Trace analysis of Tribler BuddyCast

V. Jantet, D. Epema, M. Meulpolder

Delft University of Technology

Trace analysis of Tribler BuddyCast

Inter ship report in Computer Science

Parallel and Distributed Systems group
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology

V. Jantet, D. Epema, M. Meulpolder

31st August 2007

Abstract

Most current peer-to-peer file-sharing systems consider users as independent entities. In contrast to others Bit Torrent implementations, Tribler exploits friendship and taste similarity to increase the system usability and performance. In this paper we present a trace analyzer which highlight internals mechanisms of the novel social-based Bit Torrent protocol. Finally, we discuss Tribler efficiency by exploiting analyzer results.

Preface

This document describes my intern ship project on the subject of trace analysis of the Tribler BuddyCast protocol. The research was performed at the Parallel and Distributed Systems Group of the Faculty of Electrical Engineering, Mathematics, and Computer Science of Delft University of Technology. I want to thank the people who gave me great help during my research. In the first place my supervisors Dick and Johan, for their extensive contributions to my report and research. Furthermore, Jie for providing detailed information on Bit Torrent and Tribler, and helping me to start the live analyzer.

V. Jantet, D. Epema, M. Meulpolder

Delft, The Netherlands
31st August 2007

Contents

1 Introduction

Tribler is a Bit Torrent client, developed at the Technical University of Delft. To explain the innovations of Tribler, it is important to understand the classical Bit Torrent protocol. This section introduces peer-to-peer protocols and the classical Bit Torrent protocol which is one of them.

1.1 Peer-to-peer protocols

Peer-to-peer is a decentralized protocol for file sharing, and for all data exchange, like radio or TV distribution. In this protocol, files are not stored on a central server with a large bandwidth to allow many users downloading in same time. Instead, files are stored on users' computer until they are deleted.
Each user of the protocol is called a peer. When a user wants to download a specific file, he has to ask it directly to other peers which have the full file, or a part of it. This user, also called a leecher, can download each part of the file from a different peer. As soon as he has his first part, the user can give it to everyone who wants it. When it completes its download, it becomes a seeder, and continues to distribute the file to the other leechers.
This protocol is fast, because it is possible to download many parts of a file from many peers at the same time. The solution using a central server may be faster, but is more expensive, because the server needs a large storage capacity to store files, and a large bandwidth. Furthermore, a peer-to-peer protocol is more resistant to failure, like all decentralized protocol. There isn't a central server which can paralyze the whole system in case of a crash.

1.2 Bit Torrent specifications

Bit Torrent is a Peer-to-peer protocol which defines how to find other peers which have the file wanted.
At first, the user has to find the torrent file of what he wants to download. The torrent file contains information about the file like its name, description and hash. This step is not described in the Bit Torrent protocol; an external way should be used. Usually, users use external torrent search engines. A torrent file is less than 5 KB size, so a server can be used to store a million torrent files, and to search in them. With the torrent file and the information it contains, the Bit Torrent protocol can start.
The torrent contains the address of one or more trackers. A tracker is a server which takes a censure of peers which have or want a specific file. By giving its address to the tracker, the user becomes a member of the swarm. A swarm represent the group of users which have some parts of a specific file, or want to download it. After be identified on the server, the peer obtains a list of some other peers of the swarm. It can use this list to ask a pert of the file to each other peer, using the peer-to-peer protocol.

1.3 Bit Torrent limits

Bit Torrent is a decentralized protocol, where a user downloads a file from other peers. In its current implementation, it needs a tracker which is a vulnerable point. If this tracker crashes, peers will not be able to continue to exchange data. Furthermore, the protocol needs an outside torrent search engine, which is also a vulnerable point.
To remedy these problems, Tribler, our peer-to-peer client, uses an innovating algorithm called BuddyCast. The next section is dedicated to Tribler, a social-based peer-to-peer system.

2 The Tribler peer-to-peer client

Tribler is a social-based peer-to-peer client which uses a modified Bit Torrent protocol to allow some innovations, not included in other Bit Torrent clients. This section details some of Tribler's innovations. First, we explain why it is called a social-based peer-to-peer system [1], then we present one of its uses: the cooperative download. Secondly, we describe the BuddyCast algorithm which is an improvement included in Tribler and we details the messages it uses. Finally, we show two possibilities offer by BuddyCast: peer and content discovery.

