Omar Khalil (ok77)

<u>How to run the program</u>

In the directory containing all files, type:

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

<u>High level description</u>

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.

The program connects to 3 peers from the peer list, where each peer replies back with an equal message to begin communication with the client. 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. Download is done simultaneously from each peer, resulting in a faster download time.

While the program is downloading the file, the program is also waiting for new connections from peers that want to download from us. The program connects to new peers, verifies handshakes, sends pieces downloaded already, and waits for an interested message. If an interested message is received, the program sends an unchoked message to the peer and awaits request messages. Once a request message is received, the program takes in its parameters and sends the piece to the peer.

<u>Class description</u>

RUBTClient

RUBTClient opens and reads the torrent file onto the tracker. The tracker then returns the peer list to this class, where it creates a thread for each peer to run. It also creates a new thread for the tracker and server to run. The program will only run RUBTServer if the file is already downloaded. This class waits for the user to input “quit” to stop the program, in which the program terminates and saves any piece downloaded onto a file.

RUBTServer

RUBTServer creates a serversocket to listen for incoming connections from peers. Once a connection is made, it creates a new UploadPeer class for that connection and runs it as a thread.

UploadPeer

Opens the socket for the connecting peer and receives a handshake. The handshake is verified and the program sends back another handshake generated by the client. Pieces already downloaded are sent to the peer and sends back an interested message if the peer is interested. The program unchokes the peer and reads incoming request messages. A new piece message is created from the parameters from the request message and the piece is uploaded to the peer.

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. If min_interval is found, then the tracker uses this instead. Otherwise it uses half the interval found.

Peer

Peer establishes connection with each peer found from the tracker 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. If the user stops the program, downloaded pieces are saved to a file and once the user starts the program again, it resumes on the same index it stopped on.

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.

Utils

Contains several methods such as checking if the file exists and returning the index to continue download, creating a handshake message, and verifying a piece.