# Availability in BitTorrent Systems

Giovanni Neglia[†], Giuseppe Reina[†], Honggang Zhang[*]
Don Towsley[*], Arun Venkataramani[*], John Danaher[*]

[†]D.I.E.E.T.
Università degli Studi di Palermo, Italy
giovanni.neglia@ieee.org, g.reina@gmail.com

[*]Computer Science Dept.
University of Massachusetts Amherst
{honggang, towsley, arun}@cs.umass.edu, jpdanaher@comcast.net

UMass Technical Report 06-41
First version: August 1st 2006
This version: August 26th 2006

*Abstract*— **In this paper, we investigate the problem of highly available, massive-scale file distribution in the Internet. To this end, we conduct a large-scale measurement study of BitTorrent, a popular class of systems that uses swarms of actively downloading peers to assist each other in file distribution. The first generation of BitTorrent systems used a central *tracker* to enable coordination between peers, resulting in low availabiilty due to the tracker's single point of failure.**

**Our study analyzes the prevalence and impact of two recent trends to improve BitTorrent availability: (*i*) use of multiple trackers, and (*ii*) use of distributed hash tables (DHTs), both of which also help to balance load better. The study measured over 22,000 popular torrents in four different continents spanning 1,700 trackers and 25,000 DHT nodes over a period of several months. We find that both trends improve availability, but for different and somewhat unexpected reasons. Our findings include: (*i*) multiple trackers improve availability, but the improvement largely comes from the choice of a single highly available tracker, (*ii*) such improvement is reduced by the presence of correlated failures, (*iii*) multiple trackers can significantly reduce the overlay connectivity, (*iv*) the DHT improves information availability, but induces a higher response latency to peer queries.**

## I. INTRODUCTION

Peer-to-peer file distribution is rapidly displacing traditional client-server distribution in the Internet. By some estimates [1], BitTorrent, a popular class of peer-to-peer file distribution systems, constituted about 30% of Internet backbone traffic in June 2004. BitTorrent uses active peers to assist each other in file distribution eliminating a single point of congestion, the server. Thus, the capacity of BitTorrent systems increases with the number of active peers enabling highly scalable file distribution.

Although BitTorrent eliminates a single point of congestion as regards data traffic, it continues to have a single point of failure. The first generation of BitTorrent systems employed a centralized *tracker* to enable coordination between peers. The tracker maintains the set of active peers, also called the *swarm*, interested in a specific file. A peer joins the swarm by *announcing* itself to the tracker, which returns a small random subset of peers from the swarm. Peers use this subset to connect to other peers to obtain missing chunks of the file. If the tracker fails or is unreachable, the system becomes unavailable to new peers, so they can not obtain the file or contribute resources to the system.

Measurement studies [2] confirm low tracker availability experienced by users of BitTorrent systems today. The massive prevalence of BitTorrent and recent proposals to adapt BitTorrent's techniques for more general forms of packet delivery [3] including email attachments, software updates, and security patches make BitTorrent availability an important problem. For example, unavailability of security updates distributed using BitTorrent can seriously impact the well-being of the Internet.

Two recent trends have emerged to tackle the problem of tracker availability. First, is the support for multiple trackers to increase the likelihood of at least one available tracker for a given file (introduced at the end of 2003). Second, is the integration of distributed hash tables (DHTs) with BitTorrent clients that store information across the entire community of BitTorrent users (introduced in May 2005). Section V and VI describe in detail how these two mechanisms work in practice today.

Our study investigates availability of BitTorrent systems in the light of these trends. Availability depends on several factors such as the multi-tracker or the DHT infrastructure (simply DHT in what follows), the amount of information they store, patterns of tracker and network failures, and the amount of information shared across trackers and peers. We quantitatively analyze the improvement in availability due to the two mechanisms.

Our study considered 26,000 torrents using more than 1,700 trackers and 25,000 DHT nodes over a period of several months. We find that multiple trackers as well as DHT-based trackers improve availability, but for different and somewhat unexpected reasons. Our major findings are as follows.

- Multiple trackers improve availability, but the improvement largely comes from a single highly available tracker.
- The potential improvement from Multi-tracker is reduced due to the presence of correlated failures.
- The use of multiple tracker can significantly reduce the connectivity of BitTorrent overlay.
- DHT improves information availability, but induce a higher response latency.

- Tracker and DHT show complementary characteristics features. Current trend of combining multiple trackers and DHT can provide high information availability with low information response latency.

The rest of this paper is organized as follows. In Section II we illustrate related works. After the description of the measurements sets in Section III, we show results about the trackers availability in Section IV. The improvement deriving from the use of multiple trackers and of the DHT infrastructure are respectively described in Sections V and VI.

## II. Related Works

There are now many measurement studies about BitTorrent traffic and operation. However they mainly focus on issues different from peer information availability: amount and characteristics of P2P traffic in the network [4], swarm evolution dynamics depending for example on peer arrival pattern and average connection time [5], [6], [7], [8], global downloading performance achievable by the peers [5], [8], the BT-specific content sharing algorithms like the choke algorithm or the content pieces selection algorithm [9] in particular as regards their effectiveness in promoting cooperative user behavior [10], [11].

The work most similar to ours is [2]. The authors focus on suprnova.org, which at the time of the study was the most popular website advertising BT contents (it was closed in December 2004 as a consequence of legal actions). sprnova.org was not just a website, but a complete architecture including a mirroring system to balance user requests across multiple websites, servers to store the torrent files, and human moderators to eliminate faked contents. The measurements span from June 2003 to March 2004, and the authors investigate availability of the architecture and also of the peers of a specific content. Tracker availability appears a significant problem: only half of the 1941 trackers they consider have an average uptime of 1.5 days or more. At the same time trackers appear to be more available than HTML mirrors and torrent servers in suprnova.org architecture. Our results suggest that there is a significant non-stationary effect affecting this kind of measurements. Our study also addresses new features that were not present (DHT), or not widespread (multi-tracker), during the measurement campaign described in [2].

Separate from the specific BT framework, there are some works about availability of distributed systems in the Internet [12], [13], [14], [15], [16]. In [12] the authors investigate peers availability through a measurement campaign of the Overnet file-sharing network [17]. They stress “aliasing errors” when IP addresses are considered as identifiers for the peers and show that availability of each peer significantly depends on the measurement time interval (because peers join and leave the system) and on time-of-day but is roughly independent from the availability of other peers. Even if trackers should be stable entities in the BitTorrent architecture we observed lifetime effects in our availability measurements. In [13] three different large distributed systems (PlanetLab, Domain Name Systems and a collection of over 100 web servers) are considered. The study identifies differences among temporal availability, Mean Time To Failure (MTTF), Mean Time To Repair (MTTR), Time To Failure (TTF) and Time To Repair (TTR). TTF is the expected time to failure, given that the system has already been in the working state for a specific time $T$ . They show that good availability does not necessarily imply good MTTF and MTTR and while MTTF and MTTR can be predicted with reasonable accuracy, TTF and TTR are much more difficult to predict. Besides these systems seem to exhibit large-scale correlated failures (differently from [12]). Our study confirms the presence of correlated failures among different trackers. [14] points out some limitations on using average temporal availability evaluated on long time periods and across many peers. In particular they show that temporal affinity (i.e. similar temporal pattern of peer presence in the system, due for example to day-of-time effects) and difference in the availability distribution for different peers can increase system global availability. They introduce a new metric to characterize system performance considering the number of peers in the system at a given instant and evaluate it through two traces from Kazaa and Overnet networks. Although a similar analysis could also be interesting in our case, it is out of the scope of this paper. [15] is a measurement study of Napster and Gnutella networks, trying to quantify content popularity and peers presence in the system. They also show a significant dependence of peer availability on the time of the day. [16] looks at the availability of Kazaa peers mainly to investigate potential benefits for file-sharing coming from locality-awareness.

