Jaixi, Evan, Mohamed

Decomposition Document 1:

Components:

1. Utilities
    a. Bencoding (Evan)
        i. Potential implementations to use: https://github.com/willemt/CHeaplessBencodeReader, https://github.com/cwyang/bencode
    b. Url Encoding (Jiaxi)
        i. https://stackoverflow.com/questions/5842471/c-url-encoding
    c. Sha1 checksum (use a4 code)
    d. UTF-8 encode / decode (Not sure) (Same as url encoding?)
2. Command line arguments (Jiaxi)
    a. -p: port te client will be on
        i. Optional, if not specified, choose from 6881-6889
    b. -f: metafile
    c. -c: want compact or not
        i. If the tracker only sends compact response, this field is ignored
    d. -a: ip address of the machine
        i. Optional, if not provided use machine ip
3. Data structures (Jiaxi)
    a. Files (list of files we want to download)
        i. List of download_state
        ii. Download_state
            1. File id (urlencoded 20-byte SHA1 hash of the *value* of the *info* key)
            2. Has what pieces
            3. Pieces checksum
    b. Peer (fields will be set to -1 if not applicable) (Mohamed helping)
        i. Id (20 bytes, optional)
        ii. Interested_in: 0 or 1
        iii. Am_Interested: 0 or 1
        iv. Being_choked: 0 or 1
        v. Choking: 0 or 1
        vi. Last time msg sent
        vii. Last time msg receive
        viii. Last time rate calculated
        ix. Avg upload rate for 20 sec
        x. Interval between requests
        xi. File id
        xii. Haves which indexes of the pieces
        xiii. Ip and port
        xiv. Connected Socket
4. Talking with the tracker (Mohamed)

a. Sending GET request
b. Receiving tracker response with info on peers.
c. Parsing response
d. Storing info in data structures
e. Periodically request peers from tracker

5. Talking with peers (Evan) (Mohamed helping)
   a. Handshake
      i. On initial connection to peer
      ii. Send bitfield after handshake
   b. Msg type
      i. Keep Alive (send periodically)
      ii. Choke/Not Choked Status
         1. Peer 1 stops sending/uploading data to Peer 2, but can still receive data from Peer 2
      iii. Interested/Not Interested
         1. Client 1 wants data from Client 2 (client 1 interested)
      iv. Have (may become interested in)
         1. Sent after piece is successfully received and checksum is verified
      v. Request (requesting a piece of data)
      vi. Piece (send the piece of data)
      vii. Cancel (cancel)
         1. Peer 1 wants to stop the download of data to Peer 2

Inter-component interfaces (Written by Jiaxi, Evan, Mohamed):

1. Utilities
   a. Bencoder
      i. In the main function, used to decode information about the tracker from the metafile
      ii. In the infinite loop and in the main function, decode responses from the tracker
      iii. Talking with tracker needs bencoder to decode the messages sent from the tracker
   b. Url Encoding
      i. Needed for talking to the tracker
   c. Sha1 checksum (use a4 code)
   d. UTF-8 encode / decode
      i. For the command line arguments
2. Command line arguments
   a. To process the -f option, will use bencoder in the utilities to decode
   b. Will store the tracker info provided in the -f to the peer structure
   c. -c will affect how the response from tracker is parsed
3. Data structures
   a. File id fields will store the SHA1 hash produced by SHA1 hash in the Utilities and -f in the command line arguments

    b. Has what pieces variable of files and peers data structure relies on Talking with the peers to be updated
    c. Talking with trackers initializes the list of peers data structure
    d. Talking with peers need am_interested, interested_in, choke, being choke, upload rate to decide which peer to unchoke
4. Talking with the tracker
    a. Initialize the list of peers data structure
    b. Use bencoder to encode requests to tracker
    c. Use bencoder to decode tracker responds
    d. Update data structures as more periodic responses are received from the tracker.
5. Talking with the peers
    a. Uses fields inside the peer data structure to get information about the current state of a peer, which are being updated constantly
    b. Uses am_interested, interested_in, choke, being choke, upload rate to decide which peer to unchoke from Data structures

Details of what each component will do (written by Jiaxi and Evan)

