Measuring the BitTorrent Ecosystem: Techniques, Tips, and Tricks

Michal Kryczka, University Carlos III Madrid and Institute IMDEA Networks

Rubén Cuevas, Carmen Guerrero, and Arturo Azcorra, University Carlos III Madrid

Angel Cuevas, Institut Telecom, Telecom SudParis

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 article 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 us to collect aggregated information of torrents and present very high scalability, able to monitor up to hundreds of thousands of torrents in short periods of time. Microscopic techniques operate at the peer level and focus on understanding performance aspects such as the peers' download rates. They offer higher granularity but do not scale as well as macroscopic techniques. Finally, complementary techniques utilize recent extensions to the BitTorrent protocol in order to obtain both aggregated and peer-level information. The article also summarizes the main challenges faced by the research community to accurately measure the BitTorrent ecosystem such as accurately identifying peers and estimating peers' upload rates. Furthermore, we provide possible solutions to address the described challenges.

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 used daily by dozens of millions of users. This has attracted the interest of the research community, which has thoroughly evaluated the performance and 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, that 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 article we first describe the main aspects and functionality of the complete BitTorrent ecosystem. Afterward, we present a survey of the existing BitTorrent measurement techniques. Finally, we describe the main challenges these techniques face and possible solutions to them before concluding the article.

BITTORRENT ECOSYSTEM

BitTorrent [5] is the name used by Brian Cohen to define the peer-to-peer file sharing protocol he designed a decade ago. Due to the great success of this protocol, a complex system was created around it. In this article 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.

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 with 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, and thus only uploads pieces to other peers. A *leecher* is a client that does not have a complete copy of the content, and thus uploads and downloads pieces to and from other peers, respectively.

• A *.torrent file* is a meta-information file asso-

ciated with 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.

PUBLISHING CONTENT IN BITTORRENT

In order to make available content *C* in Bit-Torrent, the content publisher creates a .torrent file associated with *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 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. Figure 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 new published torrents.

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 with *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 neighborhood of the new node. Furthermore, if a peer's neighborhood size falls below a given threshold (typically 20), it again sends an announce started request to the tracker in order to get new neighbors. 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 Bit-Torrent ecosystem relies on a few trackers that manage a large number (up to a few million) of torrents in parallel. The OpenBitTorrent and PublicBitTorrent trackers[3] are currently the most important ones.
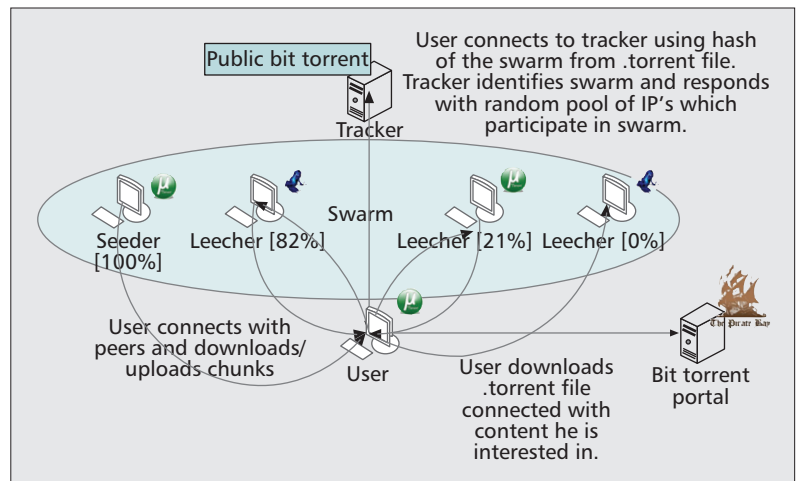
Figure 1. *BitTorrent Ecosystem basic functionality: i) the BitTorrent client contacts a BitTorrent portal to download the .torrent file associated to the desired content (the .torrent file includes the IP address of the Tracker managing the swarm associated to the desired content); ii) the BitTorrent client contacts the Tracker that provides the IP addresses of a set of peers within the swarm; and iii) the BitTorrent client connects to these peers for downloading the content.*

BITTORRENT DELIVERY PROCEDURE

In BitTorrent two peers communicate using the *peer wire* protocol. Every communication starts with an initial handshake. Just after the handshake sequence is completed (and before any other messages are sent), the peers exchange *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 its neighbors by using a *HAVE* message. Hence, every peer is aware of the chunks each neighbor has at any moment.

