Measuring BitTorrent Ecosystem: Techniques, Tips and Tricks

Michal Kryczka*†, Rubén Cuevas*, Ángel Cuevas*, Carmen Guerrero* and Arturo Azcorra*
*University Carlos III Madrid
†Institute IMDEA Networks

Abstract—**BitTorrent is the most successful peer-to-peer application. In the last years the research community has studied the BitTorrent ecosystem by collecting data from real BitTorrent swarms using different measurement techniques. In this paper we present the first survey of these techniques that constitutes a first step in the design of future measurement techniques and tools for analyzing large scale systems. The techniques are classified into Macroscopic, Microscopic and Complementary. Macroscopic techniques allow to collect aggregated information of torrents and present a very high scalability being able to monitor up to hundreds of thousands of torrents in short periods of time. Rather, Microscopic techniques operate at the peer level and focus on understanding performance aspects such as the peers' download rates. They offer a higher granularity but do not scale as well as the Macroscopic techniques. Finally, Complementary techniques utilize recent extensions to the BitTorrent protocol in order to obtain both aggregated and peer level information. The paper also summarizes the main challenges faced by the research community to accurately measure the BitTorrent ecosystem such as accurately identifying peers or estimating peers' upload rates. Furthermore, we provide possible solutions to address the described challenges.**

I. Introduction

BitTorrent [5] is the most successful peer-to-peer file-sharing application. Indeed, it is responsible for a major portion of the Internet traffic share [10] and is daily used by dozens of millions of users. This has attracted the interest of the research community that has thoroughly evaluated the performance and the demographic aspects of BitTorrent. Due to the complexity of the system, the most relevant studies have tried to understand different aspects by performing real measurements of BitTorrent swarms in the wild, this is inferring information from real swarms in real time.

Several techniques have been used in order to measure different aspects of BitTorrent so far, however to the best of the authors knowledge there is no document that compiles, describes and classifies these techniques. In this paper we firstly describe the main aspects and functionality of the complete BitTorrent Ecosystem in Section II. Afterwards, Section III presents a survey of the existing BitTorrent measurement techniques. Finally, we describe the main challenges that these techniques face and the possible solution to them in Section IV before concluding the paper in Section V.

II. BitTorrent Ecosystem

BitTorrent [5] is the name used by Brian Cohen to define the peer-to-peer file-sharing protocol that he designed one decade ago. Due to the great success of this protocol a complex system was created around it. In this paper we adopt the terminology used by Zhang et al. [16] to refer to this complex system as the *BitTorrent Ecosystem*. In this section we describe the main players of this ecosystem as well as its functionality. This is summarized in Fig. 1.

A. Description of BitTorrent Functional Elements

- A ***BitTorrent Portal*** is a server into which content publishers upload *.torrent* files and BitTorrent clients download those *.torrent* files.
- A ***BitTorrent Swarm*** is formed by a set of peers downloading a given content using the BitTorrent protocol.
- A ***BitTorrent Tracker*** is a server that maintains a list of clients forming the BitTorrent swarm associated to a given content. Furthermore the Tracker is aware of the download progress of each peer within the swarm.
- A ***BitTorrent Client or Peer*** is an entity that participates in a BitTorrent swarm by downloading and/or uploading pieces of the content. Two categories of peers may be distinguished: A *seeder* is a client that has a complete copy of the content, thus only uploads pieces to other peers. A *leecher* is a client that does not have a complete copy of the content, thus uploads and downloads pieces to and from other peers respectively.
- A ***.torrent file*** is a meta-information file associated to a content shared through BitTorrent. The *.torrent* file includes the following information: content name, file size, number and size of the pieces that form the content named *chunks*, torrent infohash (an identifier that uniquely identifies the swarm associated to the .torrent file) and IP address(es) of the Tracker(s) managing the swarm associated to the file.

B. Publishing Content in BitTorrent

In order to make available a content C in BitTorrent, the content publisher creates a *.torrent* file associated to C. After creating the .torrent file, the content publisher uploads it to a *BitTorrent portal*. A detailed analysis of the BitTorrent content publishing phenomenon can be found at [6]. There are a few BitTorrent portals such as The Pirate Bay[1] indexing millions of torrents and receiving millions of daily visits. These portals

[1]This is the current largest BitTorrent portal based on Alexa Ranking.

