Modeling Unconnectable Peers in Private BitTorrent Communities

Kornél Csernai, Márk Jelasity
Dept. of Informatics,
University of Szeged, Hungary

Johan Pouwelse, Tamás Vinkó
Department of Computer Science,
Delft University of Technology, The Netherlands

***Abstract*—In a typical BitTorrent swarm, a large proportion of the peers are behind firewalls or NATs. These peers are called unconnectable. When developing P2P applications, a main requirement is to handle unconnectable peers appropriately. One important aspect of this problem, which has not been emphasized so far, is understanding the difference between the attributes of unconnectable peers and peers in the open Internet. For example, if unconnectable peers spend much less time online, or if they download significantly more, exploiting these facts helps to optimize the implementation; and ignoring these facts can even lead to severe performance problems. Comparing open and unconnectable peers is not easy because most traces contain no information about connectability. Here we study two large traces collected in two private BitTorrent communities: FileList.org and BitSoup.org, both of which contain the connectability attribute. From these traces we extract several attributes of individual online sessions, swarms, and users. We compare the distributions of these attributes over unconnectable and open peers. We find that there are some potentially important differences, e.g., unconnectable users tend to have a lot more sessions, and they tend to spend slightly more time online. Some of our findings are in contradiction with previous results that were based on a different trace collection methodology.**

I. Introduction

It is clear that in the past years interest in developing P2P protocols that tolerate or even exploit NATs and firewalls has been increasing. Apart from technical aspects of NAT puncturing [6], protocols have also been proposed. For example, Kermarrec et al. and Leitão et al. proposed gossip protocols for environments with NATs and firewalls [11], [12], and D'Acunto et al. measure through simulations the speed gap between the two connectivity classes and they concluded that the connectable peers benefit from the presence of the unconnectable peers [4].

This increasing interest makes it important to understand the difference between the behavior of unconnectable and open clients so that simulating P2P protocols in the development phase could rely on realistic assumptions. However, although there are numerous P2P (and BitTorrent) traces available [1], [2], [8], [16], [18], very few of them have direct information about the connectability of the peers. In this paper this will be our main focus.

We are not aware of any studies that offer a thorough comparison of the different connectability classes, although some measurement studies do provide information on the ratio of peers that are behind a firewall or NAT [5], [7], [15]. Reported values range from 35% to 90%.

In this paper we look into questions that go well beyond the ratio of unconnectable peers such as the dependence of several key attributes of sessions and users (upload and download volume, session length, and so on) on unconnectability based on two datasets from two private BitTorrent communities: FileList.org and BitSoup.org.

II. Basic Notions

First let us summarize some basic notions of BitTorrent [3]. In a BitTorrent P2P network, each *peer* (user) downloads and uploads data simultaneously. The *torrent* file describes one or more files that are to be shared. The files are then split into fixed-size *chunks* or *pieces*, which are transferred in *blocks*. The peers can acquire these pieces in any order. A *swarm* is the set of peers participating in downloading a common torrent. The peers that have finished downloading all the pieces defined in the torrent are called *seed*s, whereas the ones still trying to get some of them are called *leech*es. The *sharing ratio* is defined as the *uploaded/downloaded* ratio for each session.

A *tracker* is typically a central database-driven website that coordinates the peers and keeps track of their uploads, downloads, sharing ratios, client versions, and so on.

Let us now introduce the notion of private communities. A private BitTorrent community, or a 'BitTorrent darknet' [19], [20] is a special kind of BitTorrent network. These communities restrict their membership: one can typically join only after receiving an invitation from a senior member.

Private trackers aggregate the lifetime sharing ratio of each user and enforce a sharing ratio policy. The method of sharing ratio enforcement depends on the private tracker site. For example, users that have a sharing ratio below a given threshold may have to wait hours before being able to start downloading newly added content, or may even be excluded from the community. However, a good sharing ratio can earn the user some extra privileges. For this reason users have a strong incentive to contribute to the community by uploading (seeding) as much as they can.

III. The Datasets

There are many ways of creating a trace of a BitTorrent network. One is through active measurement [10], [18], where a modified BitTorrent client is used to request peers from the tracker as a normal client. The modified client then performs a handshake with the peers, but instead of exchanging data, it disconnects itself, and stores information about the pieces each peer is reported to have. The downside

In Proc. 19th PDP, 2011, pp582–589, doi:10.1109/PDP.2011.21

of this approach is that unconnectable peers are out of reach. Although unconnectable peers can connect to the modified client, this is an extremely unreliable and inefficient way for collecting information about unconnectable peers.

Instead, we rely on data collected from the tracker. The tracker usually has a Web front-end, which is the main source of retrieving torrent metafiles. The front-end also provides the users with some aggregated statistics for each torrent (swarm) that is registered. The data is based on what the clients report to the tracker periodically. The attributes of the stored records include connectability, downloaded and uploaded amount and speed, completion (%), client version, sharing ratio, and so on. Collecting the trace consists of downloading the HTML output of each swarm's statistics page periodically and converting these HTML files into suitable formats.

The advantage of tracker-based traces is that the central view of the tracker is more complete and accurate than the former method. However, the tracker also adds a layer of obscurity and information loss, as indicated by some anomalies in the collected records. For instance, based on the traces we processed, we suspect that in some cases the client and the tracker did not agree on how to report the downloaded and uploaded amounts. Also, the reporting period for the various clients can range from 5 minutes to 1 hour. We address some of these issues in Section IV.

