Comparing BitTorrent Clients in the Wild: The Case of Download Speed

Marios Iliofotou*, Georgos Siganos†, Xiaoyuan Yang†,
and Pablo Rodriguez†
* University of California, Riverside
† Telefonica Research, Barcelona

***Abstract*—BitTorrent (BT) is currently the main P2P protocol used for sharing large files over the Internet. It is therefore important to understand the performance characteristics of existing real-world BT implementations (clients). In this work, we measure the download speed of over 10 million BT users over one month. Surprisingly, our measurements show that the two most famous BT clients, namely uTorrent and Vuze (Azureus), achieve different download speeds for the same set of torrents. In particular, we observe that uTorrent users achieve 16% faster download speed than users of Vuze in our data set. To shed light to the cause of this difference, we reverse engineer the two clients to infer their individual design choices. Our study shows that the two clients differ in two important areas: (a) how they manage their neighborhood, and (b) how they distribute their upload capacity to their peers. We speculate that the cause of the mismatch in download speeds can be attribute to these differences. We hope that our findings will open the door for new research efforts to better understand the impact of design choices in the performance of real-world BT implementations.**

I. Introduction

BitTorrent's popularity and open design has spurred a number of 3rd party BitTorrent (BT) client implementations. These clients provide different interfaces (GUIs) as well as functionalities to their users. For example, Vuze - a popular BT client - provides the ability to the user to search for new files (torrents) and allows users to directly stream video files to their computers. Even though different clients offer different functionalities, they all implement the same application-layer protocol; as initially proposed by Cohen et al. [2]. Due to the existence and popularity of many implementations, today's BT swarms[1] exhibit a remarkable "bio-diversity" of coexisting clients. Since all BT clients follow the same application-layer protocol, they are all compatible and can exchange control messages as well as files with one-another.

The most commonly used clients today are uTorrent [14], Vuze [15], Mainline [2], and Transmission [13]. From all existing client implementations, our measurement data show that uTorrent and Vuze are the most prevalent and cover 50% and 25% respectively of all BT users in Pirate Bay[2]. Our work here is motivated by the following questions: *"Are the existing client implementations different in only their interface and extra functionalities? Is there one client that can achieve faster download speeds than the other?"* To answer these questions, we conduct a large-scale measurement study on the download speed of the two most popular BT client implementations; namely uTorrent and Vuze. Our study reveals a surprising behavior. We observe that peers using uTorrent can reach on average 16% faster download speeds than the users of Vuze.

[1]A swarm represents a set of online hosts downloading the same file.
[2]Pirate Bay is the largest web-site providing torrent search functionality.

In an attempt to identify the root causes of the mismatch in download speed, we reverse engineer the two clients in order to infer their individual design choices. Our study shows that the two clients differ in two important areas: (a) on how they manage their neighborhood, and (b) on how they distribute their upload capacity to their peers. We speculate that the cause of the speed mismatch can be attributed to some or all of these differences. Evaluating the contribution of each difference to the download speed is out of the scope of this paper and is included in our future directions. We expect our work to show the way for new research in the area of real-world BT client implementations. This is particularly important, especially now that the two clients are considering to adopt different transport protocol policies by using both TCP and UDP for their file transfers [12]. Therefore, research initiatives that can evaluate such design choices will become increasingly attractive in the near future.

The highlights of our work can be summarized in the following points:

- We conducted a large-scale study measuring the achieved download speed of many millions of BT users. Our study revealed that uTorrent achieves 16% faster download speed than Vuze for the same set of torrents. For the collection of our data we used the Apollo [11] system from Telefonica Research.
- We reverse engineer the two BT clients and identified several differences in how they manage their connections with other peers and how they distribute their upload capacity.

The remaining of the paper is organized as follows. In §II we provide details about the collection of BT data using Apollo [11] and present our observations on the speed mismatch. In §III we reverse engineering Vuze and uTorrent and reveal four main implementation differences. We discuss related work in §IV. We conclude the paper with a summary and our future directions in §V.

II. Comparing Download Speeds

A. Experimental methodology

Download speed estimation. To estimate the speed of a BT user for a particular torrent, Apollo [11] uses the following methodology. It connects with the peer two times with a five minute gap. Each time it request the `bitfield` that reports which pieces the peer has at each time instance. Apollo also takes into consideration any BT `HAVE` messages

Client	*Avg.*	*Med.*	30^{th} *pct.*	60^{th} *pct.*	70^{th} *pct.*
uTorrent	176	90	65	119	165
Vuze	151	81	61	106	147

TABLE I
Comparing the speed of the two clients over the entire data set. All reported speeds are in Kbps.

it receives from the user in addition to what is reported by the `bitfield`. The speed of the peer is then calculated as the new pieces (bytes) obtained over the time period divided by the length of the period. The estimated download speed corresponds to the speed of a single BT user for a particular torrent.

