Fundamentals of Computer Networking	Project 3: Selfish BitTorrent Client
CS4700/CS5700 Spring 2011	21 March 2011

This project is due at 11:59pm on April 18, 2011.

1 Description

You will modify the trading policy of a BitTorrent client in order to improve the download performance. Unlike previous projects, the majority of your grade in this project will come from performance. Part of your grade will come from how your code performs when run against the solutions of your the course staff, and part will come from how your code performs against your peers.

In this project, you will have to do background reading on BitTorrent. Since we are providing you with a working implementation, you are *expected* to research the BitTorrent trading scheme, improvements to it, and possible attacks. *Simply turning in the code that we provide (or a slightly modified implementation) will result in a very low grade.* You should write selfish code, the goal of which is to improve your download speed.

2 Requirements

We will provide you with the code you will need to make the project functional. Thus, all you must do is increase the performance of your client. In particular, your code must

- Outperform the client provided by the TAs, in terms of download speed.
- Outperform the client provided by the TAs, in terms of the upload to download ratio.
- Outperform the the solutions of your peers.

As before, you should not assume that other peers will follow the protocol. Since these are the solutions of your peers, they may be malicious, they may upload bad data, they may lie to you, and the certainly will be greedy and selfish. Your solution should handle these errors gracefully, recover, download the file correctly, and *not* crash.

3 Your program

For this project, you will submit a complete BitTorrent program. It is recommended that you use the starter code provided by the TAs; however, you may implement a BitTorrent client from scratch if you so wish (and are crazy). If you do so, you are welcome to use a language of your choice. However, all of the code must be your own; you are not allowed to use code from other BitTorrent implementations.

The command line syntax for your sending is given below. The program takes either a filename or a url pointing to a tracker. The syntax for launching your sending program is therefore:

`Usage: java -jar snark.jar [--debug [level]] [--port <port>] [--show-peers] [--share <ip> <file>] [<url>]`

with the following details:

`--debug` (Optional) Shows some extra info and stacktraces. Argument `level` is how much debug details to show (defaults to SEVERE, other options INFO and ALL).

`--port` (Optional) The port to listen on for incoming connections (if not given defaults to first free port between 6881-6889).

`--show-peers` (Optional) If enabled, periodically prints peer information.

`--share` (Optional) Start torrent tracker with the provided file..

`file` (Required if `--share`) The file to share.

`url` (Required if no `--share`) URL pointing to .torrent metainfo file to download/share..

Luckily, the code we provide to you already does all of the argument parsing and supports these options. You should develop your BitTorrent program on the CCIS Linux machines, as these have the necessary compiler and library support. You are welcome to use your own Linux/OS X/Windows machines, but you are responsible for getting your code working, and your code *must* work when graded on the CCIS Linux machines. If you do not have a CCIS account, you should get one ASAP in order to complete the project.

4 Testing your code

In order for you to test your code, the code we provide can be used to set up a private torrent for you to test with. In order to start a torrent, you will execute

`java -jar snark.jar --share <ip> <filename>`

where `<ip>` is the IP address that you want snark to bind to, and `<filename>` is the name of the file you wish to share via the torrent. In the output, you will see the line

`Torrent available on <url>`

Leave this process running (as it is serving as the seeder). In order to connect clients to this torrent, you simply run

`java -jar snark.jar <url>`

Since BitTorrent is a scalable protocol, you can run any number of clients. When testing, it's easy to leave old snark processes running in the background. To clean these up, you can run `killall -KILL java` (note that this will kill any other Java programs you have running as well).

5 Undergraduate student version

The undergraduate version of this project can make the following simplifying assumptions: You may assume that other clients will *not* say they have pieces that they don't, and you may assume that other clients will *not* upload junk data to you (i.e., all file hashes will match).

Students enrolled in the graduate version of this project may not make either of these assumptions.

6 Shark Tank

In order for you to test your code against others', we will be providing a simulator that will periodically run your code against the code of your peers. Called the Shark Tank, it will allow you to try out new approaches to maximize your performance, to test how well your code is working, and to make sure that your code works in the test harness that we will use to grade. More details on how to submit to the Shark Tank will be provided as the course progresses.

7 Grading

Your grade in this project will be composed by the following components:

- 60% Implementation of techniques to maximize download capacity while minimizing upload required. In other words, what techniques did you employ in order to get better performance?
- 10% Performance in the Shark Tank of best version submitted on April 11, 2011 at 11:59pm
- 20% Performance of final version in the Shark Tank
- 10% Documentation and coding style

8 Advice

A few pointers that you may find useful while working on this project:

- Start by familiarizing yourself with the operation of the BitTorrent protocol. This will be covered in class, but you are responsible for learning this yourself as well. Then familiarize yourself with the code provided. You will need to modify this code, so you should take some time to familiarize yourself with it.
- Check the Blackboard forum for question and clarifications. You should post project-specific questions there first, before emailing the course staff.
- Do not, under any circumstances, test your code by downloading copyrighted or illegal content. Any such behavior is likely to result in both legal and academic penalities. Instead,

only join torrents that you create yourself, and share junk data (e.g., created via `head -c1000000 /dev/urandom`).

- If you use the snark BitTorrent implementation, note that the code is not configured to exit as soon as the download is complete. Given that you are judged by how quickly you download the file and exit, you probably want to do so. The easiest way to do this is to modify line 458 of `PeerCoordinator.java` to add a

```
    if (completed()) {
      client.interrupt();
+     System.exit(0);
     }
```

- Finally, *get started early* and come to the TA lab hours - these are held from 4:00pm - 6:00pm on Wednesdays in the lab at 212 West Village H. You are welcome to come to the lab and work, and ask the TA and instructor any questions you may have.

9 Submitting your project

You should submit your project by running the `/net/course/cs5700/bin/turnin` script. Specifically, you should create a `project3` directory, and place all of your code and README files in it. Then, run

```
/net/course/cs5700/bin/turnin <group> project3 <dir>
```

Where `<group>` is the usernames of your group concatenated together with dashes (e.g., `amislove-aghayev`, note NO SPACES), and `<dir>` is the name of the directory with your submission. If using a compiled language (e.g., C, Java, C++), you should include a `Makefile` or `build.xml` that will compile your code. The `README` should have a brief description and explanation of the approach you use, a list of properties/features of your design that you think is good, as well as examples of how to run your code.

BitTorrent resources

There is a large amount of BitTorrent literature out on the Web. A few relevant documents and papers:

- `http://wiki.theory.org/BitTorrentSpecification` Documentation of the low-level BitTorrent protocol.
- `http://www.bittorrent.org/bittorrentecon.pdf` Description on BitTorrent's original incentive mechanism.

- http://www.dcg.ethz.ch/publications/hotnets06.pdf Description of BitThief, a client which tries to never upload any data.
- http://www.cs.washington.edu/homes/arvind/papers/bittyrant.pdf Paper describing the BitTyrant client.
- http://www.cs.umd.edu/~dml/papers/bittorrent_sigcomm08.pdf Paper describing the PropShare client.