## III. The Data Sets

To share a file or group of files through BitTorrent, clients first create a *torrent* file (or simply a torrent). A torrent contains meta information about the files to be shared in the *info* section and about the tracker which coordinates the file distribution in the *announce* section. The content is identified by the *info-hash* value, obtained by applying a hash function to the info section of the torrent. In order to support multiple trackers and DHT two new optional sections have been added: the *announce-list* and the *nodes* ones.

In our study we considered more than 22,000 torrents, found mainly through `www.torrentspy.com`, one of the largest BT search engine, and through `www.btjunkie.org`. We developed a script, which automatically downloads the RSS feeds of these sites and then downloads every new torrent file indicated in the feeds. In what follows we are going in particular to refer to the following sets.

| | |
|---|---|
| TS1 | : set of 7815 torrents advertised by `www.torrentspy.com` in March 2006. |
| TS2 | : set of 4238 torrents advertised by `www.torrentspy.com` from May 15 to May 19, 2006. |
| TS3 | : set of 15275 torrents advertised by `www.torrentspy.com` from May 20 to June 30, 2006. |
| BTJ2 | : set of 1017 torrents advertised by `www.btjunkie.com` from May 11 to May 19, 2006. |

| set | torrents# | well-formed torrents # | Trackers# total | Trackers# HTTP | Trackers# UDP | Mainline DHT nodes | Azureus DHT nodes |
|---|---|---|---|---|---|---|---|
| TS1 | 7815 | 7815 | 654 | 621 | 33 | 10191 | 77 |
| TS2 | 4238 | 4238 | 525 | 491 | 34 | 4546 | 21 |
| TS3 | 15275 | 15275 | 1202 | 1146 | 56 | 18663 | 196 |
| BTJ2 | 1017 | 1017 | 349 | 329 | 20 | 1491 | 0 |
| BTJ3 | 2312 | 2312 | 532 | 494 | 38 | 3790 | 0 |

TABLE I
TORRENT SETS

BTJ3 : set of 2312 torrents advertised by `www.btjunkie.com` from May 20 to July 01, 2006.

All these torrents specify more than 1,700 trackers and more than 25,000 DHT nodes. Table I summarizes the main information about trackers and nodes we can achieve from the different sets. Note that in this paper we refer to Bram Cohen's BitTorrent client [18] as "Mainline" client[1].

Figure 1 shows the distribution of the torrents across the different trackers for sets TS3 and BTJ3. The 20 most popular trackers manage more than 50% of all the torrents and the 10% most popular ones (about 120) manage more than 73% of them. Similar results hold also for the other sets. We can also estimate the popularity of each tracker directly by querying it with a *scrape* request. This is also a HTTP GET in BT protocol, whose purpose is to collect the statistics about the torrents managed by the tracker, like number of leechers, of seeders, etc.. If the scrape URL includes the infohash of a torrent, this restricts the tracker's report to that particular torrent. Otherwise statistics for all the torrents the tracker is managing are returned. In this way we can discover the total number of torrents each tracker manages. The drawback of this method is that not all the trackers support scrape queries, because of the load they produce, and that the torrents could be very old. Figure 2 shows the distribution of the torrents across the different HTTP trackers in set TS2, evaluated from our torrent set (as in Figure 1) and from scrape queries. Only 371 trackers out of 491 in TS2 were alive when we made the queries and only 124 trackers answered correctly to scrape queries. According to the ranking based on our torrent set, the 10% most popular trackers (about 50) manage about 80% of the torrents, while according to the ranking based on scrape queries, the 10% most popular ones (12 trackers) manage about 50% of the torrents. As regards the overlap between the two rankings, 12 of the 20 most popular trackers in the ranking based on our torrent set support scrape queries and only 6 of them appears among the most popular trackers in the other ranking.

## IV. TRACKER RELIABILITY

In this section we first consider the availability of tracker itself, without considering the specific contents they manage.

There are two different kind of trackers: those using HTTP protocol for the communication with the client and those using UDP protocol. The second possibility has been introduced in

[1]Bram Cohen is the creator of BitTorrent protocol.

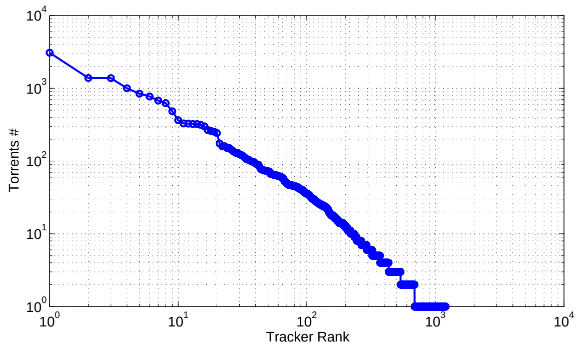

Fig. 1. Popularity of the Trackers in TS3&BTJ3

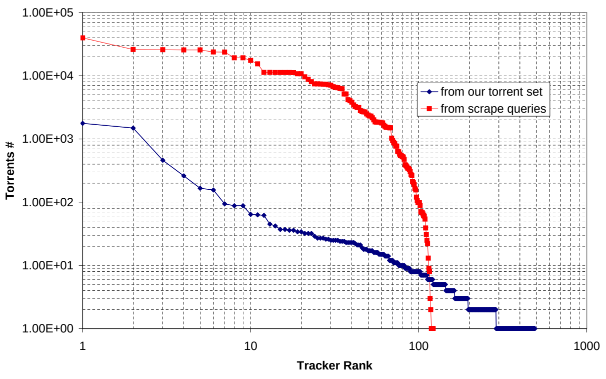

Fig. 2. Popularity of the Trackers in TS2

order to reduce the load on trackers [19]. As Table I shows, HTTP trackers are much more common. Also we noted that most of the UDP trackers are associated to a HTTP tracker (they have the same IP address).

The availability has been evaluated by probing periodically the trackers (usually every 15 minutes). If the probe shows that the tracker is working then the tracker is considered available until the following probe. Unless otherwise specified we consider the availability of the tracker as the fraction of the time it was available over the total measurement time.

