests.

[1]The torrent's ID is obtained by hashing the torrent file itself or using a *magnet link*.

III. Related work

Considering BitTorrent as a two-components application, the central-tracker and the swarm, we can argue that it has been the core study in many research works. However, almost all the studies on BitTorrent are focused on the swarm part and the tracker side but very few on BitTorrents two DHTs, the Mainline DHT and the Azureus DHT. Among those, Wolchok et al. [21] conducted a monitoring study on the Azureus DHT. They clearly show how this DHT can be crawled thanks to a Sybil attack, so as to rebuild from scratch a BitTorrent search engine as well as to monitor pirate's behavior. While monitoring the Azureus DHT, the authors of [8] considered the performance of the lookup algorithm and propose new parameters for it reducing the lookup time at the cost of a moderate overhead. Finally, Crosby et al. conducted a detailed comparative study of the two DHTs in [7]. They point out the difference of performances between the two DHTs despite the fact that they are both based on Kademlia and implement the same service. They also highlight many design and implementation problems affecting both the security and performance of these DHTs.

Regarding the security of BitTorrent's DHTs, even fewer studies exist. In our previous work [19] we showed how the Mainline DHT network is wide open to attacks that can dangerously hurt the network, putting in jeopardy users privacy as well as the network performance itself. Recently, Jetter et al. [11] proposed a self-registration mechanism, as a way to avoid a Sybil attack in BitTorrent DHTs. They limit the number of peers'ID per IP, so as to avoid an attacker to launch several peers from a single machine. However, their solution require a jump to a new network, avoiding backward compatibility.

Considering KAD, its performance has been first considered in [18] in which the authors highlight the role of K-buckets in the efficiency and reliability of the routing table. Then [16] proposed a similar study but considered alternative parameters, like the number of contacts per lookup or the time window separating the lookup phase from the service. In [3] they characterize the churn and use this model to design a new

publication policy taking into account the reliability of each peer based on its session time. More recently, [12] investigated the efficiency of the lookup process in KAD, and proposed a new approach to improve the consistency of the results.

KAD also suffered from a vast range of attacks during the past years, resulting in a wide set of proposed protections, which hardened the security of KAD clients. On the one hand, Steiner et al. [17] present how KAD can be misused by the well-known Sybil attack. They showed how this attack can take the control over information stored in the DHT, compromise the privacy of users and be used to launch a DDoS attack without further effort from a single computer. On the other hand, Wang et al. [20] proposed an alternative attack, called *reflection attack*, in which all the entries in the target's routing table contain the target's IP. They proposed an identity authentication to avoid this hijacking, currently implemented by KAD clients. In [13], the authors describe another way to eclipse a content by making the lookup process run indefinitely until timeout.

Finally, we assessed in our previous work [4] some protection mechanisms introduced in KAD. These mechanisms include flood protection, IP verification and identity verification, and make a KAD client resilient to most known-attacks, including the Sybil attack [17] and contacts overwriting [20] attacks. KAD's developers considered the network's flaws and turned it into a strong and mature P2P network. Despite that, some remaining flaws still exist involving more complex distributed attacks targeting specific contents on the KAD DHT as the one we presented in [6]. But we also proposed an efficient way [5] to detect peers trying to attack a DHT entry (which forge their KADID accordingly) and also the countermeasures to avoid them.

IV. DHTs PERFORMANCE COMPARISON

This section aims to point out the main performance characteristics of each DHT network. Despite the fact that they are based on Kademlia, the freedom let in the implementation can lead to major performance variations, as previously shown between the two BitTorrent's DHT [**?**].

We will focus our attention on three main features:

- Time to publish information.
- Overhead during publishing.
- Lifetime of the stored information.

We have chosen to measure the performance regarding the time it takes to publish, but not the time to search, since both operations are expected to be symmetric. In KAD the routing procedure is the same when storing or searching a given information, with a different service message type regarding the service the client is performing. In BitTorrent, both services use the same announce request, used both to publish a torrent and retrieve a list of peers already sharing it.

In all cases, the last version of aMule (2.2.6) was used to test the KAD network. The Vuze client, version 4.5, and its Mainline DHT plug-in, version 1.3.3.1, were chosen to test the Mainline DHT.

