CMSC417 Final Project Report

Members: William Yuan, Tyler Luk, Hunter Alabrodzinski

1. Supported Features

- Download: Parse .torrent files, contact tracker, multi-peer downloading, SHA-1 piece verification, persistent disk storage
- Upload: Listen for connections, send bitfield, implement choking/unchoking, respond to PIECE requests
- Protocol: BitTorrent handshake, message parsing, length-prefixed framing, peer state machine

2. Design and implementation choices

- **Modular design** with separate components for download, upload, and peer management
- We decided to create a bit torrent that had **two different states** while open, either downloading or uploading. It was really difficult to implement Bittorrent to send pieces while also downloading them from peers. So we just made it so that it would turn into the seeding state after completely downloading the torrent file.
- **Data Structures:** PieceBuffer array stores in-memory pieces during download/seeding; Peer tracks connection state and choking status
- **Key Decisions:** Keep pieces in RAM for seeding speed; use select() for non-blocking I/O; enforce 16KB block size limit; max 4 concurrent unchoked peers
- Implemented **peer mode** which allows a client to download exclusively from a peer (passed into arguments)

3. Testing/measurements/experiments, especially those distinct from the demo.

- We tested each part of our code by using a .torrent file and then checking that it correctly worked the way we implemented.
- For example: when we made the parser for the torrent file, we had a struct that stored the info from it, and we tested this implementation by outputting what the struct contained and seeing if it matched to what we expected.
- We added a lot of log messages and printf statements to understand where in the process we were failing
- We used test files/scripts to test various modules of our client such as contacting the tracker or seeing if our client was accepting new connections in the seeding phase.
- To measure our download speed, we would store the file size as well as how long the download took and calculate our average download speed (not displayed but for testing purposes).

4. Problems encountered (and if/how addressed)

- The torrent was really slow. We realized it was because we coded our client to iterate synchronously through the handshakes and then started requesting pieces from peers one at a time. In order to make it faster, we started requesting multiple pieces at once to make it faster, which worked.
- Another problem we encountered was that we were really struggling to get the uploading part of Bittorrent client to work. For some reason it always disconnected from peers right after completing the download and wouldn't be able to receive anything from any new peers who would try to contact our client. We did multiple tests including using wireshark to see the error. So far we were unable to figure out the issue.
- Another problem during the upload phase was that there would be random mallocs or segmentation faults. To resolve this, we used gdb and valgrind thoroughly to find where our program was crashing and to see if we had any memory leaks.

5. Known bugs or issues

- Common issues that we ran into were not being able to tell how fast our Bittorrent was at downloading, this was because the peer list for the torrent file varied at times and we were unable to get a consistent reading if our bittorrent was fast enough
- An issue we believe is occurring is that we are unable to be put on the tracker's peer list due to the port not being seen on the internet. We could not figure out how to get it visible, we suspect it is something to do with WSL or the way docker hides addresses
- If you wait too long after being prompted to enter the seeding phase, the program may exit or malloc. We don't really know why, but it is consistent.

6. Contributions made by each member

- Hunter Alabrodzinski: Coded the parser that reads the .torrent file and also the functions that create the GET request for the tracker. He also worked on parsing the message returned from the tracker and updating the struct to be used later on to contact the peers
- Tyler Luk: Worked on connecting the peers from the peer list and getting the pieces from them and verifying that the pieces were valid. He worked on making the torrent faster download wise. He also worked on the main function that brought everything together.
- William Yuan: Worked on the uploading part for Bittorrent and also wrote the pieces that were received into disk. He also updated the bit field. Also worked on main.c as well as debugging parts over the whole project.