BitTorrent uses *Tit-for-Tat* as the 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 behavior. It is a periodical operation where every 10 s a leecher selects (unchokes) *n* other leechers from its neighborhood to upload chunks to. These *n* (typically 4) unchoked leechers are those from whom the peer downloaded more chunks during the last 20 s. The rest of the neighbors 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 s (i.e., those with higher download rate).[4] In addition to the regular unchoke operation, BitTorrent implements the *optimistic unchoke* operation. Every 30 s (i.e., every 3 regular unchoke operations) both leechers and seeders randomly select one choked neighbor 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 of this neighbor. Since leechers have full knowledge about the availability of every chunk in their neighborhood, it always requests the rarest one.

BITTORRENT EXTENSION

Several extensions to BitTorrent have been proposed so far. Here we just mention those relevant to measurement studies.

[1] *This is currently the largest BitTorrent portal based on Alexa ranking.*

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

[3] *http://www.openbittorrent.org and http://www.publicbt.org*

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

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 trackerless mechanism that allows a BitTorrent user to learn 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 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 neighbors (e.g., *N*). If *N* supports PEX, it responds with the list of IP addresses of all its neighbors. Hence, by using a few PEX queries a given peer can learn the IP addresses of a large number of participants in the swarm without requesting them from the tracker.

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 perfor-

Figure 2. *Example of a 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 Gbytes), language (English), upload 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.*

mance information. A summary of different techniques is presented in Table 1.

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 Internet service providers, ISPs), the relevance of the different portals and trackers, and so on. Furthermore, the macroscopic measurements also allow some performance aspects to be studied, such as the ratio of seeders/leechers, the session time of BitTorrent users, the arrival rate of peers, the seedless state (period the torrent is without a seeder) duration, and so on.

We classify the macroscopic techniques into two subcategories: *BitTorrent portals crawling* and *BitTorrent trackers crawling*.

BitTorrent Portals Crawling — As shown earlier, the (major) BitTorrent portals index millions of torrents in a structured way. Furthermore, they provide detailed information about each indexed torrent (typically) in a specific torrent web page. For instance, in the case of The Pirate Bay, the torrent web page associated with a torrent with an assigned torrent-id equal to i can be accessed through the URL http://thepiratebay.org/torrent/i (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 a specific portal (we use The Pirate Bay as our example).

Backward Crawling — In this case the aim is to retrieve the information associated with 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 on 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.

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

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

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 1. *Comparison of main BitTorrent measurement techniques.*

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 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 time.

BitTorrent Tracker 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 (remember that a tracker managing a given swarm knows the IP addresses of all the participants).