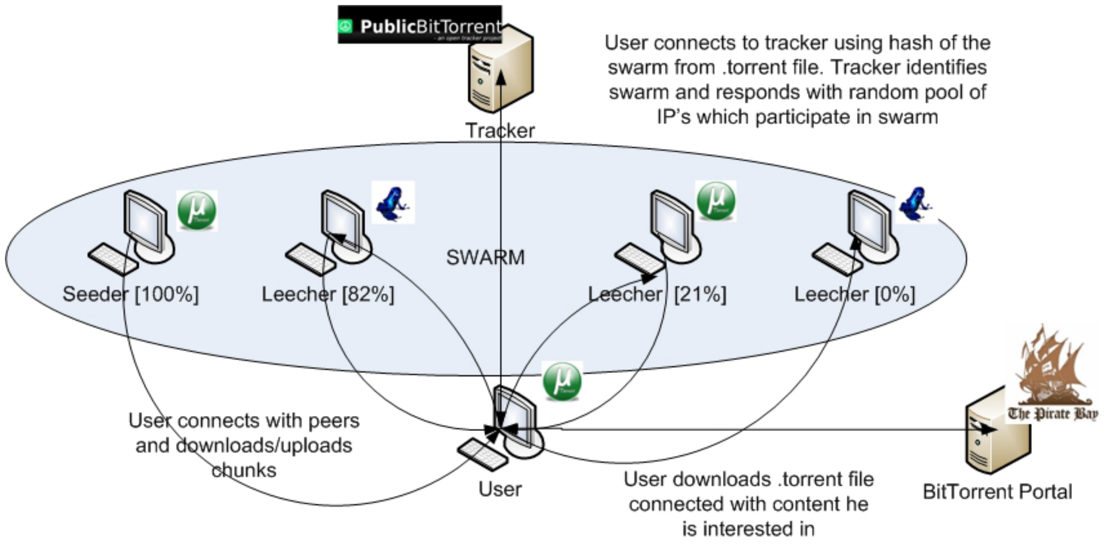

PublicBitTorrent
Tracker
User connects to tracker using hash of the swarm from .torrent file. Tracker identifies swarm and responds with random pool of IP's which participate in swarm
SWARM
Seeder [100%]
Leecher [82%]
Leecher [21%]
Leecher [0%]
User connects with peers and downloads/uploads chunks
User
User downloads .torrent file connected with content he is interested in
The Pirate Bay
BitTorrent Portal

Fig. 1: BitTorrent Ecosystem basic functionality: (i) The BitTorrent client contacts a BitTorrent portal to download the .torrent file associated to the desired content (the .torrent file includes the IP address of the Tracker managing the swarm associated to the desired content); (ii) The BitTorrent client contacts the Tracker that provides the IP addresses of a set of peers within the swarm; (iii) The BitTorrent client connects to these peers for downloading the content.

are critical for the BitTorrent ecosystem as demonstrated by Zhang et al. [16]. They offer detailed information regarding each indexed torrent. This information slightly varies from one portal to another, but in general it includes: category of the content, number of associated files, size of the whole content in the torrent, complete name of the file, uploading date, username who uploaded the torrent, number of seeders and leechers participating in the torrent swarm (this data is updated every few minutes) and a description text giving more detailed information regarding the content. Fig. 2 shows an example of a torrent web-page from the Pirate Bay portal. Finally, it is worth noting that some of these major portals offer a *Really Simple Syndication* (RSS) feed to announce the new published torrents.

C. Joining a BitTorrent Swarm and Discovering Peers

When a BitTorrent user wants to download a given content C, it looks for the .torrent file associated to C in a BitTorrent portal and downloads it. The .torrent file can be opened with any of the existing BitTorrent clients[2]. Upon opening the .torrent file, the BitTorrent client connects to one of the Trackers included in this one. A new peer first contacts the Tracker using an *announce started* request that is answered by the Tracker with the number of seeders and leechers participating in the swarm along with the IP addresses of N (between 40 and 200) randomly selected peers. These N peers form the initial neighbourhood of the new node. Furthermore, if a peer's neighbourhood size falls below a given threshold (typically 20), it sends again an *announce started* request to the Tracker in order to get new neighbours. Finally, when a peer leaves the swarm, it sends an *announce stopped* request to the Tracker that removes this peer from the list of participants in the swarm.

It is worth noting that, in practice, the BitTorrent Ecosystem relies in few Trackers that manage a large number (up to a few millions) of torrents in parallel. The OpenBitTorrent and PublicBitTorrent Trackers[3] are currently the most important ones.

D. BitTorrent Delivery Procedure

In BitTorrent two peers communicate using the *peer wire* protocol. Every communication starts with an initial *handshake*. Just after the handshaking sequence is completed (and before any other messages are sent) the peers exchange the *bitfields* by using a BITFIELD message. The bitfield indicates which chunks of the file a peer has already downloaded. Furthermore, every time a peer gets a new chunk, it informs to its neighbours by using a *HAVE* message. Hence, every peer is aware of the chunks that each neighbour has at any moment.

BitTorrent uses the *Tit-for-Tat* as incentive model for the delivery mechanism; basically each leecher uploads chunks to those other leechers from whom it is downloading more chunks. The *Choking Algorithm* is responsible for providing this behaviour. It is a periodical operation where every 10 seconds a leecher selects (unchoke) n other leechers from its neighbourhood to upload chunks to. These n (typically 4) unchoked leechers are those from whom the peer downloaded more chunks during the last 20 seconds. The rest of the neighbours are blocked (choked). In the case of a seeder, it unchokes n (typically 4) leechers to whom more chunks it uploaded in the last 20 seconds (i.e., those with higher download rate)[4]. In addition to the regular unchoke operation,

[2]http://en.wikipedia.org/wiki/Comparison_of_BitTorrent_clients

[3]www.openbittorrent.org and www.publicbt.org

[4]Note that different BitTorrent clients may implement different variations of the explained unchoke algorithms.

Property	Portal Crawling	Tracker Crawling	Peers Crawler	Own client/plugin
Category	Macroscopic	Macroscopic	Microscopic	Microscopic
Type of Information	Torrents	Demographics and High level performance	Peer Level Performance	Peer Level Performance
Cost of Crawler preparation	Low	Medium	High	High
Scalability	Very High	High	Medium	Medium-Low
Obtained details	Basic	Medium	Advanced	Very Advanced
Completeness of Torrent Population	-	High	Very High	Low

TABLE I: Comparison of main BitTorrent measurement techniques

BitTorrent implements the *optimistic unchoke* operation. Every 30 seconds (this is, every 3 regular unchoke operations) both leechers and seeders randomly select one choked neighbour to upload chunks to. Finally, when a leecher is unchoked by a neighbor, it applies the *Rarest First Policy* in order to choose which chunk to request to this neighbor. Since leechers have full knowledge about the availability of every chunk in its neighbourhood, it always request the rarest one.

