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

[§]Math Computer Science Dept.
Suffolk University
hzhang@ieee.org

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

***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 use 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 among peers, resulting in low availability 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 considered more than 1,400 trackers and 24,000 DHT nodes (extracted from about 20,000 torrents) over a period of two 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 connectivity of the overlay formed by peers, (*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 (BT), 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 BT 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 BT 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 pieces 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 BT systems today. The massive prevalence of BitTorrent and recent proposals to adapt BT techniques for more general forms of packet delivery [3] including email attachments, software updates, and security patches make tracker availability an important problem. For example, unavailability of security updates distributed using BT can seriously impact the well-being of the Internet.

Two recent trends have emerged to tackle the problem of tracker availability. The first one 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). The second one is the integration of Distributed Hash Tables (DHTs) with BT clients that store information across the entire community of BT users (introduced in May 2005). Section V and VI describe in detail how these two mechanisms work in practice.

Our study investigates availability of BT 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 more than 20,000 torrents specifying more than 1,400 trackers and 24,000 DHT nodes over a period of several months. We find that multiple trackers as well as DHT use 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 trackers can significantly reduce the connectivity of BitTorrent overlay.
- DHT improves information availability, but induces a higher response latency.
- Tracker and DHT show complementary characteristic 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 BT 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], global downloading performance achievable by the peers [5], the BT-specific content sharing algorithms like the choke algorithm or the pieces selection algorithm [7] in particular as regards their effectiveness in promoting cooperation [8], [9].

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. `suprnova.org` was not just a website, but a complete architecture including a mirroring system to balance user requests across multiple websites, servers to store torrent files, and human moderators to eliminate faked contents. The measurements span from June 2003 to March 2004, and the authors investigate the availability of the architecture and also of the peers of a specific content. Tracker availability appears to be a significant problem: only half of the 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 considered during the measurement campaign described in [2] (multi-tracker support was introduced during the measurement period, DHT support only later).

Separate from the specific BT framework, there are some works about availability of distributed systems in the Internet [10], [11], [12], [13], [14]. In [10] the authors investigate peers availability through a measurement campaign of the Overnet file-sharing network [15]. 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 BT architecture we observed lifetime effects in our availability measurements. In [11] three different large distributed systems (PlanetLab, Domain Name System 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 (in contrast with [10]). Our study confirms the presence of correlated failures among different trackers. [12] points out some limitations of 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 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 (see also remarks in Section VII). [13] 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. [14] 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 BT, 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. A client performs a HTTP GET request to the tracker specified in the announce section in order to receive a subset of peers. In this paper we refer to this request as an announce request. In order to support multiple trackers and DHT two new optional sections have been added: the *announce-list* section and the *nodes* one.

In our study we considered about 20,000 torrents, found mainly through `www.torrentspy.com`. We developed a script, which automatically downloads the RSS feed of this site and then downloads every new torrent file indicated in the feed. In what follows we refer to the following sets.

- SET1 : set of 4238 torrents advertised by `www.torrentspy.com` from May 15 to May 19, 2006.
- SET2 : set of 17198 torrents advertised by `www.torrentspy.com` from May 20 to June 30, 2006.

All these torrents specify more than 1,400 trackers and more than 24,000 DHT nodes. Table I summarizes information about trackers and nodes we can extract from the different sets. Azureus is one of the most popular BT clients together with Bram Cohen's[1] one, which is usually called the *Mainline* client [16]. The Table also specifies the length of the measurement period as regards trackers availability. While SET1 is smaller in terms of torrents and trackers, it has been investigated during a longer period of time. For this reason SET2 has been used to investigate characteristics at a given time instant,

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

set	advertised torrents#	unique torrents#	Trackers# total	HTTP	UDP	Mainline DHT nodes	Azureus DHT nodes	Availability Meas. Period
SET1	4238	4186	525	491	34	4646	21	May 26th-July 27th
SET2	17198	16900	1355	1283	72	21474	196	July 5th-July 28th