There are various ways to access 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 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. Afterward, the crawler performs as many 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 us to study some relevant demographics and BitTorrent performance features. Next, we briefly describe some of them:
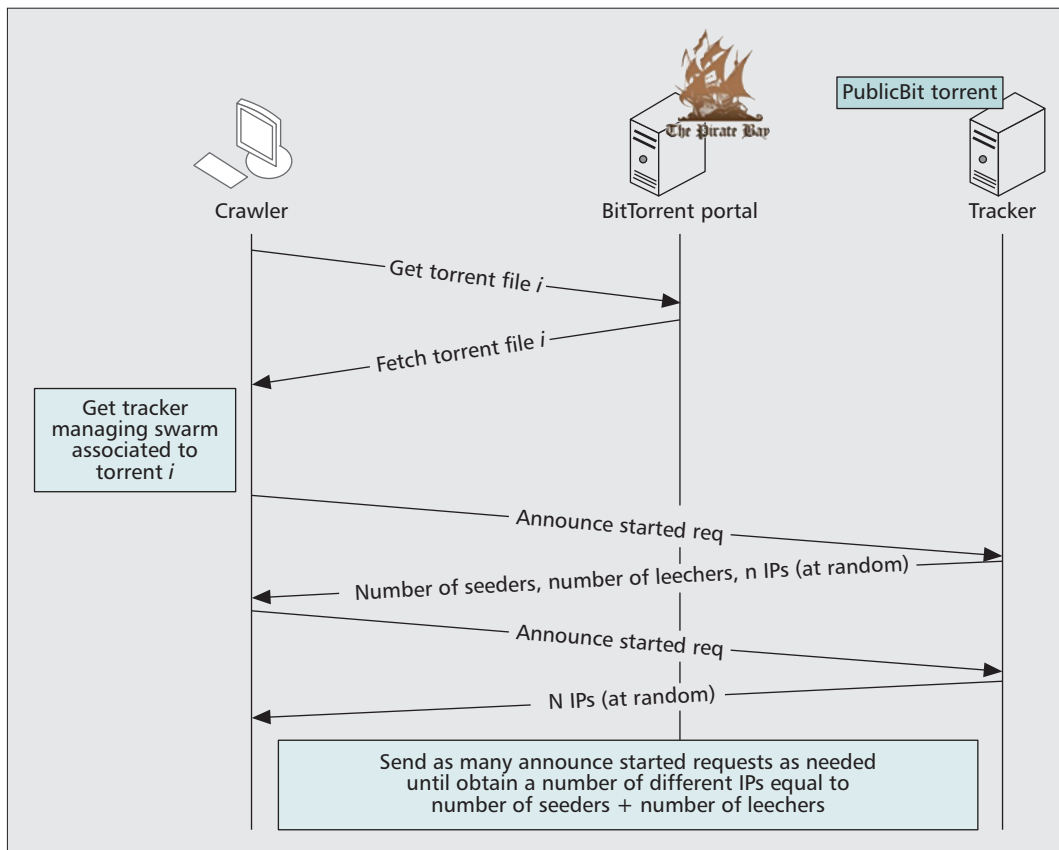
The distribution of clients per country or ISP: Some studies have applied the described crawling technique to a large number (even millions) of torrents [16]. Afterward, the IP address of each client is mapped to its country and ISP (e.g., using the MaxMind database [1]). From this data we can compute 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) present in a large number of torrents [3]. We call 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 40,000 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)

Figure 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. Afterward, it sends as many announce started requests as needed until it obtains the IP addresses of all the peers participating in the swarm.*

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 on a large number of torrents.

