The BitTorrent-protocol

A review of the technical aspects

Computer Communication and Distributed Systems

Jonas Lindquist, *820827-4954*
jonasl@etek.chalmers.se

Martin Andersson, *820511-4914*
martande@etek.chalmers.se

Page of Contents

1. Introduction

The BitTorrent protocol, referred to as the BT-protocol, is a protocol in the application layer of the OSI model used for file distribution over peer-to-peer connections. It was designed to distribute large amounts of data without the need to invoke the use of large file servers or a large bandwidth.

1.1 Glossary

Torrent – a metadata file containing the information adhering to the file or files which are to be downloaded. It holds their names and sizes as well as checksums. The torrent file also holds the uniform resource locator to the tracker. Torrent files usually has the file extension .torrent.

Tracker – a server (and the only server) in the BT-protocol. It holds information of which users have which parts of which files. A very important aspect of the BT-system is that the tracker does not hold the original file or files for download!

Peer – a user connected to a certain tracker. The user can either be a seeder or a leecher.

Seeder – a peer who has downloaded the whole file or files or is the person who initiated to first share the file or files.

Leecher – a peer who has not yet downloaded the whole file or files.

Swarm – the set of all peers connected to the same tracker.

1.2 History

The BT-protocol and client with the same name were developed by an American programmer called Bram Cohen. The idea came from a project he worked on at a company called MojoNation during the turn of the century. The idea was to chop up files of secret nature and distribute them in individually encrypted blocks between servers placed in different places. When a user wanted to get hold of the file, simultaneous downloads would take place and the secret file would be reassembled on the user's client. This meant that not only was the information hard to get hold of for an adversary but is also meant that the user got hold of the file quickly and the load was not to extensive on one of the servers compared to a server holding the entire file.

The project evolved into BitTorrent which Cohen did on his free time until he left MojoNation to work on BT full time. It quickly became popular with the Linux users around the world for sharing and distributing large open-source projects, however it was not until the word spread like a bush fire on the internet concerning the sharing of music and movies that the BT-protocol gained the popularity which it holds today.

2. The file-sharing process

First of all, we assume that someone else has a single file that we are interested in obtaining. For now, we will also assume that this someone has provided us with a metadata-.torrent-file and also has published a tracker somewhere. In this section, we will refer to him as the seeder.

2.1 The .torrent-file

The .torrent-file holds information as per below.

`announce` – a URL cohering to the tracker some where on the Internet.

`info` – holds a lot of information concerning the file. These are called keys. Some are as follows:

- `name` – holds a suggested advisory path where to save the file and what to call it.
- `piece length` – holds the size of each piece of the split up file in bytes. The file is split up into pieces in order to take advantage of the distributed system.
- `pieces` – holds a string which length is a multiple of 20. The string is then split up into substrings with the length of 20. Each of these substrings is the SHA1 hash of the piece at the corresponding index.
- `length` – only present in the .torrent-file if we are attempting to download a single file. Holds the length of the file in bytes.
- `files` – only present if we are attempting to download several files of directories.

Before looking at the content of a .torrent-file, we must understand how information is denoted. For example, a string always starts with the string length followed by a colon. I.e. `5:jonas = jonas` and `4:monkey = monk`. An integer is always padded by a leading `i` and a trailing `e`. I.e. `i3e = 3` and `i-75e = -75`. `d` denotes a set of information.

The following is an excerpt from a .torrent-file. Let's see what we can find out!

```
d8:announce39:http://torrent.ubuntu.com:6969/announce7:comment28:Ubuntu CD
cdimage.ubuntu.com13:creation datei1129187485e4:infod6:lengthi3048179712e4:name24: ubuntu-
5.10-dvd-i386.iso12:piece lengthi1048576e6:pieces!#%${[@s%/!%Y!HFG1235h1234%H#¤%h1…
```

First of all, d denotes a set { }. It holds the announce, 39 characters long, and a 28 character long comment and the creation date.

```
{"http://torrent.ubuntu.com:6969/announce", "Ubuntu CD cdimage.ubuntu.com", 1129187485}
```

Following now is the set of info keys length, name, piece length, and piece string with the file hashes.

```
{304819712, "ubuntu-5.10-dvd-i386.iso", 1048576, "!#%${[@s%/!%Y!HFG1235h1234%H#¤%h1…
```

(For means of easy reading and saving the rainforest, the hash corresponding to 304,819,712 bytes of data has been left out.)

With the information above, our BitTorrent-client can commence the download of the single file ubuntu-5.10-dvd.i386.iso.

