BitTorrent Phase II

Min Chai(netid - mxc3), Terence Williams(netid - trw63)

November 1, 2016

1 RUBTClient class

We take a torrent file and parse the data inside using the given TorrentInfo class. Then, we connect to the tracker with a randomly generatedID and get the tracker info. Bencoder2 was used to decode the tracker info. From that we make a peer list, taking those that begin with the peerID '-RU'. For each peer, we calculate the RTT and save the index of the minimum average RTT. To calculate the RTT, we ping the peer with InetAddress class 10 times and use the system time. After building the peer list, we perform a handshake to the remote peer with the lowest average RTT to try to download the file. Upon successful communication with the peer, the file will be written into a path specified by argsv[1]. RUBTClient creates a new fileoutputstream before contacting the peers. Contacting the tracker with the completed download and stopped event is also done in this class.

2 Peer class

The Peer class represents a peer retrieved from the peer list inside tracker info. Each peer is constructed by its peerID, IP, and port number. The Peer class is also used to perform handshaking. Necessary fields likes output streams and input streams are stored inside this class. SHA-1 verification is also done inside this class(which utilized the MessageDigest class). This Peer calls on the PeerMessages class. As pieces are being downloaded and SHA-1 verified, pieces are written to the specified download path. Keep Alive messages and have messages are sent here. These are implemented in the PeerMessages class. **For every successive 10 pieces downloaded, a verification message is printed out to keep the console unclogged.(Piece 10, piece 20, piece 30...piece 510). This does not mean that the last pieces are not downloaded, THEY ARE.**

3 PeerMessages class

The PeerMessages class implements the messages needed to contact the peer. This class has access to the input and output streams between the local client and the remote

peer. The Peer class uses this class to try to download pieces of the file. A peer calls PeerMessages.start() and PeerMessages.showInterest() to start the download. Choke, Interest, Keep Alive, Piece, Have, Request messages implementations and other utility functions can be found here.