The way to probe the tracker in order to check if it is working differs according to whether a tracker uses HTTP or UDP. The availability of UDP trackers has been evaluated by trying to establish an *UDP handshake* as described in the UDP tracker protocol specification [19]. The tracker is considered unavailable if three consecutive attempts fail. HTTP tracker availability has been evaluated trying to open a TCP connection to the address specified in the announce key. The tracker is considered not available if three consecutive attempts to open the connection fail. This procedure can produce wrong results. For example some trackers are implemented as modules of Apache web-servers [20]: BitTorrent requests[2] are identified from the specific URL and forwarded to the tracker module. Our measurements suggest that this is quite common. For example out of 491 HTTP URLs extracted from TS2 set, at the begin of June 2006, 364 servers were answering to generic GETs, and 209 of them were declaring themselves as Web servers (181 Apache, 15 Microsoft IIS, 10 Lighttpd, 2 LiteSpeed, 1 nginx), most of them (142) were listening on port 80 (there are 183 URLs with port 80 in the data set). In such cases we would erroneously conclude that the tracker is available if the tracker module is down, but the web-server is working and accept incoming TCP connection. The problem is not easy to solve. We thought to reveal the correct operation of a tracker from answers to Announce[3]: the tracker is operating if the answers are bencoded dictionaries, otherwise only the web-server is working; but unfortunately, web-servers often provide HTTP answers to Announce for not-existing torrents (including torrents previously managed by the tracker), even when the tracker module is working correctly. In order to overcome this impasse, we decided to rely on the heuristic we are going to describe. First we observe that it is very unlikely, at least for Apache servers, that a module is not working while the web-server is running. Hence we assumed that the tracker module is working whenever the web-server is available unless the module has been uninstalled. We assume that the tracker module has been uninstalled if all the following conditions are satisfied: 1) the web server is available, 2) Announce requests for different torrents which should be managed to the tracker[4] do not receive a BitTorrent answer, 3) ScrapeAll requests do not receive BitTorrent answers. In such a way we identified 16 web-servers where the BitTorrent module has been probably uninstalled, interestingly TCP port 80 is common to all these URLs except one, supporting the idea that these web-servers not have been deployed specifically for BitTorrent operation.

We performed tracker availability measurements for many months, probing the trackers every 15 minutes. We observed that for some trackers the availability depends on the length of the measurement time interval (a similar effect was observed in [12] for the peers of the Overnet network) and in particular decreases as the measurement time interval increases. Our hypothesis is that probably these trackers *died*, i.e., they stop operating definitely. Figure 3 quantifies this non-stationary effect. It shows the evolution of the number of live trackers during a two months period. We assume that a tracker dies

[2]BitTorrent Announce or Scrape messages are simple HTTP GETs.

[3]Some trackers do not support Scrape requests, while all of them have to support Announce ones.

[4]We consider up to 3 different torrents taken from torrentspy_new data set declaring the specific tracker.

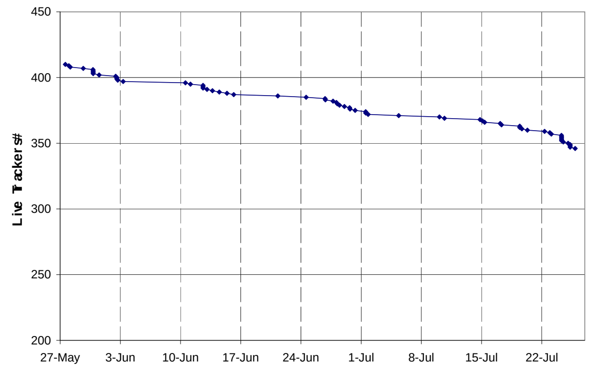

Fig. 3. Number of live Trackers

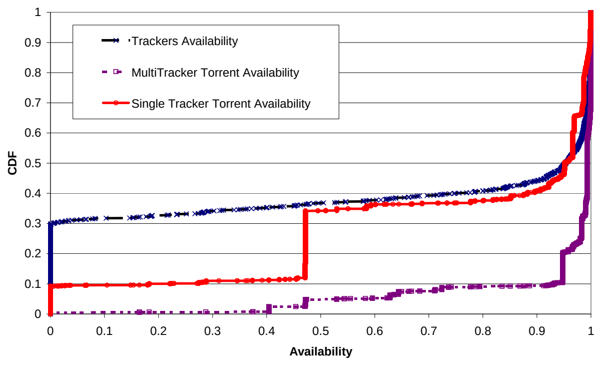

Fig. 4. Trackers and Torrents Availability

when it starts being unavailable until the end of the measurement period for at least two days. It appears that the number of live trackers decreases from 416 to 354 (about 15%) over 58 days, from the May 27 to July 24). From the data we can roughly estimate that the average tracker death rate is about $2.6 * 10^{-3}$ per day $(= 62/(416 * 58))$, hence the tracker lifetime is about 428 days.

Figure 4 shows the Cumulative Distribution Function (CDF) of the availability of TS3 and BTJ3 trackers over a 21 days period (dashed curve labelled “Trackers Availability”) starting from July 21. The curve is similar for different periods and different sets, only the number of unavailable trackers changes quite significantly depending on the measurement period (from 20% to 30%). We are more interested to characterize the availability of information for the peers in a given swarm. We briefly refer to this concept as *torrent availability*. For single tracker torrents, torrent availability coincides with the availability of the tracker specified in the torrent (see Section V for multi-tracker case). If trackers were equally represented across the torrents, torrent availability would reflect tracker availability, but we have shown in Figures 1 and 2 that tracker popularity is skewed. This effect is clearly shown by the CDF of torrent availability in Figure 4 (solid curve labelled

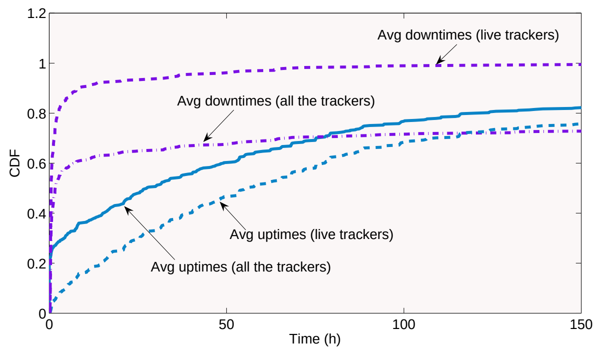

Fig. 5. CDF of average up-time and down-time over two months

"Single Tracker Torrent Availability")[5]. We note a 25% jump in the CDF, it corresponds to `www.thepiratebay.org` tracker (`tracker.prq.to`), the most popular tracker in Figure 1. The availability of this tracker changed a lot during our measurement campaign, from 0.5% during May 26-June 9 to 47% for the period which the figure refers to. If we filter out this tracker, the availability at the swarm level appears to be higher than the availability of the trackers, mainly because many of the always unavailable trackers (corresponding to the 30% initial jump in the blue curve) are not used for single-tracker torrents, but are always coupled with other trackers in multi-tracker torrents. Finally the third curve in Figure 4 refers to multi-tracker torrents, which we are going to address in the following section.

In order to investigate if there is a relation between tracker availability and the number of torrents the tracker is managing, we performed a linear regression on the data with the availability as the response variable and the number of torrents (derived from TS2 set) as explanatory variable. The estimate of the slope is $\hat{\beta} = -2.9{*}10^{-4}$ (i.e. there would be a reduction of about 3% in the availability every 100 torrents) with a 99% confidence interval equal to $[-5.4 * 10^{-4}, 0.3 * 10^{-4}]$. The correlation coefficient is quite small: 0.0216. We performed also the linear regression considering the number of torrents each tracker declares as answer to a scrape request (see Section III). The analysis does not suggest a dependence, but only one third of the trackers could be considered in this analysis, the others do not provide this kind of information.

