I]

Programming language used: 1] Python2 for client and server side codes

2] thrift as Interface Definition Language

Tools : Apache thrift ; for generating server and client side stubs from chord.thrift file. These stubs are responsible for marshalling and unmarshalling of parameters and return objects.

II]

Compiling the code and running on remote servers: Follow the following instructions

1) Make sure you are on bash by **echo $0**

2) run these commands one by one

```
export PATH=$PATH:/home/yaoliu/src_code/local/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:"/home/yaoliu/src_code/local/lib"
```

3) generate server, client side stubs and probably a header file by

thrift -r –gen py chord.thrift

(I have pushed my gen-py folder to github so this step can be skipped)

4) run the server file provided as **python server.py <port number> &** for as many nodes as desired. Change port number for every server. Record the process id to **kill pid** later. It will be easier to use the same ports again after killing the process.

5) Open nodex.txt file and add nodes into it on which your servers are running as

<ip address>:<port number>

6) run **chmod +x init** then **./init nodex.txt**

7) Run your client implementation on any of the node for example :

Python <ip address> <port number>

Ps: Every time you need to write or read a file you will first need to get successor node for that file name, that is sha256(filename).

III]

I have implemented all the 6 functions and tested them as well. After testing rigorously I have fixed almost all the bugs, but still I wouldn't be 100% sure as my implementation seems a little bit unconventional. (From DHT paper implementation.)

IV]

Implementation details:

1] Updating the contenthash along with content was not mentioned in the requirements document but I have updated it. If it is not desired just comment the line : self.dht_files[file_name_hash].meta.contentHash = _hl.sha256(rfile.content).hexdigest() in else part of writeFile() in server.py

2] I have used program memory to store the files. My filesystem is a python dictionary with filename hash as the key and rFile <file> as data.

3] In findSucc() node I check if the filename hash or the key to be precise is already present in the handler's dictionary, if so it avoids making any unnecessary RPCs and returns NodeID <self.node>. If this is not desired please comment the following lines in findSucc() function:

```
if file_id in self.dht_files:
        return self.node
```

4] Whenever a server is started it prints out the node object with its ip, id and port no.

Also on every system exception that occurs it appends the node object along with the message.

If these are not desired please remove the corresponding prints.

-------- Sachin P. Rodge

CS557