On Understanding the Existence of a Deep Torrent

Rafael A. Rodríguez-Gómez[1], Gabriel Maciá-Fernández[1], and Alberto Casares-Andrés[2]

[1]Dpt. of Signal Theory, Telematics and Communications, CITIC-UGR, Spain
rodgom@ugr.es, gmacia@ugr.es
[2]4IQ, Los Gatos, CA, USA
alberto.casares@4iq.com

***Abstract*—Nowadays, a great part of the Internet content is not reachable from search engines. Studying the nature of these contents from a cyber security perspective is of a high interest, as they could be part of many malware distribution processes, child pornography or copyrighted material exchange, botnet command and control messages, etc. Although the research community has put a big effort on this challenge, most of the existing works are focused on contents that are hidden in Web sites. Yet, there exist other relevant services that are used to keep and transmit hidden resources, such as P2P protocols. In the present work, we suggest the concept of Deep Torrent to refer to those torrents available in BitTorrent that cannot be found by means of public Web sites or search engines. We present an implementation of a complete system to crawl the Deep Torrent and evaluate its existence and size. We describe a basic experiment crawling the Deep Torrent for 39 days, in which an initial estimation of its size is 67% of the total number of resources shared in BitTorrent network.**

I. Introduction

In the early days of the Internet, crawling the Web was a relatively easy task. Search engines were able to index almost all the contents in the Web. Yet, after a few years, Web contents have considerably evolved to a more "dynamic" behaviour, *e.g.*, Web servers often use databases to build and serve dynamic Web pages. As pointed out in 1994 by Jill Ellsworth, this evolution leveraged the apparition of the *invisible* Web. In 2001, Bergman [1] divided the Web contents into a *Surface Web*, *i.e.*, the indexed content crawled by search engines, and a *Deep Web*, containing all the dynamically generated content as a response to query forms. Nowadays, the *Deep Web* is more generally defined as the informational content on the Internet that present any of the following characteristics: *(a)* it is not accessible through direct queries made by conventional search engines; *(b)* it is only accessed through specific and targeted queries or keywords; *(c)* it is either not indexed or cannot be indexed by conventional search engines; *(d)* it is somehow protected by security mechanisms, like login IDs and passwords, certificates, membership registrations, codes, etc.

Crawling the Deep Web is a challenging task, not only due to the hidden nature of its contents, but also because of its large scale. In 2001, Bergman [1] estimated that the Deep Web was 400-550 times greater than the Surface Web, and other authors [2] gave some insights about its size, including more than 307,000 sites, 450,000 databases, and 1,258,000 interfaces, describing an increase in size by 3-7 times during the period 2000-2004. In 2007, some Deep Web directory services started to index databases in the Web, although their coverage were still small, ranging from 0.2% to 15.6% [2]. In 2015, the data stored on just the 60 largest Deep Web sites was estimated to be 40 times larger than the size of the entire Surface Web [3].

From a cyber security perspective, discovering and analyzing the structure and dynamics of this huge amount of hidden contents is of paramount importance. Many illegal activities in the cyber space are based on the existence of these hidden contents. Common examples of this fact are the presence of malware propagation mechanisms, botnets communications, exchange of child pornography or copyrighted contents, etc.

The use of services like P2P is relevant for some of these illegal activities. As an example, some of the botnets studied in previous research works use *Command & Control* mechanisms that are based on existing P2P networks (parasite P2P botnets) [4]. Despite this fact, the crawling and analysis of the resources shared using these protocols has received little attention so far. Again, we can conclude that, from a cyber security perspective, studying the contents in P2P networks is essential.

In this paper, we focus on the Mainline implementation of BitTorrent, the most used P2P network nowadays, called Mainline. The publication of a resource in BitTorrent is done by somehow sharing a `torrent` file containing metadata related to the content description and location of the shared resource. These `torrent` files are either published in the public Web (specific web sites for torrent files that will be referred to as *torrent-discovery sites* in advance) or simply sent to the interested users in an out-of-band channel (email, deep web, IRC, etc.). This mechanism for publication lead us to make an analogy with the classification of contents in the Web, and divide the BitTorrent resources into two parts: those that are publicly announced in the Web, that we denote as the *Surface Torrent*, and those that remain hidden to the general public and are shared in private communities, *i.e.*, the *Deep Torrent*. Note that the concept of *Deep Torrent* does not include only *private torrents*. In fact, in this work we do not consider private torrents, as they have been analyzed in other works [5]. We specifically focus on resources that are being announced by out-of-band mechanisms instead of public web sites, while still being shared in public trackers or DHT.