2.1 Building a social network

In a social network, a peer should be able to recognize its friends, then to communicate with them. Tribler uses a Permanent Identifier to recognize each peer, and a global swarm which encompasses all peers to allow communication between them.
To recognize peers, using their IP address is not enough, because they are sometimes changing their address. That is the reason why each peer has now a PERManent IDentifier, randomly generated at the Tribler installation. This PermID is in fact the public-key of a public/private cryptography key pair [2]. It is used in an identification protocol to guarantee the identity of peers. Some peer identifiers can then be saved in a friend-list, in function of their similarity. A number of privileged operations which should only be available to friends can thus be introduced.
To allow communications between peers, a global swarm is introduced. In classic Bit Torrent, swarms are managed by trackers, and are associated with a unique torrent file. That makes communications allowed only with peers when they are downloading the same file. In the Tribler implementation, there is a virtual swarm that encompasses all peers that are using the system. This swarm is called the Overlay Swarm [3], and works without any server, it is only managed by peers themselves. The full list of peers, part of the Overlay Swarm, is not saved locally on a server, but distributed to each peer of the swarm. This OverlaySwarm allow peers to communicate with their friends even if they are not downloading the same file.

2.2 Cooperative Download

One of the best interest of this social network is the possibility to boost a specific download by asking help of friends [4]. They download some parts of the file by themselves, then forward pieces to the peer which asked help with all the bandwidth available.

Peers involved in a cooperative download have one of two possible roles: they are either coordinator or helper. The coordinator is the peer that is interested in obtaining a complete copy of the file, and a helper is a peer that is recruited by the coordinator to assist in downloading that file.

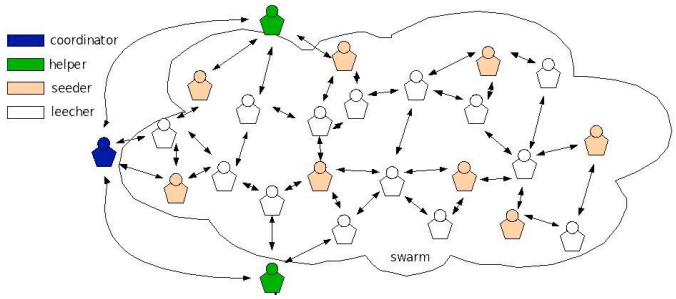


Figure 1: Cooperative Download

At first, the coordinator sends help requests to his friends and gives them the torrent. Both the coordinator and the helpers start downloading the file using the classic Bit Torrent protocol. Before downloading a file piece, each helper asks the list of missing pieces of the coordinator, and reserves one of them, preferably not already reserved. When it has successfully downloaded the part, it informs the coordinator that it got a piece and starts to download another one. Finally, when the coordinator is not using all its download bandwidth, it chooses among its helpers, and opens a standard connection with it. Connections between a coordinator and a helper are using the Normal Bit Torrent Protocol, but with the highest priority.

2.3 BuddyCast

The BuddyCast algorithm is an improvement included in the Tribler Peer-to-peer client [5]. One of its goals is to make trackers and external search engines unnecessary in the protocol. In contrast to a normal client, Tribler can work even if trackers are crashed. Furthermore, it doesn't need an external search engine to discover new content.

BuddyCast works by exchanging messages between peers, using the OverlaySwarm. These messages contain information about others peers, and torrents. Information about peers is used to find new peers with similar interest, and information about content is used to find new torrents.

Message structure

After the overlay swarm is deployed, peers are able to exchange BuddyCast messages. The BuddyCast message contains information about torrents and peers. To not saturate the bandwidth in a BuddyCast message, Tribler waits 15 seconds before sending a new message. In the last version of the BuddyCast protocol (which is the version 3), messages contain the following information:

Preferences list contains hashes of maximum 50 torrents rated by the user. You can add a rate and a textual comment to a torrent file to indicate its quality or its integrity. This rate is automatically exchanged with peers when they connect to you, but not forwarded by them to others. If the user has not enough rated torrents, some recently downloaded torrents and some recently collected torrents are included, up to 50. This list allows BuddyCast to calculate similarity between peers.

Taste Buddies list contains information of 10 similar peers. A peer is similar if it has downloaded the same files. Information about peers contains their PermID, their last IP and Port, their similarity with the emitter of the BuddyCast message and their age, i.e., the time since their last connections.

Random Peers list looks like the Taste Buddy list, but contains information about random known peers. Information about peers is the same as previously but doesn't contain the similarity ratio which is useless.

BuddyCast messages are used with two different purposes. First, they allow new peers discovery, with a high similarity, secondly, they provide new content discovery, which will interest the user.