TABLE I
TORRENT SETS

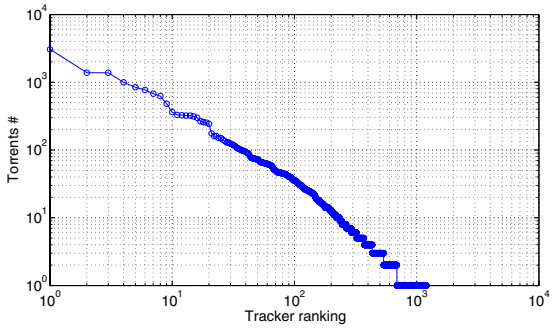

Fig. 1. Popularity of the Trackers in SET2

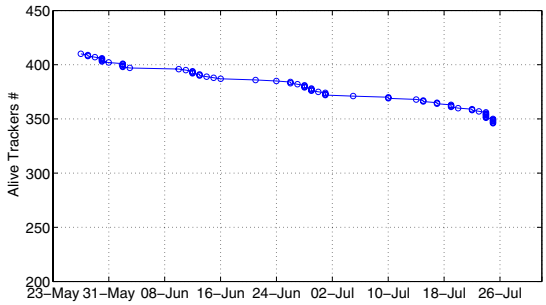

Fig. 2. Number of live Trackers

while SET1 has been used to investigate performance across time. In our study we also considered `www.btjunkie.com` (the corresponding data sets are described in [17]). Although this website declares to be the largest BT search engine, we were able to obtain fewer torrents through their RSS feed than through the RSS feed of `www.torrentspy.com`.

Figure 1 shows the distribution of the torrents across the different trackers for sets SET2 . 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 and also if we estimate the popularity of each tracker directly by querying it with an apposite *scrape* request [17].

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 order to reduce the load on trackers [18]. 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). A single machine in UMass network has been performing the task, with at most ten trackers being probed at the same time. We calculate tracker availability by dividing the number of successful probes (i.e. the number of probes the tracker responds to) by the total number of probes.

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 [18]. A probe consists of three UDP packets sent consecutively. The tracker is considered unavailable (the probe is unsuccessful) if these attempts fail. HTTP tracker availability has been evaluated by 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 (the time between two consecutive attempts is equal to 100 seconds). This procedure can produce wrong results. For example some trackers are implemented as modules of Apache web-servers and BT requests are identified from the specific URL and forwarded to the tracker module. Our measurements suggest that this is quite common (see [17]). 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 accepts incoming TCP connection. The problem is not easy to solve and we decided to rely on a heuristic to identify such cases [17]. In such a way we identified 16 web-servers where the BT module had been probably uninstalled.

We performed tracker availability measurements for two months. We observed that for some trackers the availability depends on the length of the measurement time interval (a similar effect was observed in [10] 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 finally stop operating. Figure 2 quantifies this non-stationary effect. It shows the evolution of the number of live trackers during the two months. We assume that a tracker dies 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

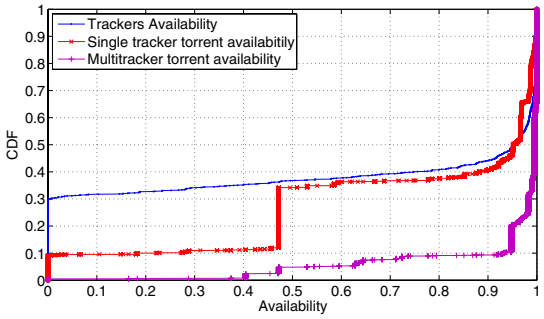

Fig. 3. Trackers and Torrents Availability

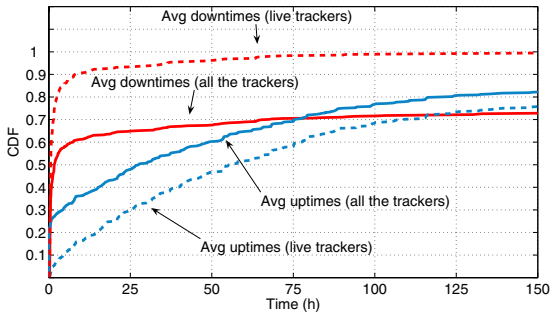

Fig. 4. CDF of average up-time and down-time over two months

15%) over 58 days, from May 27 to July 24. From the data we can roughly estimate that the tracker lifetime is about 390 days[2].