E. BitTorrent Extension

Several extensions to BitTorrent have been proposed so far. Here we just mention those relevant to measurement studies:
-Distributed Hash Table (DHT): Trackers are a single point of failure in the BitTorrent ecosystem. Indeed, they are typically threatened by legal actions. The BitTorrent developers reacted to this by designing a Tracker-less mechanism that allows a BitTorrent user learning the IP addresses of peers without contacting the Tracker. This mechanism is based on a DHT [15].
-Peer Exchange (PEX): This is a simple gossiping protocol that is used to get IP addresses of peers participating in the swarm. In more detail PEX works as follows: a given peer P sends a PEX request to one of its neighbours, e.g., N. If N supports PEX, it responds with the list of IP addresses of all its neighbours. Hence, by using few PEX queries a given peer can learn the IP addresses of a large number of participants in the swarm without requesting them to the Tracker.

III. Techniques for Measuring the BitTorrent Ecosystem

In this section we describe the BitTorrent measurement techniques defined in the literature so far. We classify them into two main categories: *Macroscopic* and *Microscopic* depending on the retrieved information. The former obtains demographic and high level performance information whereas the latter gathers peer level performance information. A summary of different techniques is presented in Tab. I.

A. Macroscopic Techniques

The main objective of these techniques is understanding the demographics of the BitTorrent ecosystem: the type of content published, the popularity of this content, the distribution of BitTorrent users per country (or ISP), the relevance of the different portals and Trackers, etc. Furthermore the Macroscopic measurements also allow to study some performance aspects such as the ratio of seeders/leechers, the session time of the BitTorrent users, the arrival rate of peers, the seedless state (period the torrent is without seeder) duration, etc.

We classify the Macroscopic techniques into two subcategories: *BitTorrent portals crawling* and *BitTorrent Trackers crawling*.

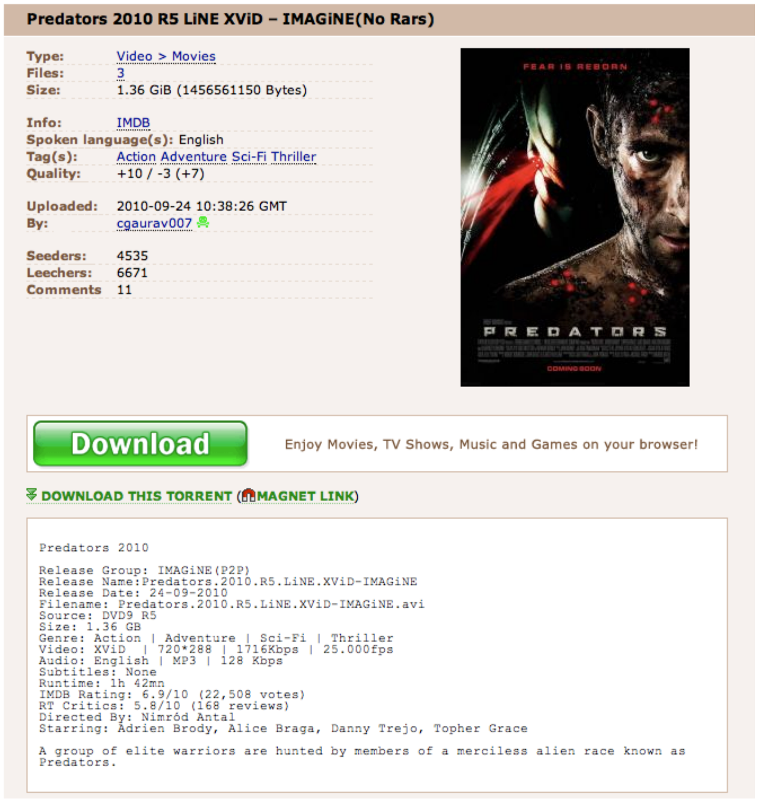
-BitTorrent portals crawling:
As shown in Section II, the (major) BitTorrent portals index millions of torrents in a structured way. Furthermore, they provide detailed information about each indexed torrent (typically) in an specific torrent web-page. For instance, in the case of The Pirate Bay, the torrent web-page associated to a torrent with an assigned torrent-id equal to i can be accessed through the url http://thepiratebay.org/torrent/i (See Fig 2). Hence, once we know the id assigned to a given torrent in The Pirate Bay, we just need to access its web-page and parse it (using an html-parser) to retrieve the torrent information. However, in order to analyze the demographics of BitTorrent we need to crawl a large number of torrents. Next we describe two types of crawling techniques that can be used in order to systematically crawl up to millions of torrents from an specific portal (we consider The Pirate Bay as example):

Backwards Crawling: In this case the aim is to retrieve the information associated to the *alive* torrents published in The Pirate Bay from a given past date to the current instant. For this purpose the Crawler sequentially parses all the torrents' web-pages from the last published torrent (http://thepiratebay.org/torrent/last_torrent_id/) decreasing up to the first torrent published in the target date, for instance with torrent id k (http://thepiratebay.org/torrent/k/). The last published torrent-id can be identified either manually or using the RSS feed.