Data collection. For each data-collection-cycle (currently one hour long), Apollo analyzes all the BT users that download the top-600 most popular torrents of Pirate Bay [10], [11]. For each BT user, our data collection system records: the estimated speed, client type (e.g., Vuze, uTorrent, etc.), and ISP (as Autonomous System Number - ASN). This data-collection-cycle is repeated for every hour. Our study covers the entire month of August 2009. In the remaining of the paper, we present data from one representative week: 08/12/09 until 08/19/09. Qualitatively similar results are observed for other days as well. Our data set contains information from 11 millions BT clients from 6,200 different ASNs involved in more than 1,100 torrents.

Fair comparison. To be able to make a fair comparison we use per torrent the same sample size on the ISP level, so that the characteristics of the torrent or the characteristics of the ISP is not biasing the result. Our selection process is as follows. For each $\{torrent, ASN\}$ pair we select the same number of peers for each client (uTorrent and Vuze). For our study, we do not include $\{torrent, ASN\}$ pairs with less than 10 users for each of the two clients. If for a particular $\{torrent, ASN\}$ one client has more users than the other, we select a random subset of the largest client so that the sets we are comparing are of the same cardinality. We repeat the random sampling multiple times (> 10) to verify that we get consistent results. By following this selection process, our final sample of peers from each client (Vuze or uTorrent) have the same overall size as well as identical sample sizes from each $\{torrent, ASN\}$ pair. Using this selection process, the number of hosts per ASN for each client varies from 10,000 up to 80,000 BT users allowing statistically meaningful comparison.

B. Download speed comparison results

1) Overall speed differences: Our measurements show that uTorrent users can achieve on average 16% faster download speed than users of Vuze. The comparison results are summarized in Table I. On average uTorrent users achieve 176 Kbps while the users of Vuze achieve 151 Kbps over the same torrents and ISPs. The difference is also present in other percentiles of the speed distribution. We see this in Table I where we report the 30^{th}, 50^{th} (median), 60^{th}, and 70^{th} percentiles of the speed distribution over all peers over one week. Clearly, uTorrent is faster over a large range of percentiles. In addition, we see that the difference between the two clients gets higher for larger percentiles. For example, the gap is 6% for the 30^{th} percentile and rises up to 12% for

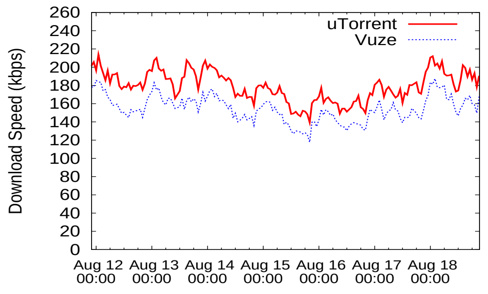


Fig. 1. Average speed in time for the two clients over all ISPs.

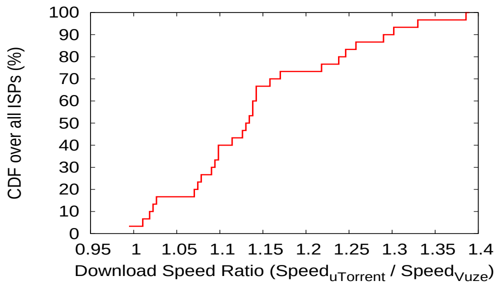
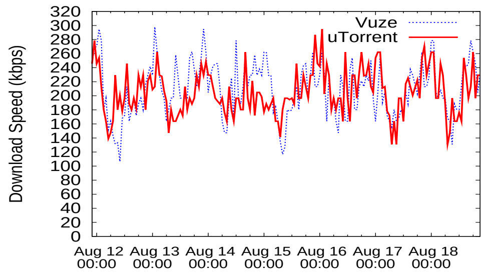


Fig. 2. Comparing among top ISPs. All ratios above 1 show that uTorrent is measured to be faster for that ISP

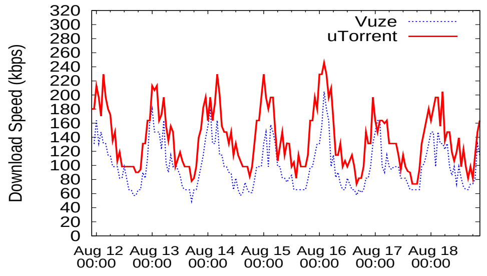
the 70^{th} percentile. This shows that uTorrent tends to achieve better speeds over Vuze for its fast clients (higher percentiles) compared to its slower ones (lower percentiles). Further study of this behavior is out of the scope of this paper and is left for future work.