For our analysis, we use two separate traces. The first one is a FILELIST.ORG trace collected by Jelle Roozenburg between December 9, 2005 and March 12, 2006 [17]. The second is a BITSOUP.ORG trace collected by Andrade et al. from April to July, 2007 [1]. Both datasets were collected using a similar methodology: the tracker website was periodically crawled (around every six minutes for FILELIST.ORG, and every hour for BITSOUP.ORG) to obtain a partial state of the P2P network.

The BITSOUP.ORG database was originally made up of multiple observation intervals of which we chose one continuous interval that resembled the FILELIST.ORG trace most in terms of length and the number of peers and swarms. In addition, the BITSOUP.ORG trace does not contain swarms of torrents smaller than 100MB due to the large crawling interval [1], so we decided to remove the torrents smaller than 100MB from the FILELIST.ORG trace as well.

Figure 1 illustrates the number of online users as a function of time. The actual number of sessions, users, and torrents (swarms) observed in the datasets can be seen in Table I along with the effects of the cleaning process which we discuss in the next section.

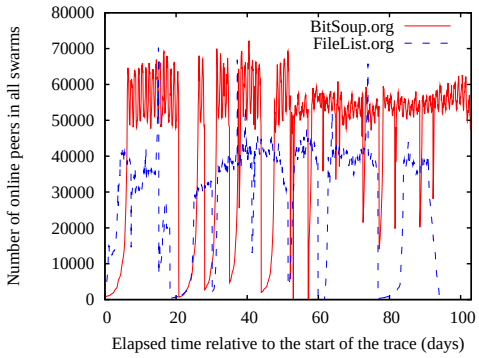

Figure 1. The number of online users as a function of time.

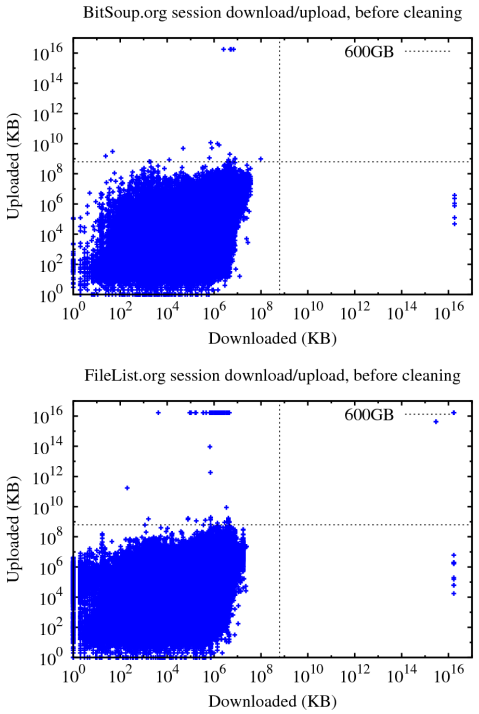

Figure 2. The motivation for cleaning the data: the download/upload scatter plot of the sessions, where every point represents a session.

IV. CLEANING THE DATA

There are a number of possible factors that can contaminate a trace, including measurement errors and malicious user behavior, where clients intentionally report incorrect information to the tracker. For a correct and unbiased result, we need to eliminate these sources of errors as best we can.

We found an insignificant number of trivially erroneous events that report negative traffic, or otherwise infeasible attribute values. Again, such events might result from bugs in the tracker or in the crawler. Accordingly, we discard these events.

Malicious user behavior typically means misreporting the amount of upload, since this results in a better sharing ratio. Most trackers do not perform feasibility checks on the reported amount of upload (including the one that produced the trace in question) so this is relatively easy to do.

Figure 2 illustrates the rule that we used to clean this type of misbehavior: we removed all users that had at least one session that reported traffic exceeding 600GB. The limit was set based on a visual inspection of the scatterplots shown in the figure. Observe that the two dataset show a very similar

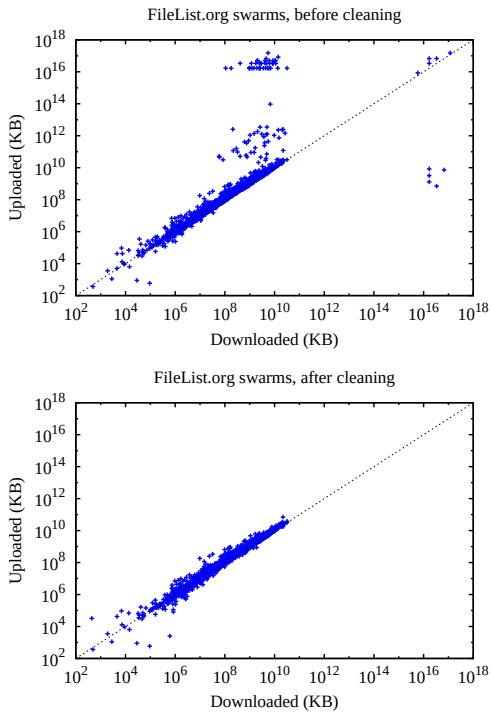

Figure 3. The effects of cleaning: total download/upload scatter plot of the swarms, where every point represents a swarm. The BitSoup.org trace shows a very similar pattern (omitted due to lack of space).

structure.

Note that all the sessions of these users were removed, not only those that exceeded this limit. This was because we decided that these users could not be trusted.

The amount of data that was removed is shown in Table I. We can see that—especially in the case of FILELIST.ORG—the amount of information loss is not significant. Table II shows the distribution of the erroneous sessions, users, and torrents that we filtered out. We can see that data removed due to "misbehavior" correlated closely with trivial errors, which might indicate that what we observe is in fact not misbehavior afterall, but an artifact of other client-tracker communication problems.

