Rohan Patel

Project Phase 1

Description:

The RUBTClient takes two parameters on start up. The first is a torrent file and the second is a file that will be created and populated with data downloaded from peers. The given torrent file is then decoded using the Bencoder2 class in the GivenTools package. The information from the TorrentInfo class is passed to a Tracker object.

The Tracker class contacts the tracker using a HTTP Get request with the started parameter and the tracker returns a response. This response is decoded using the Bencoder2 class and is checked for failure. If the response passed this failure check, the peer list and interval is extracted. The peer list is searched for a peerID with prefix –RU11 (or –RUBT11). If found a Peer object is created using the ip, port, and ip recovered from the list.

The Pinger class is used to calculate the fastest RTT. The peers found are passed to this object's method. The getLowestRTT method pings each supplied peer 10 times and calculates the average RTT. It sends back the peer with the fastest time. This peer is used for the download.

The peer class is used to connect to the peer and download the pieces for the torrent file. A TCP connection is opened by creating a socket with the given ip and port. Next, the handshake is sent to the peer. The responding handshake is verified using the hash within the torrent file and peerID. After verification, an alive message is sent to the Peer, and a AliveTask object is created. This object is a subclass of the TimerTask class and is used to periodically send alive messages to the peer. After this the getFileFromPeer method is called to start the download of pieces.

The getFileFromPeer method first creates a file using the given parameter from the user. It then creates a Message object. This object is a messenger that sends all messages to the peer(choke, unchoke, etc). After we connected to the peer, the peer sent us a BitField which we ignored for this phase of the project, since it had all the pieces. To start the download, we sent an interested method. Once the peer sent back a unchoke message, we can begin requesting pieces. The peer requests pieces in sequence beginning at index 0.

Before requesting it creates a Piece object. This object is used to maintain the blocks that are received from the peer which make up the piece. To request a piece, the Peer class uses the Message object to send a request message for a given piece. This in turn results in a piece message being sent by the peer. These piece messages are passed to the Piece object to create the full piece. Blocks within the Piece class are represented by Block objects which contain two fields: byte[] block and int offset. The byte array is the data, and the offset is the position in the full piece where this block begins.

Once the full piece has been received. The piece is verified using a SHA-1 Hash. If it is not verified the piece is requested again. If it was verified, a have message is sent to notify the Peer of receipt of the piece. After this message is sent, the piece is written to the file, and then the subsequent

piece is requested. Note: Pieces are only requested if the peer sent the unchoke message. If the choke message was received at any point, requests stop until unchoked.

After all pieces have been received and written to the file, all streams are closed as well as the socket. The last thing that needs to be done is send a HTTP Get to the tracker notifying it that the downloaded was completed.

Classes:

The RUBTClient class is the driver for the program that takes in parameters and controls flow of the program. The Tracker class communicates with the tracker and obtains the list of peers. The Pinger class pings each possible peer supplied by the tracker and calculates the fastest average RTT. It then sends the chosen peer back to the RUBTClient to commence download. The Peer class sets up a TCP connection to the Peer, downloads the pieces and writes them to a file. The Piece class stores all blocks that make up a piece. These blocks are obtained through the piece messages from the peer. The Block Class represents the individual blocks that make up a piece. Its fields are a byte array and it's offset within the piece. The Message Class is used with the Peer class to send messages to the peer. These messages include: choke, interested, request, have, etc. The AliveTask is a subclass of TimerTask that sends alive messages to the peer periodically. The TrackerTask class is a subclass of TimerTask which periodically sends updates to the tracker. Lastly, the GiveTools package hold utility classes that were provided for the project. These include Bencoder2, BencodingException, ToolKit, and TorrentInfor. These are used for decoding, handling exceptions, printing, and storing information from the torrent file.