In Fig. 1 we show the download speed over time for the two client for a period of one week. The plot shows the average download speed of the peers for each client summarized over each hour in the data set. As mentioned before in §II-A, we use the same number of BT users for each client taken over the same set of torrents and ISPs (ASNs). In the time series of Fig. 1 even though diurnal speed variations exist for both clients, uTorrent is consistently faster than Vuze. For the remaining of the paper, for each client, we summarize the download speed using the average value over all BT hosts unless otherwise stated.

2) Download speeds for different ISPs: Our goal here is to answer the following questions: *"Is uTorrent faster than Vuze in all ISPs? Is the speed difference the same across different ISPs?"* For the following experiments, we present results for the top 30 ISPs in our data ranked by the total number of observed BT users. Qualitatively similar results are observed when we include other ISPs as well. Our measurements show that uTorrent is consistently faster than Vuze for all ISPs; with Verizon (ASN 19262) being a notable exception. The distribution of per ISP speed ratio between uTorrent and Vuze is summarized in Fig. 2. A ratio above 1 means than uTorrent achieves on average faster download speed than Vuze for that ISP over the one week in our data set. From the CDF of Fig. 2 we see that for all ISPs the ratio is above 1. Moreover, we see that for the majority of ISPs (>50%) the ratio is >1.13 showing that uTorrent is 13% faster than Vuze. The difference is even higher for 10% of the ISPs where uTorrent achieves 30% faster download speeds than Vuze. The only



(a) Verizon: almost identical speeds.



(b) UPC Broadband: uTorrent is faster than Vuze.

Fig. 3. Comparing the uTorrent and Vuze clients over two ISPs.

ISP with different behavior is Verizon, where the two clients are observed to have nearly identical speeds (ratio≈1). To show this visually, in Fig. 3(a) we plot the time-series of the measured speeds for the two clients in Verizon. The time-series shows that the two clients have similar speeds without a clear winner. To contrast this behavior, in Fig. 3(b) we show the behavior of a "typical" ISP (UPC Broadband). For UPC, even though the two clients follow the same diurnal pattern, uTorrent is faster than Vuze for the vast majority of hours over the week. For these two ISPs, our week-long data set consists of 23k and 24k users per client for Verizon and UPC respectively. Investigating the root causes of this behavior with Verizon is an interesting direction for future work.

III. Highlighting Implementation Differences

This section is motivated by the following question: *"Can we identify differences between uTorrent and Vuze than can explain the observed mismatch in download speeds?"* Towards this end we use raw packet traces collected from the two clients and reverse engineer some of their basic functionalities. We acknowledge that the code of some BT implementations (e.g., Azureus) is open and we can inspect the code to understand specific design choices. However, we choose to use a trace-driven approach because is general enough to be applied to any client.

A. Experimental methodology

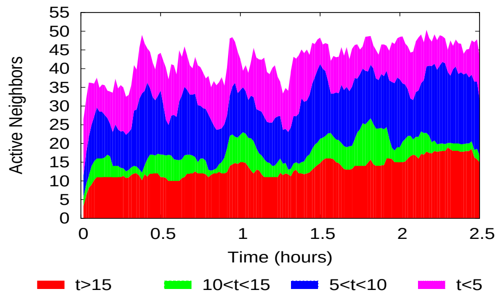
Reverse engineering Vuze and uTorrent. Initially we set a client to download a particular torrent and then capture all incoming/outgoing traffic using `wireshark`. To identify the BT flows we do a first pass over the trace and used: (a) the deep packet inspection functionalities of `wireshark` and (b) our knowledge of the listening port of the local host. All incoming flows to the assigned listening port are assumed to be incoming BT connections. From the first pass, all remote $\{IP, port\}$ pairs involved in BT are stored in a data base. We then perform a second pass over the trace and mark as BT all the flows that have a remote $\{IP, port\}$ in our data base. Having all the BT flows (UDP and TCP), we can then use a trace driven simulation and reconstruct the internal state of each BT client. This allows the estimation of: the total number of open (active) connections, the upload/download speed of each connection, when a connection is opened/closed, when a connection is actively sending/receiving payload packets, etc. Using this information we can extract internal mechanisms and parameters for each of the two client. For our study, we used the most recent versions of the two clients at that time; uTorrent v1.8.3 and Vuze v4.2.0.4. We used the default parameters for both clients since we expect most BT users to do the same. Both clients allow up to 50 open connections to other peers and have no restrictions on their download speed. Both clients support an automated method to adjust their upload rate. In uTorrent this functionality is not enabled by default while Vuze enables the auto-rate by default. Evaluating the impact of this parameter to the performance of the two clients remains an interesting direction of future work. More details regarding the automated upload rate adjustment can be found in [14], [15].