V. ANALYSIS OF THE DATA

After cleaning the traces, we converted both into three databases, the records of which describe individual online sessions, users, and users within a swarm.

The attributes of the session records we discuss in this paper are the following: *session-length* (sec); *upload* and *download*, that give the amount of uploaded and downloaded data during the session (KB); *seeding-length*, that is, the amount of time spent online after the file has been completely downloaded (sec); *seeded*, the amount of data uploaded during seeding (KB); *up-speed* and *down-speed*, that are calculated by taking the maximum of the download or upload speed, respectively, as reported by the tracker over the observation points of the session (KB/sec); and *open*, a one-bit attribute stating whether the user was open (1) or unconnectable (0), as reported by the tracker. Obviously, a session also has an associated user and a swarm (that is, a file).

Note that *up-speed* and *down-speed* are supposed to approximate the bandwidth available during the session. Obviously, this is a very rough approximation; nevertheless for our purposes it is sufficient, since the meaning and measurement methodology of the value are independent of the connectability bit.

Most of the attributes of a user record summarize session attributes that belong to the given user: for each session attribute we calculate the maximum, average, sum, variance and median w.r.t. the given user (note that the minimum is always 0). The record also contains the number of sessions (*# sessions*) and torrents (or swarms) (*# torrents*) the user belongs to. Attribute *# loss* gives the number of times a session has a smaller completion rate (i.e., it has downloaded a smaller part of the file) than the previous session in the same swarm. This happens if a user loses parts of a file, or if many users use the same account, and are downloading a file in parallel. Attributes *# seed-overlap* and *# non-seed-overlap* give the number of session overlaps within the same swarm, where the overlapping sessions are both seeding, or at least one of them is not seeding, respectively. Seeding overlaps can happen when a user is intentionally seeding a file from several servers, for example.

User/swarm records contain the same attributes as the user records, except that each record summarizes the sessions that belong to a fixed user over a fixed swarm. This way, all users have as many records in this database as the number of swarms they participate in.

We calculated and analyzed many additional attributes that we do not discuss here because we judged them too unreliable, we could not reverse-engineer their clear semantics, or we found them uninteresting or redundant.

Note that the same user can have both connectable and unconnectable sessions for a variety reasons. For example, user IDs can be shared among users, or a user can use several machines or clients at the same time. Our main focus is to analyze the difference between the records that are clearly connectable or unconnectable. For this reason, we removed those records that have an average value of attribute *open* different from 0 or 1. Table III shows what proportion of the records are clearly open or clearly unconnectable. In the case of the user databases, roughly half of the records (users) have both open and unconnectable sessions, but within one swarm user behavior is much more consistent with only around 10% of the users having mixed sessions. A single session is always clearly open or unconnectable. The proportions are rather similar in the two traces.

A. Methodology

The key question we would like to answer is *whether the attributes in the unconnectable class of the records have the same distribution as in the open class?* In this study we consider only single attributes, and ignore the comparison of covariance, and other multivariate statistics.

	FileList.org			BitSoup.org		
observation points	681,812,792			122,660,152		
number of	before cleaning	after cleaning	% diff.	before cleaning	after cleaning	% diff.
sessions	13,935,412	13,809,112	<1%	15,518,599	13,351,279	14%
users	91,745	91,579	<1%	97,943	94,633	4%
torrents	3,064	3,016	2.6%	14,837	11,710	21%
refresh interval	6 minutes			1 hour		
first measurement	2005-12-08			2007-01-27		
last measurement	2006-03-12			2007-05-10		
days	94			102		

Table I
BASIC PROPERTIES OF THE TRACES AND THE EFFECTS OF THE CLEANING.

	FileList.org			BitSoup.org		
	removed total	due to trivial error	due to misbehavior	removed total	due to trivial error	due to misbehavior
sessions	126300 (100%)	120241 (95%)	111837 (89%)	2167320 (100%)	2165082 (99.9%)	1946322 (90%)
users	166 (100%)	161 (97%)	153 (92%)	3310 (100%)	3305 (99.8%)	3204 (97%)
torrents	48 (100%)	45 (94%)	48 (100%)	3127 (100%)	3127 (100%)	3127 (100%)

Table II
THE DISTRIBUTION OF THE DATA THAT WAS FILTERED OUT. NOTE THE LARGE OVERLAPS AMONG REASONS FOR REMOVAL.

	FileList.org	BitSoup.org
database	total / open(%) / unconnectable(%)	total / open(%) / unconnectable(%)
session	13,809,112 / 69% / 31%	13,351,279 / 68% / 32%
user/swarm	2,148,871 / 62% / 27%	1,848,478 / 60% / 26%
user	91,579 / 32% / 16%	94,633 / 36% / 19%

Table III
PROPORTIONS OF CLEARLY UNCONNECTABLE AND CLEARLY OPEN RECORDS.

To compare two distributions, one can apply several statistical tests and visualizations. Since the distributions we are dealing with are very far from normal, we can apply only generic nonparametric tests such as the Wilcoxon two-sample rank-sum test [9]. After studying the results of this test, we found that for a large enough sample size it suggests that the distributions over each attribute differ significantly. However, we are not interested in whether the distributions are *exactly* the same or not: we are more interested in the nature and significance of the difference, where a test score offers little help.

We also experimented with several visualizations, among which the well-known quantile-quantile (or Q-Q) plot seemed to be the most informative. Given two data sets drawn from two distributions, the Q-Q plot shows the quantiles of one data set plotted against the same quantiles of the other. Practically speaking, one has to sort the smaller data set, which gives one coordinate, and the other coordinate has to be calculated as the corresponding quantiles of the larger data set. The Q-Q plot is useful because one can not only test whether two distributions are the same, but one can also derive the nature of the difference (scaling, shift, different skew, etc). In this paper we rely only on Q-Q plots.

