Final Report

Supported Features

- Connection to the tracker through an HTTP GET request
- Extract data from the bittorrent metafile and place it into a data structure
- Connection to other Bittorrent clients using TCP
- Able to download pieces of data from a bittorrent file and reconstruct those pieces into a singular file

Design and Implementation Choices

- Included data structures for torrent file, tracker request, tracker response.
- Peer list to store information for each peer such as peer_id, interested, choke, bytes received, have, ip address and port.
- Started with making a request to the tracker to get an initial list of peers.
- Tried to connect to peers received from the tracker and added those peers to our data structures if the connection was successful.
- Made an infinite loop where we poll for incoming connections.
- Polled for incoming data from existing peers. The data could be a handshake or message.
- Handle each message accordingly.
- Send periodic requests to the tracker to get potential new peers.
- Only unchoke peers that unchoke us first
- If a peer fails to connect or send a message to us in two minutes, our client closes the socket
- Rarest first algorithm. Find the 3 rarest pieces of the peers that unchoke us. Send the request for them to the peers.

Problems and Solutions

- Socket used to talk to the tracker needs to be bound and the listening socket also needs to be bound, but only one socket can be bound to one port.
    - Use setsockopt to make the socket able to bind to a port that's already bound.
- Tracker response size was unknown, so did not know how much memory to allocate.
    - Came up with an implementation to increase the allocated buffer and read response incrementally and does not need to know size in advance.
- Cannot download last piece successfully
    - Turns out the length field of the request message exceeds the length of the piece. Use file length, number of pieces, and piece length to find out the length of the last piece.
- Issues with sending and receiving the bitfield correctly
    - Originally, we were sending the bitfield to the clients then getting no responses. The bitfield we assumed to be the problem, but from our understanding it was correct (empty bitfield). The problem ended up being the padding of the structure

we were sending. We set the padding of the structure to one that created the desired outcome.

Known bugs

1. The hash is incorrect when combining the pieces of the bittorrent data. We believe this is due to the way we are reconstructing the data, rather than the way we are receiving and parsing the data.
2. Can only successfully connect to a full seeder. We are still unsure why this issue occurs and if we had more time we would have looked further into it. This may have something to do with the structure of the for loop or an unintended exit from the connection-to-peer loop.
3. Since sockets for talking to tracker and listening for connections are on the same port, if there are incoming connections when the socket is waiting for tracker response, there will very likely be an error.

Contributions

- Evan
    - The program structure, the talking with peer logic, the bencoder implementation, Sending and Receiving data from peers based on the message type we received or want to send
- Mohamed
    - Reading torrent file, using bencoder implementation to decode torrent file, get info from tracker using HTTP, using bencoder implementation to decode response and store relevant info in data structures.
- Josh
    - Data structure, functions to manipulate data structure, talking to peer logic