Chord

Implemented the original Chord protocol for distributed storage and fetch of files.

It has been implemented using consistent hashing for hashing the key and identifier values and storing them in the finger(hash) tables. Also implemented object access service by requesting a

random node to find a specific key.  
  
The implementation was done using Akka actors where each actor represented a node in the network.

Each node maintains a finger table of it its neighbors as designated by the protocol.

The reference to the paper : <https://pdos.csail.mit.edu/papers/chord:sigcomm01/chord_sigcomm.pdf>

The project can be run using SBT.

Build and run.

There are two inputs to the project. The number of nodes that the network should contain.

Each node in the network is set to produce a number of random requests for a file location in the network per second.

This forms the second input. It is the number of requests each node node has to make per second in the network.

**Inputs**:

numNodes - The number of nodes in the network

numRequests - The number of requests executed at each node

**Output**:

avgNumHopes - The average number of hops taken for finding the requested key.

**Working Part:**

We implemented the chord protocol using the consistent hashing and the key access part. All the functionalities are working correctly. Use the command below to execute :

Run using sbt

- run numNodes numRequests

Sample input - run 1000 5

The table below shows the results for various combinations of number of nodes and number of requests

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No. | NUMBER OF NODES | NUMBER OF REQUESTS | SEARCH TIME | TOTAL NUMBER OF  HOPS | AVERAGE NUMBER OF  HOPS |
| 1. | 1000 | 4 | 5.288s | 26398 | 6.5995 |
| 2. | 1000 | 8 | 9.224s | 53601 | 6.7001 |
| 3. | 3000 | 4 | 5.568s | 89138 | 7.4281 |
| 4. | 3000 | 8 | 9.876s | 179117 | 7.4632 |
| 5. | 4500 | 4 | 5.821s | 140559 | 7.8088 |

**Largest Network:**

Largest Network Implemented : 4500 nodes

Number of requests = 4

Results are as follows:

Total number of hops = 140559

Average number of hops = 7.8088

**Understanding:**

1) As the number of nodes remains constant and the number of requests is increased, then we observed a very slight increase(less than 1) in the average number of hops that were to be traversed in order to deliver a message but there was a significant increase in the search time.

2) As we increased number of nodes keeping the number of requests value constant, we observed a reasonable increase (more than 1) in the average number of hops that were to be traversed in order to deliver a message whereas the search time remained almost same with a slight increase.