1. Utilities
    a. Bencoder :
        i. Decode the metafile to get information about the tracker
        ii. Decode responses from the tracker
    b. Url Encoding
        i. Decode/encode the information received and set to the tracker
    c. Sha1 checksum (use a4 code)
    d. UTF-8 encode / decode
        i. Decode strings in the metafile
2. Command line arguments
    a. Similar implementation to the previous projects.
3. Data Structures
    a. Create the structure to hold the peer sockets
    b. Create struct that contains connection information about the tracker
    c. Create struct that contains information about peers
        i. Connection information
        ii. Piece information
        iii. Status in relation to our node
    d. Create structure that will house the piece data
4. Talking with the tracker
    a. Before infinite loop, successfully connect to the tracker using a HTTP get request
    b. Receive peer information from the tracker, and place into the data structures
    c. Periodically send requests/retrieve peer data from the tracker in the infinite loop
    d. During all requests and data retrieval from the tracker, encode and decode the data using bencoder respectively
5. Talking with the peers

    a. In the infinite loop, we will be running a switch case on a poll response.
        i. On Timeout, determine if periodic functions need to be run. Otherwise, do nothing.
        ii. On error, we exit out of the program.
        iii. On data received, loop through all of the sockets until we determine which socket sent the data. The msg_id is extracted from the incoming data. Based on the ID, we call a function that handles that specific case.

Who will do which component:
Evan May: Talking with peers logic, infinite loop, data structure, talking with peers functions, bencode
Jiaxi Tang: Data Structure, Command line arguments, urlencode, Talking with the peers functions
Mohamed Aamir: Talking with Tracker, Talking with peers helper functions, data structure testing

Plans for testing each component in isolation and in conjunction (Written by Jiaxi and Evan):

1. Utilities
    a. Bencoder:
        i. Print the decoded metafile information to ensure correctness
        ii. Print out the decoded responses from the encoder to ensure correctness after talking with tracker is implemented further
    b. Url Encoding:
        i. Use sample string to make sure it's encode correctly
    c. Sha1 checksum (use a4 code):
        i. Assume to be correct
    d. UTF-8 encode / decode (Not sure)
2. Command line arguments:
    a. Print out the results and examine by print statements
3. Data Structure
    a. Test add, search, delete functions by adding stuff to the data structures, making sure they are added correctly, making sure they are being searched correctly, deleting stuff, ensuring they are being deleted. Freeing all memory used.
    b. Print out all data structures and tracker response, cross examine the information
    c. As talking with tracker/peer is implemented, ensure that the data is being correctly store in the data structure
4. Talking with the tracker
    a. Use wireshark to ensure that messages are being sent from the tracker to the client
    b. Print statements to ensure correctness of tracker messages
    c. Ensure that the tracker is correctly storing and connecting with the other peers sent by the tracker by using wireshark and print statements.
    d. Ensure messages are being decoded properly.
5. Talking with peers

    a. Use wireshark to ensure that data is being sent correctly
    b. Use transmission to make sure messages are sent in the correct format
    c. Print out data that is received and ensure that it is correct
    d. As more functions are written, let the program run for longer before checking messages in print statement and wireshark

What is already coded (written by Jaixi, Evan, Mohamed):

- Argument Parser (will use from previous assignments)
- SHA-1 Checksum (will use from previous assignments)
- Bencoder (implementations exist, just need to call functions)
- URL encoder (implementation found online. Need to modify 1-2 things)

What is due by December 5th (written by Jaixi, Evan, Mohamed):

1. Utilities:
    a. Bencoder
        i. Should be written
    b. URL Encoder
        i. Should be written
    c. UTF-8 encode / decode
        i. Should be written
2. Command-line arguments:
    a. Command line arguments should be completed
3. Data Structures
    a. Data structures initialization should be written
    b. Some helper functions to insert/delete/etc in these data structures should be written (based on needs of talking w/ tracker and peers)
    c. Data structures should be tested to ensure they are working properly.
4. Talking with the tracker
    a. HTTP Get-Request should be working by this time (i.e. data the request is successfully being sent to the tracker and we are getting information back)
    b. Should be able to get the initial tracker response, parse it, and store it in data structures.
5. Talking with peers
    a. The structure of the infinite loop should be setup
    b. Some helper functions should be written
    c. A plan for the logic of (un)choke/(not) interested relationship should be understood

Pseudo code (written by Jaixi and Evan):

```
1. Read the metafile, initialize list of files data structure
2. Connect with tracker
3. Send request
4. Receive response
5. Initialize list of peers data structure
```

        a. Loop through all of them, hand shake

6. Enter infinite loop (Jiaxi, Evan)
    a. Poll for incoming connections
        i. If found, add to peer data structure and send handshake
    b. Poll for incoming data from existing connections (Jiaxi, Evan)
        i. If found, switch on msg_id
        ii. Call function to handle based on msg based on ID
    c. Loop through list of peers (Evan)
        i. If two minutes have passed in the timer, kill connection
        ii. If 30 sec pass, calculate upload rate and shuffle optimistic unchoking
        iii. Get the 4 peers with best upload rate and am interested in us

sudo apt-get install transmission to download bittorrent client on linux.

<u>Test Tracker</u> available at: `http://128.8.126.63:21212/announce`