Peer discovery

The BuddyCast algorithm can play the role of the tracker in a classic Bit Torrent implementation.
When a user wants to connect to the network, it first has to find a peer that is already connected. Of course, finding the first peer can be done by interrogating a classical tracker. But in case trackers are unavailable, other ways are necessary. It can ask SuperPeers, which are special peers, always connected to the network and using the same IP address. Their addresses are know by Tribler when it is installed. It can also ask for old peers at their previous address, and hope that they are still using the same IP and the same port. Previous statistics show that IPs are not very dynamic [6]. In further versions, it is envisioned to use other networks (like MSN network) to find peers.
After the first peer is found, the new user can request it the address of some other peers, by sending a BuddyCast message. Thus, the user collects more and more other peer addresses, and can begin to download. The interval before sending a new BuddyCast message is dynamic and can vary between 1 and 60 seconds. BuddyCast runs in four modes:

- 1 second rounds; Bootstrap mode
  Only used ”once”, the 2 minutes directly after the software is installed.
- 5 second rounds; Accelerated mode
  Used 30 minutes directly after the software is restarted. (Also following Bootstrap mode).
- 15 second rounds; Normal mode
  Default operation.
- 60 second rounds; Slow mode
  Entered when Tribler has been running for more than 24 hours.

By collecting other peers' Preferences list, a peer is able to compare these lists with its own Preferences list. Then a similarity ratio is calculated, and associated with each peer. This ratio is used in the calculation of the probability of exchanging BuddyCast message. The higher is the similarity ratio between two peers, the more often they will exchange BuddyCast messages together. That promote a fast similar peer discovery. Indeed, a social network have the Small-World property, so peers of this list are also similar peers.
BuddyCast messages also contain a list of random peers. That allows to discover new peers from other swarms, which are sometimes very similar.

Content discovery

Tribler allows users to subscribe to RSS feeds of torrent files. These RSS feeds are sometimes given by some web sites to distribute new torrent files. These torrent files are automatically downloaded by Tribler and saved into a torrent cache.
In parallel to RSS subscriptions, torrent files can be collected from other peers by the tit-for-tat protocol. If a peer discovers a hash of a torrent that it doesn't know in the list of contents exchanged in a BuddyCast message, it can ask to receive the full torrent. When new torrents appear on the network, they are quickly propagated.
All torrent files collected are indexed by their name and description, to provide a fast search engine inside of Tribler. Of course, this search engine takes a long time to find a rare torrent, but it is very efficient for well-known ones. The search engine can also sort torrents by their similarity with torrents already downloaded. This similarity rate is based on the similarity of peers which downloaded the file.

3 Analysis of Log Files

With the goal to better understand Bit Torrent users, it is important to obtain statistics and to make observations about them. There are already two scripts which produce interesting statistics of Bit Torrent users. The first one has been executed over information from an older Bit Torrent community, instead of the second one over information from the Tribler community. But both these scripts take long time to process. Analyze this information needs a few hours to obtain interesting

statistics. That's why a new analyzer was created, which allows real time analysis. This new program also uses information from Tribler's users to plot graphs.

3.1 Previous Statistics

There already exist statistics of Bit Torrent swarms. The first set, collected from an existing Bit Torrent community, contains information about how often a user changes his IP address. The second set, collected from Tribler's Log Files, contains graphs about peers, like the number of new users per day or the number of current users.

Analysis of an older community than Tribler's one

The first analysis done by S. Koolen extracted information from an existing community named "fileporn.org" [6]. This community was chosen for many reasons. Some statistics were displayed on their web site like the full list of peers currently in each of its 800 swarms. Furthermore, users are registered with a unique login, what is very important to identify them. Even if a peer changes its IP address, it is still traceable by its login. This property is indispensable to detect how often a peer changes its IP address. Unfortunately, the web site administrator decided not to publish longer time swarm information, and the analysis can't be replicated.
The methodology used was to download periodically all statistics web pages and to parse them to extract data. This information was then cross-checked with information coming from exchanges with trackers and direct connections with peers. In this way it was possible to filter accounts which are using many computers at the same time. Indeed, they are recognizable because they switch their IP address many times in a short period. These accounts are not representative of the IP change frequency.
Most important observation was that the majority of users never changes their IP address. Some users change their IP very often, sometimes many times per day. Therefore, the idea of remembering peer addresses from one day to the other is working well in the majority of cases.

Analysis of Tribler users

The second analysis, also done by S. Koolen, processes information from the log files of the Tribler SuperPeers [7]. Traces of SuperPeer processes were logged and archived to be processed by some Python scripts. A first script reads all log files and reorders information by PermID. Then the other scripts use these new files to extract information like the number of active peers at the actual date.
The most important problem was that the analyzer doesn't remember its previous results. It has to start the analysis from the beginning each time statistics files are updated. Furthermore, the volume of the log files was continuously increasing, and the duration of the script is directly linked to the volume of data. That is why a full

analysis take too long time to process and why the script could only be used for daily statistics.
An interesting result of this study is the curve of the total number of Tribler accounts for the period from March to July 2006. Dates of publication about Tribler are clearly identifiable by an impressive increase of the number of users. Unfortunately, the majority of these new users used Tribler for less than one day, and this information is not displayed on this statistics [6].