MICROSCOPIC TECHNIQUES

The described macroscopic techniques exclusively retrieve the peers' IP addresses, so only metrics associated with 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 call *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 handshake procedure. This is essential to connect to other peers. The handshake procedure can be done actively (the crawler initiates it) or passively (the crawler waits until a peer starts the handshake). 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
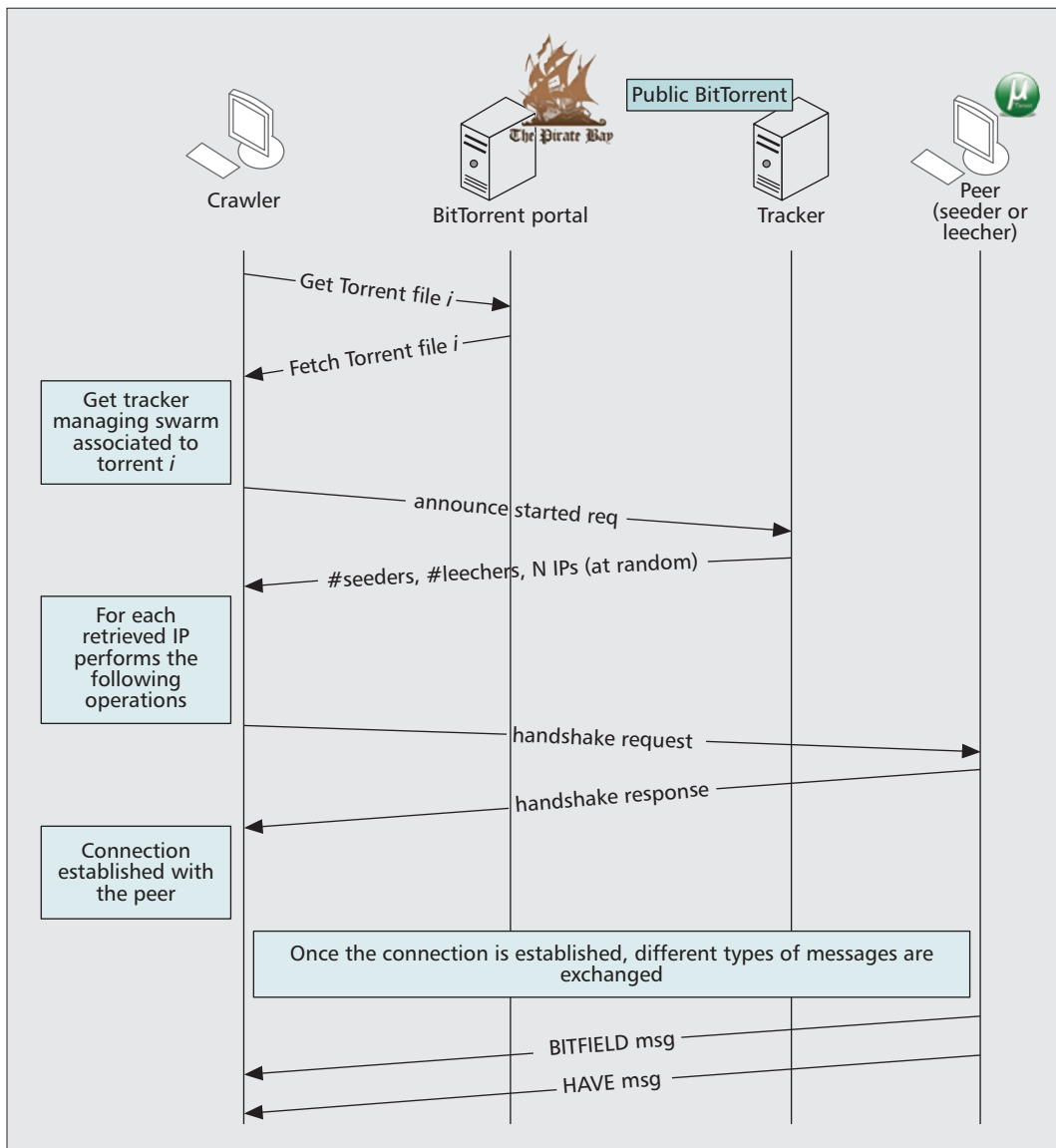
literature to measure the most important peer-level performance aspects of BitTorrent.

Peer Type — After the handshake procedure succeeds 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 network address translation (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 handshake 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., 4 Mbytes) 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, which 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

---

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

Figure 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. Afterward, the crawler contacts each individual peer, performs the handshake procedure, and exchanges different messages (BITFIELD, HAVE) to obtain different peer-level performance information.*

the peer. By repeating this operation periodically, we can obtain the evolution of the instantaneous download rate of a given peer [14].

Average Download Rate — In this case the crawler connects to a peer, obtains its bitfield, and disconnects. After some time (e.g., 1 h) 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 measure. Indeed, to the best of the authors' knowledge there is no work that has

properly measured the upload bandwidth. Rather, some have measured some parameters related to the upload rate. On one hand, Isdal *et al.* [11] measure the physical upload capacity (≥ 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 Internet Control Message Protocol (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 with BitTorrent as well as with other applications.

Chunk Distribution (Rarest First Performance) — An important aspect of the 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 bitfields of a large number of peers in a swarm (ideally all) in a relative short period of time (a few minutes). By analyzing the collected bitfields we achieve our objective: 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.

COMPLEMENTARY TECHNIQUES

Some researchers have used measurement techniques that can complement the above described macroscopic and microscopic techniques. On 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 plug-in for a popular BitTorrent client such as Vuze [4]. These clients (or plug-ins) report information to a log server. This technique complements the microscopic measurement mechanisms since it gives very accurate information regarding peer-level performance parameters; for instance, it can provide precise information about a peer's download and upload rates.