Upwards Crawling: In this case the aim is to retrieve the information associated to every torrent published in The Pirate Bay from now during a given time (e.g., 1 month). In this case, each new torrent will be assigned a torrent-id that can be learnt from the RSS feed. We will use these learnt torrent-ids to crawl the torrents web-pages.

By post-processing the retrieved data from the BitTorrent portal crawling very relevant aspects of the BitTorrent Ecosystem demographics can be characterized. Next we describe few representative examples. We refer the reader to [16] for a detailed analysis of the BitTorrent Ecosystem demographics:

- *Content Popularity Distribution*: For this purpose we obtain the number of leechers and seeders for each specific torrent from the html-parsing. Note that if we want to study the evolution of popularity for a given torrent we have to periodically parse its web-page to retrieve



Predators 2010 R5 LiNE XViD – IMAGiNE(No Rars)

Type:	Video > Movies
Files:	3
Size:	1.36 GiB (1456561150 Bytes)
Info:	IMDB
Spoken language(s):	English
Tag(s):	Action Adventure Sci-Fi Thriller
Quality:	+10 / -3 (+7)
Uploaded:	2010-09-24 10:38:26 GMT
By:	cgaurav007
Seeders:	4535
Leechers:	6671
Comments	11

FEAR IS REBORN
PREDATORS
COMING SOON
Download
Enjoy Movies, TV Shows, Music and Games on your browser!
DOWNLOAD THIS TORRENT (MAGNET LINK)

```
Predators 2010

Release Group: IMAGiNE(P2P)
Release Name:Predators.2010.R5.LiNE.XViD-IMAGiNE
Release Date: 24-09-2010
Filename: Predators.2010.R5.LiNE.XViD-IMAGiNE.avi
Source: DVD9 R5
Size: 1.36 GB
Genre: Action | Adventure | Sci-Fi | Thriller
Video: XViD  | 720*288 | 1716Kbps | 25.000fps
Audio: English | MP3 | 128 Kbps
Subtitles: None
Runtime: 1h 42mn
IMDB Rating: 6.9/10 (22,508 votes)
RT Critics: 5.8/10 (168 reviews)
Directed By: Nimród Antal
Starring: Adrien Brody, Alice Braga, Danny Trejo, Topher Grace

A group of elite warriors are hunted by members of a merciless alien race known as
Predators.
```

Fig. 2: Example of a The Pirate Bay torrent web-page. HTML parsing techniques can retrieve the following information: Content name (Predators 2010 R5), Content category and subcategory (Video and Movies), Number of files (3), Size of the whole content (1.36 GB), Language (English), Uploading date (2010-09-24), username uploading the .torrent file (cgaurav007), current number of seeders and leechers (4535 and 6671) and a text-box with further information regarding the content.

the evolution of the torrent population (i.e., number of leechers and seeders).

- *Distribution of number of published content per category and subcategory*: For this purpose we obtain the category and subcategory for each specific torrent from the html-parsing.
- *Torrents Publishing Rate per date*: For this purpose we obtain the date when each specific torrent was uploaded from the html-parsing.

By applying the described measurement study to different portals, we can perform a comparative study of the relevance of these portals in the BitTorrent ecosystem.

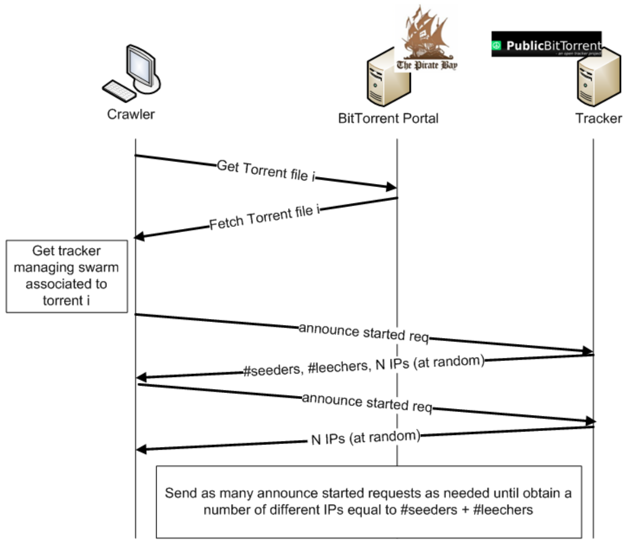
Finally, by tracking the evolution of the number of seeders and leechers for a given torrent we can also infer some performance metrics such as the seeder-to-leecher ratio and its evolution along the time.

-BitTorrent Trackers crawling:

The crawling of a BitTorrent portal gives detailed information regarding the torrents (type, publishers) and some aggregated numbers such as the number of seeders and leechers. However, this does not suffice if we aim to study more detailed demographics parameters such as the distribution of BitTorrent users per country (or ISP) or relevant performance aspects such as peers arrival rate and peers session time. In order to study these issues we need to collect the IP addresses of the peers participating in the swarms. This can be obtained from Trackers (remind that a Tracker managing a given swarm knows the IP addresses of all the participants).