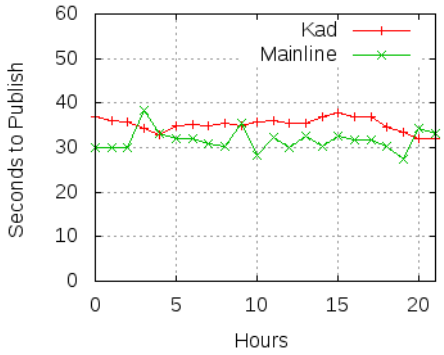

Fig. 1: Time to Publish Random Content.

A. Publishing new content

Publishing content is one of the basic function in any DHT. It mainly consists in finding the adequate peers, the closest ones in Kademlia, for that particular content and storing in them a link to the owner of this content.

The time it takes to publish a new content is a good metric to measure how well the routing algorithm performs. Therefore we conduct an experiment, aiming to measure this property, in which 1000 random IDs were published in both DHT networks during 24 hours.

In the Mainline DHT there is a type of message, **Announce Peer**, which publishes a content and retrieves a list of peers already sharing that content. This message has the same functionality that the **Announce** message in a central-tracker approach (the Announce request is sent to the tracker to join the swarm and, in the meantime, introducing the current peer and retrieving the list of seeders-leechers).

Regarding KAD, a **Store** message is used (for both keywords and files) in order to publish a given information, whilst a **Search** message provides the functionality to retrieve the sources (or files) associated to this content, opposite to the Mainline DHT, in which the same message does both operations.

An extra consideration needs to be made regarding the group of closest peers in which we will publish the content, normally called *replica set*. In KAD the replica set is formed by 10 peers, whilst in Mainline is a group of 8 peers. Therefore, to be fair enough, we will compute the time it takes a full announce in Mainline DHT (publishing in the closest 8 nodes) and the time it takes to store a content (Keyword or File) in KAD in the first 8 peers.

Figure 1 shows the experiment's results. It can be observed that both DHTs perform similar, both in the range of 30-40 seconds. Although Mainline DHT might perform a few seconds faster, KAD includes a 3-seconds time-window between it retrieves the closest peers and it sends the publish requests. It has been proved by STEINER ET AL. in [16] that this time-window can be reduced from 3 seconds to 0.5 seconds, and the routing algorithms will perform faster, without losing performance. Taking into account this time reduction, KAD and Mainline present similar times regarding the publish process.

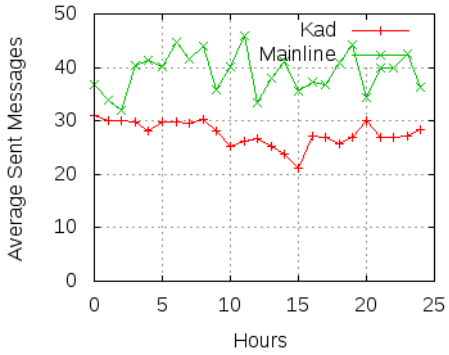

Fig. 2: Messages sent during a publish process.

B. Overhead during publishing a new content

Considering the time to publish a content is a good metric, however the number of messages used in the process needs to be taken into consideration. We compute the messages sent for the 1000 random Ids for both DHTs.

As illustrated in figure 2, the KAD DHT produces between 25 and 30 messages in average for each publication, whilst in the Mainline DHT this value claims to 40 in average. It is important to point out that these messages include routing messages and service messages. The routing messages are used to locate the closest peers, and once those peers were located, then the service requests are issued. The number of service requests is fixed, and depends on the number of peers in the replica set, therefore the difference between the number of messages is based on routing messages.

Additionally, every routing response in KAD contains 3 contacts, opposite to the Mainline DHT, in which a routing response contains 8 contacts. These extra contacts produce an unnecessary overhead, since only a sub-set of them are used to keep routing, whilst the rest are disregarded.

KAD optimizes the routing by responding with the 3 best contacts, which are sufficient to keep routing. We believe that Mainline's responses contains 8 contacts since the churn level of the network is higher, and therefore more contacts in the response might be stale.

C. Information lifetime

The lifetime of the information is extremely important, since it gives an approximation of the re-publishing rate needed to maintain the information in the network. A DHT network with a high called *churn*, needs to re-publish its data more often since the original peers which kept it in the first place might have gone off-line.