When plotting the Q-Q plots, we had to consider two issues. The first was a special property of the datasets, namely that the discrete value 0 has a high probability in almost every case. Table IV summarizes these empirical probabilities. At other locations, the distributions behave as continuous distributions. For this reason, the Q-Q plots were created with the samples of value 0 removed. In other words, we compare conditional distributions: we assume the value is positive.

The second issue is the scale of the sample values. In almost every case, the density function of the distributions is heavy tailed, often very close to scale-free. To get a usable visualization, we consider the Q-Q plots on the log-log scale. Nevertheless, it should be kept in mind that on the log-log scale the interpretation of the Q-Q plot changes slightly. For example, if one distribution is scaled w.r.t. the other, then, instead of a non-translated line with a different steepness, we get a line with a steepness of 1, but translated.

Finally, we plot the Q-Q plots using vertical lines that connect every point in the plot to the $x = y$ line to emphasize the deviation from $x = y$.

B. Discussion

Figure 4 shows the Q-Q plots for the session database. The attribute *seeded* is not shown as it is very highly correlated to *upload*. The attribute *seeding-length* is also very highly correlated to *session-length*, as can be seen in the figure.

We can see that only *download*, and the speed attributes show notable differences between the open and unconnectable classes. In the case of *download*, it is clear that unconnectable sessions tend to download significantly less, except in the very high range (but over 90% of the sessions download less than 10^6 KB).

In the case of the speed attributes it is apparent that unconnectable sessions tend to be faster when it comes to upload. The bump in the Q-Q plots shows that the highest

Statistics over the session database

	FileList.org						BitSoup.org					
	mean-1	mean-0	std-1	std-0	$P(0\|0)$	$P(0\|1)$	mean-1	mean-0	std-1	std-0	$P(0\|0)$	$P(0\|1)$
session length	18102	18259	34999	36267	0.0677	0.0537	29922	31471	61032	62665	0.0008	0.0006
upload	403781	351502	2507378	2139577	0.3416	0.3478	297791	263032	2134956	1699252	0.3400	0.4203
download	748835	651444	978731	890217	0.7169	0.6720	821859	654589	1229945	1075100	0.7861	0.7724
seeding length	18095	18341	36508	38445	0.2392	0.2878	29572	31441	62502	64274	0.1864	0.2217
seeded	392759	337736	2441262	2161164	0.4711	0.5211	276871	237969	2122849	1685898	0.4696	0.5552
up-speed	95	101	8408	6555	0.4979	0.5612	14	16	3264	1767	0.4765	0.5695
down-speed	256	277	7669	14505	0.7655	0.7363	138	90	14046	9449	0.8038	0.7969

Statistics over the user database

	FileList.org						BitSoup.org					
	mean-1	mean-0	std-1	std-0	$P(0\|0)$	$P(0\|1)$	mean-1	mean-0	std-1	std-0	$P(0\|0)$	$P(0\|1)$
# torrents	37	93	77	219	0.0000	0.0000	36	103	99	307	0.0000	0.0000
# sessions	41	107	111	307	0.0000	0.0000	37	106	109	359	0.0000	0.0000
session-length-avg	23264	22112	22682	23701	0.0038	0.0038	40303	37521	56328	53359	0.0001	0.0001
session-length-sum	872977	2034761	2237036	5639709	0.0038	0.0038	1311153	3411258	3654604	10691444	0.0001	0.0001
upload-avg	720156	479084	2942067	2874685	0.0307	0.0371	661080	384126	3085786	2522138	0.0612	0.0841
upload-sum	23349397	29824368	121485330	150603731	0.0307	0.0371	16104440	19250478	111726542	79252263	0.0612	0.0841
donwload-avg	659193	516800	738597	614973	0.0515	0.0516	702225	477704	946447	766688	0.0816	0.0757
download-sum	17500925	27638790	35100807	62204665	0.0515	0.0516	14084784	18633679	29871850	39872233	0.0816	0.0757
seeding-length-avg	17772	15025	21764	22648	0.0926	0.1101	30285	27232	53358	50532	0.1670	0.1768
seeding-length-sum	763316	1707381	2176955	5349162	0.0926	0.1101	1183659	3192954	3558187	10564417	0.1670	0.1768
seeded-avg	557374	319991	2098243	1671083	0.1172	0.1379	510757	292804	2505441	2599416	0.2143	0.2389
seeded-sum	20005238	23695184	110628312	143061014	0.1172	0.1379	14017881	16163008	114788037	73464048	0.2143	0.2389
up-speed	3969	4340	86765	61920	0.0507	0.0713	173	271	8282	15647	0.0818	0.1136
down-speed	2942	3851	31576	36112	0.0560	0.0592	488	462	28344	17970	0.0869	0.0839
# loss	2	3	3	4	0.8751	0.7476	2	2	2	4	0.9111	0.8342
# non-seed-overlap	0	0	0	0	1.0000	1.0000	7	16	12	41	0.6646	0.5238
# seed-overlap	0	0	0	0	1.0000	1.0000	25	80	100	338	0.6486	0.5720