There are various ways of accessing the information of a Tracker (i.e., IP addresses of participants in the swarms managed by the Tracker):

- Getting access to the Tracker logs [12]. This requires the Tracker owner's collaboration.
- Using a Tracker where the information is publicly available [9]. Unfortunately, only minor Trackers offer this functionality.
- Using measurement techniques, i.e., crawling the Tracker as depicted in Fig. 3. In this case we need to use a BitTorrent Crawler that implements the part of the BitTorrent protocol to communicate with the Tracker. More specifically, this Crawler works as follows: first, we define the list of torrents whose participants' IP addresses we want to obtain. This list of torrents can be retrieved (for instance) from a BitTorrent portal. For each torrent in the list, the Crawler performs an initial *announce started* request to the correspondent Tracker. From this request the Crawler retrieves the number of participants (seeders and leechers) in the swarm and an initial list of IP addresses. Afterwards, the Crawler performs as many



Crawler
BitTorrent Portal
Tracker
PublicBitTorrent
Get Torrent file i
Fetch Torrent file i
Get tracker managing swarm associated to torrent i
announce started req
#seeders, #leechers, N IPs (at random)
announce started req
N IPs (at random)
Send as many announce started requests as needed until obtain a number of different IPs equal to #seeders + #leechers

Fig. 3: BitTorrent Tracker Crawler basic functionality: The BitTorrent Crawler retrieves the .torrent file from a BitTorrent portal and obtains the IP address of the Tracker managing the swarm from it. Afterwards, it sends as many *announce started* request as needed until obtain the IP addresses of all the peers participating in the swarm.

announce started requests as needed to obtain as many IP addresses as the number of participants in the swarm.

Hence, by using any of the previous techniques we are able to collect the IP addresses of the participants in a large number of torrents. This data allows to study some relevant demographics and performance BitTorrent features. Next, we briefly describe some of them:

- *The Distribution of clients per country or ISP*: Some studies have applied the described crawling technique to a large number (even millions) of torrents [16]. Afterwards, the IP address of each client is mapped to its country and ISP (e.g., using the MaxMind database [1]). From this data we can compute the distribution of BitTorrent users per country and/or ISP.
- *Heavy Hitters*: By doing a cross-torrent inspection we can find those users (IP addresses) being present in a large number of torrents [3]. We name these users *Heavy Hitters*.
- *BitTorrent Traffic*: The authors of [7] performed the described crawling technique in a short period of time (90 min) over the most recent 40k torrents announced by a BitTorrent Portal. This can be viewed as a snapshot of a portion of the BitTorrent ecosystem. By computing the traffic flowing between the BitTorrent clients in the different torrents, the authors estimate the Intra-ISP and Inter-ISP traffic generated by BitTorrent in a large number of ISPs.
- *Peers' Arrival Rate and Session Time*: If we apply any of the described techniques periodically on a given torrent, we are able to continuously monitor the peers participating in the torrent. Therefore for each single user (i.e., IP address) we can approximately determine the instant in which it joins and leaves the torrent, thus being able to define the session time for each user. Furthermore by looking at the time between the subsequent arrivals of peers we can infer the arrival rate. Authors of [9], [13] have performed this analysis in a large number of torrents.

B. Microscopic Techniques

The described Macroscopic techniques retrieve exclusively the peers' IP addresses, thus only metrics associated to the presence/absence of the peer can be studied. Unfortunately, an IP address does not suffice to infer relevant performance metrics at the peer level such as peers' download and upload rate. For this purpose we need to apply more sophisticated (but less scalable) techniques that we name *Microscopic techniques*.

To perform Microscopic techniques we need to implement different parts of the BitTorrent peer wire protocol. Any Microscopic Crawler has to implement the functions to perform the handshaking procedure. This is essential to connect to other peers. The handshaking procedure can be done actively (the Crawler initiates it) or passively (the Crawler waits until a peer starts the handshaking). Once the Crawler is connected to a peer, it exploits different messages of the peer wire protocol in order to measure different parameters. This process is illustrated in Fig. 4. Next we describe the specific techniques proposed in the literature to measure the most important peer level performance aspects of BitTorrent:

Peer Type:

After the handshaking procedure succeed with a peer, this one immediately sends a BITFIELD message to the Crawler. By analyzing the bitfield, the Crawler classifies the peer as seeder or leecher [13], [14].

Furthermore, when using an active Crawler there are some peers that do not respond to the Crawler's handshake messages. These peers are typically located behind a NAT or a firewall that prevents the establishment of incoming connections. Thus, these peers are classified as *NATed* [13], [14]. In order to infer if a *NATed* peer is a seeder or a leecher we need to apply passive techniques and wait until the peer contacts the Crawler.

Instantaneous Download Rate:

After the handshaking procedure is completed (either passively or actively) the Crawler waits until it receives two HAVE messages from a given peer. The size of the chunk (e.g., 4MB) used in a given torrent is well-known[5]. Furthermore, the Crawler measures the time between the reception of these two consecutive HAVE messages from a peer, that is approximately the time needed to download a chunk. Hence, by dividing the size of the chunk by the time needed to download it we can infer the instantaneous download rate of the peer. By repeating this operation periodically we can obtain the evolution of the instantaneous download rate of a given peer [14].

[5]This information is available in the .torrent file.

