Computer Networks: Final Project - JBJ torrent

Jeffrey Lai, Benjamin Tong, Jefferson Zou

May 10, 2016

1 Project Overview

For this project we implemented a simplified torrenting program. In place of a torrent file, we have a more streamlined and simplified protocol, with the initialization done when we first communicate with the server. The server keeps track of the files being torrented and the clients trying torrent each file. In order to have a file listed on the server, at least one client needs to seed it (as we want to guarantee for the sake testing/demoing that we can complete the final transfer). Each client first communicates with the server to get its neighbors. We assume that the contents being sent are not corrupted or malicious.

2 Techniques and Design

To establish peer to peer connections we use Java's built in java.net Sockets, which use an underlying TCP protocol. On top of this we defined our own protocol which is used to send control messages and mark data. Our goal was to try different strategies for behaving in the peer to peer network to maximize performance. Thus we chose the rather small piece size of 256 bytes so that we could have more pieces without having to transfer enormous files for testing and demoing.

For our server, we had a Hashmap keeping track of the different files being torrent, and associated with each file was a list of clients. We considered making the server more modular and multithreaded, however for the purposes of testing, we were not going to run more than a dozen clients at a time, and the server is not frequently talked to, thus the server did not need to be extremely performant.

For our client, we also kept a Hashmap for each of the files that we were torrenting. We broke the client down into two halves, each with its own logical task. One side was the listening half, which dealt with incoming messages and responded according to our response strategy. The other half dealt with initiating messages as dictated by our requesting strategy. This was done in a generic fashion so that new strategies could be added easily.

3 Problems and Possible Improvements

We ran into issues with trying to construct good strategies for the network. In the end we have two strategies (basic and random) which are built into the client. However a larger project would include a suite of strategies, as well as automated test scripts to create and destroy clients throughout and execution to simulate stress conditions. One issue we ran into was that we printed out a lot of information regarding what was being sent e.g. control messages, data, etc. This artificially slowed down the file transfer. The problem was that we wanted to monitor how the file transfer progressed, so removing these print statements is contrary to the goals of the project. The print statements can be turned off by editing the Debug file and turning off the `_debug` flag. Overall there is a lot more to be explored for this kind of project, however it is outside the scope for a project of this size.