In the present work we focus on demonstrating the existence of such a Deep Torrent and evaluating its size. We propose the use of a combined crawler for both the *Surface* and the *Deep Torrent* based on: (*i*) a Mainline monitoring module, and (*ii*) a

Web crawler for torrent-discovery sites. The system combines the output of these two modules to obtain a list of Deep Torrent resources. We make an evaluation of the system for a 39 days period extracting experimental results about the Deep Torrent. Up to our knowledge, there is no previous research published on analyzing or describing this phenomena.

The rest of the paper is organized as follows. In Section II, some related work is presented. In Section III, some fundamentals of BitTorrent-based networks are given. After that, the overall functional architecture for the proposed Deep Torrent crawling system is detailed in Section IV. Section V describes the preliminary results obtained from our proposed system. Finally, Section VI draws the main conclusions and points out directions for future work.

II. Related Work

Since the work by Bergman in 2001 [1], there have been some efforts in the research community to investigate the magnitude and features of the Deep Web. These efforts have been concentrated towards two directions.

The first one is related to the understanding of the nature of hidden contents and the methodologies to automate the data extraction from the Deep Web sites [6]. The second direction of research is related to optimizing the number of queries used to dig the Web in order to obtain the maximum percentage of the hidden contents [7]. In the first prototype of our system we are aligned with the first direction of research, as we are really interested on getting an overview of the features of the Deep Torrent, without caring too much about efficiency.

Additionally, there exist works specifically focused on crawling *torrent-discovery sites* [8], [9]. These works are not really focused on extracting information about the Deep Torrent, as they are only able to get information about the torrent files publicly published in torrent-discovery sites (*Surface Torrent*). Among torrent-discovery sites, it is worth to mention the existence of the so called DHT search engines, which publish information (magnet links instead of torrent files) about resources being shared in the BitTorrent DHT. The first engine capable of searching BitTorrent DHT was `btdigg`. This engine was active during our research period and closed in June 2016 for several months. It is currently active again under a different domain (https://btdig.com/). In December 2016, a new DHT search engine called `Alphareign` (https://alphareign.se/) have just appeared. Up to our knowledge, these are the only DHT search engines up to date.

Regarding the research efforts related to monitoring activity in BitTorrent, in our previous work [4] we developed a monitoring system to detect files belonging to P2P parasite botnets. There are other similar approaches in the literature, like [10], [11], [12]. Unluckily, all of them are aimed to analyze some features of BitTorrent files without paying attention to the Deep Torrent phenomena.

Finally, it is important to highlight that the combination of both modules, *i.e.*, a Surface Torrent crawler in the Web and a BitTorrent monitor, in a complete system to crawl the Deep Torrent is not present in the literature and it represents a contribution of this work.

III. BitTorrent General Concepts

The BitTorrent protocol is used to share resources among peers in a large network. For every resource shared in BitTorrent, the nodes of the network can play different roles: *seeders* are nodes that contain a complete copy of a shared resource; *leechers* are those that have partially downloaded the considered resource —note that leechers really download the parts of a resource not only from seeders, but also from other leechers—; finally, *trackers* are special nodes in the network that keep track of the leechers and seeders for every shared resource.

To locate the resources shared in the network, the BitTorrent protocol uses `torrent` files, which contain metainformation about resources and, when necessary, about their corresponding trackers. The 20 bytes SHA-1 hash of the `info` section of a `torrent` file is called `infohash`, and it uniquely identifies a resource in the network.

`Torrent` files are stored in torrent-discovery servers (normally Web based) that allow users to search contents and then get the corresponding `torrent` file to start the corresponding download. Some examples of these torrent-discovery sites are: https://thepiratebay.org/, https://torrentdownloads.me/ or http://extratorrent.cc/, among others.

Since 2005, the BitTorrent protocol implements a distributed operation mode that does not require the participation of trackers. It was first implemented in the Azureus torrent client (currently known as Vuze). In this operation mode, a *distributed hash table* (DHT) is used to store the correspondence among the resources and the peers that share them. Here, we could say that each peer plays the role of a tracker. Currently, there exist two different incompatible implementations of DHT: Vuze and Mainline. Both are specific implementations of Kademlia [13]. In this paper we focus in Mainline [14], as its use is more widespread [8].