Finally Figure 5 shows the Cumulative Time Distribution (CDF) of the average uptime and downtime evaluated for all the trackers in TS2 and BTJ2 and considering only the trackers alive at the end of the measurement period. If we consider all the trackers then almost half of the trackers appear to have an average uptime smaller than 1.5 days as observed in [2][6], but if we restrict to live trackers the average availability increases significantly and about 70% of the trackers show an average uptime longer than 1.5 days. As regards the distribution of the downtime itself, 25% of the downtimes last more than half an hour, 20% more than 1 hour and 10% more than 2 hours. This suggests that tracker unavailability is due often to software or machine crash rather than to temporary network problems.

## V. Multi-Tracker Feature

Multi-Tracker feature allows two or more trackers to take care of the same content [21]. In addition to the mandatory announce section in the torrent file, which specifies the tracker URL, a new section, announce-list has been introduced. It contains a list of lists of tracker URLs. Trackers in the same list have load-balancing purpose: a peer randomly chooses one of them and announces to it. All the trackers in the same list exchange information about the peers they know. The different lists of trackers are intended for backup purpose: a peer tries to announce to a tracker in the first list, if all the announce requests to tracker in the first list fail, it tries to contact a tracker in the second list and so on[7]. On the next announce, it repeats the procedure in the same order. Trackers in different lists do not share information. There are two common way to use multi-tracker feature: only for backup purpose when the announce-list contains lists with a single tracker, and only for load balancing purpose when the announce-list contains a single list with many trackers. In our sets about 35% of the torrents specify multiple trackers: 60% for backup, 25% for load balancing and 15% for both backup and load balancing.

Multi-Tracker feature is clearly intended to improve the availability of the information about the peers in the swarm. In what follows we are going to quantify this improvement.

### A. *Correlation among different trackers*

In order to quantify the benefit of multi-tracker we first need to check if availabilities of different trackers can be considered independent. From our measurements it appears that trackers availabilities are more correlated than one could expect.

This result is similar to the conclusion in [12] for Planetlab machines and webservers, and opposite from the remarks in [13] for Overnet peers. In [12] the authors simply show that the number of near-simultaneous failures does not seem to follow a geometric distribution[8], nor a beta-binomial distribution which should be more suited to account for correlated failures. In [13] the authors consider for all the host pairs (A,B) the difference between the a priori probability that host A is available and the same probability given that host B is available. They observe that the difference is between 0.2 and -0.2 for 80% of all the host pairs and conclude that there is significant independence, even if there is a significant diurnal pattern in single host availability.

Our analysis is based on 4 weeks availability measurements for live trackers (trackers which where not completely unavailable during the measurement period) in TS2 and is more accurate from the statistical point of view. For all the tracker pairs[9] we considered the contingency table and performed a

[5]The CDF of torrent availability *weights* the availability of each tracker in the set with its number of presences in the torrents.

[6]The authors do not address the issue of dead trackers.

[7] Some clients announce to one tracker in each list.

[8]A limitation of their analysis is that they assume a unique failure probability for all the machines.

[9]We consider a tracker identified by IP address, protocol and port number

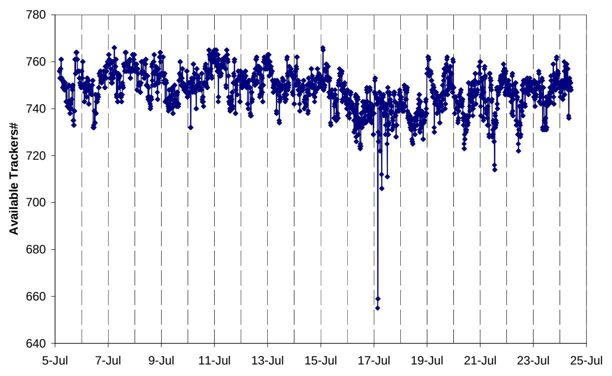

Fig. 6. Number of Available Trackers Time Plot

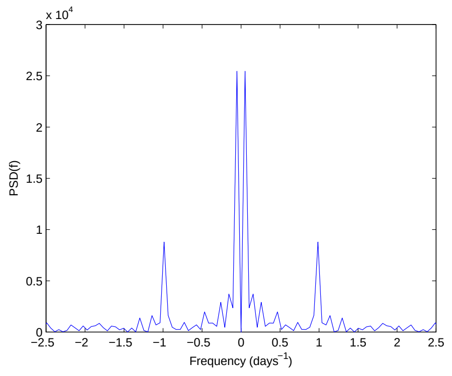

Fig. 7. Power Spectral Density of the Number of Available Trackers

G-test [22]. We tested the null hypothesis that availabilities of different trackers are independent with a Type I risk equal to 5% and 1%. In order to use the G-test we had to discard 65% of the pairs. The test supported statistical dependence for 40% of the pairs and 30% of the pairs respectively with the 5% and 1% Type I risks. The G-test is an approximation of the exact Fisher test, which is a combinatorial test, and hence is unfeasible for the number of samples we are considering. Nevertheless in order to check the G-test results we empirically evaluated with Montecarlo method the statistical distributions needed to perform the Fisher test. This approximate Fisher test overcomes some limitation of the G-test allowing us to consider a larger set of pairs (86%). The results of the G-test are confirmed also on this larger set

One simple cause of correlation is that trackers could be hosted in the same machine. Among the 406 trackers considered, there where 26 groups collecting 73 trackers having the same IP. For all these pairs (except two) the G-test refused the independence assumption, but they represent less than 0.2% of the total number of pairs considered, hence this justifies only a minimum part of the correlation individuated by the tests.

We think that this correlation is due to the fact that many trackers show a strong daily pattern in their up-times and down-times. This can be due to user behavior or a consequence of tracker failures that can be recovered only when the user is present. This thesis is partially supported by data in Section IV showing that in many cases tracker unavailability is due to software or machine failures rather than network unreachability. Figure 6 shows the total number of available TS3 trackers for three weeks in July 2006 with a 10 minutes resolution. The daily pattern is confirmed by Figure 7, where the spectral density, evaluated with the unmodified periodogram method (see chapter 4 in [23] also for a discussion about the goodness of this method in order to estimate spectral lines), exhibits a peak corresponding to a 1-day periodicity[10].

## B. *Availability Improvement*

The presence of multiple trackers in the torrent clearly increases peers information availability for the swarm because it is sufficient that at least one of the trackers is available. If failures at different trackers were independent we could simply evaluate the unavailability of a group of trackers as the product of the unavailabilities of each tracker. This assumption is not corroborated by the data in the previous section, so we have to consider for each tracker its availability temporal sequence and then check if at a given time instant there is at least a tracker available. We call this method to evaluate the availability of a group of tracker “time-aware”.

The CDF of the time-aware availability for multi-tracker torrents is plotted in Figure 4 (dotted curve). This picture shows a significant improvement coming from multi-tracker. We note that this improvement does not derive from the combination of many trackers with low availability, but mainly from the presence of a highly available one in the set of trackers. This claim is supported by Figure 8. The figure shows the availability improvement using all the trackers, in comparison to the availability of the best tracker. The availability has been evaluated both considering trackers availabilities independent (dashed curve) and considering the availability time sequences for all the trackers (solid curve). The figure suggests two main remarks. *First* if we consider the time-aware curve the gain in comparison to the most available tracker is quite small: below 0.6% in 83% of the cases and below 2% in 95% of the case. *Second* the availability correctly evaluated considering the temporal sequence is smaller than that evaluated under independence assumption. This was also expected because tracker availabilities mainly exhibit a positive correlation due to time-day effect: trackers tend to be available during the same time periods.