In order to measure this lifetime in both DHTs, we publish 1000 random Ids and keep the replica set. Every 30 minutes, we check how many of these peers are still alive. Figure 3 shows the measured aliveness of the peers through time.

Regarding the KAD network, it can be observed that after the first 30 minutes, 84% of the peers in the replica set are alive, and after 5 hours only 50% remain on-line. Finally after a whole day, 72% of the peers in the replica set are dead. Although these churn values might seem high, KAD adapted its publication process accordingly. In KAD there are two types of content to publish, *keywords* and *files*, respectively re-published, every 24 hours and 5 hours. The keyword-file association is permanently refreshed by all the peers sharing a given file and is unlikely to be lost. 50 % of alive peers after 5 hours is an good value that allows to retrieve the source of a file between two scheduled re-publishing.

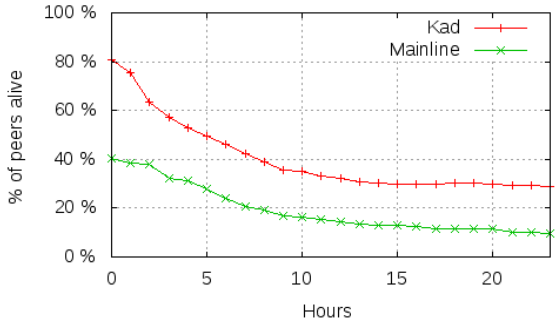

Fig. 3: Measured aliveness of nodes.

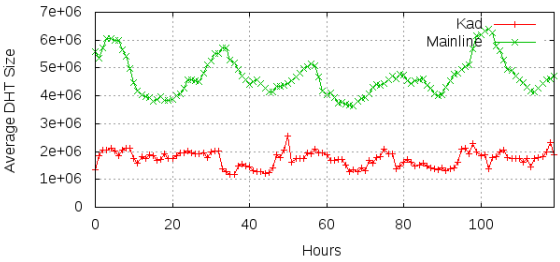

Fig. 4: Measured Population.

On the other hand, Mainline DHT has higher churn levels. After the first 30 minutes, only 41% of the replica set is alive, and considering that this replica set is formed by 8 peers, it means that in average, only 3 peers are on-line. This value decreases to only 9% after one day. A difference with KAD, is that there is not a time window to re-publish a content, and it depends to the original seed to re-announce itself, otherwise the content will be lost.

As long as the seed is on-line, the client will periodically publish the torrent, similar to a periodically announce to a central-tracker. However, if the original seed does not re-announce the content, it is highly likely that new peers do not find this seed, and therefore they can not start the download. This situation is similar when the tracker goes off-line, in a central-tracker approach.

Hence, high level of churn might produce stale downloads. In order to understand this churn behaviour, we computed the population variation of both networks for 5 days. Figure 4 shows the obtained results.

The Mainline DHT network presents an important variation of its population, indicating that it is more a snatch-and-go network, contrary to KAD which presents a more stable behaviour due to the nature of its participants, which stay connected long periods of time. Clearly this high dynamism of peers is linked to the low information availability we previously obtained with our experiment.

Despite the fact we saw differences in the implementations of both DHTs, according to the Kademlia design, there is not a significant gap in terms of performance. This slightly gap is not sufficient to motivate a radical change in the current BitTorrent architecture.

We will show that Mainline DHT presents other weaknesses among security flaws and lack of some features, which will definitely motivate this change.

V. DHTs SECURITY COMPARISON

This section will be organized as follow. Firstly, we will introduce the KAD protection mechanisms that the latest aMule client includes. Secondly, a Mainline client will be tested against a basic routing table attack. Finally, conclusions regarding both DHTs will be presented.

A. Protections Mechanisms in KAD

As we showed in our previous work [4], KAD contains some protections at the routing table level which have been progressively introduced from (0.49a / aMule 2.2.1). Every protection adds more restrictions when adding new contacts, as presented in the following subsections.

1) Flood Protection: A flood protection mechanisms is achieved by keeping a history of all the packet received the last 12 minutes. A threshold is set as a way to restrict the maximum number of messages per peer, and in case this threshold is surpassed, the incoming messages are dropped, unless the threshold is highly overcome, in which case the peer is banned. Additionally, the client uses this protection to drop unrequested messages.