Statistics over the user/swarm database

	FileList.org						BitSoup.org					
	mean-1	mean-0	std-1	std-0	$P(0\|0)$	$P(0\|1)$	mean-1	mean-0	std-1	std-0	$P(0\|0)$	$P(0\|1)$
# torrents	1532	1534	879	878	0.0008	0.0011	76783	76413	7455	7945	0.0000	0.0000
# sessions	5	5	11	13	0.0000	0.0000	5	6	10	13	0.0000	0.0000
session-length-avg	23875	24220	29453	32010	0.0044	0.0050	43250	44467	69162	73118	0.0001	0.0001
session-length-sum	89047	99121	353072	288869	0.0044	0.0050	154435	199036	303931	381510	0.0001	0.0001
upload-avg	555239	446469	2873477	2461495	0.0612	0.0862	645595	473339	3662854	2595672	0.0928	0.1292
upload-sum	1708088	1553984	13465809	12424122	0.0612	0.0862	1439515	1272199	6928342	6065491	0.0928	0.1292
download-avg	565829	522494	770871	704590	0.1111	0.1236	700061	587136	1063323	948435	0.1681	0.1666
download-sum	1460824	1513681	3618277	4011277	0.1111	0.1236	1433483	1331936	2063635	1996053	0.1681	0.1666
seeding-length-avg	21219	20621	29683	32307	0.1753	0.2203	40400	40924	70350	73647	0.2651	0.2959
seeding-length-sum	86540	93964	381729	312751	0.1753	0.2203	163095	213663	327817	412933	0.2651	0.2959
seeding-avg	506295	382149	2703333	2258770	0.2359	0.2952	588845	399961	3643943	2542001	0.3176	0.3756
seeding-sum	1634034	1417370	13414912	13000970	0.2359	0.2952	1405743	1206209	7244536	6457105	0.3176	0.3756
up-speed	298	305	15368	12684	0.1303	0.2081	32	44	1682	3852	0.1408	0.2036
down-speed	377	488	9706	20826	0.1223	0.1380	204	147	17098	13396	0.1797	0.1802
# loss	1	1	1	1	0.9796	0.9693	1	1	0	1	0.9867	0.9803
# non-seed-overlap	0	0	0	0	1.0000	1.0000	3	4	3	4	0.9117	0.8802
# seed-overlap	0	0	0	0	1.0000	1.0000	7	9	13	19	0.8461	0.8213

Table IV
STATISTICS OVER EACH DATABASE. THE POSTFIXES 0 AND 1 STAND FOR THE UNCONNECTABLE AND THE OPEN CLASS, RESPECTIVELY. $P(0|0)$ AND $P(0|1)$ ARE THE EMPIRICAL PROBABILITIES OF THE VALUE 0.

density region of the speed is higher for the unconnectable sessions. In the case of *down-speed*, we also see that in the low-speed region unconnectable sessions have a higher density. In other words, low-speed unconnectable sessions tend to be slower, while high speed ones tend to be faster than low-speed and high-speed open sessions, respectively: the distribution of unconnectable download speed is broader. Note that the second bump in the Q-Q plots is most likely due to errors in the trace because it corresponds to unrealistic speeds; but it involves only relatively few outliers.

Figure 5 reveals another important difference: unconnectable users tend to have significantly more sessions, and participate in significantly more swarms. In addition, we can also see that the increased number of sessions is largely explained by participating in more swarms, since within a swarm the difference is much smaller, although unconnectable users tend to have slightly more sessions within a swarm as well.

The distribution of the session length in the two classes is very similar in the user and user/swarm databases as well (we show only the user database). The behavior of the total session length is more interesting (see Figure 5). An unconnectable user has a larger total online time, but the difference is much smaller than what we could expect from the large difference in the number of sessions. The reason is that the two attributes: *# session*, and *session-length* are not independent. Those users that have a larger number of sessions tend to have more shorter ones. In addition, within a swarm unconnectable and open users have a practically identical distribution of online time. Our results show that seeding time is correlated with online time in the aggregated case as well, so we do not discuss this attribute separately.

Let us now examine the aggregated traffic related attributes (Figure 6). In this case there is no dramatic difference between the user and user/swarm database, so we show only the user database due to lack of space. In addition, our seeding traffic related attributes are again highly correlated with the upload related attributes, so they are not discussed separately.

