Report 5: Chordy

Yanghang Zeng

October 12, 2022

1 Introduction

This assignment is about the implementation of a distributed hash table(DHT). A distributed hash table is a decentralized storage system that provides lookup and storage schemes, which can be utilized in distributed systems, e.g., P2P network. It is important in distributed systems as a way to spread resources and perform precise lookups with good performance. This implementation uses a simplified version of the Chord protocol. What is done was to complete the procedure of building and maintaining a ring structure, and to build local storage for each node which enables lookup functionality.

2 Main problems and solutions

There are several main problems:

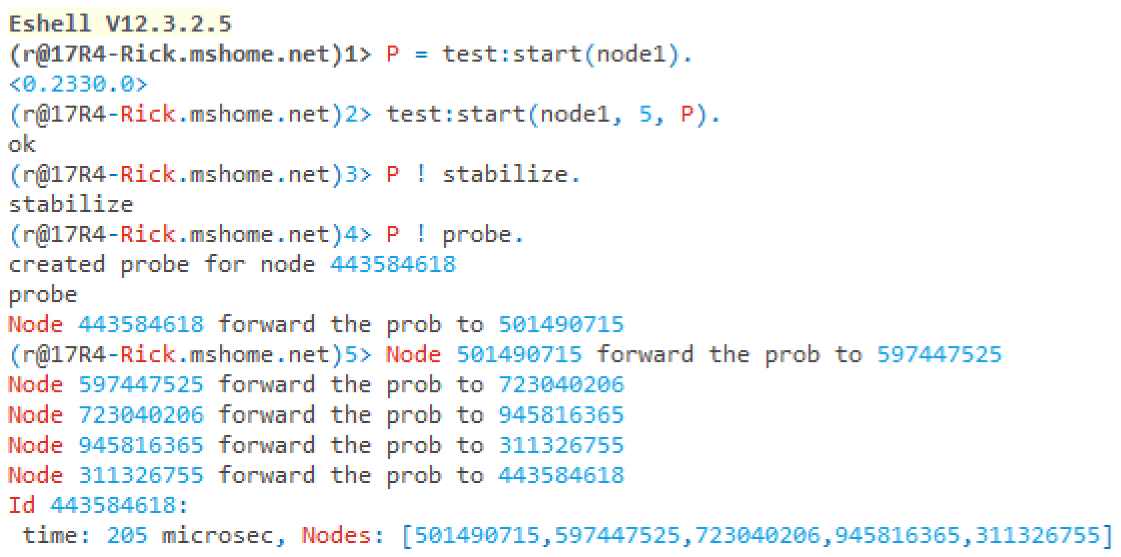
The first problem is to implement a ring structure and keep it in the correct state when adding nodes to it. To achieve this, each node keeps a record of its predecessor and successor, and the node periodically checks if the predecessor of its successor is the node itself. If something went wrong, the node will notify its successor to correct the predecessor pointer. This is the stabilization procedure, which will continuously make sure the ring is a closed one after adding new nodes.

The second problem is completing the lookup function. Each node has local storage, which keeps tuples of key, value pairs. When performing a lookup, a node will check if it is responsible for the queried key, if not, the node will forward the request to its successor, otherwise it will send the result to the client's request.

3 Evaluation

To test the implementation, the test module provided by the course staff is used. The test creates some nodes to form a ring and verify the state of the ring, and also test the lookup function. In addition, a probe mechanism is used to observe the correctness of a ring structure.

