GROUP 24
Yaman Saadi, ysaadi
Ramzy Saleh, rs1064
Rohan Vernekar, rav84

Write-up

Package GivenTools

Bencoder2.java
This class parses bencoded byte arrays and returns Maps, Lists, ByteBuffers, and Integer objects. This class is used in communication with tracker, to get the list of peers.

BecondingException.java
This class handles all exceptions thrown by Bencoder2.java.

ToolKit.java
This class contains many methods that are used for debugging and developing programs using the Bencoder2 methods.

TorrentInfo.java
This class parses data from torrent files and turns the data into fields that are required for use. These are in public fields, to be used by RUBTClient. This class works only with single-file torrents.

Package RUBTClient

Message.java
This class implements messages to used to communicate with the peer. One of its constructors is used to implement all messages after the handshake (choke, unchoke, interested, etc.). This method also generates a general method to create a handshake message. The decodeMessage method sends back a string stating what type of message a passed byte array is. For example, if a message has message ID = 6, the method returns, “request”.

Peer.java
This class helps create Peer objects. It contains the information of a peer, like IP address, peer_id, and port number. Peer.java also has a method to generate a handshake message for a given peer. This class provides getMethods (for IP addresses, peer_id’s, and port numbers). RUBTClient uses this class when the peer list received is decoded, creating peer objects for all the peers.

RUBTClient.java
This is home to the main method. It create tracker, server, and client objects. Using these objects it executes, in parallel, the upload, download, and update of the tracker. It also contains an updateTracker class, which is responsible for periodic updates at the correct interval time.

Server.java
This class creates a ServerConnection for each peer that connects to our client. It loops until the user inputs “-1” to terminate the program. A server socket is created and accepts each peer that tries to connect. A ServerConnection object is created for each of these peers, which starts the upload process.

GROUP 24

Yaman Saadi, ysaadi
Ramzy Saleh, rs1064
Rohan Vernekar, rav84

ServerConnection.java

This class is responsible for uploading pieces of the file to peers. It first reads and verifies the handshake from the client. If the handshake is verified, it then sends its own handshake to the peer. A "have" message is sent to the peer for each piece of the file that is already downloaded. If the peer sends an interested message, an unchoke message is then sent to that peer. A loop begins to read "request" messages from the peer and the corresponding blocks are sent to the peer as "piece" messages. Finally, the connection to the peer is closed.

Client.java

This class is responsible for download from the two peers we are suppose to download from. It does it by requesting pieces, at the same time, from both .131 and .132 peers. It saves the downloaded and verified pieces to a temporary file on the disk. It also saves the file to the disk when the download is completed. It contains some helper methods findAPeer (which finds a peers with the desired IP addresses), verifyHandshake (which verifies the info_hash of the handshake messages), and verifyPiece (which verifies the SHA-1 of a piece passed as parameter). This method also contains a checkFileState, which loads the previously downloaded pieces from the temp file and verifies them. This method ensures that previously downloaded pieces don't go to waste.

Tracker.java

This class is responsible for all communication with the tracker. It reads the torrent file and extracts the information into a TorrentInfo object. It then sends a HTTP Get to the tracker to announce the URL. Then, it decodes the peer list and stores the peers and their information into an array. It contains generateRandomString (which generates a random string of length passed as parameter).

RarestPiece.java

This class is responsible for determining the piece that is the most rare among peers. It also updates rarity values, accepts bitfield messages sent by peers to update rarity values, and sorts pieces by their rarity values. RarestPiece also tells the Client which piece to download next, keeping tracker of pieces that were successfully downloaded.

PeerManager.java

This class is responsible for connecting to the peers determined by the tracker. It contains the keep-alive method, which sends keep alive messages every two minutes. It also is responsible for finding the peers that we are to download from.

Who Did What:

Message.java – Ramzy, **Server.java** – Yaman, Rohan
ServerConnection.java – Yaman, Rohan, **Tracker.java** – Rohan
Client.java – Ramzy, Rohan, **RUBTClient.java** – Ramzy, Yaman
Peer.java – Ramzy, Rohan, Yaman, **RarestPiece.java** – Ramzy, Rohan,
PeerManager.java – Ramzy, Rohan, Yaman

GROUP 24
Yaman Saadi, ysaadi
Ramzy Saleh, rs1064
Rohan Vernekar, rav84

Dependency Diagram