Quite surprisingly, the sums of uploaded and downloaded

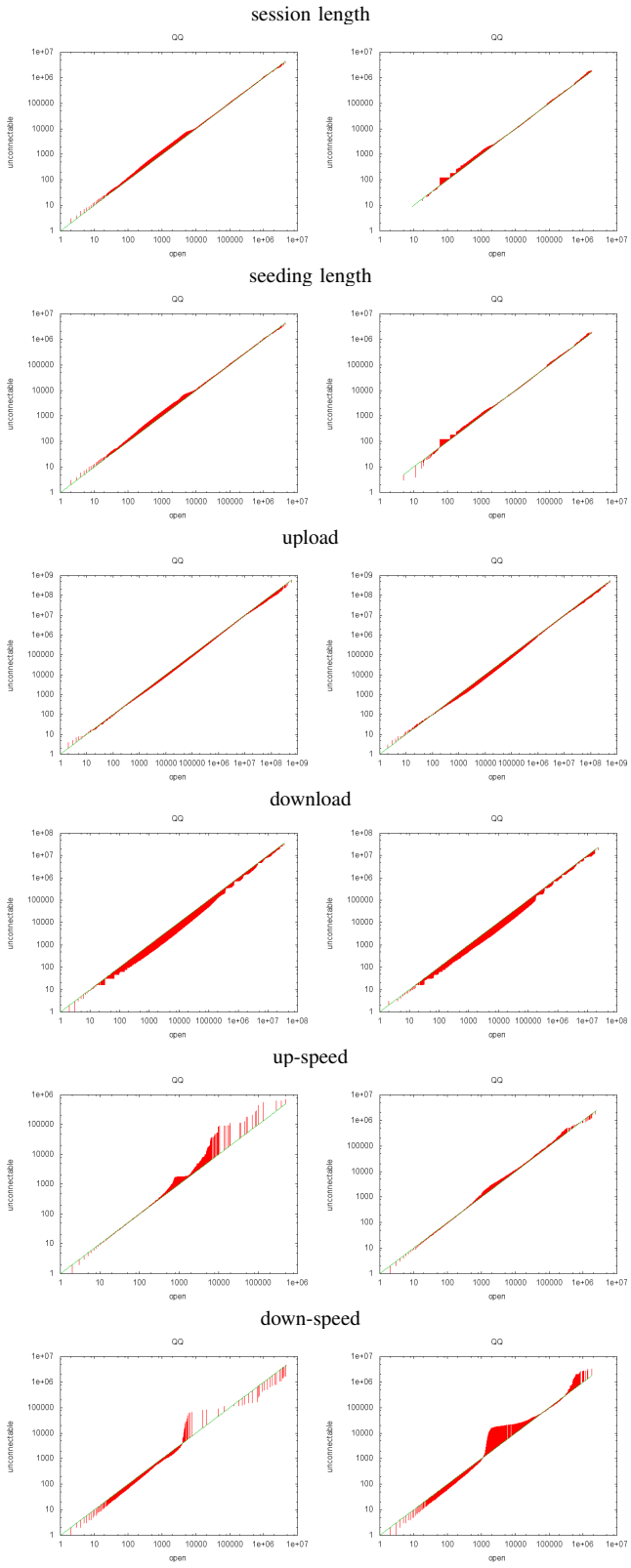

Figure 4. Session database Q-Q plots. Left column: BitSoup.org, right column: FileList.org.

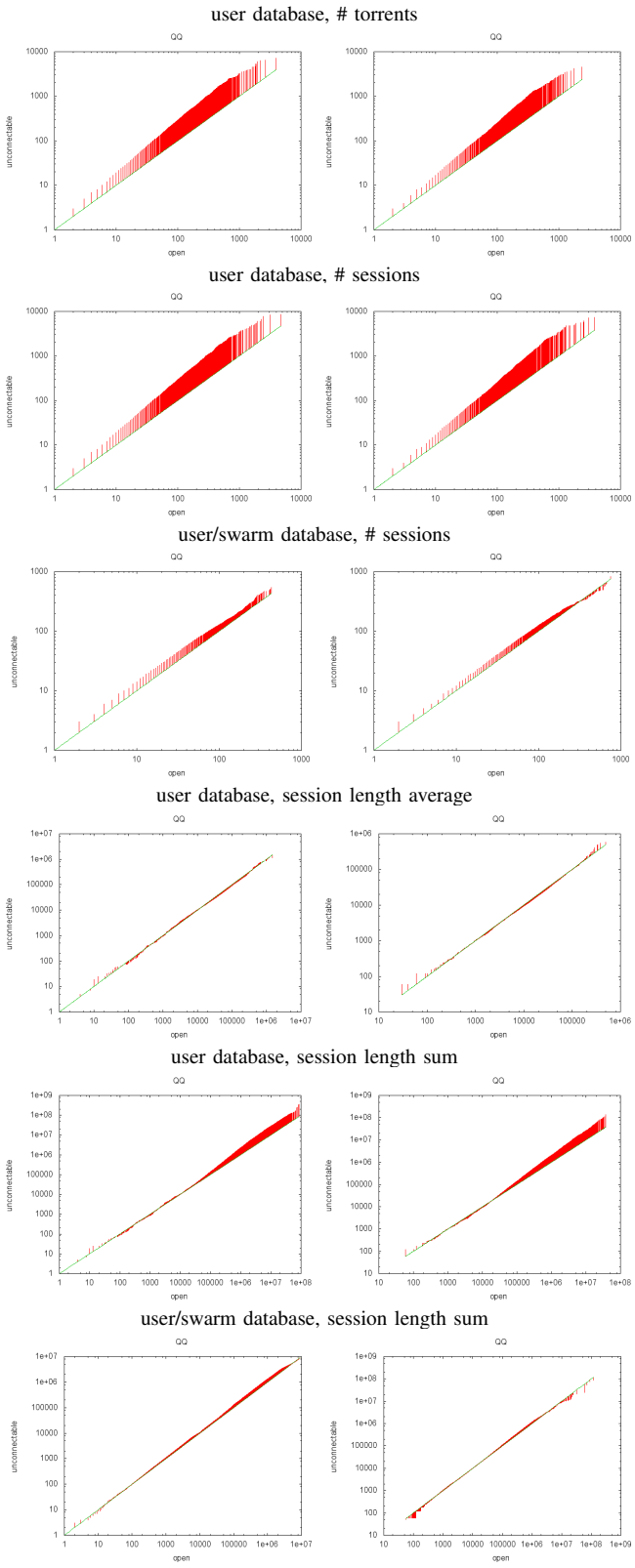

Figure 5. User and user/swarm database Q-Q plots. Left column: BitSoup.org, right column: FileList.org.