Mainline uses 20 bytes unique identifiers for both nodes and resources (`infohash`) in the DHT network. In the case of nodes, they are known as `nodeID`s, and are randomly generated the first time a BitTorrent client is initiated. These identifiers will not change unless a user manually uninstalls the BitTorrent application or changes its configuration file. Even if a user changes its IP address, its `nodeID` will remain and, for this reason, we can assume that `nodeID` is a unique identifier per user.

In Mainline, a metric for the closeness between a DHT node and a resource is defined as the XOR operation between their corresponding identifiers: `nodeID` and `infohash`. The DHT nodes that are closer to a resource are in charge of keeping track of the list of peers that are sharing it.

There are four queries in the Mainline DHT protocol:

- `ping`: verifies if a peer is alive and responsive.
- `find_node`: requests a node for the list of closest nodes to a given `nodeID` in its routing table. A response message is issued with the IP address, port, and `nodeID` of every node in this list.
- `announce_peer`: announces that a peer holds the resource (or a part of it) identified by its `infohash`.

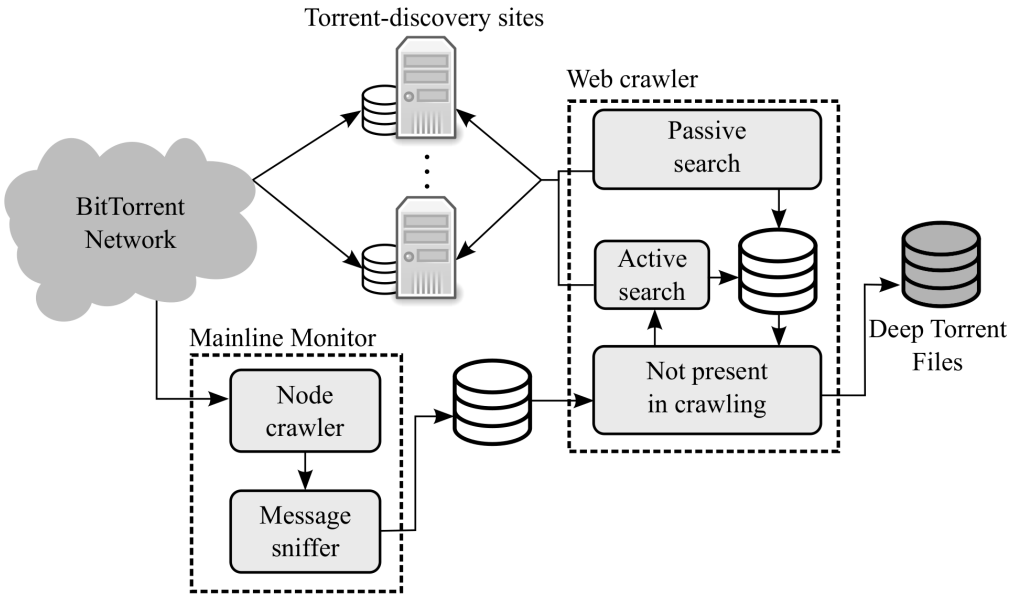

Fig. 1: Functional architecture of the Deep Torrent crawler.

- `get_peers`: get a list of peers associated with a `infohash`. If the queried DHT node does not have this information, it returns the eight nodes in its routing table closest to the `infohash` supplied in the query.

Then, if a peer wants to announce that it has a copy of a given resource $\texttt{infohash}_i$, it has to find first the list of peers that are closest to $\texttt{infohash}_i$. With this purpose, it sends `get_peers` messages that iteratively reach the nodes in the DHT containing this information, thus getting the response. After that, an `announce_peer` message is sent to the nodes in the list of peers. As this information expires after a time-out that depends on the client implementation (around 30 minutes), the announcing peer is responsible for re-announcing the tuple $\texttt{<IP:port,infohash}_i\texttt{>}$ over time.

Note that announcing the resources is a necessary condition to allow other nodes in the Mainline network to download them. Based on this fact, we reduce the problem of monitoring the shared resources to that of monitoring `announce_peers` messages in the network. In what follows, we describe how we manage to achieve this.