3.2 SuperPeers Log File format

Within Tribler swarms, there are some special peers called SuperPeers. These SuperPeers looks like normal peers, they download files and help other peers downloads. But more importantly, they produce log files about their actions which can be analyzed. The current version of the BuddyCast log files is version 3. These log files are the only way to understand what happens in the network. To optimize Tribler's time constants like the delay before two messages are sent, it's important to get a mathematical model from real statistics.
Standard logs created by a SuperPeer process are grouped per day. The average size of a log file for one day is about 20 MB.
In a log file, each line represents one action done by the SuperPeer process, and all actions are ordered in the chronological order. There are five different types of actions, which are logged using the following syntax:

```
<Time> CONN_TRY <IP> <Port> <PermID>
<Time> CONN_ADD <IP> <Port> <PermID> <Version>
<Time> SEND_MSG <IP> <Port> <PermID> <Version> <MessID> <Message>
<Time> RECV_MSG <IP> <Port> <PermID> <Version> <MessID> <Message>
<Time> CONN_DEL <IP> <Port> <PermID> <Reason>
```

Where:

Time is the timestamp in seconds since the epoch. It is a float with three digits precision in older log versions, and with one digit in the newest version.

IP is the remote IP address, i.e., the one of the peer when it connects to the SuperPeer. It is written in the usual string format with the point separator.

Port is the remote port used to establish the connection.

PermID is the PERManent IDentifier given to each peer. Since the IP address sometimes changes, this identifier is needed to allow peers to recognize the others.

The five logged tuples concern actions done by the SuperPeer, the connections it opens and the messages it sends and receives. They don't concern actions done by others peers, so we can know when a peer opens a connection with the SuperPeer, but not when it opens a connection with another peer.

CONN_TRY is the type of action raised when a peer attempt to establish a connection with the SuperPeer. This action is not one of the most interesting, that's why it was not logged anymore in recent log files versions, but can appear in old log files.

CONN_ADD is the action raised when the connection between the client and the SuperPeer is opened. The connection will be used to dispatch a BuddyCast message, but needs an agreement of the two peers to choose the protocol version. `<version>` then represents the most recent version of the protocol, known by the two peers, and chosen to exchange messages.
The last version of the BuddyCast protocol at the time of writing is version 3.

SEND_MSG is the action raised when the SuperPeer sends a message to a peer. `<version>` represents the version of the BuddyCast protocol used to encode the data into a message. This version should be the same as the version of the connection which was opened to dispatch the message to the other peer. The `<MessID>` is a random identifier of the message, used to detect duplication or message lost. Finally, `<message>` is the full BuddyCast message exchanged between peers. It contains the Torrent info list, the Taste Buddies list and the Random Peers list.

RECV_MSG is similar to the `SEND_MSG` action, and is raised when the SuperPeer receives a BuddyCast message from a peer.

CONN_DEL is the action raised when the connection is closed. `<Reason>` then represents the reason. The reasons can differ from one BuddyCast version to another, but two reasons are dominant: the message is corrupted because an error occurred somewhere, or the connection times out after five minutes of inactivity.

3.3 New log file analyzer

Like the log analyzer of S. Koolen, a new log analyzer has been developed. It is composed of Python scripts. A first script is in charge of reading log files, the others are processing its results to make graphs, plotted by GnuPlot. The major innovation is the possibility of the script to restart the analysis at its previous point. It can be launched every minutes to make live statistics.

Logs reader

The main script is named "log2data" and has the goal to read log files, then reorganize and regroup information. It works in three steps. At first, it selects log files never analyzed. Then it parses each line of them to count the number of peer currently active, their actions and some other information. Finally it writes results and temporary settings to a specific directory.
At first, when the script is launched, it opens the data directory to recover previous results. One of these previous results is the name of the last log file opened, and the date of the last action read. It uses this information to filter the list of log files in the log directory, by keeping only new ones. Each new log file is opened, in chronological order, and each line is processed by three different functions. The first function is named "DB_PermIds" and looks for the PermID of users. For each

requester, it counts the number of actions done and the number of times the peer changed its IP address. The second function is named "DB_Time" and processes the date to counts each type of action happened in each period of ten minutes, of an hour, of a day and of a month. The last function is named "DB_Connections" and detects a pair of opening and closing connections, to verify that the BuddyCast protocol is well respected.

Database by PermID

The function "DB_PermIds" keeps a database of each Tribler user up to date. It is able to determine the Tribler installation day, because an unknown PermID appears in the log files. For each user, information stored is as follows:

PermID is the identifier of the user, also the index of the table.
first_Time is the date of the first discovery of this peer by the SuperPeer. Therefore it is also the date of Tribler installation.
First_IP is the IP used at the first connection. It is kept only to know in which country the user is localized.
Last_Time is the date of the last appearance of the peer over the network. It is used to discover users who don't use Tribler anymore.
Last_IP is the last IP used by the peer, used to count how many times it changes its IP.
IP_Change is the number of times the peer changes its IP.
Some other information, like the number of BuddyCast messages sent and received from the peer, are also counted and saved in the table.