Figure 9 gives some more insight. The figure shows the gain distribution across all the tracker groups specified in the set[11]. The gain has been normalized to the maximum achievable improvement in comparison to the most available tracker. For example if the most available tracker has a 95% availability, and the presence of the other trackers raises the availability up to 97%, the normalized improvement is 0.4 $(= 2/(100 -$

[10]The other peak corresponds to the total measurements scale and it is mainly due to the average decrease of available trackers between July 16th and July 18th.

[11]Differently from Figure 8 two torrents which specify the same group of trackers are considered as one.

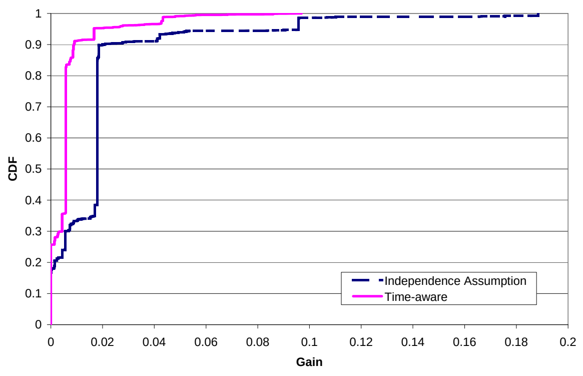

Fig. 8. Multitracker Gain Distribution

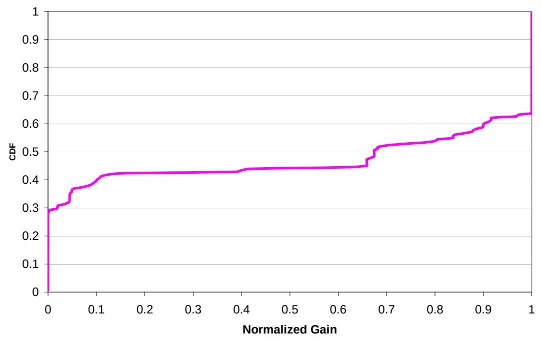

Fig. 9. Multitracker Normalized Gain Distribution for the different group of trackers

$95)$). The figure shows that two situations occur very often. For 30% of the groups (left part of the curve) there is no gain in comparison to the most available tracker, as it was already underlined by Figure 8. At the same time for 27% of the groups (right part of the curve) the presence of the other trackers raises the availability up to 100%, but we know from Figure 8 that the absolute value is small.

### C. Problems related to multitracker: swarm splitting

When the announce-list specifies a group of trackers for load balancing, all the trackers should know all the peers in the swarm. When the group of trackers is for backup, at a given time only one tracker should know all the peers in the swarm (but see also footnote 7). In reality things can be different due to peer arrival and departure, tracker failures, time intervals between consecutive tracker updates. Besides there are also some bad implementations of the torrent maker or of the client, some examples are reported in [21], so that the peer swarm could even be split into disjoint subsets. This would be clearly harmful for content spreading. In what follows we use the term "subswarm" to denote the subset of the swarm each tracker manages, i.e., all the peers it knows about.

In order to evaluate if the risk of disjoint subswarms is realistic, we considered all the 568 multi-tracker torrents in TS3. On July 14th for each torrent we made multiple Announce requests to each tracker in the announce-list in order

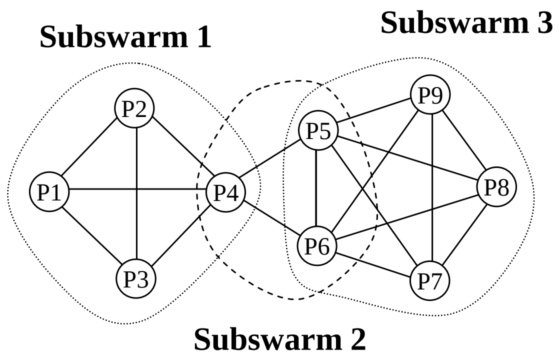

Fig. 10. Potential Neighbors Graph

to discover the subswarm it was managing, i.e. the (IP, port) pair of all the peers the tracker knew about. The whole process took about 5 hours and collected more than 22,000 peers. Once we had the subswarms, we built a graph as it follows: each node in the graph corresponds to a peer and a link between two nodes indicates that there is at least a subswarm that includes the corresponding peers. Note that if two peers (say P1 and P2) belong to the same subswarm then they could be neighbors in BitTorrent overlay, this occurs when the tracker managing the subswarm includes P1 (/P2) in the list of peers of the response to an announce of P2 (/P1). For this reason we call this graph as "potential neighbors graph". An example is shown in Figure 10: there are three partially overlapping subswarms with peers $4$, $5$ and $6$ included in more than one subswarm. Clearly if the graph has more than one component than the subswarms are disjoint. Only 17 torrents (about 3%) exhibited this problem: 16 had two components, 1 three. The peer communities were quite small ranging from the 3 to 24 peers. In such cases if a piece of content was available only at a single peer, it could be propagated only inside the subswarm the peer belongs to (as far as the graph does not change). The specific purpose of multi-tracker in these 17 torrents was backup for 9 torrents and load balancing for 7 torrents.

Even when the graph is completely connected, we can quantify subswarm overlap and then the possibility to spread the content across the community. In particular we considered two other performance metrics evaluated on graphs (beside the number of connected components). One performance metric is the connectivity degree: the number of links in the graph divided by the maximum number of links, i.e. the number of links of a fully meshed graph. For example the connectivity of the graph in Figure 10 is $0.5$, because there are $18$ links out of $36$ possible links in a $9$ nodes graph. This metric refers to the graph in its entirety. The other metric quantifies how much connected is the worst connected subswarm. We adapt the idea of graph conductance and we define the conductance of a non-empty subswarm $S$ ($g_S$) as the number of links connecting nodes of the subswarm ($N_S$) with nodes outside ($N_{S^c}$), normalized by the product $N_S N_{S^c}$, i.e. the maximum number of links. When $N_{S^c}$ is equal to $0$, we consider $g_S =$

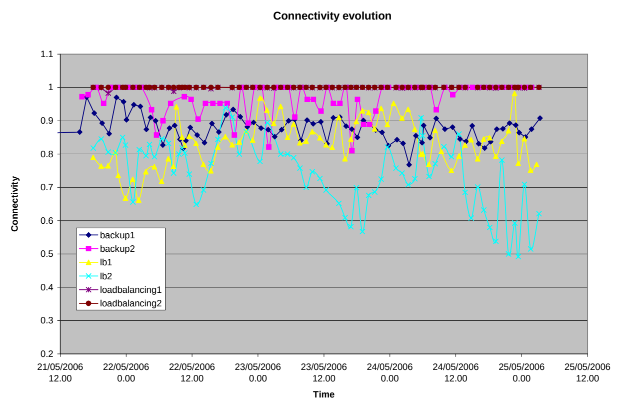

Fig. 11. Connectivity Time Plot

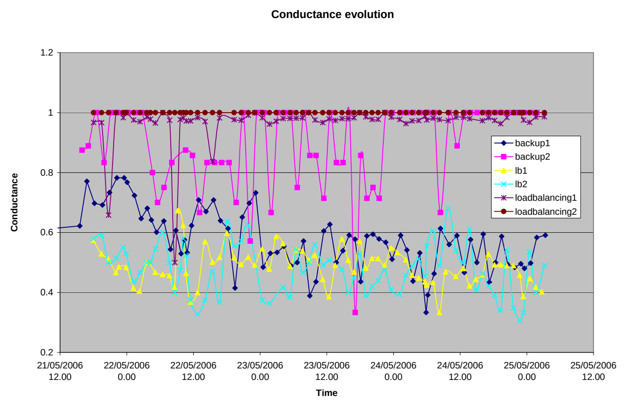

Fig. 12. Conductance Time Plot

$1$[12]. Then we define the conductance of the community as the minimum value of $g_S$ among all the subswarms. For example the conductances of the three subswarms in Figure 10 are $g_{S_1} = 2/(4*5)$, $g_{S_2} = 9/(3*6)$ and $g_{S_3} = 2/(5*4)$ and the community conductance is $0.1$.

