

# Tribler Download Core Improvement

## Revised Project Planning

Rick van Hattem  
R.D.T.vanHattem@student.tudelft.nl

Raynor Vliendhart  
R.Vliendhart@student.tudelft.nl

Created: June 12, 2007  
Modified: June 13, 2007

### Abstract

This document presents a short summary of the progress made so far and the changes to the project. The introduction summarizes why these changes were made. In the two sections following the introduction, more details of the progress and the changes are described.

## 1 Introduction

After injecting the logging code into the existing clients (which took more time than expected) and measuring the download performance of several clients, the shortcomings of Tribler became clear. However, while the major problems are identified, Tribler did not perform poorly. On the contrary, it was almost performing as well as Azureus (the fastest client in the tests).

Based on these results, Johan Pouwelse suggested to change the project's focus. Instead of focusing both analysis and improvement of Tribler's download performance, we had to focus only on analysis. This analysis would include more BitTorrent clients, e.g. the closed source  $\mu$ Torrent client. The actual improvement of the Tribler download core would be future work. Reasons for delaying the improvement were also related to possible large structural changes to the core.

The remainder of this document will present a summarized form of the progress made so far, followed by a revised project description and planning reflecting the made changes.

## 2 Progress so far

The download progress measurement provided us the most insight in improving Tribler's performance. As can be seen in Figure 1, Tribler needs a bit more time than Azureus to reach maximum speed. This initial phase of the download however does not seem to have much impact on the download performance, as Tribler is only about 15 to 20 seconds behind Azureus.

Instead, the most critical phase is the end phase of the download. The progress graph clearly shows that near the end Tribler's download speed collapses, while Azureus almost retains maximum speed. The inspection and the comparison of the two client's source code gave us an explanation. Azureus uses a different piece request algorithm in the end phase of the download, the so called endgame mode. This special mode is described in [1].

While inspecting the source of Azureus, we also discovered that Azureus does optimistic unchokes between choke/unchoke rounds (which take place every ten seconds) when not all upload slots are in use. This means that when an upload slot becomes unused, Azureus does not wait till the next choke/unchoke round to fill the slots. Instead, it will check every second if it can perform any so called *immediate unchokes*. Adding immediate unchokes to Tribler might improve the initial phase of the download.

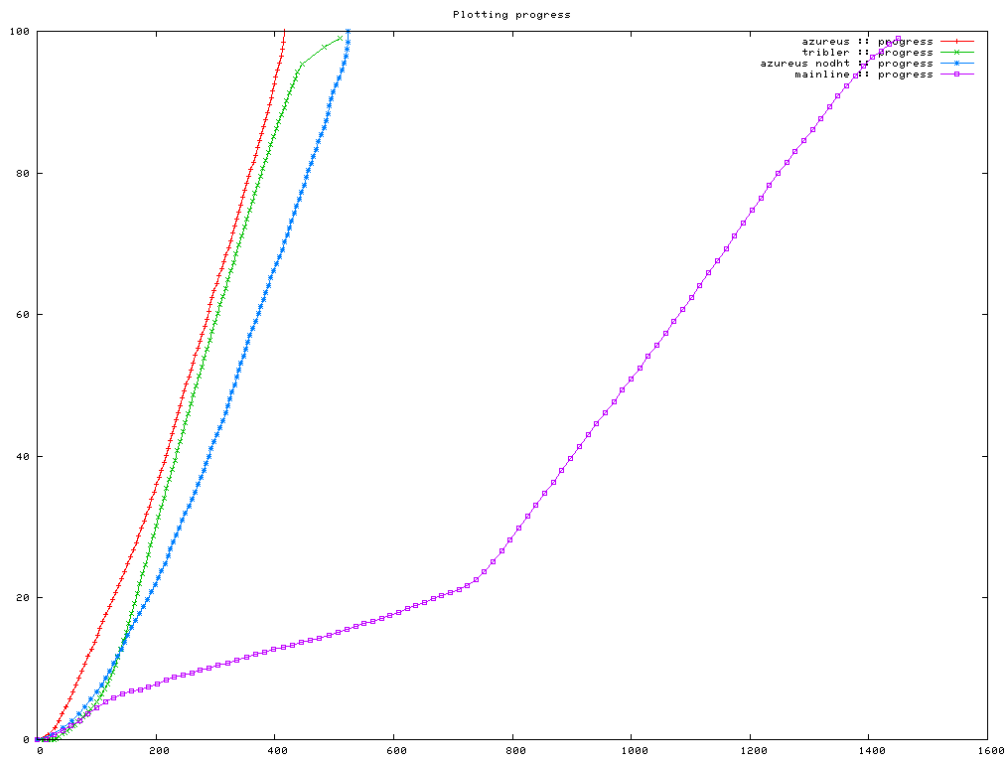


Figure 1: Progress graph

---

## 3 Project Revision

### 3.1 Revised Goal

The revised goal of the project is to analyse the performance of more clients and create a software tool for measurements.