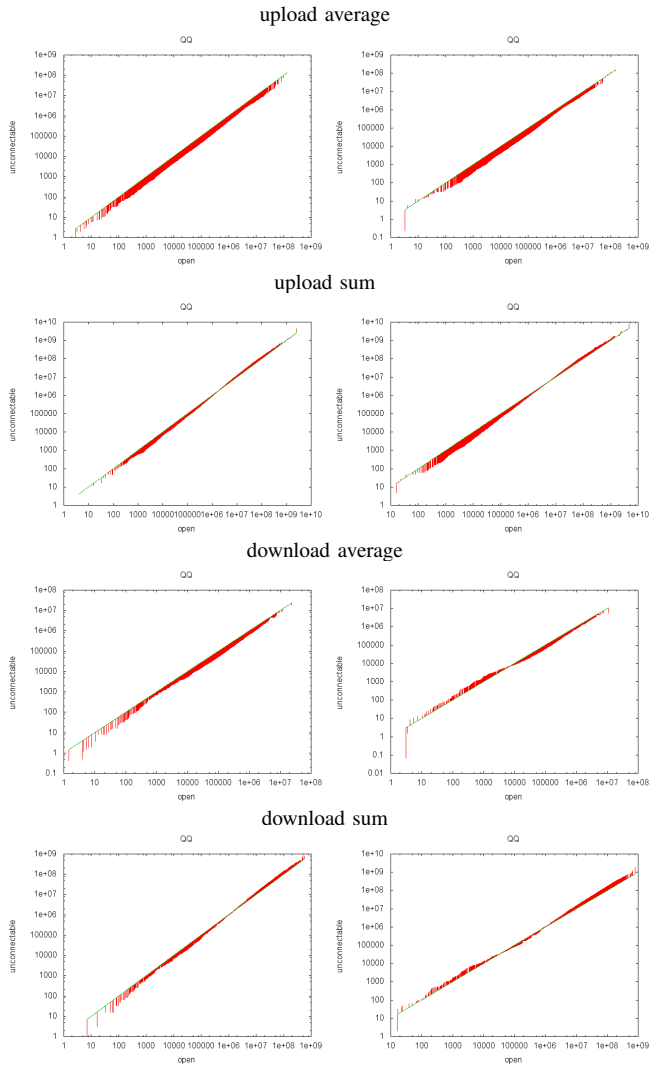

Figure 6. User database Q-Q plots. Left column: BitSoup.org, right column: FileList.org.

data have very similar distributions, despite the large difference in the number of sessions and the fact that sessions have the same length distribution. A possible explanation is that those who have many sessions tend to have smaller transfers in them. In other words, people still want to download and upload the same amount of data, only they seem to use more sessions if they are unconnectable.

Another issue worth discussing is a peculiar behavior of unconnectable users, that we found accidentally while analyzing the traces. We found that some users run sessions in parallel in the same swarm, that is, they participate in the swarm with more than one client. We speculated that this might be for at least two reasons: sharing the user ID with others, or boosting one's sharing ratio via seeding from multiple servers after completing the download. We defined attributes *#non-seed-overlap* and *#seed-overlap*, as described previously, to see how this behavior is correlated

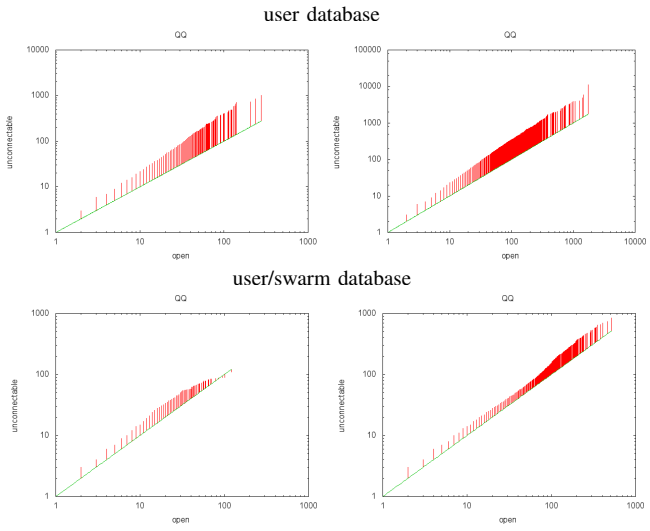

Figure 7. User and user/swarm database Q-Q plots. All plots belong to BitSoup.org. Left column: *#non-seed-overlap*, right column: *#seed-overlap*.

with being unconnectable or open. In fact, Figure 7 shows that there are significant differences between the two classes for these attributes (although note that only a small fraction of the users have parallel sessions, see Table IV). We have overlap data only in the BITSOUP.ORG trace, since in the FILELIST.ORG trace overlaps have been artificially removed.

The main conclusion regarding overlaps is that unconnectable users have a significantly stronger tendency to run overlapping sessions. In the user database this very strong effect is partly explained by the larger number of sessions, but after removing this effect (that is, looking inside one swarm) the effect is still visible.

We did not shown Q-Q plots of speed attributes for the user and user/swarm databases because they show similar patterns to those of the session Q-Q plots.

VI. CONCLUSIONS

We performed an extensive statistical analysis of the difference between unconnectable and open peers, based on two large traces from two different BitTorrent communities. We found several interesting differences. For example, unconnectable users tend to have many more sessions, and participate in more swarms, but at the same time they upload and download similar amounts of data. Furthermore, more peers have high-speed connections among the unconnectable peers than among the open ones.

We also suspect that the class of unconnectable peers is not homogeneous. For example, a company server is typically behind a firewall, and is thus unconnectable, just like a naive user behind a NAT device, who is not able (or does not wish) to configure the client properly. This could explain why we see broader distributions in speed with unconnectable nodes; but this requires further analysis.

It is worth mentioning here that our conclusions are rather different from some of the conclusions presented in [15].

For example, we found that unconnectable users are actually more active than open ones, while Mol et al suggested the opposite. The differences are probably due to the different methodology for collecting the trace: our data comes from the tracker, while Mol et al used active measurement technology, which is not ideal for approximating properties of unconnectable peers. It is also known that private trackers in general have a higher ratio of connectable peers, higher speed, and longer seeding times [13], [14], so public swarms can be expected to behave differently. At the same time, we found a strong agreement between the FILELIST.ORG and BITSOUP.ORG traces, which further strengthens the reliability of our conclusions concerning private trackers.

