Project 3 Report

Group Members:
Joseph S Rayan, (UFID: 61022245), josephsrayan@ufl.edu
Sai Krishna Anugu, (UFID: 42266064), saikrishnaanugu@ufl.edu

Design and Chord Implementation:

We decided to utilize the SHA-256 hashing algorithm to hash the values of each Process ID to get the Node ID and the key that needs to be looked up. The finger table of each node stores entries based on the value of *m*. The value of *m* scales as the number of nodes increases, therefore this value is directly affected by the user input. Each Node will maintain its own finger table with several successors based on its Hash value.

Insertion of node

Whenever a new node is inserted into the network, another node that is already present in the network will be requested to check the predecessor and successor of the inserted node. As soon as it finds its immediate neighbors the initialization of the finger table for the inserted node gets started. Suppose the random node that has been asked about the predecessor & successor of the inserted node doesn't have the information, then it asks its neighbors to find and this search continues until the predecessor and successor of the inserted node are known. After the finger table of the newly inserted node is set up then the finger table of other nodes gets updated because of the changes that occurred in the network due to node addition.

Lookup request

When a lookup request is initiated from a random node, the node will check whether the key lies within its own Node ID and its successor Node ID. If the key lies within this range, the successor node is responsible for this key. If the key doesn't lie within this range, then the finger table of the node is used to find the closest node in the table and the lookup request is sent to that node. This other node will perform the same operation to determine where the lookup request should be directed.

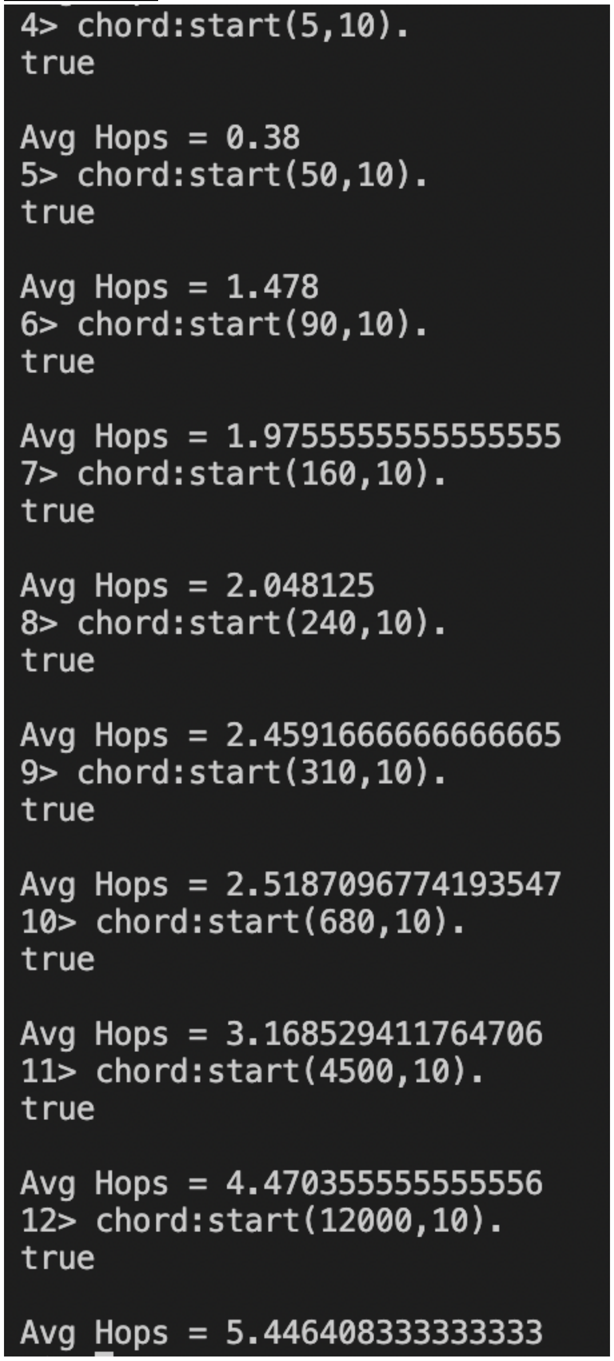
Number of nodes	Average number of lookups
5	0.38
50	1.478
90	1.9755
160	2.048125
240	2.4591
310	2.5187
680	3.1685
4500	4.4703
12000	5.4464
15000	5.4441

The above is the average number of hops taken for different numbers of nodes for a lookup request of 10.

Additional Observations:

The decision of use SHA-256 was due to the problem of recurring Node ID values when using the SHA-1 hashing function. In our testing, the SHA-256 performed a lot better in terms of reducing the number of duplicate Node IDs

Appendix:



```
4> chord:start(5,10).
true

Avg Hops = 0.38
5> chord:start(50,10).
true

Avg Hops = 1.478
6> chord:start(90,10).
true

Avg Hops = 1.9755555555555555
7> chord:start(160,10).
true

Avg Hops = 2.048125
8> chord:start(240,10).
true

Avg Hops = 2.4591666666666665
9> chord:start(310,10).
true

Avg Hops = 2.5187096774193547
10> chord:start(680,10).
true

Avg Hops = 3.168529411764706
11> chord:start(4500,10).
true

Avg Hops = 4.470355555555556
12> chord:start(12000,10).
true

Avg Hops = 5.446408333333333
```