2) IP Limitations: The IP limitation aims to mitigate the Sybil attack from an attacker owning a single public IP. Before adding a new contact, the IP is checked, and in case it is already used, the contact is dropped. This makes a Sybil attack much harder to perform, since an attacker will need several public IPs to have a significant effect. This IP limitation also affects contacts from the same /24 subnet, since it is no possible to add more than ten contacts to the routing table coming from this subnet. Moreover, this ten contacts need to be in different K-buckets, so as not to allow an attacker owning a /24 subnet to position Sybils very closed to the target ID, and launch a localized attack.

3) IP Verification: Finally, an IP verification aims to avoid identity spoofing (both IP and ID) by introducing a three-way handshake before adding a contact.

This three protections stack mitigates a Sybil attack from a single IP or even a /24 subnet. It is still possible to launch a distributed attack using several IPs, however we proposed in [5] an ID distribution analysis which can successfully avoid this kind of attacks and fully compatible with the KAD network.

B. Assessing Mainline DHT

In order to establish the protection level in Mainline DHT, we ran the attack described by STEINER ET AL. in [17], commonly known as a routing table poisoning from a single peer, which is a basic attack. Successfully achieving this attack will prove that the network is open to the most basic vulnerabilities, and further complex attacks can be easily performed.

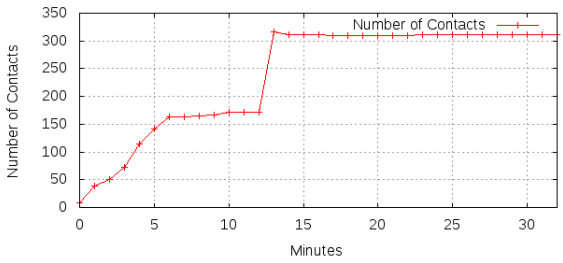

Fig. 5: Number Sybils of contacts during attack on Mainline.

The idea of the attack is to add as many fake contacts, the Sybils, into the target's routing table. The target peer will eventually route requests through these Sybils, but especially respond with these to routing requests from other peers.

The attack consists in sending several *ping* requests (equivalent to the *Hello* request in KAD) to the same target peer, using an unique IP and different ports. A *ping* request contains the IP and port of the receiver and the ID of the sender. In our case, every *ping* message will contain a random ID varying from 0 bits to 159 bits in common to the target peer ID.

Figure 5 shows the number of contacts of the target peer. Firstly, we let the target peer bootstrap and reach a steady number of contacts, normally around 170 contacts, which are positioned in the first 22 buckets (sharing between 0 bits and 22 bits in common). At the minute 12 the attack begins, sending 160 *ping* requests. It can be observed that the routing table gets filled instantly with approximately 140 new contacts (Sybils), reaching around 310 contacts in total.

Because the target peer's first buckets are already full with normal (not Sybil) contacts, no new contacts can be added. However, launching the attack as soon as the target peer is on-line will successfully poison the first buckets, achieving a total number of 160 Sybils (one Sybil per bucket).

This first experiment consists in positioning one Sybil per bucket. However, we performed a second experiment aiming to fully fill each bucket, which means 8 Sybils per bucket. Therefore, we sent 1280 (8 * 160) *ping* request properly scheduled to the target peer. Notwithstanding, only one Sybil succeeded to enter a bucket, because Mainline DHT does not allow two contacts in the same bucket from the same IP. This feature protects the routing table from a full poisoning (8 Sybils per bucket), but not from a partial poisoning (1 Sybil per bucket). However, this behaviour resembles an implementation limitation, more than a real security mechanism to avoid routing table poisoning. However, in the lowest buckets (the buckets close to the target), only Sybils can be found and will be returned at the end of each lookup process which is actually the major threat.