For example in Figure 11 and Figure 12 we show the temporal evolution of these metrics during a 3 days period for 6 torrents: two of them declare multiple trackers for backup purpose, two for load balancing and two for backup and load balancing.

Figures 13 and 14 show respectively the CDFs for the connectivity and the conductance. In each figure there are 4 curves, one considers all the multi-tracker torrents, the others refer to backup torrents, load-balancing ones and torrents for both the purposes (LB&B in the legend). As it was expected the performance are very good for pure load balancing, in fact in this case the trackers periodically communicate each other their subswarms. Performance can be bad for backup, especially if we look at the conductance in Figure 14. It appears that 27% of the worst connected subswarms have a conductance smaller than 0.5, which indicates that on the average nodes in the subswarm can at most discover half of the nodes outside the subswarm. Figure 15 shows that connectivity and conductance are highly correlated: usually a graph with a low connectivity has also a low conductance, and vice versa.

[12]Note that $g_S$ is always less than or equal to one.

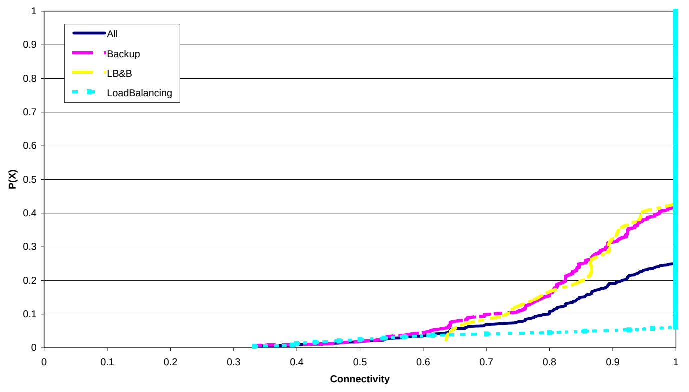

Fig. 13. Connectivity Cumulative Distribution Function

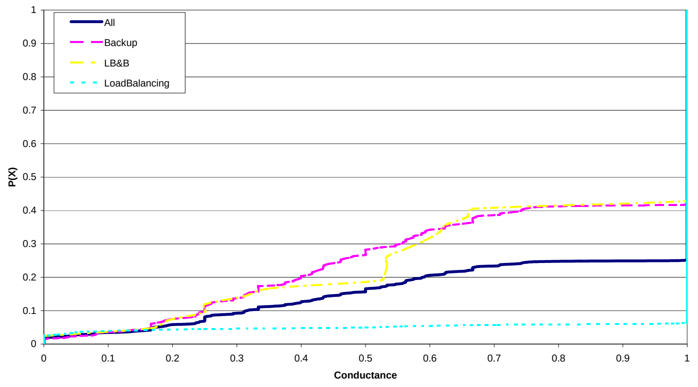

Fig. 14. Conductance Cumulative Distribution Function

## VI. Distributed Hash Tables

The latest versions of the most popular clients (Azureus, Mainline, BitComet, $\mu$Torrent, BitLord and BitSpirit) implement the functionalities of a DHT node, so that all the peers, independently from the content they are interested into (i.e. from the swarm they are in) can form a single DHT infrastructure. The purpose of the DHT is to store the information needed to contact peers interested into a specific content: according to the common DHT language the *key* is the infohash of the torrent, while the *value* is the contact information (e.g. the IP and the port of a peer client). Theoretically the DHT could completely replace the tracker, permitting the operation of *trackerless* torrents.

We said that all the BT clients could form a single DHT, in reality this is not true, because there are currently two different incompatible implementations (even if both are based on the Kademlia model [24]): the Mainline one, and the Azureus one. Except Azureus all the other clients are compliant with

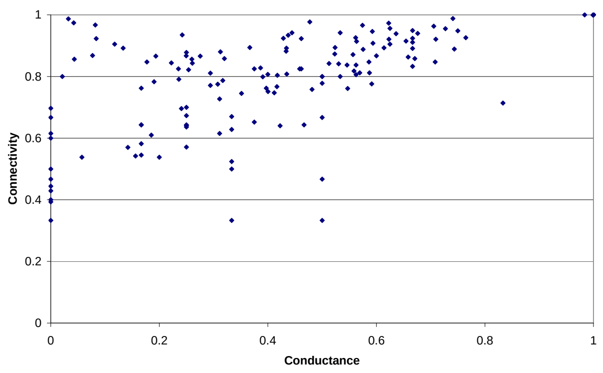

Fig. 15. Connectivity vs Conductance

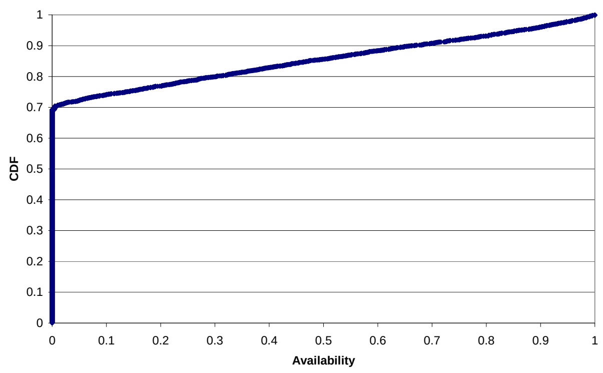

Fig. 16. Cumulative Distribution Function of DHT nodes availability

Mainline DHT specifications. Our measurement study focuses on the Mainline DHT.

### A. A Brief Overview of DHT Operation

When a new torrent is created, the program usually allows the user to insert some DHT nodes. The DHT nodes can be manually specified or are just randomly picked up from the set of "good" (highly available) DHT nodes from the routing table of the client[13] . These DHT nodes act as bootstrap nodes, in fact they are used to initialize the client routing table. The routing table is updated from time to time according to the protocol description in [25]. There are also other ways to discover DHT bootstrap nodes to initialize the routing table, for example if the peer is already in a swarm and is connected to another peer, they can exchange DHT related informations.

In order to download the content, the BitTorrent client can send requests to the set of DHT nodes in its routing table closest[14] to the infohash. The contacted nodes will reply with the contact information of peers interested into the content, if they know any, or with the contact information of the DHT nodes in their own routing table closest to the infohash. The timeout for a request is 20 seconds in a Mainline client.