IV. Deep Torrent Crawler

The proposed Deep Torrent crawler is based on two modules (see Fig. 1): (*i*) a Mainline monitor, and (*ii*) a Web crawler for *torrent-discovery sites*. The Mainline monitor module is in charge of obtaining the resources that are being actively announced in the BitTorrent network. In parallel, the Web crawler extracts resources that can be found in *torrent-discovery sites* (Surface Torrent). Finally, both data are combined to find the resources that are being really announced in the BitTorrent network but cannot be found in the *torrent-discovery sites*. Following our own definition, these would be the resources that belong to the Deep Torrent.

A. Mainline Monitor Module

The monitoring module for the Mainline network is based on our previous work [4] and it is composed of two submodules: *(a)* a node crawler, and *(b)* a message sniffer.

1) Node Crawler: The purpose of the node crawler is to obtain all the active peers in a specific zone of the Mainline network and to maintain this list updated. A zone of the network is defined as all the identifiers with a common prefix. For example, the crawler can monitor a eight bits prefix zone by extracting all the active nodes in the network whose `nodeID` begins with the same eight bits.

The crawling process starts getting a list of nodes in the monitored zone, `known_list`, by recursively sending `find_nodes` messages to some hardcoded bootstrap nodes. Once it has a minimum number of known nodes, two threads are launched. One is periodically asking the nodes in `known_list` about new ones, and the other thread is receiving their answers and registering the new nodes into `known_list`.

2) Message Sniffer: Its aim is to include our monitor node in the routing tables of the DHT nodes previously collected in `known_list`. To accomplish it, this module periodically sends `ping` messages to the DHT nodes, indicating that it is alive and responsive. In this process, we forge the source `nodeID` so that many different sybil nodes are included in routing tables. As we are interested in receiving the same messages as the nodes in `known_list`, the fake `nodeID`s are chosen so that they are close to them.

In summary, the Mainline monitoring procedure works as follows. First, we obtain the active nodes of a specific zone by using the node crawler module. After that, we try to be inserted into the routing tables of these nodes by including our sybils as neighbors. As a result, legitimate nodes send `announce_peer` messages to our sybils when they are

sharing a resource with `infohash` in the monitored zone. We log all these `announce_peer` messages into a database, registering the `infohash` of the announced resource, IP address, port, `nodeID` of the announcer node, and the message arrival timestamp.

Note that this module does not alter anyway the proper operation of the monitored zone. The only effect is that real nodes in the monitored zone will send some extra messages to our sybils.

B. Web Crawler Module

Recall that torrent-discovery sites publish `torrent` files that are used to start the download of a specific resource. These sites usually have a query interface that allows users to obtain information related to the searched torrent resources. Based on this information, a user is able to decide which is the best `torrent` file for downloading a given resource.

In order to make our crawler capable of extracting this knowledge, we use two methods:

- *Passive search*: information announced in the torrent-discovery sites is obtained by using Rich Site Summary (RSS) feeds.
- *Active search*: we also query special Web sites for the resources already identified in the monitoring of the Mainline network and focus on those that have not been previously identified in the RSS data source.

Regarding passive search, RSS feeds of the monitored sites are periodically queried by our crawler and all the announced resources are stored in a database of known resources. The information stored in this database is: *a)* unique identifier of the resource (`infohash`); *b)* name of the resource; *c)* size in bytes; *d)* number of seeders and leechers; *e)* timestamp at which this resource started to be shared; *f)* Web site from which this information was obtained; *g)* timestamp of the instant at which the crawler got the information.

The idea of the active search is leveraging the information already extracted from the Mainline monitor module to make a deeper search of indexed resources in the torrent-discovery sites. Here, all the resources identified in the Mainline monitor module that have not been found in the queried RSS are first identified. For each of them, using the `infohash` announced in Mainline, a new specific query is launched to certain Web sites that allow searching a torrent by its `infohash`. Only when a resource is not found at this point, it is labeled as a hidden resource and stored in the Deep Torrent resources database.

V. Measurement Results

We have monitored a part of the Mainline network during 39 days, from March 16th 2016 until April 24th 2016. During this period, two Mainline monitors have been launched to monitor the zones with a 8-bits prefix equal to `0x09` and `0x10` respectively. This represents a part out of 128 of the complete Mainline network (2 zones of a size of one out of 256 each). As `nodeIDs` are randomly assigned, we consider that this sample is representative of the behavior of the whole Mainline network. The main reason to use two different sensors is to check if the obtained results are biased for a specific zone.

