COP5615 Distributed Operating System Principles: Project - 3 P2P Chord DHT

Submitted By:

Rachit Rathi, UFID: 8089-6039

Rishabh Das, UFID: 8045-9065

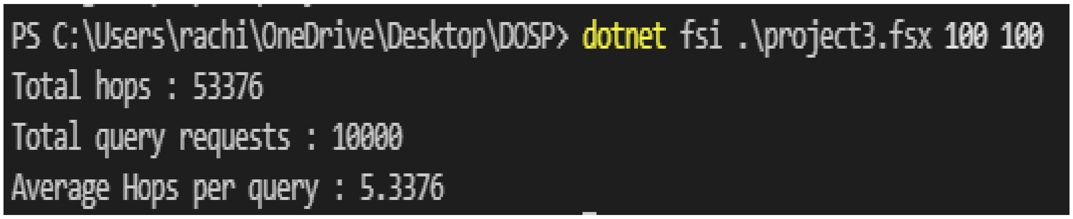
Background:

In this project, we have implemented a distributed hashing routing table using the Chord Algorithm. We have simulated the distributed nature of the routing table using the “AKKA actor framework” implemented in FSharp. Essentially, the routing table acts as an efficient object access service that stores the data of the succeeding neighboring hop for any given node.

Steps to run:

1. The name of the project is project3.fsx
2. To run the project extract the file “Project3.zip” and copy the file “project3.fsx” in a directory that is accessible by dotnet.
3. To run the program, use the command: **dotnet fsi project3.fsx <numOfNodes> <numOfRequests>**
4. Make sure dotnet.exe path is defined in $PATH

Sample Output:



```
PS C:\Users\rachi\OneDrive\Desktop\DOSP> dotnet fsi .\project3.fsx 100 100
Total hops : 53376
Total query requests : 10000
Average Hops per query : 5.3376
```

<u>What is working:</u>

1. File Distribution: The nodes in the system are distributed with names encrypted by the SHA-1 encryption.
2. Routing: Each node has its own associated routing table which directs it to the succeeding node.
3. File search: A node initiates a self-search in order to find a particular file. If the search succeeds, the node requesting search is sent the id of the node it requests.
4. Network Join: Each node in the DHT table is associated with another node ie: a node can hop to another node in a circular arrangement of nodes "joined" or connected together. Therefore, each node can send or receive messages in the system in the ring called the Chord ring. Each key in the DHT which is essentially the SHA-1 key directs to the node that it succeeds.

<u>Results:</u>

A. Query = 10

Num of Nodes	Total Hops	Hops / Query
50	2247	4.494
100	5313	5.313
500	33429	6.6858
1000	68315	6.8315
5000	398392	7.96784

B. Query = 50

Num of Nodes	Total Hops	Hops / Query
50	11175	4.47
100	26732	5.3464
500	167468	6.69872
1000	341310	6.8262
5000	1991703	7.966812

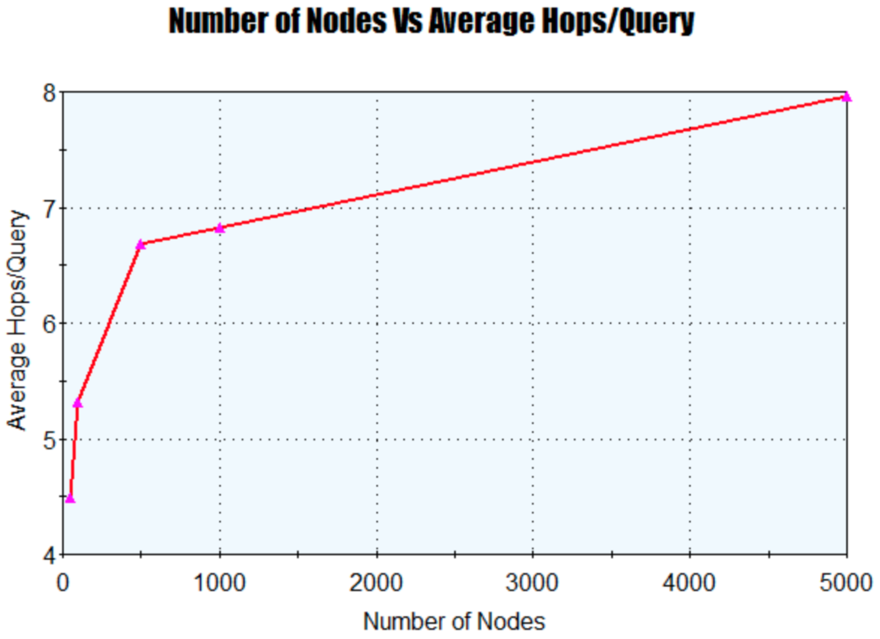
C. **Query = 100**

Num of Nodes	**Total Hops**	**Hops / Query**
50	22631	4.5262
100	53392	5.3392
500	333812	6.67624
1000	683175	6.83175
5000	3982278	7.964556

Number of Nodes Vs Average Hops/Query

Number of Nodes	**Average Hops/Query**
50	**4.49**
100	**5.32**
500	**6.68**
1000	**6.82**
5000	**7.96**

Plot for Number of Nodes Vs Average Hops/Query



Observation:

The number of Nodes and Average Hops/Query are logarithmically related. The above graph has been created from the tabular values above with combinations for different nodes and number of requests.

Largest Network

The largest network that we tested for is one with 5000 nodes, 100 queries. Any nodes upwards of the tested network took a considerable amount of time to terminate.