Table I shows the number of DHT nodes we found in the torrents of our data sets. The higher number of Mainline nodes is mainly due to BitComet torrent-maker, which adds by default 10 nodes to each torrent.

### B. Information availability through the DHT

Similarly to what we did for trackers, we have been measuring the availability of DHT nodes. The DHT protocol [25] implements a specific request, called *DHT ping*, in order to check if a DHT node is available, so we resort to DHT pings. We considered a node unavailable when it did not answer to three consecutive DHT pings. Figure 16 shows the Cumulative Distribution Function of nodes availability evaluated during one week: 70% of the nodes were always unavailable, while the others show an availability nearly uniformly distributed between 0% and 100%. The availability of the bootstrap nodes clearly influence the speed of the query process.

In order to investigate the effectiveness of DHT networks, we customized a Mainline client and conducted experiments on a set of 2569 torrents, those of TS3 with DHT nodes[15].

The description of our experiment follows. For each torrent, we first erase the routing table and all the files that keep the information of previous content downloading. Namely, the client starts with a clean slate for each torrent. Then we let the client start contacting the DHT nodes in the torrent file and trying to recover information about the peers. In the mean time, all the nodes in the routing table are logged (recall that the routing table is updated frequently). The measurement stops when the client gets the first valid peer and the next torrent is considered. Our experiment started at 20:15 on July 22, 2006, and it took about 34 hours to finish.

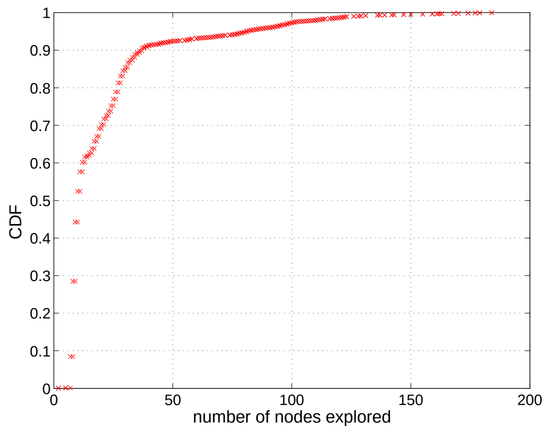

Fig. 17. The cumulative distribution of the number of DHT nodes ever explored before finding the first valid peer in a swarm.

[13]Note that each BitTorrent client is at the same time a peer and a DHT node.

[14]Kademlia DHT uses the XOR metric to compare keys and DHT nodes identifiers.

[15]9 torrents out of 2569 were incorrectly encoded torrents, as our client reported the following error: "bad data in responsefile - total too small". This happens when the product of number of pieces and the size of a piece is larger than the total size of the file. All these three numbers are reported in the .torrent file. Please refer to StorageWrapper.py in Bram Cohen's BitTorrent source code for details.

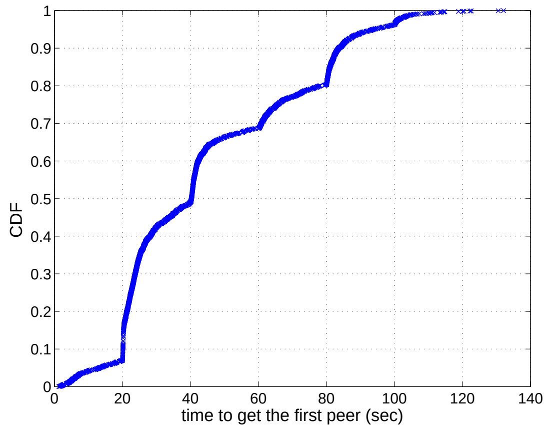

Fig. 18. The cumulative distribution of the time needed to find the first valid peer in a swarm.

Figures 17 and 18 respectively show the CDF of the number of DHT nodes ever explored and of the time elapsed before finding the first valid peer. We see that DHT is pretty effective because for about $93\%$ of the torrents a peer can be found by our client by exploring less than $50$ DHT nodes and in less than $88$ seconds. In the worst case the time needed was $140$ seconds and $184$ DHT nodes were explored. Figure 19 shows the scatter plot time needed versus number of DHT nodes explored for each torrent, it appears evident that there is a strong correlation between these quantities.

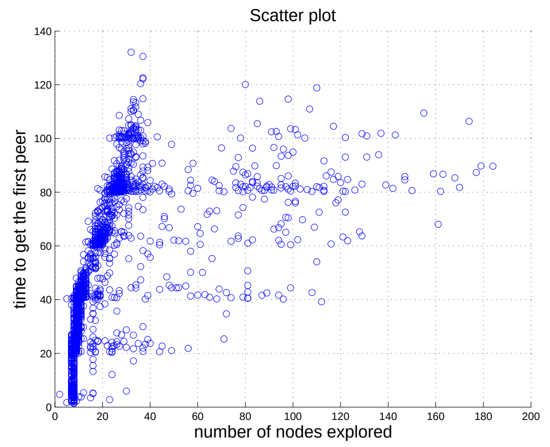

Fig. 19. Time needed to discover the first peer vs Number of DHT nodes explored.

For comparison, we also investigated the time needed to find the first valid peer by just contacting trackers in the same data set[16]. We put an upper limit of $300$ seconds for contacting a tracker. That is, our client stops announcing to the tracker after 300 seconds, even if the tracker does not answer. Our experiment started 21:33 on July 24, 2006, and finished at 22:54 on July 27, 2006. The CDF of the time needed to find a peer for both trackers and DHT is plotted in Figure 20. As expected, usually tracker can respond with valid peers faster than DHT, in less than one second. However, note that about $30\%$ trackers do not respond at all within $300$ seconds. On the contrary in these experiments our client was always able to get peers from the DHT in less than $140$ seconds. However, we need to be cautious because our tracker experiment was conducted one day later after we finished DHT experiments.

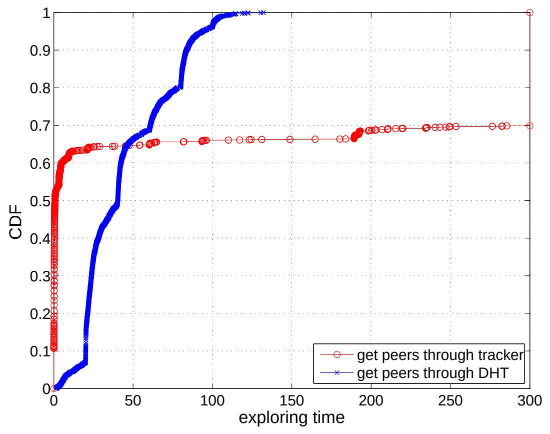

Fig. 20. Comparison between DHT and Tracker. The cumulative distribution of the time needed to find the first valid peer in a swarm.

Finally we compare the number of peers that can be obtained by the tracker (or the trackers) specified in the torrent and by the DHT (using the DHT nodes in the torrent as bootstrap nodes). It is difficult to define the framework for a fair comparison between DHT and trackers, we need to choose the time to collect the peers through the DHT, the number of queries to the tracker/trackers and the time between two consecutive queries (if more than one). We considered the number of peers harvested through the DHT in a 20 minutes time interval and the number of peers achieved through a single query to the trackers[17]. Figure 21 shows the results of our experiments for 117 torrents. The DHT was able to provide some peers in 16 out of 17 cases where trackers were unreachable. Nevertheless when trackers are available they usually provide more peers (only in 22 cases the DHT

[16]$275$ out of $2569$ torrents are not considered valid for the experiment In this case we considered only $2294$ valid torrents out of $2569$ torrents. $9$ torrents are incorrectly encoded, for other $266$ torrents our client reported the following error message: “Aborting the torrent as it was rejected by the tracker while not connected to any peers.” It is interesting to note that for these $266$ swarms which were not handled by trackers, our client could still find valid peers in the previous DHT experiment.

