CS550 Advanced Operating Systems
Programming Assignment 2
Design Documentation

Directory structure:

```
/Lalit_Patil-------|
                   |
                   /Node1---------------------------- Peer.java
                                                      TestPeer.java
                                                      config.properties
                   |
                   /Node2---------------------------- Peer.java
                                                      TestPeer.java
                                                      config.properties
                   |
                   /Node3---------------------------- Peer.java
                                                      TestPeer.java
                                                      config.properties
                   |
                   /Node4---------------------------- Peer.java
                                                      TestPeer.java
                                                      config.properties
                   |
                   /Node5---------------------------- Peer.java
                                                      TestPeer.java
                                                      config.properties
                   |
                   /Node6---------------------------- Peer.java
                                                      TestPeer.java
                                                      config.properties
                   |
                   /Node7---------------------------- Peer.java
                                                      TestPeer.java
                                                      config.properties
                   |
                   /Node8---------------------------- Peer.java
                                                      TestPeer.java
                                                      config.properties
```

1. Peer.java

Both client and server are handled here in Peer class. It acts as both Server that holds hash table as well as node that queries keys.

Two threads run concurrently:
1. Server thread
2. Client thread

Server thread waits for clients to connect and serve clients Put, Get, Delete queries.

Client thread provides user with a menu to access the put, get, delete APIs. It's a continuous menu.

In client part there is a function 'hashIt' It has key as parameter and it returns calculated hash value as integer. All put, get and del function call this function to generate hash value and then corresponding node is connected and respective operation is performed.

Node is generated by mod operation on hash value. In this program we have 8 nodes that are holding keys so node is generated by hash%8.

In Server I have created a concurrent hash map that keeps record of all the key-value pair. Being concurrent hash map it can respond to multiple concurrent queries.

2. TestPeer.java

TestPeer class is having same functionality as Peer class.

TestPeer class is created for testing functionality with 100k put, 100k get and 100k del requests.

In TestPeer I have removed user input for keys.

It keeps track of the response time and prints average response time for each thread.

3. config.properties

config.properties file holds the port list and host names. The values are read in to the hash maps in constructor.

4. build.xml

build.xml is ant build file. It provides compile, run and clean operation.

5. output.txt:

output.txt has sample output of all three operations.

Tradeoffs/known Problems:

1. I could have used more modular approach.
2. The menu is not working properly. There is some problem with 'newline' character. One or more extra 'newline' character are causing menu to load more than one time before allowing user to enter choice.
3. While running put in test mode, the last node we run operation concurrently runs faster than others. The output is wayward for that node.
4. Resilience is not provided. If any node is down, the program wont work.
5. The program is designed such that all nodes are need to be up before performing any task. Otherwise wont work.

Improvements/Extension:

1. Need to make this more modular
a. Can make different class for hash function
b. Client server can be separated for more flexibility
2. Resillence can be added by duplicating the key-value pairs on different nodes. this can be achieved either by duplicating all DHTs on every other node or by making pair of nodes for mirroring.