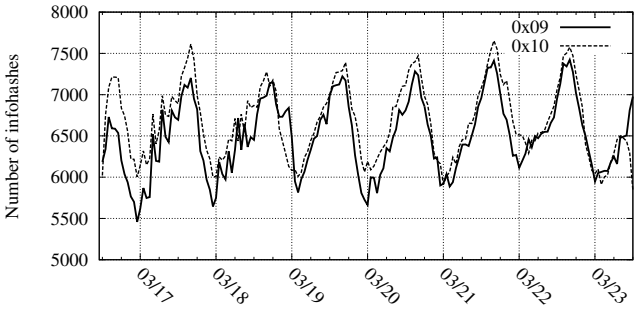

Fig. 2: New infohashes discovered every hour during the first week.

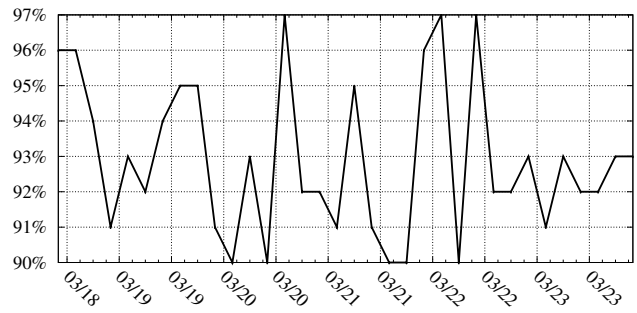

Fig. 3: Percentage of coincidence between infohashes monitored by three different sensors in the zone `0x09`.

We have conducted a preliminary experiment to check the accuracy of our crawler when sniffing the infohashes announced in a given zone. Three different instances of the Mainline monitor have been launched in the same zone (`0x09`). In this setup, our estimation for the percentage of resources monitored by the Mainline crawler is the percentage of resources observed by the three sensors. Thus, any resource monitored by only one or two sensors is considered to be a non-observed resource in the monitoring (worst case). In Fig. 3 we can see that our estimation is that more than 90% of the resources are being monitored. Note that this number is coherent with the performance already indicated in other works [11].

A total of 321,962 different resources have been monitored during this period, 166,035 in the `0x09` zone and 155,927 in the `0x10` zone. We can see in Fig. 2 the evolution of the new infohashes discovered every hour during the first week in both zones. The similar behavior of both monitors lead us to the conclusion that these results could be generalized to other zones.

For each of the monitored resources we have stored every `announce_peer` message received, logging the `infohash`, origin `nodeID`, IP address and port and the timestamp. In Fig. 4 we can see the evolution of the number of peers communicating within the `0x09` zone for the first week of our monitoring period. The number of peers exhibit a periodic behavior with an increasing mean value that stabilizes

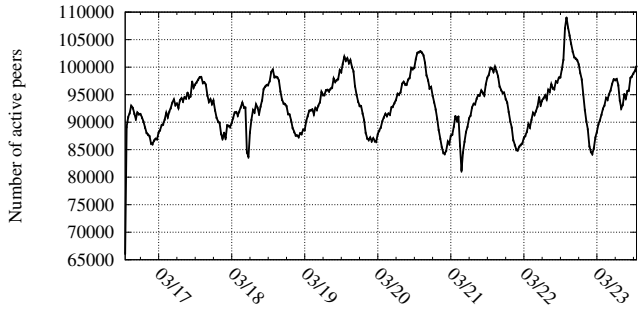

Fig. 4: Evolution of the number of connected peers (0x09 zone) during the first week.

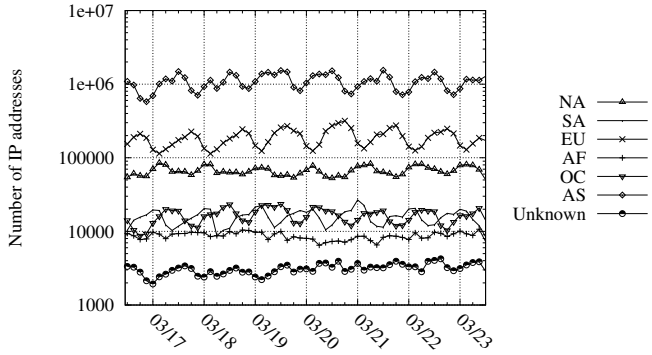

Fig. 5: Evolution of the number of different IP addresses grouped by continent during the first week (0x09 zone).

after some days. Depending on the time of the day, around 95,000 peers are actively sending/receiving messages to/from our system.

