Christopher Rios
Ben Green
Manuel Lopez

Project Overview

The ‘main’ method of the program is contained in RUBTclient.java. Arguments taken in are the torrent filename and the desired output filename (if a file already exists with that name, the user is asked if they should overwrite).

The torrent file is parsed into a TorrentInfo object. Once parsed, the extracted info_hash, our randomly generated peer_id, uploaded, downloaded amounts and amount left are composed into a GET request and sent to the provided Ip and port, that that is also extracted from the .torrent file.

The response to the GET request is decoded and stored in a Response object which contains an array of Peer objects.

After a valid peer is extracted from the list of peers, RUBTClient creates a new Peer and Destfile object.

RUBTClient gets a handshake message from Message.java, passes it to our Peer who then handshakes with a valid peer on the list. Once we receive a handshake back, and validate the info hash sent by the response tracker and then after we receive the bitfield, we send an interested response, assuming the peer has any pieces we want, composed in our Message class.

The peer then waits until an unchoke is sent, or disconnects if it is not given one after 2 minutes. Once an unchoke is sent, our Peer class sends a started message to the tracker, and creates request messages using our Message class. It then sends a request for each piece that we need, split into two blocks. Once a successful piece message is returned from the peer, we make the data,offset and piece number into a piece object.

As each piece is generated, its hash is verified against the piece_hashes array in the torrentinfo object. If the hash matches, the RandomAccessFile writes the piece’s data to the proper location in the file. Once all pieces are downloaded and verified, a completed event is sent to the tracker from RUBTClient.java, followed by closing the filestream. If one of the pieces are hashed and do not match, peer catches this and closes connections and sends a stopped event to tracker.

Class descriptions

GetRequest
The GetRequest class manages the communication between the client and the tracker. The constructUrl() method takes in the required information from torrentinfo and makes the url that will serve as the get request for the tracker to return a list of peers. GetRequest also manages the creation of the random alphanumeric peer_id for the client, as well as the event messages that are sent to the tracker when the client has begun, finished, and stopped the download process. GetRequest is called by the main method in RUBTClient.java

Peer
The Peer class manages the communication between the client and the peer, including the socket connections and input/output streams. Peer takes in data created from Message objects and sends them through the output stream to peer and reads in the data on the input stream. Peer manages the proper responses to each reply from the host, such as handling any chokes the peer might send. When downloading chunks, the peer handles combining them into a full piece and hands them off to destFile for verification

Message

The message class handles the composition of the different messages that will be sent to the peer, stored as a byte array. This classes methods are called by RUBTClient.java to compose the initial handshake and interested message, and Peer.java uses it to compose request messages to send through the socket.

Piece

The piece class builds a piece object, containing the piece data, piece number, offset. A piece is created every time peer gets a successful piece response. Once a piece is made its hash is verified by DestFile, and is then added into our destination file to compose the final download.

DestFile

This class manages a RandomAccessFile used to create the final output file as well as a torrentinfo object for reference. Methods include: addPiece to write Piece data, generateBitField to make bitfields more accessible, and verify, which verifies a Piece's data hash against the hashes within the DestFile's torrent info.

Response

This class analyzes the tracker’s response to the initial GET request and extracts the bencoded dictionary containing peers, intervals, etc. Functions to print the peer list in a readable form and to select peers with the ‘RUBT’ prefix and IP are included.