Jonathan Chu 804141479

Eileen The 104141878

Joshua St. Clair 904187688

# Project 1 Part 2: BitTorrent Protocol

This project was designed to allow us to learn about the BitTorrent protocol, a popular peer to peer (P2P) protocol. This differs from a Client-server protocol, which just downloads from a single server. This allows a person to instead download pieces from peers, offsetting the total load to the server. In addition, this allows for a faster download speed for the person acquiring the file.

The overall design of the project is as follows: The person trying to get the file downloads a torrent file, which gives it information to grab from a tracker. This tracker acts as a way to acquire metainfo. After communicating with the tracker, the tracker responds with a list of all of the peers that are also interested in the file. These peers will have downloaded some pieces already downloaded, which can then be requested from them. After getting this peer list, the downloader sends a handshake to each of the peers, connecting with them. After receiving the handshake, the downloader then sends back a bitfield: a number of bits that tell what pieces it currently has. At the beginning, it will start will all 0’s. The peer replies with its own bitfield. The downloader then decides whether it’s interested in what the peer has or not; if it is, it sends an interested signal to the peer. If the peer is willing to send over more information, it will send an unchoke back; this means that the downloader can start requesting data. The downloader then sends a request for a piece, which the peer will then attempt to send back.

Since this occurs with every peer, it is possible to request different pieces from each peer. The way we implemented this was to keep a global integer that kept track of what the most recently requested piece was; this allows us to avoid requesting more than one thing at once. In addition, we periodically talk to the tracker to let it know our current status. This allows the tracker to ensure that we are still a peer. The entire premise of this protocol is that not only do we act as a downloader, but we also act as a peer. This means that other people will be able to request things from us, as well as send us interested, which we send back with an unchoke. As we receive more pieces, we send HAVE packets out to all of our peers, so they can keep track of our bitfield in case they are interested as well. Finally, we check whether the piece is valid using a checksum.

Our actual implementation used object orientation. For each peer that we got from the tracker, we created a peer object, with information such as a socket, its bitfield, etc. Then we used these parameters to decide whether or not we were interested: if we are, we send an interested request to them. We then switch off to the next peer and do the same thing. Once we figure out with whom we are interested, we can then request from them using our global integer mentioned earlier. We continue to ask different peers until we finally get the entire file. Ultimately, we end up receiving all of the bits, which we then tell the tracker.

The project was split up. Eileen mostly worked on the tracker request and getting all the information, as per part 1. In addition, she worked on the hash function which was used to verify whether a piece was actually valid or not. Josh worked on the object orientation of the peer, as well as populating the peer list. Most of the algorithm with regards to how we should request things and in what order was also his responsibility. Jonathan worked on sending the bit field, as well as figuring out how to parse and send messages.   In addition, he worked on sending the packets of the HAVEs, and figured out how to request the next piece correctly.

This project was rather daunting, and we were unable to finish as much as we were hoping to.  We were unable to figure out how to multithread correctly, which made it so that a lot of the downloads didn’t run fast enough for the test cases.  In addition, the tracker never responded with what we expected, so we ran into issues regarding getting the proper upload/download numbers.  We send them, but no response is ever given.  While we may not have passed many test cases, this project has taught us all a lot about peer to peer systems, and we now have a thorough understanding of how they work.