DOS Project 3 - Chord: P2P System and Simulation

Rupayan Das - 63992113
Pirouz Naghavi - 30158432

Getting Started

Requirements

- **.NET 5 SDK on both machines**

Running the project

- *dotnet fsi Project3.fsx <numberOfNodes> <numberOfRequests>*
- eg: dotnet fsi Project3.fsx 1000 10

What is working

We have implemented the joining and routing of the nodes as per the Chord paper using the API functions described in the Chord algorithm for distributed hashing. The nodes are assigned an ID that is taken by modding the hash value of their index (SHA-1) and the total size of the identifier space (2^m). Each node generates a random request (string) which then gets converted into a key using SHA-1 and makes hops to arrive at the destination node using the finger table. Requests are sent until each node receives confirmation that the request they sent has arrived at the corresponding destination *numRequest* times. We assigned m = 3 * log(N) where N is the number of peers defined by the user.

Join:

- We set up the initial network with (numNodes - 10) nodes. And while they start sending requests, we make the remaining nodes join the network as well.
- The joining node gets a randomly selected existing node in the network from the dispatcher and asks it to find its successor. Once it gets the successor and the successor list, it runs the stabilize and the notify function along with fixFingerTable. This way the newly joining node becomes an active participant in the network.
- The other nodes get to know about new peers as they keep running the stabilize and fixFingerTable functions periodically.
- The stabilize function includes updating the successor list as well in addition to checking the successor's predecessor.

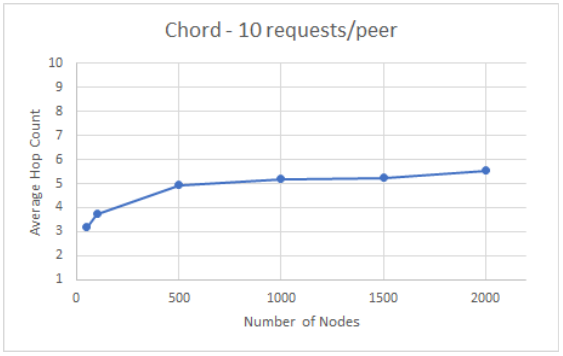
Routing:

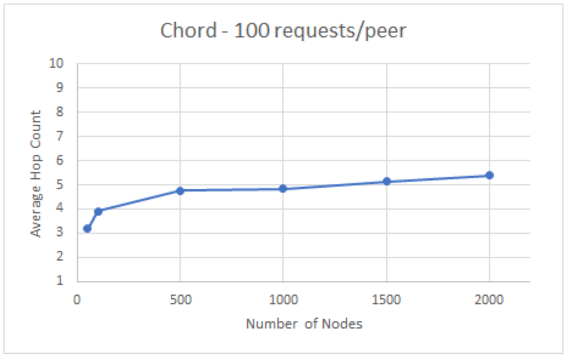
- Routing is implemented by a node sending a request every second.
- Whenever a node sends a request to the findDestination function, we first check if the request key exists between the predecessor and node. If it does, then the current node is the destination itself.
- Then we check between the node and the successor. If it does not exist in between, then we do a lookup in the combined finger table and successor list and return the closest preceding node. (IV - E.3)
- As the request message traverses through the network, the hops are also calculated. Once the destination node is reached, the no. of hops taken is sent to the dispatcher where the average number of hops are calculated once all the peers are done sending requests. On average it takes O(logN) hops to reach the destination as proven in the paper.

What is the largest network you managed

Largest network: {Nodes: 10,000; Requests: 100; AvgHops: 8.034546}

Nodes	Avg Hop Count - 10 req/node	Avg Hop Count - 100 req/node
50	3.19	3.19
100	3.72	3.72
500	4.94	4.94
1000	5.19	5.19
1500	5.24	5.24
2000	5.53	5.53


Chord - 10 requests/peer
Average Hop Count
Number of Nodes


Chord - 100 requests/peer
Average Hop Count
Number of Nodes