2.2 The tracker

Once you commence the download, your software generates a 20 character long random string. This is refereed to as the `peer_id`. Your client software sends your `peer_id`, IP and port to the tracker. The tracker saves your `peer_id`, `IP` and `port` as subsets in the set `peers`. The tracker also provides this set `peers` to you and a key called `interval` which tells your software how long it should wait before requesting the set again. The later is however optional and the `peers` can be requested whenever.

2.3 File transfer

Now that you have a `peer_id` and a set of `peers`, the actual acquiring of data can begin! One peer connects to another over TCP and initiates a handshake. This starts with a decimal (,) and the string "BitTorrent protocol". Then follow eight for the future reserved bytes which all today represent zeros. Then comes the 20 byte SHA1-hash on binary encoded form and the `peer_id` and as long as both peers hold the same hash, and they both are in the `peers` set, they are good to go! As the connection is symmetrical, they share what they have with each other, as the other party requests. For good TCP performance, it is recommended that the peers keep several trailing piece requests queued one after the other. This is however not a criteria in the protocol.

When the above peer-to-peer connection has been made, you can initiate another peer-to-peer connection. This way, you can increase your download speed and share what you already have downloaded to other peers.

2.4 Peer-to-peer interaction

As we know by now, the main objective of the BT-protocol and system likewise, is to ease the load of the user with the originating file. This means that the main goal is to make the existence of the file replicate among the peers as fast as possible. The way in which this is achieved may not necessarily be a fair method towards the leechers but decreases the load on the original seeder as fast as possible. In reality this means that a peer applies two policies to select which peers to communicate with.

The first policy states that if I get a high download speed from a peer, he gets a higher priority amongst the peers that I upload to. This policy is applied to effectively punish peers that do not share enough and thus negatively affect the swarm.

The second policy states that if a peer has a high download speed from me (in other words, I upload at a high speed to him), he also gets a higher priority amongst the peers that I upload to. This policy is applied to maximise the rate at which files replicate within the swarm as a peer with a slow connection will not positively replicate the file faster than peers with a high speed connection. This method may seem unfair to peers with a slow connection but at the same time generates seeds at a higher rate which in turn positively affects the file replication speed within the swarm.

There is also another mean that the peer takes in action to achieve the main goal and that is how the pieces for download are selected. This is done by applying a method called "rarest first". This means that a peer always seeks to download the rarest piece within the swarm and thus maximising the entropy of pieces within the torrent, which results in a good distribution of pieces and a high total throughput within the swarm. When a peer has no pieces, i.e. begins to download, the "rarest first" rule cannot be applied since this would increase the latency of the download considerably. Instead a first piece is chosen at random before the "rarest first" is applied.

2.5 Making .torrent-files

At this point, you may see that this is not an issue, as the metadata-file holds very little information. Even the most novice programmer can write a .torrent-file maker as long as there are good methods for hashing the file content.

Furthermore, you need a tracker. This is not something that is very hard do set up, but there are several public trackers available on the Internet today, so setting up your own may be somewhat in excess.

2.6 Trackerless systems

As stated per above, a tracker is needed to keep track of the peers sharing a file or files. However this may pose as a problem to some users how want to distribute large files they have made themselves and do not have the ability or want to use a public tracker. For that reason there is a method called a trackerless system or decentralised tracker. This means that every peer acts like a lightweight tracker in a distributed hash table, based on the Kademlia system. This system does not give the same possibilities of monitoring and statistics as the tracker based system, nor is it as reliable and guarantees no quality of service; however it requires no bandwidth or system resources for a tracker from the original publisher of the torrent-file.

2.7 BitTorrent-client example

Taking the example from above with the `ubuntu-5.10-dvd.i386.iso` file, this is a look at a popular BitTorrent-client called Azureus.