After the database is built, two other scripts are executed. The first script named "permid2graph" uses dates of first and last peer connections, to count the number of peers active by day. The second one named "permid2geolocal" works with the IP address to Geo-localize the peers around the world. Because the database may have totally changed, it needs to be fully read each time statistics are updated. This step is not the longest one, because the PermID database is less than 10MB in size since one year, and will grow very slowly. Furthermore, it is possible to remove very old PermID from this databases.

The script "permid2graph" opens the PermID database in the data directory to update the graph data in the graph directory. Each peer is analyzed and classified into categories. Peers which hadn't established two complete connections were excluded because they are not representative of the Tribler community. The script generates two different sets of data:
The first set organizes peers as a function of the number of times they change their IP. Table 1 in next section shows the partition of peers by the number of times they change their IP.
The second set is more interesting because it shows the number of active peers, i.e., the number of peers which have already used Tribler, and continue to use it.

There is a count which is incremented each time a new peer is discovered on the network, i.e., when a peer connects for the first time, and decremented each time a peer disappears from the network, i.e., when the peer disconnects for the last time. The result given in Figure 2 in next section shows the number of active peers during one year.

The script "permid2geolocal" tries to Geo-localize each peer, to understand where they are from. To Geo localize a peer, its IP address is needed, that is why the IP used by a peer for its first connection is stored in the database. The script which detect the users localization is very long to process. That is why it is launched over an IP address which never changes (the first one) instead of over an IP address which can change (the last one). By the way, the geo-localization is only done one time per peer.
With the IP of a peer, there are many ways to get its country, and sometimes its approximate longitude and latitude. The script asks for a server named "hostip.info". For each request, it is a bit longer to ask to a remote server, than a local database of Geo localize, but using the server is free, easy, and we have the assurance that it's always up to date. The server returns all information it founded:

- The name of the origin country, and its abbreviation in a two letters code.
- The city of the origin.
- The longitude and latitude of the IP.

It then becomes possible to count the number of peer from each country and from each city. Figure 3 in next section shows the results of this step.

Databases by period

The function "DB_Time" groups the actions by periods. There are four different periods: a month, a day, an hour, and 10 minutes. To limit memory space usage, only summaries of the periods currently in process were kept in memory. For each period, this summary contains:

- The number of new users detected.
- A count of how many actions of each type have happened.
- The full list of PermIDs met during the period.

The full list of PermIDs met is useful to know how many different users made a connection with the SuperPeer.
After all log files were read, another script named "time2graph" orders information of the temporary file, to allow GnuPlot to make graphs.

- 10 minutes summaries are grouped in a one day file.
- Hour summaries are grouped by one day file, and also cumulate in a file which presents the difference between hours of days.

- Day summaries are grouped by one month file, and also cumulate in a file which presents the difference between days of week.
- Month summaries are grouped by one year file.

These files can now easily be plotted. The results are shown in the next section.

Database for connections

The function "DB_Connections" detects pairs of opening and closing connections to measure the length of the connection. Between the opening and closing of the same connection, the script counts the number of messages exchanged. Because of the live analysis, all connections opened are not closed at the moment of the script process. That is why the script saves the list of opened connections not yet closed. Next time the script will proceed, it will read in the log files the closures of connections previously opened, and then be able to measure connection length. Figure 6 in next section shows the number of connections grouped by their length.

4 Results of the analysis

There are already more than five important SuperPeers in the network, and each one produces log files since it was started, about one year ago. For the real time analysis, a dedicated SuperPeer is running, whose logs are immediately processed by the analyzer, and not archived. Live statistics are very interesting to see the improvement done by an optimization. For the presentation of the results in this report, log files from an older SuperPeer was used. The SuperPeer with the longest period of archived log files is "jip.cs.vu.nl", whose first log file is from May 2006 and the last one from May 2007. During more than one year, log files were saved, and archived, which allows now to make a full analysis. Results are organized in three sections. The first is about the Tribler's community, the second about temporal statistics, and the third about the connections done.

4.1 Graphs of the Tribler community

During a full year period, about 85.000 different users (identified by their PermID) have made a connection with the "jip" SuperPeer. A user is not counted if he doesn't make a connection with the SuperPeer. Because of the BuddyCast protocol, peers and SuperPeers IP addresses are exchanged permanently, even if they are not subscribed to the same swarm. Furthermore, the addresses of all SuperPeers are included in the Tribler default installation, to improve the first friend discovering. That's why a peer had a lot of chance to exchange a BuddyCast message with the SuperPeer. We discovered by merging statistics from others SuperPeers for the same period, that more than 100.000 different users make a connection with one SuperPeer. This proves that not all users can be detected by the SuperPeer.