TECHNIQUE COMPARISON

In this subsection we compare the different measurement techniques introduced above, stating the pros and cons of each. Macroscopic techniques are the most scalable, allowing us to analyze up to hundreds of thousands of torrents. These techniques are valid to retrieve:
• Aggregated information at the swarm level
• Specific information regarding the presence of a peer (represented by the IP+port) in a given swarm
Therefore, they are useful to characterize important information such as the content publishing phenomenon (i.e., which users are responsible for making available the content shared through BitTorrent), popularity distribution of torrents, seeder/leecher ratio, peers' session time, and cross-torrent interactions (e.g., peers participating in multiple torrents). However, these techniques cannot provide information at the peer level (e.g., a peer's download progress, download rate, chunk distribution) since this requires contacting the peer. Microscopic techniques were designed to perform peer-level analysis. For this purpose the measurement software connects periodically to a large number of peers (potentially to all the peers participating in the set of analyzed torrents). This makes microscopic techniques to scale up to analyze (at most) a 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. 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 query 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, measurement tools based on traditional tracker crawling are simpler than enhanced versions that implement a PEX and/or DHT module.

On the other side, the collection of data based on a specific client or plug-in 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 (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 plug-in), which means we have a partial view of the analyzed torrent. Moreover, the retrieved data is only representative of a specific client with a specific implementation, so the obtained results may not be generalized to other clients. Traditional microscopic techniques lack the level of accuracy of client-based measurement but offer 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.

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. Thus, 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 IP address+port to identify a single user across multiple sessions. This works for all those users having static IP addresses. How-

ever, most 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 NAT [8], so using the IP address to identify a peer may lead to wrongly mapping several users as a single peer.

Possible Solutions — 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/plug-in. 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. Another option is to get access to the logs of private trackers. In most 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 log in, and thus can be uniquely identified across sessions. Unfortunately, both described techniques have scalability limitations. On the other hand, identifying a 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, so 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 cause the crawler's IP address to be banned by the tracker. In some studies [3, 6, 16] the crawler performs 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, which is detected by the tracker. The reaction of the tracker is to block the IP address showing this anomalous behavior. Therefore, the crawler has to limit its 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 of 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 750,000 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, so 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 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 handshake procedure with the target peer. Afterward, it retrieves the needed information (e.g., the bitfield), and can then either stay 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 another peer in its neighborhood. Once the crawler has been removed from the peer's neighborhood, 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 a few times (two or three), this peer also blacklists the crawler's IP address. The IP addresses in the blacklist have a timer associated with them; after this timer expires, an IP address is removed from the blacklist. This means that the crawler can contact a given peer in intervals ≥ blacklist timer.[7]

Possible Solution — When 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 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 *magnet links*. Basically, these are ids that allow 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 clients using magnet links to access a swarm are still a minority, this brings some difficulties in obtaining the complete list of peers participating within 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

> In the case of microscopic measurements the crawler always performs the handshake procedure with the target peer. Afterward, it retrieves the needed information, and can then either stay connected or disconnect and reconnect after a while.

be available from both). Unfortunately, this problem is even more complicated. Some torrents are associated with multiple trackers that form separated swarms. In short, to retrieve the whole set of peers downloading 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 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 in 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.

CONCLUSION

In this article 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, 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, as well as other environments.

REFERENCES

[1] MaxMind-GeoIP, http://www.maxmind.com/app/ip-location.
[2] Tribler, http://www.tribler.org.
[3] S. Le Blond *et al.*, "Spying the World from Your Laptop," *LEET '10*, 2010.
[4] D. R. Choffnes and F. E. Bustamante, "Taming the Torrent: A Practical Approach to Reducing Cross-ISP traffic in Peer-to-Peer Systems," *SIGCOMM Comp. Commun. Rev.*, vol. 38, no. 4, 2008, pp. 363–74.
[5] B. Cohen, "Incentives Build Robustness in BitTorrent," *Proc. 1st Wksp. Economics of Peer-to-Peer Sys.*, Berkeley, CA, June 2003.
[6] R. Cuevas *et al.*, "Is Content Publishing in Bittorrent Altruistic or Profit-Driven?," *ACM CoNEXT 2010*, 2010.
[7] R. Cuevas *et al.*, "Deep Diving Into Bittorrent Locality," *Proc. IEEE INFOCOM '11*, 2011.
[8] M. Ford *et al.*, "Issues with IP Address Sharing," draft-ietf-intarea-shared-addressing-issues-05, 2011.
[9] L. Guo *et al.*, "Measurements, Analysis, and Modeling of Bittorrent-Like Systems," *Proc. ACM IMC '05*.
[10] Ipoque, Ipoque Internet Study 2007, http://www.ipoque.com/userfiles/file/internet_study_2007.pdf.
[11] T. Isdal *et al.*, "Leveraging Bittorrent for End Host Measurements," *PAM*, 2007.
[12] M. Izal *et al.*, "Dissecting BitTorrent: Five Months in a Torrent's Lifetime," *Proc. PAM '04*.
[13] S. Kaune *et al.*, "Unraveling BitTorrent's File Unavailability: Measurements, Analysis and Solution Exploration," *IEEE P2P '10*, 2010.
[14] G. Siganos, X. Yang, and P. Rodriguez, "Apollo: Remotely Monitoring the Bittorrent World," tech. rep.; http://research.tid.es/georgos/images/apollo_imc09.pdf, Telefonica Research, 2009.
[15] M. Steiner and E. W. Biersack, "Crawling Azureus," tech. rep., Institut Eurecom; http://www.eurecom.fr/~btroup/BPublished/RR-08-223.pdf, 2008.
[16] C. Zhang *et al.*, "Unraveling the BitTorrent Ecosystem," *IEEE Trans. Parallel and Distrib. Sys.*, 2010.

