Code Progress Report

Ryan El Kochta `relkocht` Andrei Kotliarov `akotlia1`

December 5, 2022

1 Summary

Since the last report, we made solid progress on individual components. The file I/O interface has been restructured a bit to fit our needs better, but it is basically fully functional. We can submit downloaded blocks to it, which are centralized in a File struct. We have the logic to figure out whether a certain piece is completed, and which ones still need to be requested. The SHA-1 hashing is also functional, providing protection against corruption. We have tests which operate using the tempfile crate, which verify all of the above properties. This was primarily written by Andrei.

We have also written much of the message encoding/decoding logic. We can read a `.torrent` file and compute its info hash, using the serde and bendy crates. We have structs and easy-to-use interfaces for this as well. There are also tests that verify that this component works. The encoding/decoding for torrent files was written by Andrei, while the encoding/decoding for tracker responses was written by Ryan.

Much of the tracker interaction code is working as well. We have a fully functional HTTP/1.1 client, complete with a test that makes a connection to the provided Flatland tracker. It encodes the URL path and all of the appropriate query parameters, sends a request over a raw TcpStream, and receives the response. It returns it in a convenient HttpResponse data structure. The wrapper around this that deals with tracker requests/responses, built on the HTTP/1.1 client, is also mostly working. It is able to send a request and receive a response. This was written by Ryan.

Finally, we have started the multithreading work, although we'll admit not as much work is done here as we'd hoped. We can spawn peer threads and a special thread for accepting connections. We have created a MPSC channel for the main thread to communicate with the threads (send requests to them, receive responses, etc). The multithreading work was split pretty much equally between Ryan and Andrei. Andrei also wrote a convenience wrapper around the libc poll() function, which we may end up using in the peer threads.

2 Notes

We did switch around some of the assignments. Originally, Ryan was to do the file I/O components and Andrei was to do the HTTP/1.1 client. This was reversed.