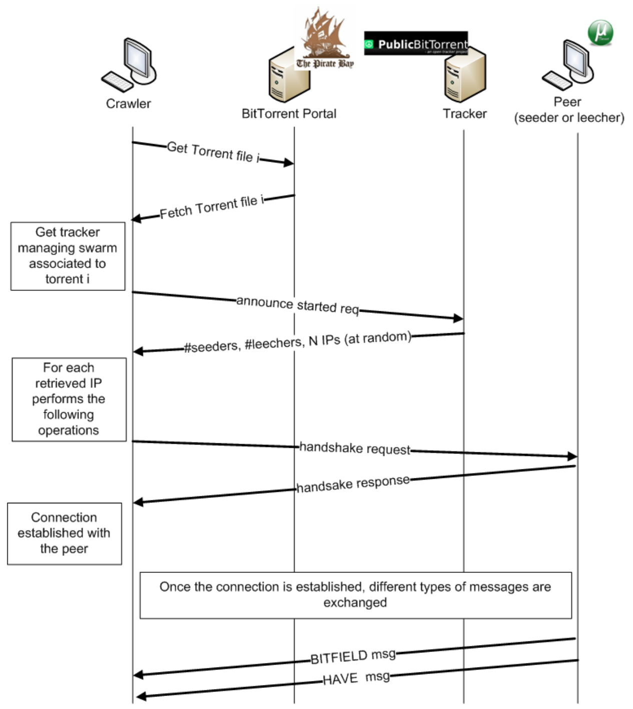

Crawler
BitTorrent Portal
The Pirate Bay
PublicBitTorrent
Tracker
Peer
(seeder or leecher)
Get Torrent file i
Fetch Torrent file i
Get tracker managing swarm associated to torrent i
announce started req
#seeders, #leechers, N IPs (at random)
For each retrieved IP performs the following operations
handshake request
handsake response
Connection established with the peer
Once the connection is established, different types of messages are exchanged
BITFIELD msg
HAVE msg

Fig. 4: BitTorrent Peer Crawler basic functionality: The Crawler retrieves the IP addresses of peers participating in a given swarm as explained in the Macroscopic Tracker crawling technique. Afterwards, the Crawler contacts each individual peer, performs the handshake procedure and exchange different messages (BITFIELD, HAVE) to obtain different peer-level performance information.

Average Download Rate:

In this case the Crawler connects to a peer, obtains its bitfield and disconnects. After some time (e.g., 1 hour) the Crawler repeats the same operation on the same peer. Then, by comparing the two bitfields, we can compute the number of downloaded chunks between the two connections to the peer. Since we know the size of each chunk (S), the number of downloaded chunks (D) and the time between the two connections to the peer (T), we can easily compute the peer's average download rate as (S*D)/T.

Upload Rate:

This is probably the hardest parameter to be measured. Indeed, to the best of the authors knowledge there is no work that has properly measured the upload bandwidth. Rather some few works have measured some parameters related to the upload rate. On the one hand, Isdal et al. [11] measure the physical upload capacity ($\geq$ upload rate dedicated to BitTorrent). For this purpose, the authors implement a passive Crawler that measures the peers' upload capacity using the chunks sent by these peers to the Crawler during optimistic unchokes. On the other hand, Siganos et al. [14] measure the number of IP packets sent by a node. For this purpose, the authors implement an active technique that uses a special type of ICMP message. The peers' answer to this ICMP packet includes the number of IP packets sent since the last time the computer was switched on. Hence, this Crawler sends two of these ICMP packets separated a given time T. The answers to the first and second ICMP messages indicate a number of packets equal to P1 and P2. Therefore the rate of IP packets sent by the peer is computed as (P2-P1)/T. Note that this rate includes IP packets associated to BitTorrent but also to other applications.

Chunk distribution (Rarest First performance):

An important aspect of BitTorrent delivery mechanism is the Rarest First Algorithm. In order to study its performance we have to analyze how the distribution of the number of available copies of each chunk in a swarm looks like. For this purpose we implemented a Crawler that collects the bitfield of a large number of peers in a swarm (ideally all) in a relative short period of time (few minutes). By analysing the collected bitfields we achieve our objective, i.e., computing the number of available copies of each chunk in the swarm and calculating its distribution. We performed this study in [13] demonstrating that the Rarest First Algorithm guarantees a uniform distribution of pieces.

C. *Complementary Techniques*

Some researchers have used measurement techniques that can complement the above described Macroscopic and Microscopic techniques. On the one hand, some Crawlers [13], [14] have implemented the DHT and/or PEX functionalities in order to learn the IP addresses of the peers participating in a given swarm. On the other hand, some research groups have implemented their own client [2] or a plugging for a popular BitTorrent client such as Vuze [4]. These clients (or pluggings) report information to a log server. This technique complements the Microscopic measurements mechanisms since it gives very accurate information regarding peer level performance parameters, for instance it can precisely informs about a peer's download and upload rate.

D. *Techniques comparison*

In this subsection we compare the different measurement techniques introduced above, stating the pros and cons of each one of them. *Macroscopic* techniques are the most scalable ones allowing to analyze up to hundreds of thousands of torrents. These techniques are valid to retrieve: (i) aggregated information at the swarm level and (ii) specific information regarding the presence of a the peer (represented by the IP+port) in a given swarm. Therefore, they are useful to characterize important information such us content publishing phenomeon (i.e., which users are responsible for making available the content shared through BitTorrent), popularity distribution of torrents, seeder/leecher ratio, peers' session time, cross-torrent interactions (e.g., peers participating in multiple torrents), etc. However, these techniques cannot provide information at the peer level (e.g., peer's download progress, peer's download rate, chunk distribution, etc) since this requires to contact the peer. *Microscopic* techniques were designed to perform the peer level analysis. For this purpose the measurement software connects periodically to a large number of peers (potentially to all the peers participating in th set of analyzed torrents). This makes *Microscopic* techniques to scale up to analyze (at most) few thousand torrents in parallel. This means at least one order of magnitude less than the *Macroscopic* counterpart.

