Richard Gerdes

Rahul Purwah

Robert Williams, III

CS 352: RUBT Phase 2

The Program:

The way that our program works is that the RUBT client takes in 2 command line arguments as per the assignment guidelines. These command line arguments are the torrent file and the file where the object which we receive will be downloaded to. Then the file where the download will happen is set up and the byte array is initialized for the file's data. This is where the core of the program begins. A new torrentInfo and Tracker type is initialized using the command line input. A Peer type is also initialized. After this the program goes into a while loop. Inside of this while loop is where we retrieve our list of peers from which we will be downloading the object. The while loop also checks to make sure that the IP address for the peer equals either 128.6.171.130 or 128.6.171.131. Our program also has implemented error check for if the peer is unable to connect to. We print out information regarding the peer once the connection has been successfully achieved. The other classes besides RUBTClient are all there to support the main program which is being run through RUBTClient. Peer is a class which is dedicated to each individual peer and implements all the different functions which are needed to connect to a single peer. PeerManager class is there to manage all of the peers as a group. The Tracker class gets the list of peers from the torrent file which is given to the program as a command line argument.

Classes:

RUBTClient: The Driver that runs the program. This takes in the command line arguments that are given by the user. This class sets everything up for the program to run correctly.

Peer: This is a class which sets up each individual peer. This contains methods which are used to communicate with the peer such as the handshake.

PeerManager: This class is designed to keep track of all the different peers which me must communicate with.

Piece: This class is used to implement a type "piece" which contains a byte array of information and other methods which are implemented to use this information stored in the byte array.

Tracker: The tracker class performs the HTTP Get requests and obtains the list of Peer IPs that the user is supposed to attempt to connect with. It then creates Peers.

Message: The message class contains all base code for the generation of the byte[] message that will be sent to the other tracker. While the message class performs validation checks for the Handshake, it leaves the checking of pieces to the Piece class.

Feedback

We felt that we were a bit more comfortable coding phase 2 since we learned a lot from the problems which we faced during phase 1 but we still had a decent amount of problems. One of our major problems was the handshake connection to the peer. For some reason our program was unable to connect despite many efforts. We finally got it to run by stripping down our program to find what the issue was. One of the Issues ended up being that we were trying to reconnect after a failure too quickly which caused us to

fail again. We were stuck in this vicious cycle for a while. We did not know of this issue since our documentation did not warn us about such a situation. Another issue was how to multi thread the connection to peers so our program can continuously make sure that the peers are working as per our needs.