A list of clients to be

- Tribler 3.6
- Azureus 2.5.0.4
- Mainline 5.0.7
- $\mu$ Torrent 1.7.2585 beta
- BitComet 0.89
- BitTyrant 1.1
- BitThief 0.1.6.879

### 3.2 Revised Demands

The software tool for measurements should be written in Python and should be easy to modify.

### 3.3 Revised Deliverables

The following documents and software should be produced:

- Design document of the download measurement tool
- The download measurement tool and its manual
- Analysis end report on the performance

These documents will be made available on the Tribler project site.

### 3.4 Revised Risks

The project was estimated to take about ten weeks, but there was some delay in the Problem Analysis phase caused by unforeseen difficulties in adding logging statements and performing measurements. This resulted in a one week setback.

For the remaining revised phases, we don't expect any problems that could delay the project. The design of the new software to be made is estimated to be far more simpler than the design of a new download core.

### 3.5 Revised Schedule

ID	Task Name	Week	Start	May 2007					June 2007					July 2007	
				23-4	30-4	7-5	14-5	21-5	28-5	4-6	11-6	18-6	25-6	2-7	9-7
1	<b>Problem Analysis Phase</b>	<b>1-4</b>	<b>30-4-2007</b>												
2	Reading BitTyrant Paper	1	30-4-2007												
3	Writing Measurement Plan	1	30-4-2007												
4	Performance Measuring	2-3	7-5-2007												
5	Analysis Download Algorithms	2-4	7-5-2007												
6	<b>Reflection on Analysis</b>	<b>5-6</b>	<b>28-5-2007</b>												
7	Study of Produced Results	5	28-5-2007												
8	Meeting	5	30-5-2007												
9	Replanning	6	4-6-2007												
10	Meeting	6	7-6-2007												
11	<b>Design Phase</b>	<b>6-7</b>	<b>7-6-2007</b>												
12	Draft Proposal	6-7	7-6-2007												
13	Meeting	7	13-6-2007												
14	Design	7	13-6-2007												
15	<b>Implementation Phase</b>	<b>8-9</b>	<b>18-6-2007</b>												
16	Measurement Tool	8-9	18-6-2007												
17	<b>Performance Analysis</b>	<b>10</b>	<b>2-7-2007</b>												
18	Measuring Performance	10	2-7-2007												
19	<b>End Report</b>	<b>11</b>	<b>9-7-2007</b>												
20	Study of Measured Performance & Writing Document	11	9-7-2007												

### 3.6 Revised Phases

After the old Analysis Phase, we reflected on the newly gained insights. This reflection brought us to the revision of this project. The revised project will know the following remaining phases: design, implementation, performance measurement and analysis, end report.

### 3.6.1 Design

The sixth and seventh week is where the design will take place. At the time of writing, the sixth week has already passed and was used for exploration of Python libraries.

*Documents to be produced:*

- Design documents

### 3.6.2 Implementation

For the actual implementation, we have allotted two weeks. This is one week less than the old Implementation Phase. This is because the software to be made will be a new standalone program and does not have to work within in a fairly complex, existing system like the download core.

*Documents and software to be produced:*

- Download measurement tool
- Manual

### 3.6.3 Performance Analysis

After implementation, we will perform measurements again with more clients to give us more insight and better reference material. This should not take longer than a week.

*Documents to be produced:*

- Performance measurement report

### 3.6.4 End Report

The last week of the project will be used to gather all results of the previous phases and compile it into an End Report. This report will present the gained insights and an efficiency ranking of the tested BitTorrent clients. Based on these two things, recommendations for future work will be included in the report.

*Documents to be produced:*

- End report

## References

- [1] COHEN, B. Incentives Build Robustness in BitTorrent. *Workshop on Economics of Peer-to-Peer Systems 6* (2003).