On the other hand, figure 6 shows the number of Sybil contacts during the same attack on a KAD client. It can be observed the quick increase of Sybils during time, in the case

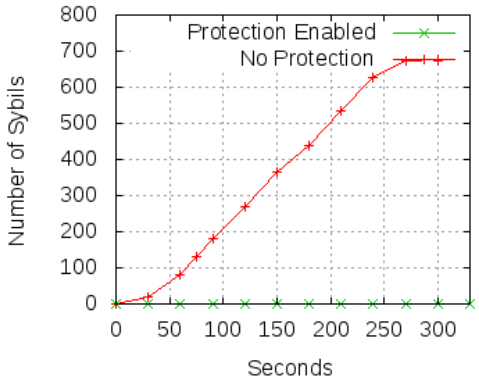

Fig. 6: Number of Sybils contacts during attack on KAD.

no protection is activated. On the opposite side, when the client enables the protections, no Sybil are place in the routing table.

We have shown that the Mainline DHT is open to the basic attack from a single machine, on the contrary to KAD, in which the IP limitation takes care of this attack. We strongly believe that this lack of protection in the network is the main reason to migrate to a more secure DHT.

VI. Download Protocol

The performance characteristics along with its security features at a DHT level make KAD an excellent network. Not only because its routing algorithm achieves good lookup times, but because it is resilient to most known attacks. The churn level measured in this network is a key feature to take into consideration, since it strongly affects the final performance, even though it does not depends on the DHT design itself.

Although the DHT level is considerably important, the final users will not perceive its performance in all cases, but the performance of the download algorithm. It is pointless to have a client that can successful publish and search for a group of sources for a given file in record time, but takes a long time to download the file. Clearly, the performance of the search procedure, a DHT-based in this case, will be overshadowed by a poor download protocol.

In order to determinate the performance of KAD and BitTorrent download protocols, we conducted the following experiment. We used 50 peers from PlanetLab's infrastructure and a single 700 MB random file, to measure the time it takes to download the file for an aMule client and a Vuze client. We ran two experiments, varying the number of initial sources, or initial seeders, in the case of BitTorrent. Both terms refer to the same concept (a peer having the entire content) and will be used indistinctly.

A. Download time with one Source

Starting with one source is the normal behaviour, since the original publisher initially uploads itself as the only source, and eventually new peers complete the download and become new sources. Figure 7 shows a graphical comparison between BitTorrent clients (Vuze clients) and KAD clients (aMule clients). While BitTorrent clients achieved a total download

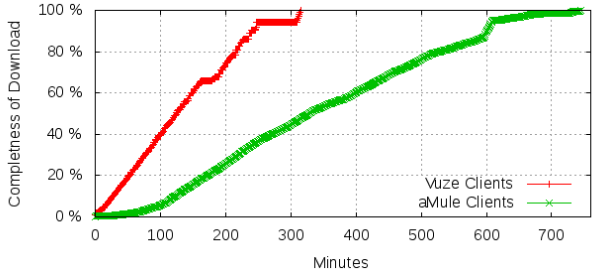

Fig. 7: Download Times with 1 Source.

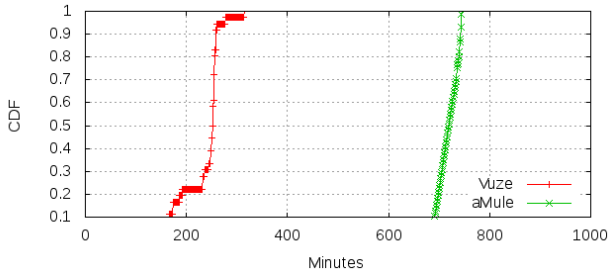

Fig. 8: Peers with download complete (1 source).

of the file in 315 minutes, KAD clients achieved it in 745 minutes, a bit more than 12 hours.

Additionally, figure 8 shows the percentage of clients that complete the download through time. aMule clients present a more lineal distribution, which indicates that the completeness of the download is achieved at the end by all the peers. This characteristic is expected since all the peers present the same ADSL-like bandwidth (1000kbit/s down and 300kbit/s up) and similar initial status (none of them contains the file nor pieces of it).

On the other hand, Vuze clients present a step-style distribution. This can be due to the nature of the peers selection process, however a study of the piece exchange protocol and BitTorrent's algorithms exceeds the scope of this work. Further characteristics of the BitTorrent's algorithms and an extensive analysis can be found in the work done by Legout et al. in [2].

At this point, we can observe that BitTorrent's download protocol performs better and achieve the download in 42% of the total download time regarding KAD.