Data collection. For collecting our traces we used two different machines connected at two different DSL lines of the same service (i.e., same download/upload capacities) and from the same ISP (Telefonica Spain). At any point in time the two computers use a different BT client (uTorrent or Vuze) and download the same torrent over the same time interval. To eliminate characteristics of the DSL line that might effect the results, we interchanged which client is running at each computer over each iteration of the experiment. Each experiment lasted from three to eight hours depending on the torrent size and the download speed of the peers. Our monitoring system automatically terminated all downloads as soon as the entire file is downloaded. The actual download file is then deleted. Only the section of the trace where our local client act as a leecher is of interest, since we are only concerned in the download activity of the peers. We collected traces over different times of the day, over different days of the week over two months; July and August 2009. For downloading, we always selected a torrent from the top 20 most popular torrents in Pirate Bay.

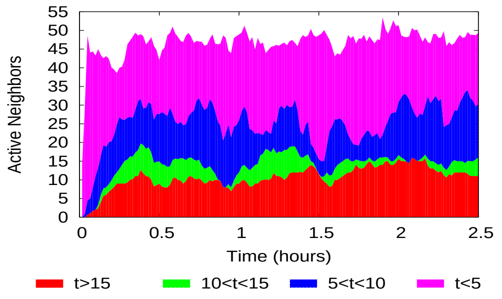
B. Implementation differences

From our measurements we observed qualitatively the same results over all our experiments. For presentation purposes, in the remaining of the paper we will present specific results from a single iteration of the trace-collection taken on the 8th of August 2009, starting at 6PM Spain local time.

1) *Neighborhood set stability:* The two clients are different in how they manage their neighborhood. In the following experiments, both clients are set to have at most 50 simultaneous open connections (default configuration). At any time, we refer to the set of connected remote peers as the **neighbors** of the local peer. We further group the neighbors according to how long their connection with the local peer will last. In particular, we group the neighbors in four groups based on their connection duration as follows: (i) below five minutes

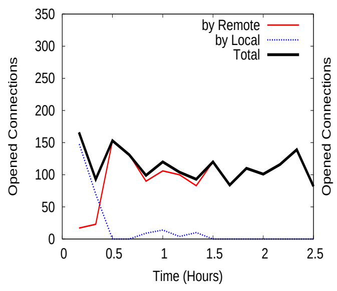


(a) uTorrent.

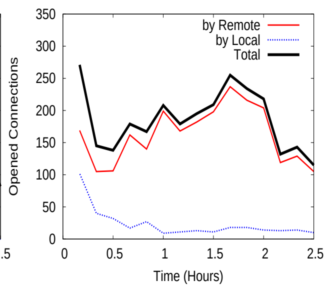


(b) Vuze.

Fig. 4. Comparing the stability of the neihboorhood for each client.



(a) uTorrent.



(b) Vuze.

Fig. 5. New connections opened over time. In the two plots, we also show how many of the new connections are initiated by the local host and how many from the remote hosts. The number of connections is summarized over bins representing 10 minutes in our traces.

(short-lived connections), (ii) from 5 to 10 minutes, (iii) from 10 to 15 minutes, and (iv) more than 15 minutes (long-lived connections). The results are summarized in Fig. 4. From the time-series we see that the two clients have differences in how they manage their neighborhood size. Vuze constantly tries to keep 50 open connections, while uTorrent seems to have higher variability in its neighborhood set. Moreover, uTorrent has more neighbors that stay longer connected than Vuze. Next, we present additional clues to support these observations.