As reported in the Sandvine 2015 report [15] Asia (AS) is the continent with a higher percentage of BitTorrent usage. This fact is reflected in Fig. 5, where we show the monitored IP addresses grouped by its continent geolocation. Note that, due to this greater percentage of users from Asia, we obtain a periodic wave showing the typical evolution in a day/night traffic in Fig. 4. In fact, using the time UTC+8 (China) the maximum number of peers is reached at 9 PM and the minimum at 4 AM.

These resources have been shared by 86,915,611 different nodes (different nodeIDs) with 23,417,933 different IP addresses from all the continents. Note the huge difference between the number of nodes and IP addresses. This could be either due to the existence of NAT boxes or the use of sybil mechanisms. For example, DHT search engines like bitdigg make use of sybil procedures to collect information from network, in a similar way that we are doing in our Mainline crawler. A prior inspection of these data showed that there are certain IP ranges that comprise a huge number of nodes. As an example, two IP ranges from Russia and Kazajstán contain 8M and 6M of nodeIDs respectively, presenting a mean value of 14K nodeIDs per IP. Due to this size, we consider that it is more likely that they have a sybil behavior than they are NAT boxes.

Estimation of the Deep Torrent Size. Using the Web crawler module, we have conducted our *passive search* since December 20th 2015, receiving information from some of the most common torrent-discovery sites. First, we have chosen a meta-search engine like torrentz (https://torrentz.eu/), due to the fact that it allows to search information in a large list of other torrent-discovery sites. During the monitoring period, torrentz comprised a list of 29 torrent sites[1]. In addition, we have also checked directly some of the more relevant torrent-discovery sites, like http://bitsnoop.com/, https://piratebay.to/, among others. Finally, we have also decided to collect information from https://btdigg.org/, the only DHT search engine at the time of the experiment.

Each of these sites generates a periodic report with the newest torrent resources, which are subsequently downloaded and stored by our crawler. Depending on the torrent-discovery site, the frequency of the crawling varies between 24 and 48 hours. As a result, we have stored in our database a total of 22,174,122 resources. Out of the 321,962 resources collected in the Mainline monitor module, we found 80,869 (25.12%) within the 22 millions of resources obtained by the Web crawler.

For the rest of the resources (a total of 241,093), we conducted an *active search* using some of the most common torrent-discovery sites that allow to find torrent resources by using their infohash. After this, we only found information about 23,878 additional resources of our set. At the end, we have 217,215 unidentified resources, which supposes a 67.47% of the monitored resources. This is our estimated size of the Deep Torrent. Note that these results are only a proof of concept, as more exhaustive search methodologies for the Surface Torrent could be followed. Anyway, the obtained percentage points out that the size of Deep Torrent is not negligible at all.

Exploring Features of Shared Resources. One application of the Web crawler is to explore the meta-data included in torrent files to draw conclusions about the contents and the sharing mechanisms.

For example, we wanted to inspect the *active duration* of the sharing of resources, in order to find out possible differences between the Deep (DT) and Surface Torrent (ST) resources. This duration is defined as the number of hours during which the monitor receives messages announcing a specific resource. The results can be seen in Fig. 6. First, note that DT resources are shared during less time. This is an expected result, as these resources are not being publicly published in torrent-discovery sites and, therefore, they are not expected to be so popular. Indeed, there exist a total of 159,195 DT resources with less than 5 hours of active duration, which supposes a 73.29% of the total amount of DT resources, while the number of ST with less than 5 hours of active duration is 25,015 (23.88% of the total amount of ST resources). Yet, it is notable that many of the resources in the DT are still being shared during a long time.

[1]See complete list at http://web.archive.org/web/20160323031455/http://torrentz.eu/help)

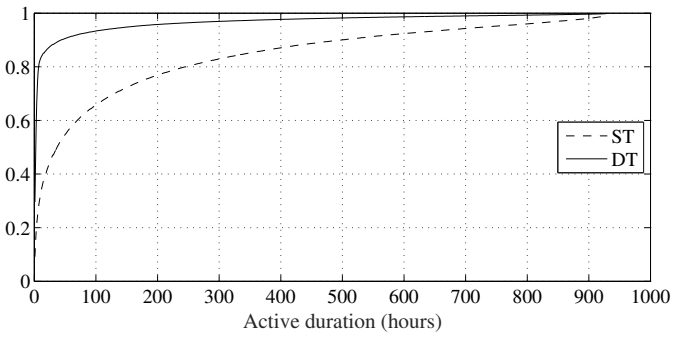

Fig. 6: Normalized cumulative histogram of the active duration of Deep Torrent (DT) and Surface Torrent (ST) resources.