However, most of the clients trying to download a file will find themselves with a higher number of sources, since in the case of popular contents, the total number of sources climb very fast. Therefore, we conducted a second experiment, with 10 sources and measured the download times.

B. Download time with ten Sources

Although ten sources is not close to the average number of sources for a popular content (around 1000 or even more for real popular content), it might be accurate enough for small torrents, or KAD files. These small torrents or files might be alive for long periods of time, despite a low number of sources.

Figure 9 shows the time it takes to download the same 700 MB file from the Vuze clients and the aMule clients. In this

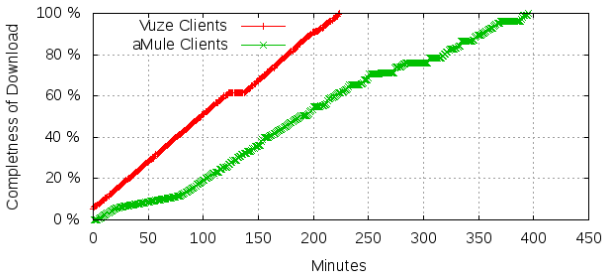

Fig. 9: Download Times with 10 Sources.

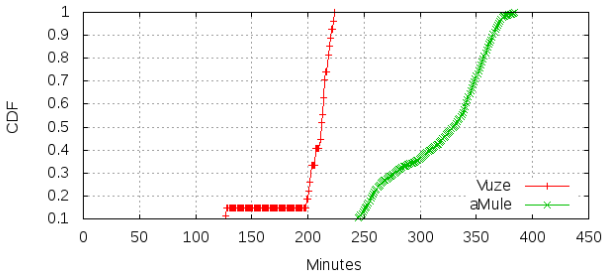

Fig. 10: Peers with download complete (10 sources).

case Vuze clients take 224 minutes to complete the download, whilst aMule clients take 395 minutes. Even though in this case the Vuze clients performed the download in 57% of the total download time regarding KAD, BitTorrent's download protocol clearly surpassed KAD's.

Figure 10 presents the cumulative distributions of peers with their downloads complete. In this case Vuze clients present a linear-like distribution, showing that a small group of peer (14% of them) complete the download at the beginning, whilst the rest complete the download close to the end.

As a conclusion it is observed that the BitTorrent's download protocol surpassed KAD's. We measured the time to download when 2% of the peers are sources (1 source among 50 peers) and when 20% of peers are sources (10 peers out of 50). In every case, BitTorrent's clients achieve the download in 50% of the time it takes to the KAD's clients to complete it. However, this experiments did not take into account concurrent downloads, in which the final results might vary.

Although we can not generalize this behaviour for every swarm configuration, we have shown the BitTorrent's download protocol performance comparing to KAD's.

VII. Merging the two networks: Proposed Architecture

After an extensive analysis of these well-known P2P networks, we proposed the architecture shown in figure 11, so as to merge the best features of each approach namely, (1) KAD's DHT security properties and its 2-level indexation solution and (2) BitTorrent's efficient download protocol.

The integrated architecture has two main components: the indexation block and the download block. The download component is based on the BitTorrent download protocol, including its tit-for-tat rewarding mechanism. The indexation component is implemented using the KAD double-indexation mechanism.

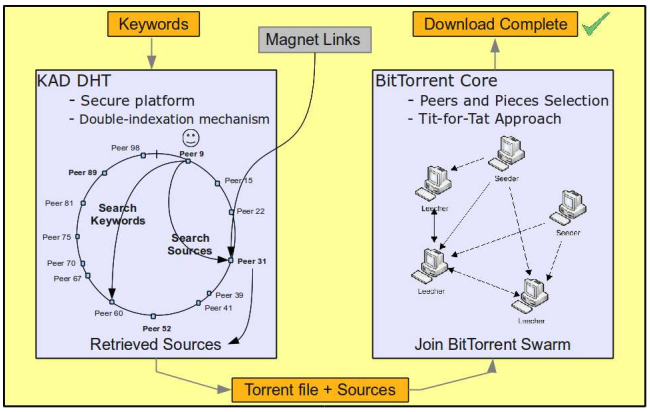

Fig. 11: Graphical view of the joint approach.

