Anthony Castronuovo(ajc320), Jake Taubner(jdt97)

RUBT Phase 3 Write-Up

Brief Overview:

This program works first by basic checking to see if the command line arguments are valid. Once the checks have been performed, the valid torrent file is decoded/parsed using the TorrentParse class to gain access to the metadata. Once the parsing is complete, we call the Tracker class to perform the actions to first call the tracker, and receive the initial GET request. Once the tracker gives us the response we want, in the form of the peer list and other information, we call the Peer class. Peer deals with all of the connections to the peer. We send the appropriate messages to the peer, get the appropriate responses back, and call multiple threads to download and upload, and then put together the file. Once the Downloader class finished downloading we return those bytes to RUBTClient, where the file is created according to the second command line argument, and the bytes written to that file. The file has been downloaded by this point.

Classes:

-- RUBTClient: This class holds the main method that calls the other classes in order to run the program fully

-- Peer: This class holds all the functions that deal with interacting with the peer. From here, we call the Downloader class that deals with messages relating to the peer, and downloading the pieces

-- Tracker: This class is responsible for interacting with the tracker. The initial get request is done here, as well as updating the tracker with the event status.

-- TorrentParse: This class is responsible for the parsing/decoding of the torrent file. This class calls the given classes, specifically Bencoder2 and TorrentInfo, to parse the torrent file

-- Downloader: This class is the class that deals with all the downloading. From Peer, this class is called as a thread so we can download multiple pieces at the same time

-- Uploader: This class is the class that manages all the connections coming from outside peers that wish to download the file. If and messages are out of order, or do not correspond to the correct messages that should be sent at that time, the connection to the peer will be closed. These are also called as threads. Once from the Peer class, and once inside the thread, more threads are called when appropriate

-- Pause: This manages the pause feature. When pause is called the program will cease downloading and wait for user input to either continue, or end the downloading and exit the client. This is called as one thread that runs alongside all other threads so that it can be called anywhere, or anytime while the client is downloading a file.