To test the validity of the ring structure, a probe is created for a node, and it will forward it to its successor. If the ring is closed, the probe will eventually reach the first node again. The probe also keeps time information to evaluation the time performance. The result of a random test is as in Figure 1:



```
Eshell V12.3.2.5
(r@17R4-Rick.mshome.net)1> P = test:start(node1).
<0.2330.0>
(r@17R4-Rick.mshome.net)2> test:start(node1, 5, P).
ok
(r@17R4-Rick.mshome.net)3> P ! stabilize.
stabilize
(r@17R4-Rick.mshome.net)4> P ! probe.
created probe for node 443584618
probe
Node 443584618 forward the prob to 501490715
(r@17R4-Rick.mshome.net)5> Node 501490715 forward the prob to 597447525
Node 597447525 forward the prob to 723040206
Node 723040206 forward the prob to 945816365
Node 945816365 forward the prob to 311326755
Node 311326755 forward the prob to 443584618
Id 443584618:
 time: 205 microsec, Nodes: [501490715,597447525,723040206,945816365,311326755]
```

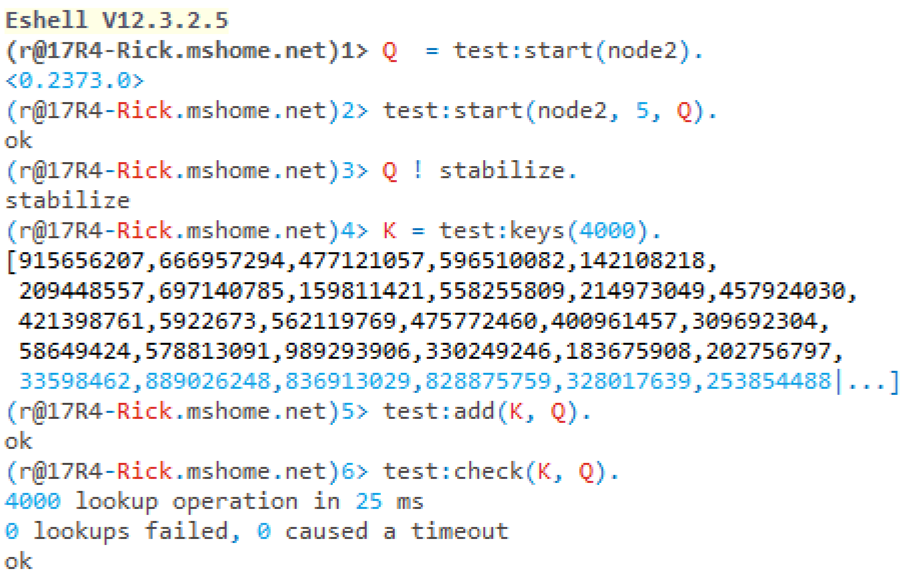
Figure 1: test for node1

From the result, we can see that the original node forwards the probe to its successor, and the probe will be forwarded to the node itself which proves the ring is in the correct state. Therefore the ring implementation is successful.

The result of the test to verify the version with lookup functionality is as in Figure 2. From the result, we can see that the lookup functionality is done perfectly.

4 Conclusions

In this assignment, I learned about how to implement a simple Distributed Hash Table using the Chord protocol. I also get to know the design challenges of a DHT system. It is one of the significant techniques to build a distributed network.



```
Eshell V12.3.2.5
(r@17R4-Rick.mshome.net)1> Q  = test:start(node2).
<0.2373.0>
(r@17R4-Rick.mshome.net)2> test:start(node2, 5, Q).
ok
(r@17R4-Rick.mshome.net)3> Q ! stabilize.
stabilize
(r@17R4-Rick.mshome.net)4> K = test:keys(4000).
[915656207,666957294,477121057,596510082,142108218,
 209448557,697140785,159811421,558255809,214973049,457924030,
 421398761,5922673,562119769,475772460,400961457,309692304,
 58649424,578813091,989293906,330249246,183675908,202756797,
 33598462,889026248,836913029,828875759,328017639,253854488|...]
(r@17R4-Rick.mshome.net)5> test:add(K, Q).
ok
(r@17R4-Rick.mshome.net)6> test:check(K, Q).
4000 lookup operation in 25 ms
0 lookups failed, 0 caused a timeout
ok
```

Figure 2: test for node2