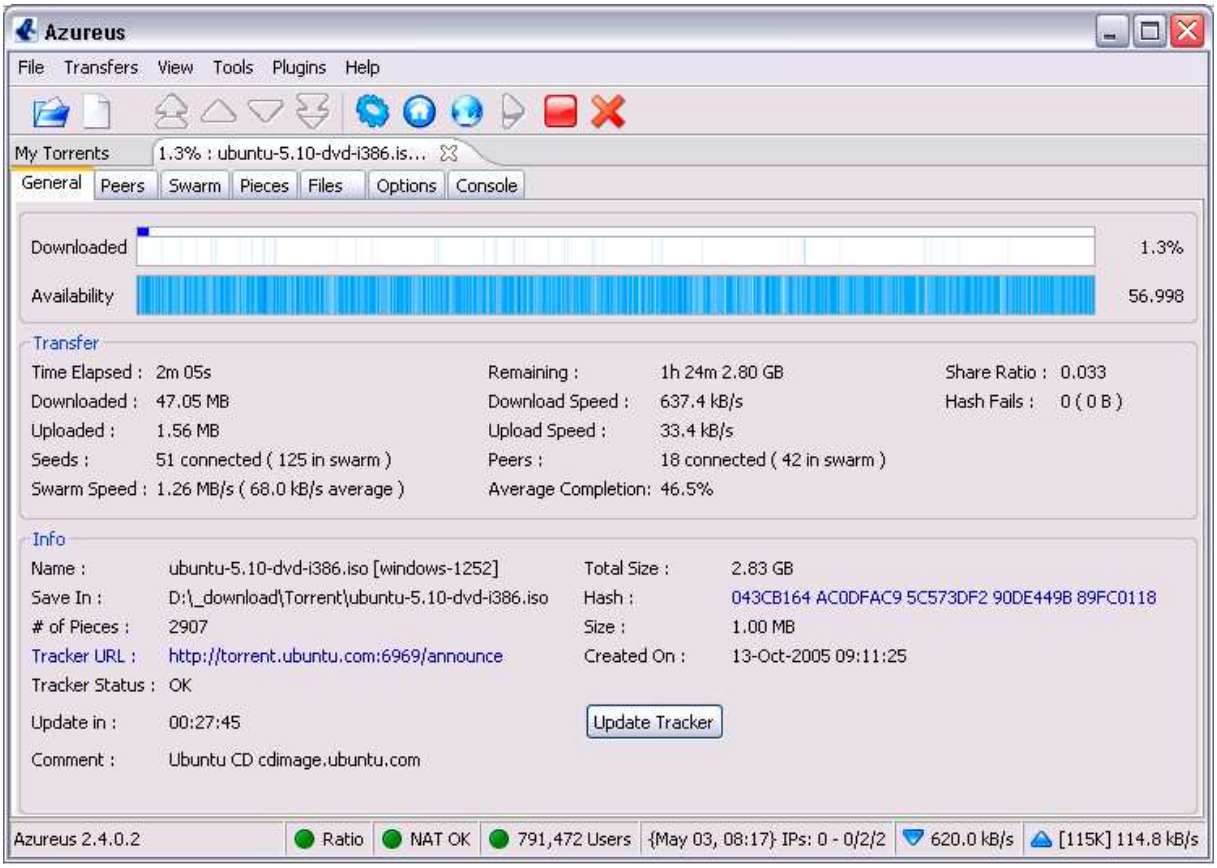

Figure 1

In figure 1 we can see that the information from the metadata-file from above has been correctly decoded. For example, the metadata-file states that piece size is 1048576 bytes, which is the same as 1.00 MB as stated in figure 1. Futhermore we see that we are connected to 69 peers, where 51 of them have completed the download. The download speed at the time of the screenshot was 637,4 kB/s which gives an average of 9,24 kB/s per peer. As one easily can understand, it means that the load on the peers from me is not so high.