Active Users

The graph with the number of active peers in the network (shown in Figure 2) is very interesting. It shows for each day of the period the number of PermIDs which will never be seen later and the number of peers active. It tries to split users between those who still use Tribler and those who use Tribler for a while, then delete it after. The reason of peers disappearing may simply be a holiday, or more interesting, a Tribler uninstallation. However, it had the problem previously cited, that a SuperPeer is not aware of everything that happens over the network. Of the users detected by the "jip" SuperPeer, some connect only once, while others make many connections.

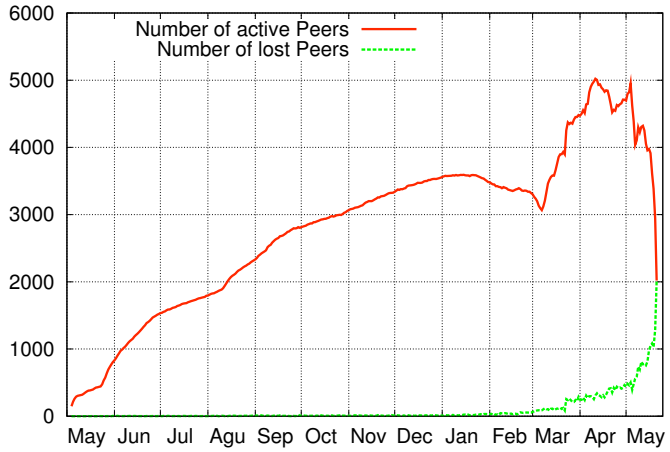


Figure 2: New and active users from May 2006 to May 2007

It is normal that the curve of lost peers grows during the ten lasts days of the analysis. Indeed, after the end of the analysis, none of the peers connects again. They disconnected for the last time some days ago, and seem not to use Tribler anymore. The curve of new peers discovered per day is similar to the curve of lost peers, and is not displayed on Figure 2.
The curve of active peers per day is the difference between new peers and lost peers. It uses the right scale, instead of the left one, because it goes higher than the bars. Of the 85,000 users discovered by the SuperPeer, we discover that there are only about 5,000 users really active. The curve goes down to zero at the end of the period, but only because we don't know if peers are still active after, and suppose they are not. The impressive increase of this curve which appears of the 8 of March 2007, is due to the update of the SuperPeer, to version 3 of the BuddyCast protocol. This new version allow faster peer discovery than the previous one, because of the random peer list exchanged, and the shorter time delay between two messages. The same update happened inside Tribler community, and each user got a new PermID. That is an explanation for the fast increase of new PermID after this date, and also for the slower decrease of the number of active peers before this date. Indeed,

many of the users who used to make connection with the SuperPeer seem to have uninstalled Tribler, whereas some new users came to replace them.

IP addresses change frequency

This analysis allows the same kind of observation as the previous analysis of S. Koolen [6]. Table 1 shows the frequency of the IP address change of users, as a function of how long they use Tribler.
In this table, it was decided to exclude peers who make less than two complete connections. It is normal that they never change their IP, since they make only one connection. It is interesting to exclude, peers which use Tribler over many computers in the same time as well. This was done for the previous statistics of S. Koolen, but wasn't reproduced here.

Using Tribler	IP never changed (71%)	IP had changed (29%)
Less than one day (37%)	11224	1288
Less than one month (43%)	8990	5667
More than one month (21%)	4005	3016

Table 1: IP addresses change frequency

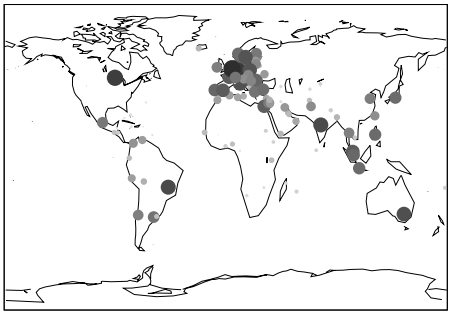
This table shows that about 60% of the Tribler PermIDs were really used. The 40% remaining were not real users, because they used Tribler during only one day. Maybe they just wanted to try it, but kept on using another Bit Torrent client.
We can also see that 70% of the users always keep the same IP address. This ratio decreases if only long term users are taken into account, but it is still up to 50% for users with more than one month of usage.