Figure 3 shows the Cumulative Distribution Function (CDF) of the availability of SET2 trackers over a 21 days period (curve labelled "Trackers Availability") starting from July 5. 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 in characterizing the availability of information for the peers in a given swarm. We briefly refer to this concept as *torrent availability*. For single tracker torrents, the 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, the CDF of the torrent availability would coincide with the CDF of the tracker availability, but we have shown in Figures 1 that tracker popularity is skewed. This effect is clearly shown by the CDF of the torrent availability in Figure 3 (curve labelled "Single Tracker Torrent Availability")[3]. 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% during the period which the figure refers to. If we filter out this tracker, the torrent availability 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 3 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 response variable and the number of torrents (derived from SET1) 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 $[-5.4 * 10^{-4}, 0.3 * 10^{-4}]$ and the correlation coefficient is quite small, 0.0216. This analysis does not suggest a dependence between the two variables. We performed also a linear regression considering the number of torrents each tracker declares as answer to a scrape request. The conclusion is the same [17].

Finally Figure 4 shows the CDF of the average uptime and downtime evaluated for all the trackers in SET1 and considering only the trackers alive at the end of the measurement period (from May 26th to July 19th). If we consider all the trackers then only 45% of the trackers appear to have an average uptime smaller than 1.5 days. This is similar to what observed in [2][4], but we note that if we restrict to live trackers the average availability increases significantly and about 60% 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 often due to software or machine crash rather than to temporary network problems[5].

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 sends it an announce request. 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 contact a tracker in the first list, if all the announce requests to trackers in the first list fail, it tries to contact a tracker in the second list and so on. On the next announce, it

[2]Under the assumption of exponential independent lifetimes, the lifetime can be estimated as the inverse of the average tracker death rate $(62/(416 * 58) \approx 2.6 * 10^{-3}$ per day).

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

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

[5]The measurement study in [19] shows that only 10% of the path failures last more than 15 minutes, and only 5% more than half an hour. The older study from Paxson [20] shows even shorter durations.

repeats the procedure in the same order. Trackers in different lists do not share information. There are two common ways 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, among which 60% are 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 [11] for Planetlab machines and web-servers, and opposite from the results in [10] for Overnet peers. In [11] the authors simply show that the number of near-simultaneous failures does not seem to follow a geometric distribution[6], nor a beta-binomial distribution which should be more suited to account for correlated failures. In [10] 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 an evident diurnal pattern in single host availability.

Our analysis is based on 4 weeks availability measurements for live trackers (trackers which were not completely unavailable during the measurement period) in SET1 and is more accurate from the statistical point of view. For all the tracker pairs[7] we considered the contingency table and performed a G-test. 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.We performed also an approximate Fisher test which overcomes some limitations of the G-test and so allow us to consider a larger set of pairs (86%). The results of the G-test are confirmed also on this larger set [17].

One simple cause of correlation is that trackers can 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 found by the tests.

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

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

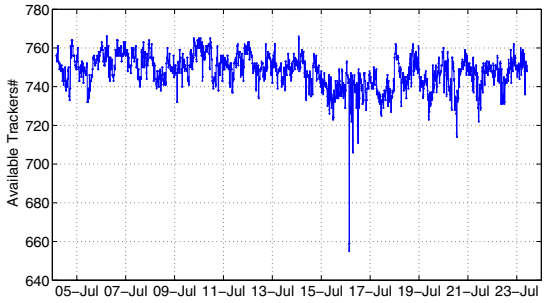

Fig. 5. Number of Available Trackers Time Plot

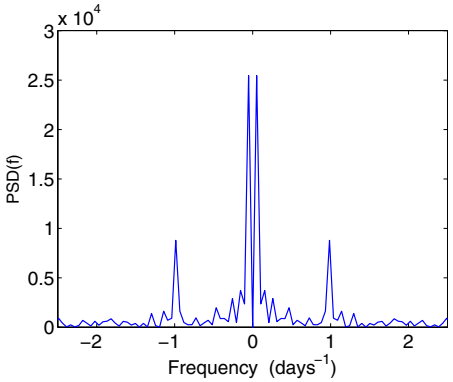

Fig. 6. Power Spectral Density of the Number of Available Trackers

We think that this correlation can be due to a daily pattern in tracker availability. This can be a consequence of user behavior (churn) or of tracker failures that can be recovered only when the user is present or of network failures [20]. Figure 5 shows the total number of available SET2 trackers for three weeks in July 2006 with a 10 minutes resolution. The daily pattern is confirmed by Figure 6, where the spectral density, evaluated with the unmodified periodogram method, exhibits a peak corresponding to a 1-day periodicity[8].

B. Availability Improvement

The presence of multiple trackers in a 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 refer to the availability evaluated in this way as *time-aware availability*.