BIOGRAPHIES

MICHAL KRYCZKA received his M.Sc. in computer science from the Technical University at Lodz, Poland, and his M.Sc. in telematics engineering from the University Carlos III of Madrid, Spain, in 2008 and 2009, respectively. Since 2008 he is a research assistant at Institute IMDEA Networks and a Ph.D. candidate in the Telematic Engineering Department at University Carlos III of Madrid. His current research lines include peer-to-peer technologies, Internet measurements, social networking, and content distribution.

RUBÉN CUEVAS obtained his Ph.D. and M.Sc. in telematics engineering and M.Sc. in telecommunications engineering at University Carlos III of Madrid in 2010, 2007, and 2005, respectively. Furthermore, he received his M.Sc. in network planning and management from Aalborg University, Denmark, in 2006. He is a member of the Telematics Engineering Department and NETCOM research group at University Carlos III of Madrid since 2006, where he is currently an assistant professor. He has been involved in several international and national research projects. Moreover, he is coauthor of more than 20 papers in prestigious international journals and conferences such as *IEEE Network*, Elsevier *Computer Network*, ACM CoNEXT, IEEE INFOCOM, IEEE P2P, IEEE ICC, and IEEE VTC. His main research interests are content distribution, P2P networks, online social networks, and Internet measurements.

ANGEL CUEVAS received his Master's in telecommunication engineering, and M.Sc. and Ph.D. in telematics engineering from the University Carlos III of Madrid in 2006, 2007, and 2011, respectively. He is currently a postdoc researcher at Institut Telecom SudParis in the Department of Wireless Networks and Multimedia Services. His research is focused on the areas of sensor networks, peer-to-peer systems, and social networking. He has published more than 20 papers in high-quality international conferences and journals. He was recipient of the Best Paper award at the ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems 2010.

CARMEN GUERRERO received her Telecommunication Engineering degree in 1994 from the Technical University of Madrid (UPM), Spain, and her Ph.D. in computer science in 1998 from the Universidade da Coruna (UDC), Spain. She is currently associate professor since 2003 at Universidad Carlos III de Madrid, teaching computer networks courses. She has been involved in several national and international research projects related to green networking, future Internet, and content distribution. More details are available at http://www.it.uc3m.es/carmen.

ARTURO AZCORRA received his M.Sc. degree in telecommunications engineering from UPM in 1986 and his Ph.D. from the same university in 1989. In 1993 he obtained an M.B.A. from the Instituto de Empresa. He has recently been appointed director general at the Centre for the Development of Industrial Technology (CDTI), an agency of the Spanish Ministry of Science and Innovation. As a result, he is on leave from his double appointment as full professor (with chair) at the Telematics Engineering Department of University Carlos III of Madrid and director of IMDEA Networks. He has participated in and directed 49 research and technological development projects, including European ESPRIT, RACE, ACTS, and IST programs. He has served as a Program Committee member in many international conferences, including several editions of IEEE PROMS, IDMS, QofIS, CoNEXT, and IEEE INFOCOM. He has published over 100 scientific papers in books, international magazines, and conferences.