Design Document

Basic Theory:

We use Java RMI to implement the communication between Peers. Since the RMI has already realized the thread for each peer client communication with peer server, we only need to create threads in Peer, where thread "peerClient" is used for showing menu for the user and thread "executeQuery" is used for running query.

There we list each code with design description.

1. PeerI.java

The Remote interface for Peer. It lists three methods: query (), queryhit () and obtain().
Other Peers need this interface to receive and know the object from this Peer.

2. Peer.java

The code for one Peer.

Attributes:

- PeerUrl: the url address for this peer.
- Port: the port for this peer
- filepath: the relative path where the files are located.
- peerid: the unique id for each peer.
- messageNo: the sequence of message, each peer start message No from 1.
- MaxTTL: the start TTL for every peer.
- Looptime: the sequence request time. Default is 1. Used for test.
- msgToPeer: the Map messageToPeer record the key of message ID with value of its upstream peer ID.
- msglist: the Queue maintain the sequence of message. when its size is msgsize and new massage is coming, the oldest one will be removed.
- msgsize: the max size of msglist.
- msgHit: the Map messageToPeer record the key of message ID with value of its hit peers ID for future obtaining.
- hitlist: the Queue maintain the sequence of message hit. when its size is hitsize and new hit is coming, the oldest one will be removed.
- hitsize: the max size of hitlist.
- msgResponces: the Map msg_responces record the key of the message ID start from this peer with the value of its query start time and all returned query hit times.
- filemap: The filemap record the key of the file name with the value of the newest query message ID.
- neighbours: The static list of neighbor peers for this peer.

Methods and classes:

- query():
  - This method receive the query call from upstream peer. The messageID contains both the real message ID and the upstream peer Url, so firstly it will be splited. Then we should check whether this message has been queried to this peer before by checking msgToPeer.

If it contains, we just skip this query(), otherwise we will maintain the msgToPeer and check whether this peer contains this file with “filename”. If this peer keep this file, it will return queryhit() back to the start peer. If this peer does not have this file and the TTL is over 1, this peer should spread the query() to other neighbors with TTL-1, besides the upstream peer.

- queryhit(): This method call queryhit from the peer which contains the required file. The messageID is the real message ID. If msgResponce contain this ID, it means this peer is the start position who calls query function. Under this condition, the reach time of the queryhit will be saved into the queue in the msgResponce and the aimed peerUrl will saved into msgHit. Additionally, it will display the “message xxx hit” when first hit for specific message come to this peer. If the peer is not the beginning and TTL is over 1, then this peer will transmit the queryhit back to the upstream peer.
- obtain():
  - This method receives the name of file wanted to be downloaded. Then this peer will search the files on the default path ”files/”. If the required file does not exist, return null. If has, it creates an array of bytes to read all bytes from the required file, then return this bytes array.
- peerClient:
  - This thread class will show the menu for users and supply corresponding methods.
  - It provides four options to user.
    - Option 1: Set loop time. It used for testing the performance as a large sequence requests. It will affect the loop time on Option 2 and 4.
    - Option 2: Query files. Firstly, it prompt user to input the file name. Then, it starts executeQuery theard. The executeQuery thread will call the query() to all its neighbor peers and record the start time of query into msgResponces map.
    - Option 3: Calculate the average response time. It will simply call calResponce().
    - Option 4: Obtain files. Firstly, it prompt user to input the file name. Then, it calls executeDownload() to download the file.
    - Option 5: Set a timer to call Lookup. We can appoint a time to start the lookup function on that time.
    - Option 6: Fresh menu. Just break, print the menu again and ask for option input.
    - Option 7: Exit. Close scanner and call Sysetm.exit(0) to quit the program.
  - It has one child class (thread) and four methods to support those options.
    - executeQuery: This thread class executes the query function with initializing the maps in this peer. It will run query() "looptime" times. Each time it will make a unique message ID since the messageNo is increasing. Then the this new messageID will be used for initializing the msgHit, hitlist and msgResponces. Additionally, the start time for each message will firstly recorded into msgResponce. Finally, it will call query() to all of its neighbor peers with maxTTL time.
    - calResponce(): It will calculate the average time of latest executeQuery. It will simply get start time and every end time (of each hit) for specific message, calculate the average time. In this way, it will calculate all the average times for messages made by latest executeQuery, and calculate the average of these average times.
    - executeDownload(): This method will firstly get the newest query message ID for

this file name. Then it will check the msgHit whether it contains this message ID. If it has, we can get the list of all the peers URLs returned hit for that query from masgHit and save the list to a String array. It will require user to choose one IP and get the peer object from the selected Peer. Then call obtain() to get the file data and write it to the new local file.

- getSelection(): This method get a integer from user with the limit that the number should between lower bound and upper bound. First, it prompt user to input an integer and check whether it is a integer. If it is, then check whether it is be between lower bound and upper bound. If all these are right, return the number. Otherwise it requires user to input again.
- TimerLookup(): This method will require user to set a time(HH:MM:SS) and create a timer based on this time with today's year, month and day. In addition, the millisecond will set as 000 uniformly. Then prompt user to input a file name. When the system time reach the set time, timer will call run method which will call executeQuery(). In this way, we can run multiple server on same time and get the performance (average time).

- main(): It creates a Peer object, registers RMI on specific port (e.g. 1101, each peer keep different port), and bind the server on specific URL (e.g. rmi://127.0.0.1:1101/Peer). Then it is waiting for other Peers to connect.

Possible Improvement:

1) The peer only passively receives queryhit time using RMI, although there is the "message xxx hit" shown on screen, the peer itself don't know when all the message are hit back during one loop. Maybe we can improve it to initiatively receive and notice user whether to obtain when all the queryhit finished.
2) This time we only distinguish the files by its name, since the functions registry and lookup are also only based on file name. However, in the real situation, there may exist different files with same name. Maybe we can calculate MD5 to compare whether different.