Moreover, we have briefly defined a set of *Complementary* techniques. On the one side, the usage of DHT and/or PEX to learn the IP addresses of the peers participating within a swarm compete directly with the traditional technique of learning the peers from the Tracker. PEX and DHT allow the measurement software to speed up the IP addresses collection and eliminate the risk of being blacklisted by the Tracker (see Section IV). However, the Tracker provides relevant information such as the number of peers participating in the swarm, thus even when using PEX and DHT to learn peers, it is strongly recommended that the measurement software queries the Tracker regularly in order to have an estimation of the the number of peers participating in the swarm. An important aspect to consider is the simplicity of the measurement tool. In this case, those measurement tools based on a traditional Tracker crawling are simpler than those enhanced versions that implement a PEX and/or DHT module.

On the other side, the collection of data based on a specific client or plugging implementation competes directly with the *Microscopic* techniques. Using a client that reports logs to a server is the most accurate method to measure the activity of a peer (e.g., download rate, upload rate, etc) and surely provides more accurate results than the traditional *Microscopic* techniques. On the downside, the scalability of this technique is limited to the number of clients running the BitTorrent client (or plugging), this means that we have a partial view of the analyzed torrent. Moreover, the retrieved data is only representative of a specific client with a specific implementation, thus the obtained results may not be generalized to other clients. Traditional *Microscopic* techniques lack of the level of accuracy of the client based measurement but offers a better scalability and coverage of the analyzed torrents.

Finally, it is important to highlight that the described techniques are not necessarily exclusive. Therefore it is strongly recommended to perform a study of the required data to be collected and then decide which of the described techniques the measurement software has to implement.

IV. Challenges

In this section we enumerate the main challenges faced by the previously described techniques as well as possible solutions for some of them:

Peer Identification:

Description: In BitTorrent the peers do not have a permanent Peer-ID. Every time a BitTorrent client is started a new random Peer-ID is generated. Then, it is not possible to follow a peer across multiple sessions using its Peer-ID. Most of the studies performed so far utilize the IP address or the IP address+port to identify a single user across multiple sessions. This works for all those users having a static IP address. However, most of the BitTorrent users are residential users with a dynamic IP address that is frequently changed by their ISP. Hence, identifying these peers by their IP addresses introduces inaccuracies in the obtained data. Furthermore, in the current Internet a single IP address may be shared by multiple users located behind a Network Address Translator (NAT) [8], thus using the IP address to identify a peer may lead to wrongly map several users as a single peer.

Possible Solutions: On the one hand, one way of guaranteeing the correct identification of a peer across sessions is using measurement techniques based on the implementation of your own BitTorrent client/plugin. Each installed instance of your client is assigned a unique and permanent ID (different than the Peer-ID used in the swarms) which is used by the client to report the logs to the log server. Other option is to get access to the log of private Trackers. In most of the private Trackers the users are required to register with a username and password. Each time a user initiates a session in the Tracker it has to login, thus it can be uniquely identified across sessions. Unfortunately, both described techniques have scalability limitations. On the other hand, identifying the peer by the combination of IP+port typically eliminates the problem of wrongly mapping users with the same IP address as a single peer. BitTorrent clients typically select a random port to operate, thus it is unlikely that two peers behind a NAT select the same port.

Crawler's IP address banned by the Tracker:

Description: The described Macroscopic Tracker crawling technique may produce the Crawler's IP address being banned by the Tracker. In some studies [3], [6], [16] the Crawler performs a large-scale crawling by continuously sending *announce started* requests to a specific Tracker for a large number (e.g., tens of thousands) of torrents. Then, the rate of *announce started* requests is very high what is detected by the Tracker. The reaction of the Tracker is blocking the IP address showing this anomalous behaviour. Therefore the Crawler has to limit the *announce started* requests rate to avoid being banned by the Tracker.

Possible Solutions: LeBlond et al. [3] describe a technique to avoid being banned while keeping a very high rate of *announce started* requests. The technique consists on sending an *announce stopped* just after the *announce started* request. Then, the Tracker removes the IP address of the Crawler from its log just after answering the *announce started* request. By using this simple technique, the authors report that they are able to crawl up to 750k torrents in around 30 min.

A second option is using an anonymization service such as TOR[6]. By using this service, the messages sent by the Crawler pass through an overlay of proxies before reaching the Tracker. Then, the IP address seen by the Tracker is that of the egress node from the proxies overlay, thus the Tracker cannot block the actual Crawler's IP address. Note that TOR is used by tens of thousands of BitTorrent clients in order to preserve their privacy while downloading content through BitTorrent. This is well-known by the Trackers administrators that do not ban

[6]http://www.torproject.org/

the TOR proxies IP addresses. Furthermore, the load created by the BitTorrent measurement tools in the TOR proxies is low compared to that created by the tens of thousands of BitTorrent clients using this service.

Finally, we can increase the rate of requests to the Tracker using several instances of the Crawler distributed among different machines with different IP addresses.