VI. Conclusions and Future Work

This paper explores the Deep Torrent, *i.e.*, torrents available in BitTorrent that cannot be found by means of public Web sites or search engines. We discuss the necessity for studying its properties, proposing a system to crawl Deep Torrent resources that combines:(*a*) a Surface Torrent crawler for the Web, and (*b*) a BitTorrent (Mainline) monitor.

For demonstrating the usefulness of the crawler, we have collected information from a part out of 128 of the Mainline network during 39 days, identifying a total of 321,962 resources. Among them, 32.53% belongs to the Surface Torrent, *i.e.*, they can be found in torrent-discovery sites; and the remaining 67.47% are part of the Deep Torrent. We have shown how the information obtained from the crawler is proven to be useful to extract interesting characteristics of the Deep Torrent.

Despite these results, we consider that there are many interesting details and questions to be solved as part of future work. Specifically, we plan to work on:

- Extending the monitoring period and the number of monitored zones to derive more general results.
- Thoroughly studying the features of the resources in the Deep Torrent and comparing them to those of the Surface Torrent.
- Including a new module in our system to automatically download Deep Torrent resources in order to study them in a posterior phase.
- Trying other techniques for the crawling of Surface Torrent by our Web crawler.

Acknowledgments

This work has been partially supported by Spanish Government-MINECO (Ministerio de Economía y Competitividad) and FEDER funds, through project TIN2014-60346-R and the Corporacion Tecnológica de Andalucía through project CTA 15/795.

References

[1] M. K. Bergman, "White Paper: The Deep Web: Surfacing Hidden Value," *The Journal of Electronic Publishing*, vol. 7, Aug 2001.

[2] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang, "Accessing the Deep Web," *Communications of the ACM*, vol. 50, pp. 94–101, May 2007.

[3] D. Sui, J. Caverlee, and D. Rudesill, "The Deep Web and the Darknet: a look inside the Internet's massive black box," tech. rep., Oct 2015.

[4] R. A. Rodríguez-Gómez, G. Maciá-Fernández, P. García-Teodoro, M. Steiner, and D. Balzarotti, "Resource monitoring for the detection of parasite P2P botnets," *Computer Networks*, vol. 70, no. 0, pp. 302–311, 2014.

[5] X. Chen, Y. Jiang, and X. Chu, "Measurements, analysis and modeling of private trackers," *2010 IEEE 10th International Conference on Peer-to-Peer Computing, P2P 2010 - Proceedings*, 2010.

[6] M. Balduzzi and V. Ciancaglini, "Cybercrime In The DeepWeb," in *Black Hat 2015 EU, Amsterdan*, 2015.

[7] Y. He, D. Xin, V. Ganti, S. Rajaraman, and N. Shah, "Crawling deep web entity pages," *Web Search and Data Mining*, pp. 355–364, 2013.

[8] C. Zhang, P. Dhungel, D. Wu, and K. W. Ross, "Unraveling the BitTorrent Ecosystem," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, pp. 1164–1177, Jul 2011.

[9] H. Jin, H. Jiang, S. Ibrahim, and X. Liao, "Inaccuracy in Private BitTorrent Measurements," *International Journal of Parallel Programming*, vol. 43, pp. 528–547, Oct 2013.

[10] M. Steiner, T. En-Najjary, and E. W. Biersack, "A global view of KAD," *Internet Measurement Conference*, 2007.

[11] G. Memon, R. Rejaie, Y. Guo, and D. Stutzbach, "Montra: A large-scale DHT traffic monitor," *Computer Networks*, vol. 56, pp. 1080–1091, Feb 2012.

[12] M. Varvello and M. Steiner, "DHT-based traffic localization in the wild," in *2013 Proceedings IEEE INFOCOM*, pp. 3141–3146, Apr 2013.

[13] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pp. 53–65, 2002.

[14] "Mainline DHT Implementation." http://bittorrent.org/beps/bep_0005.html. [Online; accessed May-12th-2016].

[15] Sandvine, "Global Internet Phenomena Asia-Pacific & Europe," 2015.