2) Opening new connections: The two clients have also differences in how they open new connections. At the early download stages (roughly the first 20 minutes) both clients initiate new connections to explore their neighbors. After this initial stage, the uTorrent client makes new connections with remote peers that send an incoming request and very rarely initiates new connections. On the other hand, even though Vuze also "discovers" most of its neighbors from its incoming requests, it constantly initiates new connections over the entire duration of its operation. We show this behavior in Fig. 5

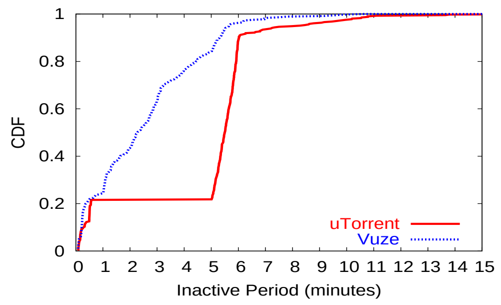


Fig. 6. Comparing the connection termination policy for each client. The CDFs summarize the inactivity period before a connection gets closed. Inactivity period is defined as the time-interval since the remote hosts sent the last packet with payload to the local host.

where we plot the time-series of new opened connections for both clients. The figure shows the total number of open connections over 10 minute bins. It also shows how many connections are initiated by the local peer and how many by the remote peers. From the time-series we see that overall Vuze opens more connections than uTorrent. This observation agrees with the data presented in Fig. 4, where Vuze has more neighbors at each given point in time and most of those peers have a connection duration of less than five minutes. That is, Vuze opens more short-lived ephemeral connections than uTorrent.

3) Connection termination policy: A very important local policy is when to terminate a connection with a remote peer. Our measurements show that the connection termination is related to when the local peer received the most recent payload packet from the remote peer. To show this, we do the following. For each connection closed by the local peer, we measure the duration passed since the last packet with payload (not BT control messages) was transfered from the remote to the local peer. We refer to this time interval as *inactivity period*. In our data, the vast majority of control messages are transferred using packets of size smaller than 600 bytes. Therefore, we mark a packet as containing payload if its size is larger than 1,000 bytes. The CDFs of the *inactivity period* right before a connection termination are plotted in Fig. 6. The shapes of the CDFs clearly show that the two clients have different termination policies. The uTorrent client is more "tolerant" and almost never closes a connection unless it stays open for at least five minutes. Only 20% of the connections are terminated before this deadline. On the other hand, Vuze is less tolerant and almost 80% of the connections are closed even before the peer stays inactive for five minutes. Perhaps this can explain the higher number of connections in Vuze with life-time less than five minutes (see Fig. 4).

4) Upload bandwidth distribution: Another difference is how the two clients distribute their upload capacity. To measure this, we slice the trace into small time-slots of five seconds and record to how many remote peers each client is uploading payload packets in each time-slot. The CDF over the entire trace is presented in Fig. 7. The uTorrent client seems to be more willing to send data to more remote peers. On the other hand, Vuze stays focused to a small subset of its peers and uploads only to them. The CDF shows that in more than 90% of the time-slots uTorrent uploads to more than four

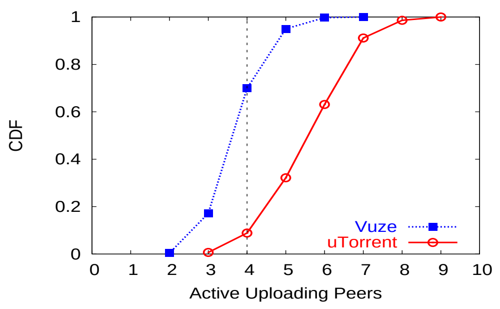


Fig. 7. Comparing the number of active uploading connections.

peers. This number is close to 30% for Vuze. Moreover, the mode of the distribution for Vuze is four while for uTorrent is six.

IV. Related Work

To the best of our knowledge, our study presents the largest in scale measurement study on the download speed of different real-world BT client implementations. Other works explored changes to BT aiming at improving its performance [8], [6]. The authors in [8] introduced a modified BT client, named BitTyrant. They show that BitTyrant can improve the performance of existing BT implementations by making better decisions in the upload bandwidth allocation of the peers. The BT modifications presented in BitTyrant [8] and ProbShare [6], can be used to explain the mismatch between uTorrent and Vuze and this is an interesting direction for future work.

In a recent work by Choffnes et al. [1], the speed of multiple Vuze clients is used as a metric for evaluating performance while reducing cross-ISP traffic. To achieve this, they required multiple users to install a plugin to the Vuze client. Our approach is non intrusive and does not require any modification to existing clients. Moreover, in contrast to [1] where only Vuze is used, our approach is client independent.