A user introduces a set of keywords which defines a given content and retrieves a list of peers having that content (sources for that content). Once the user retrieves a list of peers for that content, it will join them, in order to start a download.

The KAD DHT will now also keep data from the BitTorrent network as well, allowing users to store (or search) torrents by keywords, and store (or search) sources for a given torrent. Moreover, KAD's tags can be used during the indexation process. Tags enable the user to accurately select a torrent, based on the keywords it has associated. In case of torrents, this information is already specified by the torrent's detail, so a smooth transition can be made to generate tags. Additionally, magnet links will be still useful, since the user can retrieve the list of sources, skipping the first level of indexation.

This new keyword-level indexation will complement the web-based lookup for torrents, and hopefully will replace the jungle of torrent search websites.

This architecture will allow KAD users to benefit from the BitTorrent download protocol, which has been proven to achieve excellent download times, whilst maintaining a secure and strong DHT support. On the other hand, BitTorrent users can finally leave behind an almost obsolete central-tracker approach for a fully distributed tracker, based on a well-know and mature distributed hash table.

We can see a last benefit of merging those networks. When a file has been downloaded through the BitTorrent part of the merged approach, it can then automatically be shared on the regular KAD part. In that way, even if the torrent file of the content is lost (due to lost of popularity and therefore no longer reachable in the BitTorrent network), the content is still reachable through KAD own file sharing process. As a result, KAD keeps a memory of those files that once were popular in BitTorrent, even though the torrent file is no longer available.

VIII. Discussion

In this section we will discuss (1) Why we do not to patch up the current implementation of the Mainline DHT, and (2) Why we do not to consider well-known solutions, like identity-based approaches or a central authority for authentication.

From a security point of view, there are no disadvantages in implementing the security protections in the Mainline DHT,

even though it will require that every client implements these protections. The main reason to use the KAD DHT, and not the Mainline DHT is that the first one has the double-indexation mechanism. This feature allows the indexation of torrents, one of the main features of the proposed joint architecture.

The KAD DHT is, currently, vulnerable to distributed attacks, in which an attacker uses several nodes to achieve the same Sybil attack. However, the IDs distribution analysis proposed in [5] successfully mitigates this attack. There are alternative mechanisms, such as the external certification service propose by Fantacci et al. [9], the *Likir* framework proposed by Aiello et al. [1], and the set of security considerations by Sit et al. [15]. However, none of these approaches consider backward compatibility, and all of them require several modifications to the DHT protocol. Although a modified DHT will be suitable to fulfill the security problems, it is not viable to migrate million of users.

We consider backward compatibility as a mayor design goal, and the IDs distribution analysis solution can be implemented in a every client, without requiring a major change in the network. Each client can progressively incorporate this protection, making the network each time more resilient.

IX. Conclusion

BitTorrent and KAD are two popular P2P networks used by millions of active users. Originally, BitTorrent used a central tracker to retrieve a set of peers in order to start a download, whilst in KAD this process is originally decentralized. However, in the past years and due to legal threats against central trackers, the BitTorrent community evolved to support decentralized trackers. In this paper we focused our attention on the Mainline DHT, which is the most widely used decentralized tracker in BitTorrent.

We compared the Mainline and the KAD DHT in terms of performance characteristics, security features and download properties. We demonstrated through real experiments the lack of security mechanisms in the Mainline DHT, lack which makes the network prone to the most basic P2P attacks. We also showed how KAD is resilient to these attacks. On the other hand, we reached the conclusion that BitTorrent's download protocol is around twice faster than KAD's one in a real network environment.

Finally, we presented a joint approach, merging these two great networks into a single one, integrating the KAD security mechanisms and BitTorrent's outstanding download performance. The result is a robust and strong P2P network, with a mature DHT support and an excellent download protocol. This wedding also introduces new features like keyword indexation of torrents and a memory of contents without torrent. We strongly believe that such an architecture brings together the best of both P2P networks.

Future work will be based on testing the implementation of the associated implementation on a real network environment and the provision of smooth transition mechanisms.