Users countries

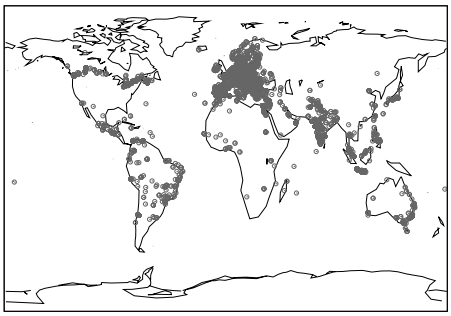
By using the IP address of a peer, we can retrieve its origin country, and sometimes its latitude and longitude. This information is summarized in Figure 3. However, this information can be erroneous. The server used is able to relate a city to an IP address in only 40% of cases. Furthermore, some Internet operators (like AOL) distribute the same IP for many countries. And finally, if the user works behind a proxy, the IP address will be the one of the proxy, and so will be the country.
Despite of all these problems, we expect that these statistics are characteristic of the Tribler's community.
Table 3a shows the first 15 most common origin countries. Of course, because Tribler is developed in Netherlands, this country is in a good position in the list, but it is not the first one. The United States are dominant, with more than a quarter of the users.
Figure 3b reuses the information from the table to present them on a map. The sizes of the circles represent how many users come from this country. Figure 3c is

Country	Users	Percent
United-States	12298	26.56%
Netherlands	4131	8.92%
England	3877	8.37%
Germany	2577	5.57%
Canada	2390	5.16%
India	1338	2.89%
Brazil	1310	2.83%
Australia	1109	2.40%
Italy	1042	2.25%
Sweden	958	2.07%
Poland	950	2.05%
France	696	1.50%
Malaysia	674	1.46%
Spain	647	1.40%
Belgium	646	1.40%
OTHERS	11651	25.17%

(a) Principal origin countries



(b) Country origin



(c) City origin

Figure 3: Users distribution around the world

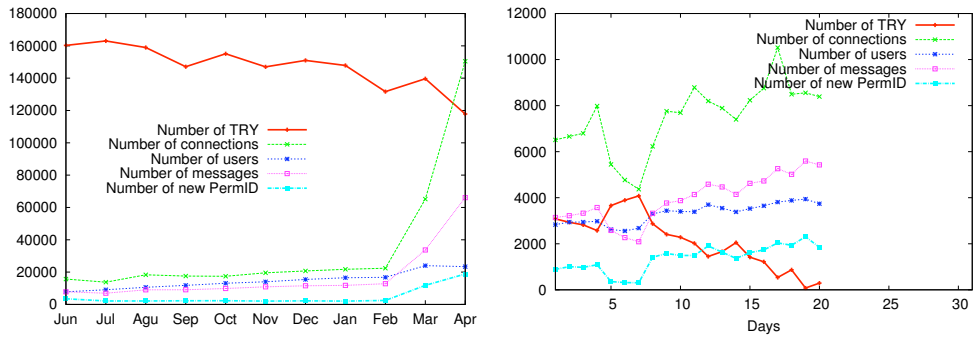
more detailed. Users are no more grouped by country, but by origin city. The high density of peers from Europe and India is clearly visible on the map. The United-States are not as dense as Europe. The explanation is that all users in United States come from only a few different cities.

4.2 Temporal Statistics

The first priority of the analyzer is to make temporal statistics (shown Figure 4), to understand what happens during Tribler's execution.

Figure 4a shows the whole period of the analysis, from May 2006 to April 2007. This period contains the date of the update to version 3 of the BuddyCast protocol, on March 22, 2007. It is very interesting to see what exactly this new version does and how it improves the protocol. We can see an increase of all curves since March, with about the same ratio. The new protocol is faster in peer discovery and message exchange.

Figure 4b is specific for the month of May 2007. It shows for each day of the month how many actions of each type were done. This month was not finished when the SuperPeer was shut-down, that's why there is no information after the 20th of May. Really interesting in this graph is the decrement of the curve of connection tries. The connection try actions, which never happens in the new version of BuddyCast,



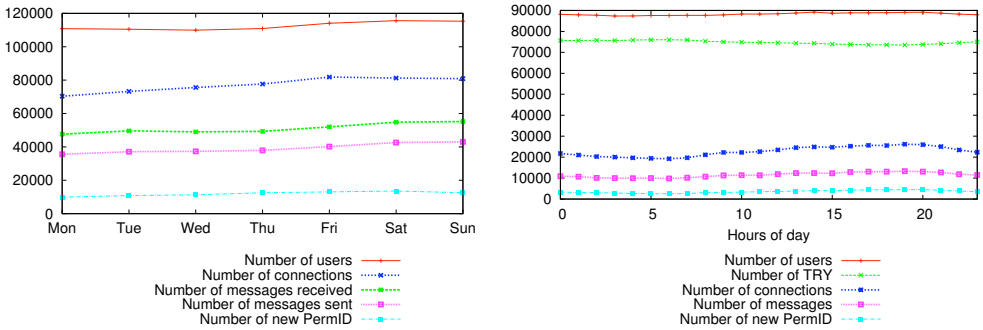
(a) Full analysis period

(b) Month of May 2007

Figure 4: Temporal statistics

show the ratio of users who still use the previous version. The less connection tries, the less users of version 2 of BuddyCast. We can suppose thats this reduction start in March, but it is not clearly visible on the graph by month.