In this work, we examined distributions of single attributes, but in our discussion we often had to refer to positive or negative correlations between different variables; for example, between *upload* and *seeded*, or between *# sessions* and *session-length*, etc. For a complete picture, it would be helpful to apply multivariate techniques, but this is left as a topic for a future study.

ACKNOWLEDGMENT

M. Jelasity was supported by the Bolyai Scholarship of the Hungarian Academy of Sciences. This work was partially supported by the Future and Emerging Technologies programme FP7-COSI-ICT of the European Commission through project QLectives (grant no.: 231200), and TÁMOP-4.2.2/08/1/2008-0008. We thank Nazareno Andrade for providing us with the BITSOUP.ORG dataset.

REFERENCES

[1] N. Andrade, E. Santos-Neto, F. Brasileiro, and M. Ripeanu. Resource demand and supply in bittorrent content-sharing communities. *Computer Networks*, 53(4):515–527, 2009.

[2] A. Bellissimo, P. Shenoy, and B. N. Levine. Exploring the use of BitTorrent as the basis for a large trace repository. Technical Report 04-41, University of Massachusetts Amherst, Dept. of Computer Science, June 2004.

[3] B. Cohen. Incentives build robustness in BitTorrent. In *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems (P2PECON)*, Berkeley, CA, 2003.

[4] L. D'Acunto, M. Meulpolder, R. Rahman, J. A. Pouwelse, and H. J. Sips. Modeling and analyzing the effects of firewalls and NATs in P2P swarming systems. In *Proceedings IPDPS 2010 (HotP2P 2010)*. IEEE, April 2010.

[5] L. D'Acunto, J. A. Pouwelse, and H. J. Sips. A measurement of NAT and firewall characteristics in peer-to-peer systems. In L. W. Theo Gevers, Herbert Bos, editor, *Proc. 15th ASCI Conference*, pages 1–5. Advanced School for Computing and Imaging (ASCI), June 2009.

[6] B. Ford, P. Srisuresh, and D. Kegel. Peer-to-peer communication across network address translators. In *Proceedings of the USENIX Annual Technical Conference (ATC'05)*, pages 179–192, Berkeley, CA, USA, 2005. USENIX Association.

[7] A. Ganjam and H. Zhang. Connectivity restrictions in overlay multicast. In *Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video (NOSSDAV '04)*, pages 54–59, New York, NY, USA, 2004. ACM.

[8] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. Measurements, analysis, and modeling of BitTorrent-like systems. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement (IMC'05)*, Berkeley, CA, USA, 2005. USENIX Association.

[9] M. Hollander and D. A. Wolfe. *Nonparametric Statistical Methods*. Wiley-Interscience, 2nd edition, 1999.

[10] A. Iosup, P. Garbacki, J. Pouwelse, and D. Epema. Correlating topology and path characteristics of overlay networks and the Internet. In *In 6th Int'l Workshop on Global and Peer-to-Peer Computing (GP2PC), held in conjunction with the IEEE/ACM CCGrid'06*, 2005.

[11] A.-M. Kermarrec, A. Pace, V. Quema, and V. Schiavoni. NAT-resilient gossip peer sampling. In *Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems*, pages 360–367. IEEE Computer Society, 2009.

[12] J. Leitao, R. van Renesse, and L. Rodrigues. Balancing gossip exchanges in networks with firewalls. In *Proceedings of the 9th International Workshop on Peer-to-Peer Systems (IPTPS'10)*, 2010.

[13] Z. Liu, P. Dhungel, D. Wu, C. Zhang, and K. W. Ross. Understanding and improving incentives in private p2p communities. In *Proc. 30th International Conference on Distributed Computing Systems, ICDCS*, Genoa, Italy, 2010.

[14] M. Meulpolder, L. D'Acunto, M. Capota, M. Wojciechowski, J. Pouwelse, D. Epema, and H. Sips. Public and private bittorrent communities: A measurement study. In *IPTPS 2010*, 2010.

[15] J. J. D. Mol, J. A. Pouwelse, D. H. J. Epema, and H. J. Sips. Free-riding, fairness, and firewalls in P2P file-sharing. In K. Wehrle, W. Kellerer, S. K. Singhal, and R. Steinmetz, editors, *Proc. 8th IEEE International Conference on Peer-to-Peer Computing*, pages 301–310. IEEE Computer Society, Sept. 2008.

[16] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The BitTorrent P2P file-sharing system: Measurements and analysis. In *Peer-to-Peer Systems IV (IPTPS 2005)*, number 3640 in Lecture Notes in Computer Science, pages 205–216. Springer, 2005.

[17] J. Roozenburg. Secure decentralized swarm discovery in Tribler. Master's thesis, Parallel and Distributed Systems Group, Delft University of Technology, 2006.

[18] B. Zhang, A. Iosup, J. Pouwelse, D. Epema, and H. Sips. Sampling bias in BitTorrent measurements. In P. D'Ambra, M. Guarracino, and D. Talia, editors, *Euro-Par 2010 - Parallel Processing*, volume 6271 of *Lecture Notes in Computer Science*, pages 484–496. Springer Berlin / Heidelberg, 2010.

[19] C. Zhang, P. Dhungel, D. Wu, Z. Liu, and K. W. Ross. BitTorrent darknets. In *Proceedings of IEEE INFOCOM 2010*, 2010.

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