References

[1] Luca Maria Aiello, Marco Milanesio, Giancarlo Ruffo, and Rossano Schifanella. Tempering Kademlia with a robust identity based system. In *P2P '08: Proceedings of the 2008 Eighth International Conference on Peer-to-Peer Computing*, pages 30–39, Washington, DC, USA, 2008. IEEE Computer Society.

[2] Stevens Le Blond, Arnaud Legout, and Walid Dabbous. Pushing BitTorrent Locality To The Limit. *Computer Networks*, In Press, Corrected Proof:–, 2010.

[3] Damiano Carra and Ernst W. Biersack. Building a Reliable P2P System Out of Unreliable P2PClients: The Case of KAD. In *CoNEXT 2007,December 10-13, 2007, New York, NY, USA*, 2007.

[4] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Evaluation of Sybil Attacks Protection Schemes in KAD. In *AIMS 2009 Scalability of Networks and Services*, Enschede Pays-Bas, June 2009. University of Twente.

[5] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Efficient DHT Attack Mitigation Through Peers' ID Distribution. In *HotP2P 2010*, Atlanta, United States, April 2010. IEEE International Parallel & Distributed Processing Symposium.

[6] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Monitoring and Controlling Content Access in KAD. In *International Conference on Communications - ICC 2010*, Capetown, South Africa, May 2010. IEEE.

[7] Scott A. Crosby and Dan S. Wallach. An Analysis Of BitTorrent's Two Kademlia-based DHTs. Technical report, Rice University, 2007.

[8] Jarret Falkner, Michael Piatek, John P. John, Arvind Krishnamurthy, and Thomas Anderson. Profiling A Million User DHT. In *ACM SIGCOMM '07*, pages 129–134, New York, NY, USA, 2007. ACM.

[9] Romano Fantacci, Leonardo Maccari, Luigi Chisci, Luca Maria Aiello, and Marco Milanesio. Avoiding Eclipse Attacks on Kad/Kademlia: An Identity Based Approach. In *ICC '09: Proceedings of the International Conference on Communications*, 2009.

[10] Ipoque. Internet Study 2008/2009. http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009.

[11] Oliver Jetter, Jochen Dinger, and Hannes Hartenstein. Quantitative Analysis of The Sybil Attack And Effective Sybil Resistance In Peer-to-Peer Systems. In *IEEE ICC*, 2010.

[12] Hun Jeong Kang, Eric Chan-Tin, Nicholas Hopper, and Yongdae Kim. Why Kad Lookup Fails. In *Peer-to-Peer Computing*, pages 121–130, 2009.

[13] Michael Kohnen, Mike Leske, and Erwin P. Rathgeb. Conducting And Optimizing Eclipse Attacks In the KAD Peer-to-Peer Network. In *NETWORKING '09*, pages 104–116, 2009.

[14] Petar Maymounkov and David Mazières. Kademlia: A Peer-to-Peer Information System Based On The XOR Metric. In *IPTPS '01*, pages 53–65, 2002.

[15] Emil Sit and Robert Morris. Security Considerations for Peer-to-Peer Distributed Hash Tables. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 261–269, London, UK, 2002. Springer-Verlag.

[16] Moritz Steiner, Damiano Carra, and Ernst W. Biersack. Evaluating And Improving The Content Access In KAD. *Springer "Journal of Peer-to-Peer Networks and Applications", Vol 3 N2, June 2009*, 2009.

[17] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. Exploiting KAD: Possible Uses And Misuses. *SIGCOMM '07*, 2007.

[18] Daniel Stutzbach and Reza Rejaie. Improving Lookup Performance Over A Widely-Deployed DHT. In *INFOCOM '06*. IEEE, 2006.

[19] Juan Pablo Timpanaro, Thibault Cholez, Isabelle Chrisment, and Olivier Festor. BitTorrent's Mainline DHT Security Assessment. In *NTMS '11*, Paris, France, February 2011. IEEE.

[20] Peng Wang, James Tyra, Eric Chan-Tin, Tyson Malchow, Denis Foo Kune, Nicholas Hopper, and Yongdae Kim. Attacking The KAD Network. In *SecureComm '08*. ACM, 2008.

[21] Scott Wolchok and J. Alex Halderman. Crawling BitTorrent DHTs for Fun And Profit. In *Proc. 4th USENIX Workshop on Offensive Technologies*, August 2010.