Jacob Rizer

151004895

Phase 2 Write-Up

Description of Program:

The RUBTClient class contains my main method which reads in the torrent file and filename of the file that I will save to my local drive. I read the torrent file into a byte array and then create a torrentInfo object with this byte array so that I can obtain the announce_url and other metainfo items.

Now in my main method I create a Torrent object. The Torrent class is a class I wrote for the bulk of the operation. It contains info for the trackerURL which is the URL I must generate in order to send a GET request to the tracker. It parses the metainfo from the torrentInfo class to obtain this information. In order to get the info_hash, I had to decode it using the Bencoder2 class provided. As for my peer_ID, I generated a random 20 byte ID and to ensure that it does not begin with "RU" I hardcoded the first byte to be the character 'x'. After assembling this URL I make an HTTP GET request to the tracker and store the map that it sends me in my Torrent object. With this map I can get the list of peers by calling get() with constant ByteBuffers that I hardcoded at the top of the class. I then generate a list of peers, which I also store in the Torrent class. The list of peers only contains the peers that have the prefix "RU" in their peer_ID. Now I call the method getFastestPeer() where I ping each peer 10 times and calculate their average RTTs. The peer that gets the fastest RTT is the one that I download from. I first send a handshake message and parse the response to ensure that the peer ID and info_hash match. I then send an intested message and get the unchoke message in return. Upon receiving the unchoke, I send a request to the tracker with the "event=started" field. I now enter my while loop where I continuously send requests to the peer in order of piece indices. Each piece requires two blocks because the block length is 2^14 bytes and the pieces are double this. For each request sent, I receive a piece message back with the raw data at the end that I write to the file. Upon finishing the download, I print out the time that the download took, send the completed message to the tracker, and close all connections.

The last class I created was the Peer class which simply allows me to create Peer objects to populate my list of peers with. The Peer class only contains data: the peer_ID, IP, port of the peer, as well as a boolean for whether it is unchoked or not.