Crawler's IP address blacklisted by the client:

Description: In the case of Microscopic measurements the Crawler always performs the handshaking procedure with the target peer. Afterwards, it retrieves the needed information (e.g., the bitfield) and then it can either keep connected or disconnect and reconnect after a while. In the first case since our Crawler does not provide any chunk to the peer, due to the *optimistic connect* algorithm implemented by the most important BitTorrent clients, the peer is likely to substitute the Crawler by other peer in its neighbourhood. Once the Crawler has been removed from the peer's neighbourhood, it is typically hard to reconnect since the peer recognizes the Crawler as a useless peer. In the second case after the Crawler connects and disconnects from a given peer few times (2 or 3), this peer also blacklists the Crawler's IP address. The IP addresses in the blacklist have a timer associated, after this timer expires the IP address is removed from the blacklist. This means that the Crawler can contact a given peer in intervals $\geq$ blacklist timer[7].

Possible solution: In the case we want to monitor the peers with a higher resolution than that imposed by the peer's *blacklist* timer, the solution is using several instances of our Crawler, each one with a different IP address and contact a given peer following a round robin schedule [13]. We could also use TOR, if two connections to the same destination are at least 10 min apart, TOR establishes a new overlay path with a new egress node. Thus, TOR guarantees a 10 min resolution.

Completeness of a torrent population

Description: BitTorrent developers have recently implemented the *magnet links*. Basically, this is an id that allows a peer to learn IP addresses of peers participating in the swarm directly from the DHT service without connecting to the Tracker. Therefore, the Tracker is unaware of the presence of these peers. Although the clients using magnet links to access a swarm are still a minority, this brings some difficulties to obtain the complete list of peers participating within an specific swarms because we need to obtain those that are available from Tracker information and those that are available from the DHT service (note that some peers will be available from both). Unfortunately, this problem is even more complicated. Some torrents are associated to multiple Trackers that form separated swarms. In short, to retrieve the whole set of peers downloading a given content we should collect the peers from each of the Trackers and those available through the DHT service.

Possible Solution: In order to retrieve the whole set of peers downloading a given content, we need to crawl all the Trackers included in the .torrent file. Furthermore we have to retrieve the list of peers that use the DHT instead of using a Tracker. This crawling can be quite costly since some torrents can use tens of Trackers.

Upload Rate Estimation:

Description: We have discussed above the difficulties for measuring the upload rate and what other parameters have been measured as an approximation of the upload rate so far.

Possible Solutions: The only available technique that allows to accurately measure the upload rate of a peer is the one based in our own BitTorrent client (or plugging) implementation.

[7]This timer value varies among the different clients. A conservative estimation based in our studies is 2 hours.

V. Conclusion

In this paper we have presented and classified the main measurement techniques applied in order to understand different aspects of one of the largest-scale systems in the current Internet, i.e., BitTorrent. We believe that the described techniques can constitute the basis for the design of measurement tools for the analysis of current and future large-scale systems in the Internet, but also other environments.

References

[1] MaxMind- GeoIP. http://www.maxmind.com/app/ip-location.

[2] Tribler. http://www.tribler.org.

[3] S. Le Blond, A. Legout, F. Lefessant, W. Dabbous, and M. Ali Kaafar. Spying the world from your laptop. *LEET'10*, 2010.

[4] David R. Choffnes and Fabián E. Bustamante. Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems. *SIGCOMM Comput. Commun. Rev.*, 38(4):363–374, 2008.

[5] Bram Cohen. Incentives build robustness in BitTorrent. In *Proc. of First Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, USA, Jun 2003.

[6] R. Cuevas, M. Kryczka, A. Cuevas, S. Kaune, C. Guerrero, and Rejaie.R. Is content publishing in bittorrent altruistic or profit-driven? In ACM CoNEXT 2010, 2010.

[7] R. Cuevas, N. Laoutaris, X. Yang, G. Siganos, and P. Rodriguez. Deep diving into bittorrent locality. Proc. of IEEE INFOCOM'11, 2011.

[8] M. Ford, M. Boucadair, A. Durand, P. Levis, and P. Roberts. Issues with IP Address Sharing. draft-ietf-intarea-shared-addressing-issues-05, 2011.

[9] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. Measurements, analysis, and modeling of bittorrent-like systems. In *Proc. of ACM IMC'05*.

[10] Ipoque. Ipoque internet study 2007, 2007. http://www.ipoque.com/userfiles/file/internet_study_2007.pdf.

[11] T. Isdal, M. Piatek, Krishnamurthy. A, and Anderson T. Leveraging bittorrent for end host measurements. In *PAM*, 2007.

[12] M. Izal, G. Urvoy-Keller, E.W. Biersack, P.A. Felber, A. Al Hamra, and L. Garces-Erice. Dissecting bittorrent: Five months in a torrent's lifetime. In *Proc. of PAM '04*.

[13] S. Kaune, R. Cuevas, G. Tyson, A. Mauthe, C. Guerrero, and R. Steinmetz. Unraveling BitTorrent's File Unavailability: Measurements, Analysis and Solution Exploration. In *IEEE P2P'10*, 2010.

[14] Georgos Siganos, Xiaoyuan Yang, and Pablo Rodriguez. Apollo: Remotely monitoring the bittorrent world. Technical report, available from: http://research.tid.es/georgos/images/apollo_imc09.pdf, Telefonica Research, 2009.

[15] M. Steiner and Biersack E. W. Crawling azureus. Technical report, institut eurecom. available from: http://www.eurecom.fr/~btroup/BPublished/RR-08-223.pdf, 2008.

[16] C. Zhang, P. Dhungel, D. Wu, and K.W. Ross. Unraveling the bittorrent ecosystem. *IEEE Transactions on Parallel and Distributed Systems*, 2010.