Readme Proj -3

Problem Statement
You have to implement the network join and routing as described in the Chord paper (Section 4) and encode the simple application that associates a key (same as the ids used in Chord) with a string. You can change the message type sent and the specific activity as long as you implement it using a similar API to the one described in the paper.

Requirements

Input - The input provided (as command line to yourproject3.erl) will be of the form:

```
project3 numNodes  numRequests
```

numNodesis the number of peers to be created in the peer-to-peer system and numRequests is the number of requests each peer has to make. When all peers performed that many requests, the program can exit. Each peer should send a request/second.

Output
Print the average number of hops (node connections) that have to be traversed to deliver a message.

Group Members
1. Divyajyoti Ukirde, UFID: 30260576, Email: divyajyotiukirde@ufl.edu
2. Yashwanth Venkat Chandolu, UFID: 11429359, Email: ychandolu@ufl.edu

Project Description
The project consists of a single file – chord.erl

Instruction Details
This project has been developed using VS Code/ Atom Text Editor.

Please make sure you have erlang is installed in the system.

Compile the file using c(chord). in the erlang terminal. Post that, start the file with chord:start(NumNodes, NumRequests). where NumNodes is the number of nodes and NumRequests is the number of requests that has to be satisfied by each node.

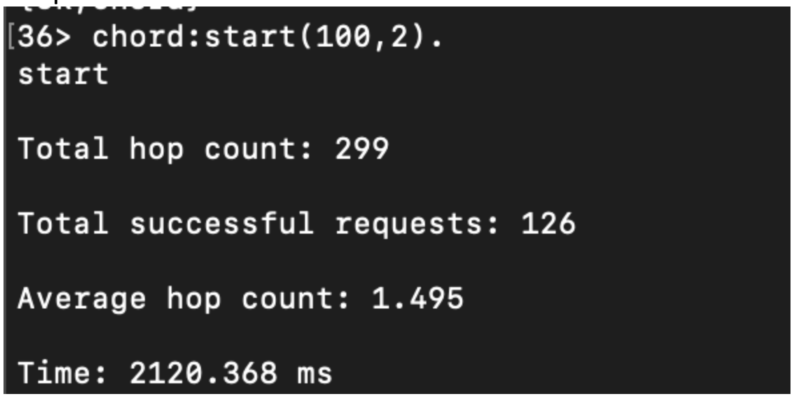
Machine Details

We tested the code on two machines:
1. 8th Gen Intel 2.3 GHz Quad Core Processor
2. Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz   2.70 GHz

Observation and result

- We observed from our result that average hop time is approximately equal to $(1/2)\log_2(10)$.
- The value of the constant term (1/2) can be understood as follows. Consider a node making a query for a randomly chosen key. Represent the distance in identifier space between node and key in binary. The most significant (say ith) bit of this distance can be corrected to 0 by following the node’s ith finger. If the next significant bit of the distance is 1, it too needs to be corrected by following a finger, but if it is 0, then no $i - 1$st finger is followed—instead, we move on the the $i-2$nd bit. In general, the number of fingers we need to follow will be the number of ones in the binary representation of the distance from node to query. Since the node identifiers are randomly distributed, we expect half of the bits to be ones. After the logN most-significant bits have been fixed, in expectation there is only one node remaining between the current position and the key. Thus the average path length will be about $(1/2)\log_2(10)$.

Output:



```
[36> chord:start(100,2).
start

Total hop count: 299

Total successful requests: 126

Average hop count: 1.495

Time: 2120.368 ms
```

Total Hop Count : Gives the number of hops that all the nodes have encountered to satisfy the NumRequests.
Total Successful requests : Gives the number of requests that were satisfied.
Average Hop Count : is given by the formula start/(successful requests * Hop Count).

Maximum Number of nodes that we managed to deal with: 10k nodes.