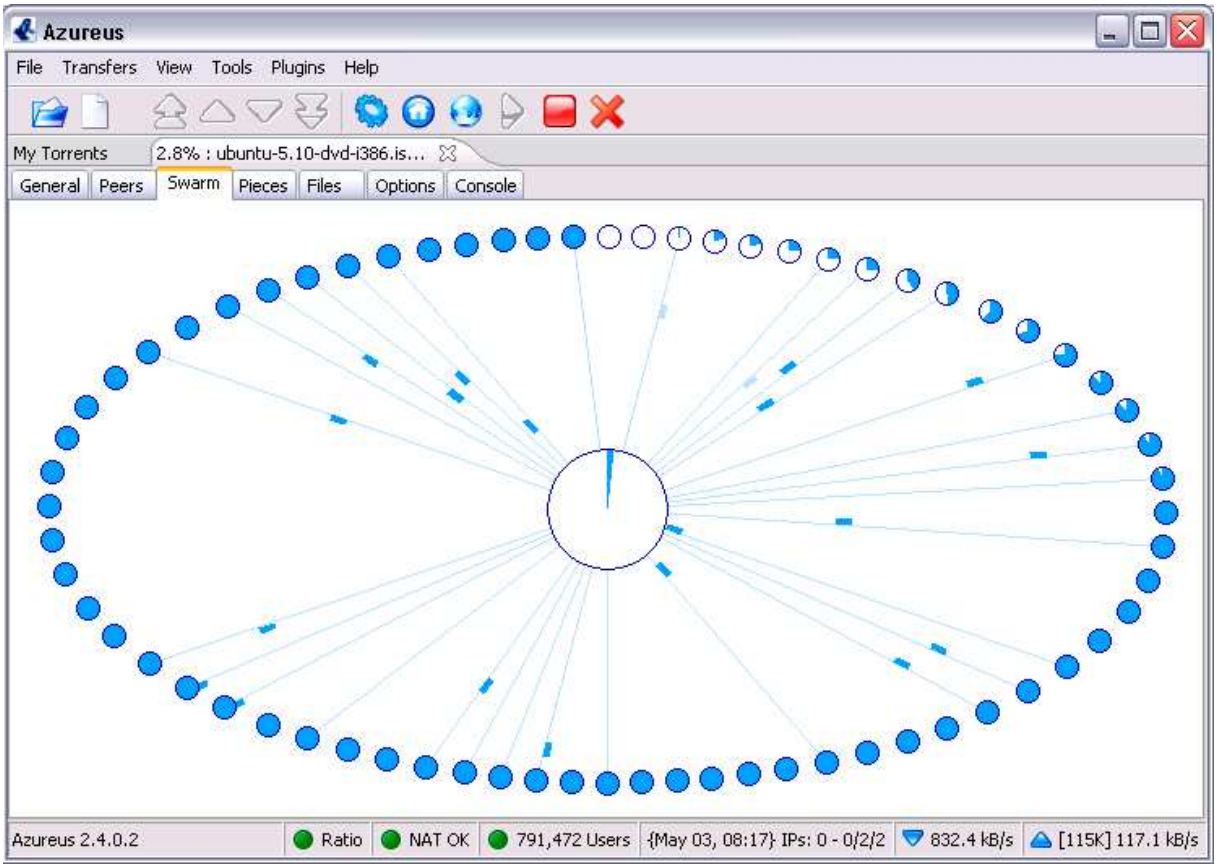

Figure 2

Above, in figure 2, is an illustration displaying the swarm of peers. The pies symbolise how much of the file has been downloaded to each peer's client. The dark blue packets indicate a downloaded piece and the light blue indicate an uploaded piece. Please note that there is a time difference between the screenshots, why the number of peers may not be the same as in figure 1.

2.8 Long term BitTorrent evaluation

We have looked at the results from a long term test done by six scientists from Eurecom who evaluated different aspects of a highly loaded torrent-file during 2003. The torrent evaluated was the Linux RedHat 9 distribution, with a size of 1.77GB. Their data consisted of logs from the tracker under april – august 2003 and a client which was online 13 consecutive hours during this time. This torrent was a good choice as it had a total amount of 180,000 peers with a peek of 51,000 simultaneous peers.

"A fundamental performance metric for BitTorrent is the average download rate of leechers. Over the 5 months period covered by the tracker log, we observed an average download rate consistently above 500kb/s. … BitTorrent exhibits good scalability: during the initial flash-crowd, the average download rate was close to 600kb/s and the aggregate download rate of all simultaneously active clients was more than 800Mb/s." [Izal]

The above quote shows that someone is able to offer a very high bandwidth without providing it himself, which is exactly the goal of the BitTorrent protocol.

3. BitTorrent and the future

A large number of media providers who want to profile themselves on the Internet, such as radio- and television stations, face the issue of limited bandwidth. Since the bandwidth needed is linear with the number of users it is serving it is virtually impossible to reach the same amount of people as broadcasted media would, as the costs of such a bandwidth would be too extensive.

Looking at current trends within the Internet community, a lot of people find the need and possibility to watch or listen to a certain programs at times of their convenience important compared to the older way of scheduled broadcasts. By means of implementing a BitTorrent-like network, media companies can provide that service without an extensive amount of costly bandwidth. This can also result in smaller stations, with a limited budget, being able to reach a wider audience giving a higher diversity and thus a tougher battle for market shares.

Taking this one step further, it may be possible to implement this in near to real-time online broadcasts in the future.

4. References

- Bittorrent.org >> Protocol Specification. Various visits, May 2006.
  - http://bittorrent.org/protocol.html

- Wikipedia, the free encyclopedia. Various visits, May 2006.
  - http://en.wikipedia.org/wiki/DHT
  - http://en.wikipedia.org/wiki/Kademlia
  - http://en.wikipedia.org/wiki/BitTorrent
  - http://en.wikipedia.org/wiki/Bram_Cohen
  - http://en.wikipedia.org/wiki/Comparison_of_BitTorrent_software
  - http://en.wikipedia.org/wiki/Distributed_hash_table

- wiki.theory.org – Bit Torrent Specification
  - http://wiki.theory.org/BitTorrentSpecification

- Izal M et Al, "Dissecting BitTorrent: Five Month in a Torrents Life". Institute Eurecom, France. 2004. Conference paper from PAM2004.
  - http://www.pam2004.org/papers/148.pdf

- Personal contact with Anders Gidenstam, Chalmers University of Technology, Sweden. May 2006.