The CDF of the time-aware availability for multi-tracker torrents is plotted in Figure 3. This picture shows a significant

[8]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 shown in Figure 5.

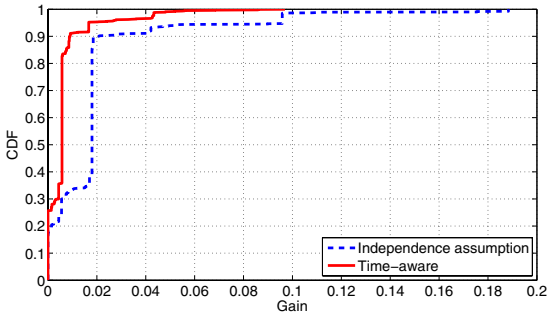

Fig. 7. Multitracker Gain Distribution

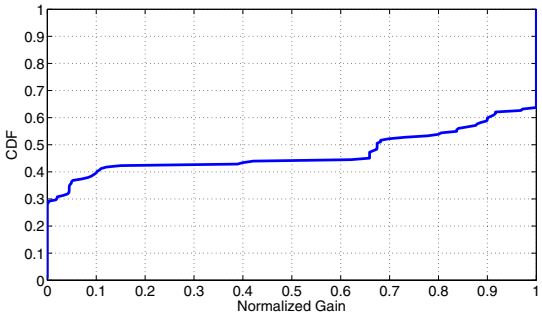

Fig. 8. Multitracker Normalized Gain Distribution for the different group of trackers

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 7. The figure shows the availability improvement using all the trackers, in comparison to the availability of the best 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 improvement (gain) is equal to 2%. The availability has been evaluated both considering trackers availabilities independent (dashed curve) and considering the availability temporal 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: trackers tend to be available during the same time periods.

Figure 8 gives some more insight. The figure shows the gain distribution across all the tracker groups specified in the set[9]. The gain has been normalized to the maximum possible improvement (e.g. in the above example the normalized improvement is $0.4 = 2/(100 - 95)$. 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 7. 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 7 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. 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 torrent makers and BT clients, which can cause a swarm to split into disjoint subsets [21]. This would be clearly harmful for content spreading. In what follows we use the term *subswarm* to denote the subset of the swarm a tracker manages, i.e., all the peers it knows about.

[9]Differently from Figure 7 two torrents which specify the same group of trackers are considered as one.

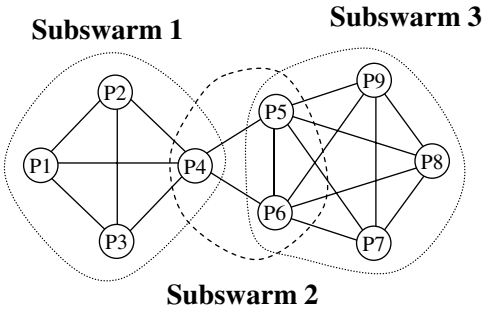

Fig. 9. Potential Neighbors Graph

In order to evaluate if the risk of disjoint subswarms is realistic, we considered all the 568 multi-tracker torrents in SET2. On July 14th for each torrent we made multiple announce requests to each tracker in the announce-list in order 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 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 BT overlay, this occurs when the tracker managing the subswarm includes P1 (P2) in the response to an announce request from P2 (P1). For this reason we refer to this graph as *potential neighbors graph*. An example is shown in Figure 9: 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 then 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 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).

Even when the graph is completely connected, we can quantify subswarm overlap and then the possibility to spread the content across the swarm. 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 9 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 $g_S = 1$[10]. Then we define the conductance of the swarm as the minimum value of g_S among all the subswarms. For example the conductances of the three subswarms in Figure 9 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.

Figures 10 and 11 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. As was expected the performance are very good for pure load balancing. In fact in this case trackers periodically communicate with each other their subswarms. Performance can be bad for backup, especially if we look at the conductance in Figure 11. It appears that 27% of the worst connected subswarms have a conductance smaller than 0.5, which indicates that on the average peers in the subswarm can at most discover half of the peers outside the subswarm. Data in [17] show that connectivity and conductance are positively correlated.

VI. Distributed Hash Tables

The latest versions of the most popular clients (Azureus, Mainline, BitComet, μTorrent, BitLord and BitSpirit [16]) implement the functionalities of a DHT node, so that all peers, independently from the content they are interested in (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 in a specific content. According to the common DHT language the *key* is the info-hash 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.

