Chord Peer to Peer Implementation

Imthiaz Hussain (imthiazh.hussain@ufl.edu)
Lohit Bhambri (lohit.bhambri@ufl.edu)

Problem Statement:

The objective for the project is to implement network join and routing algorithm referred as Chord.

1. Chord is a protocol and algorithm for a peer-to-peer distributed hash table.
2. A distributed hash table stores key-value pairs by assigning keys to different nodes; a node will store the values for all the keys for which it is responsible.
3. Chord specifies how keys are assigned to nodes, and how a node can discover the value for a given key by first locating the node responsible for that key.

Run Project:

Run the following commands for the project

1. Compile the project

   c(node).

   c(main).

   c(hopCalculator).

2. Call the main function

   main:chord_start(1000,3).

where 1000 represents NumberofNodes and 3 represents NumberOfRequests.

Explanation:

Our project will start from the chord_start(TotalNodes,Requests) function.
We will *calibrate* the total number of spaces in the chord to the highest nearest power of 2^n.

Example:

totalNodes = 9

nearest highest power of 2 which is greater than 9 is 16 i.e. 2^4 where n = 4

Now we will generate the nodes (a.k.a. Actors) in following:

Random_ID = rand:uniform(1000000000),

Hashed_data = crypto:hash(sha, <<Random_ID>>),

<<Hash_to_int:160/integer>> = Hashed_data,

Identifier = Hash_to_int rem TotalSpaces,

The above code snippet explains how we will generate the hashed node id. Once generated will spin a node via function call node:startLinkand assign the hashed identifier.

Finger Table:
In chord protocol each node maintains a data-structure entry for message distribution called as *finger table*. Our message distribution from a random node to the specified nodes work upon the neighbor lookup in the finger table and transmitting the message to the actor.
If the *specified key isn't available* inside the finger table, we will delegate the call to *the nearest responsible node* to distribute the message in an efficient way.

Working Objectives:

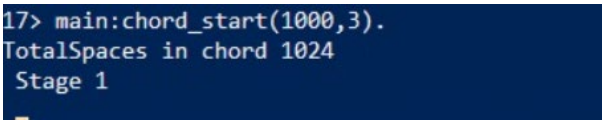
1. We are able to establish the chord network
2. We are able to populate the finger tables for each node (i.e. ActorPid) in the network.
3. Each node is able to send queries to the appropriate node in the network through a series of *hops*.
4. We are able to achieve the objective of closest node-id handling the responsibility calls of an inactive node-id.

Largest Achievable Network:

The largest achievable network is of 1000 nodes with average time of 7.74 (approx) seconds If we take the absolute log time for calculating 1000 base 2 we get 10 seconds. So our result is close to the absolute threshold value due to long traversal paths.

Result Screenshot:

Input:



Output: