# BitTorrent Project README

## Implemented Features

### Downloading

The client is able receive a torrent file and retrieve the files specified by the torrent file from multiple peers simultaneously.

### Resume Downloading

The client is able to resume downloading any previously interrupted torrent files given that the same torrent file and same targetting directory is provided again.

### Downloading Multiple Torrents Concurrently

The client is able to receive commands as torrent files are downloading, and therefore supports downloading multiple torrent files simultaneously.

### Downloading Status

The client can also show which torrent files are active and display their current status. Status includes the download progress, approxiamted speed of download, number of peers connected and time elapsed since beginning of the download session.

## Design Documentation

### Core Components

- TorrentFile
    - The TorrentFile object encapsulates information of an actual .torrent file. It keeps track of all the pieces of the torrent file, as well as a bitmap of which piece has been completed
- Piece
    - The Piece object represents a piece specified in the BitTorrent protocol. It contains all blocks within the piece, as well as hashing and SHA1 information regarding whether the piece has been correctly downloaded.
- Block
    - The Block object is used to encapsulate the actual payload data with a single block.
- PeerConnection
    - The PeerConnection object encapsulates state and connection between the client and one of its peer. It uses TCP sockets to communicate the BitTorrent protocol with the connected peer in order to send and receive messages. The PeerConneciton object spawns two thread after initialization to handling sending and receiving. It uses Queues to talk to the main client thread to receive download jobs and passing back downloaded pieces.
- Client

- The client waits and receives user input to start downloading torrent files. Upon receiving a new torrent file to download, it spawns a new thread and starts the download process.

### Initialization and Connecting to Peers

Upon starting a new download job, the client talks to the tracker server to retrieve detailed information of the torrent. It then calculates indices and construct the pieces and blocks of the torrent without the actual payload data.

Information of peers are also retrieved from the tracker server. The client then tries to connect to each of the retrieved peers to initialize a handshake. Once the handshake process is completed with the peer, the peer is treated as a functional peer that the client can download data from.

### Downloading Strategy

Once a peer is fully connected, the receiving thread of the PeerConnection will listen for incoming messages. One of the first message received would be the BITMAP message specifying which piece does the peer have. The bitmap is recorded within each of peer connections respectively.

The downloading thread looks at all missing pieces for the torrent file being downloaded, as well as all connected peers and their available pieces. It then distributes missing pieces among connected peers via synchornized Queues, only if the peer has the piece in question. After completing the piece, the peer passes back downloaded data to the downloading thread via a Queue.

The downloading thread also maintains which peer is downloading which piece. If a peer does not complete a piece within a speicified period, the download thread will redistribute this piece as a a retry.

Each downloaded piece is verified against the SHA1 hash retrieved from the tracker server. All pieces on disk are verified again when the torrent file finishes.

### Resume from Pervious Download Session

Before downloading of a torrent file starts, the client scans the download directory and uses the SHA1 hash to verify completed data on disk. It then calculates and downloads only the missing pieces. This implies that the client does not re-download any data that have been previously downloaded to the disk.

## Performance

The downloading thread sends out multiple REQUEST messages to its peers simultaneously. Each peer connection can have at most 20 un-answered REQUEST messages in-flight.

The observed speed of the client is approximately 300Kb/s for each functional peer it connects to. For the given test torrents, the clients finished the bub_ht torrent is about 40 seconds, the Hamlet torrent in about 1000 seconds and the InPraiseOfIdleness torrent in about 150 seconds.