Group 22
Manan Doshi - 145000508
Mehul Doshi - 145000507
Christopher Klinchik – 141002235

Classes:

Peer – Stores peer information in one object so that it may easily be accessed. It stores the port, IP, and the peerID.

Message – Stores the messages sent to and received from the peer in one object. It stores the actual byte message, and also the message type for easy reference.

Downloader - This class extends Thread, and is a thread that downloads from a peer. The constructor takes in a peer, . For Stage 2, we only download from one peer, but multiple Downloaders can be made to download from multiple peers. It utilizes the ReentrantLock class to lock and unlock, so that multiple threads are not accessing the same arraylists/methods at the same time. The downloader first sends the peer a handshake. After receiving the peer's handshake, it verifies the hash, and begins messaging by sending repeating 'interested' messages and 'request' messages. This is done in Downloader's download() method, and loops until all pieces are downloaded in the run() method.

Uploader - This class extends Thread, and is a thread that uploads to a peer. We unfortunately took so long to get the downloader working, we do not yet have the uploader working. However, we will have it ready in time for the final stage. The idea is the same as downloader, which is that we create an Uploader thread for each peer that wants to connect and download from us. Then, the Uploader's run() method listens to the peer and returns 'piece' messages at their request.

RUBTClient – This is the main class that does all the client/peer connections and communication. After taking in the .torrent and .mov file name in arguments, it opens and decodes the torrent file using Bencoder2. It then sends a GET to the tracker and decodes the list of peers. It chooses the given peer that has the file, and creates a Downloader thread to download it. This class also has methods like whatNext(), parseMessage(), parseHex(), and setHex() which both threads can use for the messaging part. The method addPiece() takes the ArrayList of downloaded pieces (which is a global variable) and puts them all into the file with the given file name. The global int numActiveThreads tracks how many threads are still active, and only stops running once all started threads are inactive.

Class Interactions:

The RUBTClient interacts with Peer in both the extract() and download() methods. The Peer object stores peer information, and uses it in the download method to create a socket and establish a connection to the peer.

The RUBTClient interacts with Message using the message() method, after the socket connection and handshaking with the peer is already done. The message() method is used to easily call the correct message that is going to be sent to the peer. For example, message(1,2) returns an 'interested' message.

The RUBTClient also interacts will all of the given classes. It uses Bencoder2 to decode the .torrent file, and uses BencodingException to catch errors that may occur with the Bencoder. The TorrentInfo class is used in the download() method. The method takes a torrentinfo object in, and can then access all the information and parse it. Finally, the ToolKit is used by RUBTClient to print out byte arrays and ByteBuffers.

The RUBTClient interacts with Downloader and Uploader when messaging. After getting the peer list, it creates Downloaders and Uploaders to interact with the peers, and only ends once all the Downloader/Uploader threads are no longer active.

Downloader and Uploader both interact with Peer. The constructors for both require a Peer, so that it knows which peer to download from or upload to.

Workload Distribution:

Manan - Made Uploader and Downloader class. Worked on big download method, helped with hex encoding and messaging. Created and implemented locking. Created sockets and input/output streams for peer messaging.

Mehul - Did most work in big download() method, whatNext(), addPiece(), and main(). Created the ArrayList that stores the pieces, and got it to put it all together and make the .mov file. Did most peer communication work.

Chris - Did the whole handshake() method, most of tracker communication, extract() method, Went through and organized some of the code and also added some more exception handling. Made the Peer class for easy storage of peer information.