We have also grouped information by days of week and hours of day, as shown in figure 5.



(a) Days of week

(b) Hours of day

Figure 5: Time statistics

Figure 5a shows the SuperPeer activity for each day of the week. We can observe a small increase of the SuperPeer activity during the weekend, but this is not really emphasized.

Figure 5b shows the SuperPeer activity for each hour of the day. There is a small increase of the activity on the network during the end of the afternoon. It can mean that Tribler is more used during the day, and the afternoon, than during the night. However, this increase is a bit strange because the time used in the log files is the one of the SuperPeer, which can be different from the one of the peer if it is located in another country. It would be interesting to use the Geo-localization of the IP address, to calculate the local time of the user, and use it to make the graphs.

Maybe the contrast between day and night will be more important.

4.3 Graph of connections lengths

The last part of the analysis concerns the connections done with the SuperPeer. We observe from the connections' statistics that the concept of exchanging BuddyCast messages by both peers is respected. Less than 1% of connections are used to transmit only one BuddyCast message in one way, and not the answer of the receiver. Half of the other connections are used to exchange a BuddyCast message in each way, which is the normal procedure. The other half of the connections are opened then closed without transmitting any messages.

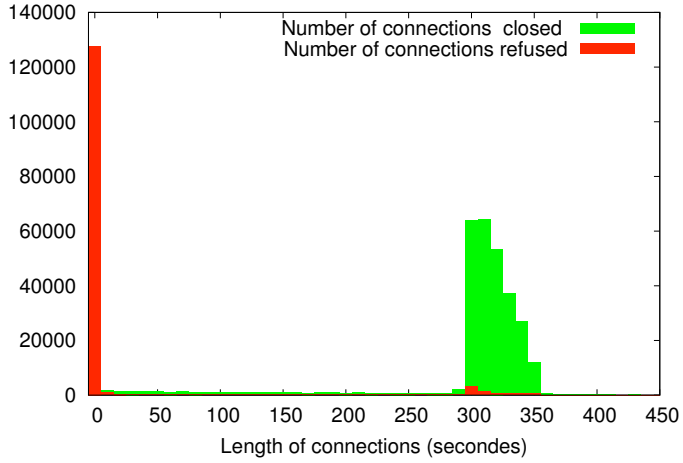


Figure 6: Connections length

Figure 6 groups connections by length and shows the number of connections in each interval. It's easy to observe that the majority of connections are between 300 and 360 seconds length. This comes from a time-out of 5 minutes (300 seconds) which closes inactive connections. Therefore, the majority of connections are active only during the first minute of their opening. If the time-out is reduced to 1 minute, the number of parallel connections open on the SuperPeer will be reduced by five, without making a difference in the protocol, nor increasing the number of connection failures.

5 Conclusions

From all this analysis, interesting remarks can be done over the Tribler community. It's a pity that statistics from Tribler client can't be compared with statistics from an other Bit Torrent client. What can be compared is the improvement done by BuddyCast version 3 with BuddyCast version 2. Furthermore, we know from previous statistics [6] that BuddyCast version 2 was already better than a basic Bit

Torrent implementation. We obtain an idea of how will work Tribler when it will be as widespread as an older client.

Bibliography

[1] J.A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D.H.J. Epema, M. Reinders, M.R. van Steen, and H.J. Sips *Tribler: A social-based peer-to-peer system*. Concurrency Computat.: Pract. Exper. 2007.

[2] A. Bakker *Permanent Identifiers in Bit-Torrent*. (February 2006) `https://www.tribler.org/attachment/wiki/PermID/PermIDs-v1-20060224.pdf`

[3] A. Bakker *The Overlay Swarm Extension*. (February 2006) `https://www.tribler.org/attachment/wiki/OverlaySwarm/OverlaySwarm-20060227.pdf`

[4] A. Bakker *Cooperative Download Extension*. (February 2006) `https://www.tribler.org/attachment/wiki/CooperativeDownload/CooperativeDownload-20060227.pdf`

[5] J. Pouwelse, J. Yang, JunWang, A. Bakker *The Buddycast Extension*. (Version 2006-05-23) `https://www.tribler.org/attachment/wiki/DecentralizedRecommendation/Buddycast-20060523.pdf`

[6] S. Koolen *IP Discovery*. MSc Thesis Tu Delft Chapter 5, 2006

[7] J. Roozenburg *Overview of Measurements and Software*. MSc Thesis Tu Delft Appendix B, 2006