CMSC 417 Computer Networks **Fall 2024**

Final Project: BitTorrent

Assigned: November 17 *Due: December* $5, \geq 12$

1 Description

For this project, implement a BitTorrent client. Successful implementations need to interoperate with commercial/open-source ones. Your project will be graded [partly] on its performance compared to a reference client. You need to devise experiments to demonstrate that your client's performance is 'fast enough' and 'stable' in comparison to a reference BitTorrent client.

You shall work in self-organized groups of 3–4 students. If you cannot find enough peers for your group, reach out to the TAs. When you're established, one member must send an email to the TAs (CC'ing other members) which includes all full names and **directory IDs**. The email subject should be **"[cmsc417] project group"**. We will grant access to a new GitLab repository.

At the first deadline, tag [1] your main branch and submit the following in a report:

1. Decomposition, including interfaces between components
2. Who's doing what, how it's going
3. Tentative testing/experiment scheme
4. EC plans, progress

At the actual deadline, you must submit a report that details:

1. List of supported features
2. Design and implementation choices
3. Testing/measurements/experiments, especially those distinct from the demo.
4. Problems encountered (and if/how addressed)
5. Known bugs or issues
6. Contributions made by each member

2 Features

2.1 Core Features

1. Communicate with the tracker (with support for compact format)
2. Download a file from other instances of your client
3. Download a file from official BitTorrent clients

2.2 Extra Credit

If you successfully implement the core features of this project, you may earn extra credit by optionally implementing one or more features below, sorted roughly by difficulty:

1. Support for UDP trackers.
2. Support for HTTPS trackers (see [5] for an intro to HTTPS which happens to be at 417-level). The intention is that you wrap your hand-coded HTTP with a TLS library and not use an HTTPS library, but talk to us if this smells.
3. Multi-file mode.
4. A rarest-first strategy (see "Piece Downloading Strategy" in [2]).
5. Endgame mode (see "End Game" in [2]).
6. BitTyrant mode (see [3]).
7. PropShare [6] along with performance comparisons.

We may allot bonus points for other aspects that go above and beyond; post if you have questions about what could qualify.

3 Resources

3.1 Specification Information

You can find information on the BitTorrent specification in [2, 4].

3.2 Libraries

You are generally prohibited from using third-party libraries, except for a bencoder/bdecoder library of your choosing. See "Implementations" in [2] for possible bencoder/bdecoder libraries. If your standard library contains features such as HTTP, do not use that. Ask your Dr. about async features. When in doubt, post.

If you use C, you can use any data structures or libraries you like, so long as they are "equivalent" to the libraries in Python in terms of power. It's a good idea to ask, but ideally you'd have "equal power" no matter what language you choose.

3.3 Reference Client

Here are a couple reference BitTorrent clients for Windows/MacOS/Linux:

`https://www.transmissionbt.com/`

`https://www.qbittorrent.org/`

Such a client may help you understand the protocol (e.g., via packet captures) and act as baseline in your experiments.
On Debian, you should be able to install by running: sudo apt-get install {transmission, qbittorrent}

3.4 Other Resources

1. Torrent metafile details: `https://fileformats.fandom.com/wiki/Torrent_file`
2. Sample metafiles: `https://github.com/webtorrent/webtorrent-fixtures/tree/master/fixtures`

4 Grading

At the end of the semester, each group will meet with the TAs to demonstrate their implementation. Additionally, each group will discuss the information contained in their report during this meeting. Watch Piazza for further details, including a rough rubric.

5 Additional Requirements

1. Your code must be submitted as a series of commits to the main branch of your team repository. We consider your latest commit prior to the due date/time to represent your submission.
2. You may implement the project in any language(s) of your choosing, minding the restrictions on libraries in §3.2.
3. Your project must include the equivalent of a makefile – namely, a package manifest (pyproject.toml, cargo.toml, etc.), and ideally installation instructions for the language's build tools. This should be enough that we could run it in the Docker, but that is not critical for your submission or demo. Or you could yourself containerize it.
4. If setup/use is non-obvious, your project should have README(s) with instructions, and/or accomplish the same in UI.
5. You are not allowed to copy code from any source.
6. Your reports must be placed inside the repository root, named "{Checkin, Report}.{pdf, txt, ipynb, html, tex, etc.}", with instructions/build if need be. Try not to make it run too long!

References

[1] Tags - GitLab. `https://docs.gitlab.com/ee/user/project/repository/tags/`.

[2] BitTorrentSpecification - Theory.org Wiki. `https://wiki.theory.org/BitTorrentSpecification`.

[3] BitTyrant. `http://bittyrant.cs.washington.edu/`.

[4] The BitTorrent Protocol Specification. `http://www.bittorrent.org/beps/bep_0003.html`.

[5] The HTTPS-Only Standard. `https://https.cio.gov/`.

[6] Dave Levin, Katrina LaCurts, Neil Spring, and Bobby Bhattacharjee. BitTorrent is an Auction: Analyzing and Improving BitTorrent's Incentives. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 243–254. ACM, 2008.