The popularity of BT motivated many researchers to study the performance characteristics of BT-like protocols [5], [7], [9], [4], [3]. In [5] the basic mechanisms of BitTorrent were evaluated using instrumented BT clients. Various characteristics of live BT users was the topic in [9] using active measurements similar to our approach. In their study [9], they focused on performance characteristics of various torrents (e.g., the existence of flash-crowds) and measured the download speed of over 50k peers but they did not compare the download speed between different BT implementations. A different approach for evaluating P2P systems using a testbed environment is presented in P2PLab [7]. By using BitTorrent, they show how P2PLab can help in evaluating various protocol algorithms and parameters. Finally, a framework for passively monitoring various BT client implementations is presented in [3]. In contrast to our approach, the method in [3] requires modifications to the source code of a BT client, which makes it impractical for monitoring closed-source clients, such as Vuze. We believe that all research efforts presented here can help in the evaluation of design choices of real BT implementations and this is included in our future work.

V. Conclusions and Future Work

Using the Apollo tool [11], we have conducted a large-scale measurement study involving more than 11 million BT clients over six thousands ISPs and observed that uTorrent is on average 16% faster than Vuze. To shed light to the root of these differences, we have reversed engineered the two clients using passive trace measurements and identified differences in: (a) how they manage their neighborhood size, (b) how they discover new peers (open new connection), (c) when they decide to close a connection, and (d) how they distribute their upload capacity. We speculate that the differences in download speed can be attributed to all or some of the above differences.

There are many directions for future work that we plan to pursue. In order to evaluate the impact of each implementation difference to the performance of each client, we want to deploy real BT clients in both a control testbed environment as well over a planetwide testbed such as PlanetLab. Moreover, we plan to include additional BT clients (e.g., Transmission and Mainline) to our study.

This work does not aim to design the best BT client, but to bring to the attention of BT implementors and users that some design choices have a significant effect in practice. Even though in our study uTorrent appears to achieve faster speeds than Vuze, we do not claim that uTorrent is the way to go. We hope that our preliminary findings will open the door for new research efforts to better understand the impact of design choices in the performance of real-world BT implementations. Ultimately, we see such research efforts leading to the design of better P2P systems.

References

[1] D. Choffnes and F. Bustamante. Taming the Torrent: a Practical Approach to Reducing Cross-ISP Traffic in Peer-to-Peer Systems. In *ACM SIGCOMM*, 2008.

[2] B. Cohen. Incentives Build Robustness in BitTorrent. In *1st Workshop on Economics of Peer-to-Peer Systems*, 2003.

[3] R. Deaconescu, R. Rughinis, and N. Tapus. A BitTorrent Performance Evaluation Framework. In IEEE *ICNS*, 2009.

[4] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. A Performance Study of BitTorrent-Like Peer-to-Peer Systems. *IEEE Journal on Selected Areas in Communications*, 25(1):155–169, 2007.

[5] A. Legout, G. Urvoy-Keller, and P. Michiardi. Rarest First and Choke Algorithms are Enough. In ACM *IMC*, 2006.

[6] D. Levin,K. LaCurts, N. Spring, and B. Bhattacharjee. BitTorrent is an Auction: Analyzing and Improving BitTorrent's Incentives. In ACM *SIGCOMM*, 2008.

[7] L. Nussbaum and O. Richard. Lightweight Emulation to Study P2P Systems. *Concurrency and Computation: Practice and Experience*, 20(6):735–749, 2008.

[8] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do Incentives Build Robustness in BitTorrent? In *NSDI*, 2007.

[9] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips. The BitTorrent P2P File-Sharing System: Measurements and Analysis. In USENIX *IPTPS*, 2005.

[10] G. Siganos, J. Pujol, and P. Rodriguez. Monitoring the Bittorrent Monitors: A Bird's Eye View. In *PAM*, 2009.

[11] G. Siganos, M. Iliofotou, X. Yang, J. Pujol, and P. Rodriguez. Apollo: Tapping into the Bittorrent Ecosystem. *Under Submission*, 2010. http://research.tid.es/georgos/images/apollo_submitted.pdf

[12] http://www.bittorrent.org/beps/bep_0029.html. uTorrent Transport Protocol.

[13] http://www.transmissionbt.com/. The Transmission BT Client.

[14] http://www.utorrent.com/. The uTorrent BT Client.

[15] http://www.vuze.com/. The Vuze BitTorrent Client.