Omar Khalil (ok77)

How to run the program

In the directory containing all files, type:

```
> javac RUBTClient.java
> java RUBTClient test1.torrent <save-file>
```

High level description

RUBTClient is a simple BitTorrent client designed for peer-to-peer sharing and allows to exchange data using the BitTorrent protocol. In this program, a torrent file is taken in as its first argument and its data is passed to the tracker as a TorrentInfo object. Using this information, the tracker creates a URL and sends an HTTP GET request; the response is then decoded using a Bencoder2 decoder and a peer list, along with their information, is returned.

After calculation which peer had the best average RTT, the program connects to the fastest peer and sends a handshake message. The peer replies back with an equal message to begin communication with each other. The client sends an interested message and waits until the peer sends an unchoked message. Once the peer sends an unchoked message, the download begins, and every piece, after it is verified of its SHA-1 contents, is saved onto a new array, which is then saved to a file named by the second argument.

Class description

RUBTClient

> RUBTClient opens and reads the torrent file onto the tracker. The tracker then returns the peer list to this class, where it takes the fastest peer and creates a new thread for that peer to run. It also creates a new thread for the tracker to run and get updated during download.

Tracker

> Tracker generates a random peer ID for the client to use and connects to a port ranging from 6881 to 6889. This class creates a URL from the torrent file metainfo and sends an HTTP GET request. The response is decoded using Bencoder2 in which it creates a peer list. While the program is running, the tracker is updated at each interval (capped at 180) taken from the metainfo.

Peer

> Peer establishes connection with the fastest peer, which is found by computing the average RTT time of each peer. The client sends/receives a handshake to the peer to begin communication. The client sends an interested message and waits until the peer sends an unchoked message. After receiving an unchoked message, the download begins and each piece is verified of its SHA-1 contents and saved onto a byte array. If the client stops the download and then resumes later, the program will read the save file, if it exists, and load downloaded pieces onto the new byte array. The index where the download stopped from before is returned and the program continues to download the rest of the file at that index. Every two minutes, a keep alive message is sent to the tracker to prevent peers from closing connections.

Message

This class contains all types of messages for the peers to send/receive. Each message, other than the keep alive message, is composed of a byte ID and an integer length, which is converted as a 4-byte big-endian. Types of messages include: choke, unchoke, interested, uninterested, and keep alive. There are other types, but those have their own class because they require a payload.

Have

This class is an extension of class Message, in which its payload is the zero-based index of a verified and downloaded piece.

Bitfield

This class is an extension of class Message, in which its payload is composed of a single byte array containing a single bit for each piece.

Request

This class is an extension of class Message, in which its payload contains three parts. The index is an integer specifying the zero-based piece index, begin is the offset within the piece, and length is the request length of the piece (usually 16384 bytes).

Piece

This class is an extension of class Message, in which its payload contains three parts. The index is an integer specifying the zero-based piece index, begin is the offset within the piece, and block is a block of data specified by the index.