[17]Most of the trackers specify a minimum time interval between two announce queries equal to 30 minutes, 1 hour or 2 hours (even if they usually do not enforce it). Hence a client should not announce more than once in a 20 minutes interval if the tracker is available.

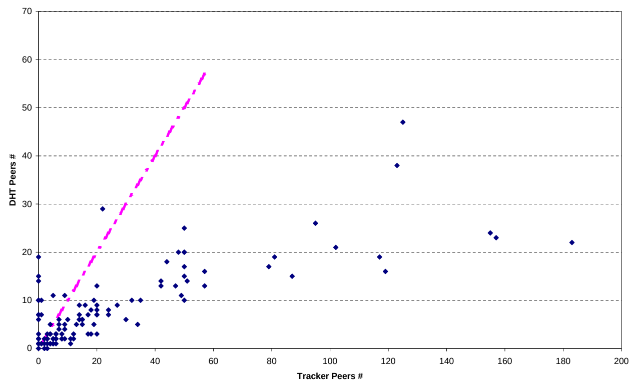

Fig. 21. Number of Peers obtained by the DHT in 20 minutes vs Number of Peers obtained by one query to the Trackers

outperformed an available tracker). From the figure it appears also that there is a strong correlation between the number of peers achievable in the two ways.

## VII. Conclusions and Future Research

From a traditional distributed systems perspective, BitTorrent is a complex system using three different forms of failure robustness: a primary-backup (the tracker) as well as a structured peer-to-peer overlay for the control plane (the Kademlia DHT infrastructure) and an unstructured peer-to-peer overlay for the data distribution plane. Our measurement study is a first step towards understanding the interaction of diverse fault-tolerance and scalability paradigms to provide a single massive-scale distributed service. In particular we have analyzed the prevalence and impact of the use of multiple trackers and DHT as regards the availability of information about the peers. The main conclusion of our study from the system design point of view is that trackers and DHT should be both considered in order to architect highly available BitTorrent systems.

A distinguishing feature of our study in comparison to previous works is the focus on the information availability rather than on the peers itself. At the same time one of its limitations is that we do not check the "quality" of the information we receive (e.g. if the peers provided by the trackers or by the DHT are updated) and the effect of lack of information or bad information on the spreading of the content (e.g. in the case of multiple trackers how low conductance slow down the file diffusion). We repute these issues meaningful and we deserve them for future research.

## VIII. Acknowledgements

The authors would like to thank Luca Scalia for his help. This research has been supported in part by Italian MIUR projects Famous and Mimosa and by NSF under grant awards ANI-0085848, CNS-0519998, CNS-0519922, and EIA-0080119. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

## References

[1] "Cachelogic," http://www.cachelogic.com.

[2] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The bittorrent p2p file-sharing system: Measurements and analysis," in *Proc. of 4th International Workshop on Peer-to-Peer Systems (IPTPS'05)*, February 2005.

[3] N. Tolia, M. Kaminsky, D. G. Andersen, and S. Patil, "An architecture for internet data transfer," in *Proc. of the 3rd Symposium on Networked Systems Design and Implementation (NSDI '06)*, 2006.

[4] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos, "Is p2p dying or just hiding?" in *Proc. of IEEE Globecom*, November 2004, dallas, TX, USA.

[5] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. A. Hamra, and L. Garcés-Erice, "Dissecting bittorrent: Five months in a torrent's lifetime," in *Proc. of the 5th Passive and Active Measurement Workshop*, April 2004.

[6] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurement, analysis, and modeling of bittorrent-like systems," in *Proc. of ACM SIGCOMM Internet Measurement Conference, (IMC'05), New Orleans, LA*, October 2005.

[7] D. Erman, D. Ilie, and A. Popescu, "Bittorrent session characteristics and models," in *Proc. of HET-NETs 05 - 3rd International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks*, 2005.

[8] J. R. Nicoll, M. Bateman, A. Ruddle, and C. Allison, "Challenges in measurement and analysis of the bittorrent content distribution model," in *Proc. of International Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting*, 2004, liverpool John Moores University.

[9] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Understanding bittorrent: An experimental perspective," EURECOM,INRIA, Tech. Rep., November 2005, http://hal.inria.fr/inria-00000156/en/.

[10] N. Andrade, M. Mowbray, A. Lima, G. Wagner, and M. Ripeanu, "Influences on cooperation in bittorrent communities," in *Proc. of 3rd Workshop on Economics of P2P Systems (P2P Econ)*, August 2005, philadelphia, PA, USA.

[11] N. Liogkas, R. Nelson, E. Kohler, and L. Zhang, "Exploiting bittorrent for fun (but not profit)," in *Proc. of 5th International Workshop on Peer-to-Peer Systems (IPTPS'06)*, 2006.

[12] R. Bhagwan, S. Savage, and G. M. Voleker, "Understanding availability," in *2nd International Workshop on Peer-to-Peer Systems*, February 2002, berkeley, CA, USA.

[13] P. Yalagandula, S. Nath, H. Hu, P. B. Gibbons, and S. Seshan, "Presence-based availability and p2p systems," in *First Workshop On Real Large Distributed Systems (WORLDS)*, 2004.

[14] R. J. Dunn, J. Zahorjan, S. D. Gribble, and H. M. Levy, "Presence-based availability and p2p systems," in *Proc. of the 5th IEEE International Conference on Peer-to-Peer Computing*, September 2005, konstanz, Germany.

[15] J. Chu, K. Labonte, and B. N. Levine, "Availability and popularity measurements of peer-to-peer systems," in *Proc. of ITCom: Scalability and Traffic Control in IP Networks*, July 2002, boston, MA, USA.

[16] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," in *Proc. of 19-th ACM Symposium on Operating Systems Principles*, October 2003, bolton Landing, NY, USA.

[17] "Overnet website," http://www.overnet.com.

[18] "Bram cohen's bittorrent client," http://www.bittorrent.com/index.html.

[19] "Udp tracker protocol specification," http://xbtt.sourceforge.net/udp_tracker_protocol.html.

[20] "mod_bt," http://www.crackerjack.net/mod_bt/.

[21] "Multitracker description," http://wiki.depthstrike.com/index.php/P2P:Protocol:Specifications:Multi%tracker.

[22] T. Dunning, *Accurate Methods for the Statistics of Surprise and Coincidence*, C. Linguistics, Ed., March 1993, vol. 19.

[23] P. Stoica and R. Moses, *Introduction to Spectral Analysis*. Upper Saddle River, NJ, USA: Prentice-Hall, 1997.

[24] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *Proc. for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, 2003, 7-8 March 2002 - MIT Faculty Club, Cambridge, MA, USA.

[25] "Dht protocol specification," http://www.bittorrent.org/Draft_DHT_protocol.html.