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

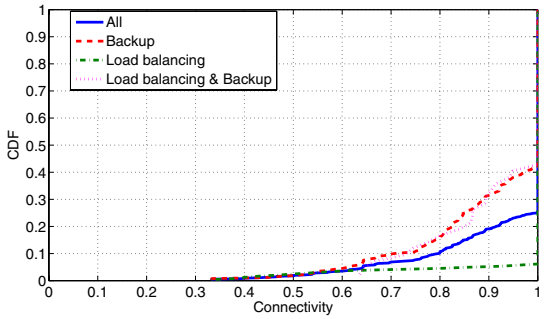

Fig. 10. Connectivity Cumulative Distribution Function

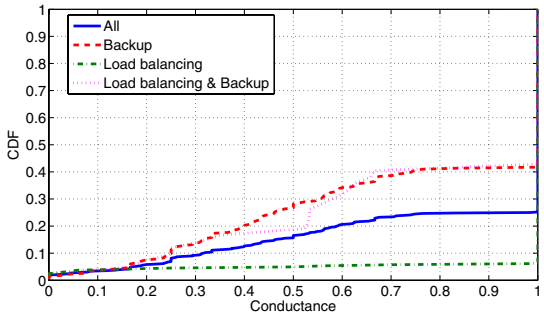

Fig. 11. Conductance Cumulative Distribution Function

We said that all the BT clients could form a single DHT, but in reality there are currently two different incompatible implementations (both based on the Kademlia model [22]): the Mainline one, and the Azureus one. Except Azureus all the other clients are compliant with Mainline DHT specifications. Our measurement study focuses on the Mainline DHT.

A. A Brief Overview of DHT Operation

When a user creates a new torrent, the program usually allows him 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[11]. These DHT nodes act as bootstrap nodes, in fact they are used in order to initialize the client routing table. The routing table is updated from time to time according to the protocol description in [23]. There are also other ways to discover DHT bootstrap nodes to initialize the routing table. For example if a peer is already in a swarm and is connected to another peer, they can exchange DHT related information.

In order to download the content, the BT client can send requests to the set of DHT nodes in its routing table closest[12] to the infohash. The contacted nodes will reply with the contact

[11]Each BT client is at the same time a peer and a DHT node.

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

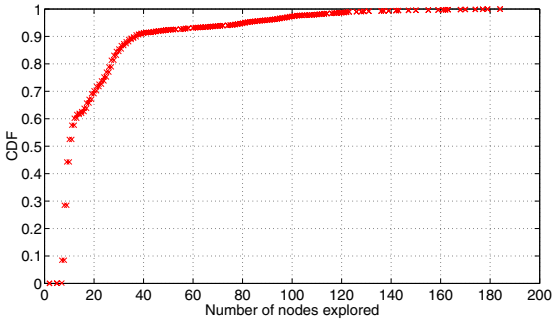

Fig. 12. The cumulative distribution of the number of DHT nodes ever explored before finding the first valid peer in a swarm.

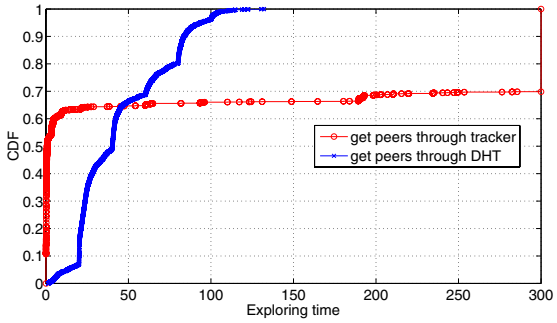

Fig. 13. Comparison between DHT and Tracker. The cumulative distribution of the time needed to find the first valid peer in a swarm.

information of peers interested in 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 [23] 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 DHT pings sent consecutively. Due to space constraints, we do not show any plot [17]. We simply mention that 70% of the nodes were always unavailable, while the others showed 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 SET2 with DHT nodes. For each torrent, we first erased the routing table and all the files that kept the information related to contents previously downloaded. Namely, the client started with a clean state for each torrent. Then we let the client start contacting the DHT nodes in the torrent file and trying to recover information about peers. In the meantime, all the nodes in the routing table were logged (recall that the routing table is updated frequently). The measurement was stopped after the client had received the first valid peer and the next torrent was considered. Our experiment started at 20:15 on July 22, 2006, and it took about 34 hours to finish.

Figures 12 and 13 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

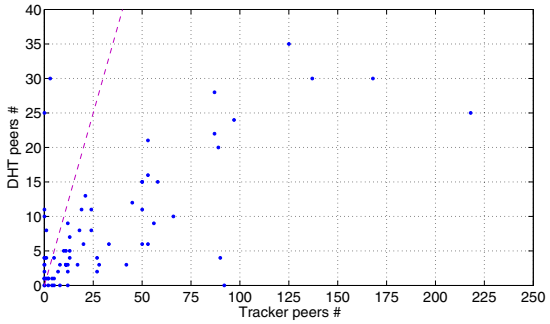

Fig. 14. Number of Peers obtained by the DHT in 20 minutes vs Number of Peers obtained by one query to the Trackers

than 88 seconds. In the worst case the time needed was 140 seconds and 184 DHT nodes were explored. There is a strong correlation between the number of DHT nodes explored and the time elapsed in order to find a peer [17].

For comparison, we also investigated the time needed to find the first valid peer by just contacting trackers in the same data set. 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 13. As expected, usually tracker can respond with valid peers faster than DHT, in less than one second. However, note that about 30% of 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.

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[13]. Figure 14 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 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.

The conclusion of these two experiments is that trackers in general provide more information and faster, but the DHT can significantly increase the availability of the whole system.

VII. Conclusions and Future Research

From a 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 whether this information is updated (e.g. if the peers provided by trackers and DHT are effectively online), 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 slows down file diffusion). Also, it could have been interesting to weight the information availability with the number of peers interested into this information (as in [12]). We repute these issues meaningful and we deserve them for future research. We observe that if we would have collected the data needed to address these issues on the same data sets and with the same time granularity, the load on the trackers would have been much higher (see [17] for more details). We would also like to carry out a new measurement campaign using more measurement points, this would help us to distinguish the different causes of tracker unavailability.

VIII. Acknowledgements

The authors would like to thank Luca Scalia at D.I.E.E.T. 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.

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

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*, Febr. 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*, May 2006.

[4] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos, "Is p2p dying or just hiding?" in *Proc. of the 47th IEEE Global Telecommunications Conference*, Nov. 2004.

[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 the 5th ACM SIGCOMM Internet Measurement Conference*, Oct. 2005.

[7] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Understanding BitTorrent: An Experimental Perspective," INRIA, Tech. Rep. 00000156, 2005.

[8] 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*, Aug. 2005.

[9] 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*, 2006.

[10] R. Bhagwan, S. Savage, and G. M. Voleker, "Understanding Availability," in *Proc. of the 2nd International Workshop on Peer-to-Peer Systems*, Febr. 2002.

[11] P. Yalagandula, S. Nath, H. Hu, P. B. Gibbons, and S. Seshan, "Beyond Availability: Towards a Deeper Understanding of Machine Failure Characteristics in Large Distributed Systems," in *Proc. of the 1st Workshop On Real Large Distributed Systems*, 2004.

[12] 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*, Sept. 2005.

[13] 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 II Conference*, July 2002.

[14] 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 19th ACM Symposium on Operating Systems Principles*, Oct. 2003.

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

[16] "Wiki on bittorrent clients," http://en.wikipedia.org/wiki/Comparison_of_BitTorrent_software.

[17] G. Neglia, G. Reina, H. Zhang, D. Towsley, A. Venkataramani, and J. Danaher, "Availability in BitTorrent Systems," UMass, Tech. Rep. 06-41, June 2006, ftp://gaia.cs.umass.edu/pub/Neglia06availability_bt06-41.pdf, http://www-sop.inria.fr/maestro/personnel/Giovanni.Neglia/publications/Neglia06availability_bt06-41.pdf.

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

[19] N. Feamster, D. G. Andersen, H. Balakrishnan, and F. Kaashoek, "Measuring the Effects of Internet Path Faults on Reactive Routing," in *ACM Sigmetrics - Performance 2003*, 2003.

[20] V. Paxson, "End-to-end routing behavior in the Internet," *IEEE/ACM Trans. on Networking*, vol. 5, no. 5, pp. 601–615, 1997.

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

[22] P. Maymounkov and D. Mazieres, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *Proc. of the 1st International Workshop on Peer-to-Peer Systems*, March 2002.

[23] "DHT protocol specification," http://www.bittorrent.org/Draft_DHT_protocol.html.