Report 5: Chordy - A Distributed Hash Table

Ziheng Zhang

October 12, 2021

1 Introduction

In this assignment, I implemented a distributed hash table, or DTH, according to instructions and the Chord paper.

2 Main problems and solutions

2.1 A Basic Ring

The first tack is to build a basic ring and handle the growing number of the new nodes. The flowing it the process of adding a new node in the ring.

1. build a new node
2. stabilize it
3. update the *finger tables*

When the new node is created, the successor and predecessor need to be notified, so the `notified` function will be called. After that, `stabilize` function will help the new node to build double link between it and its successor and predecessor.

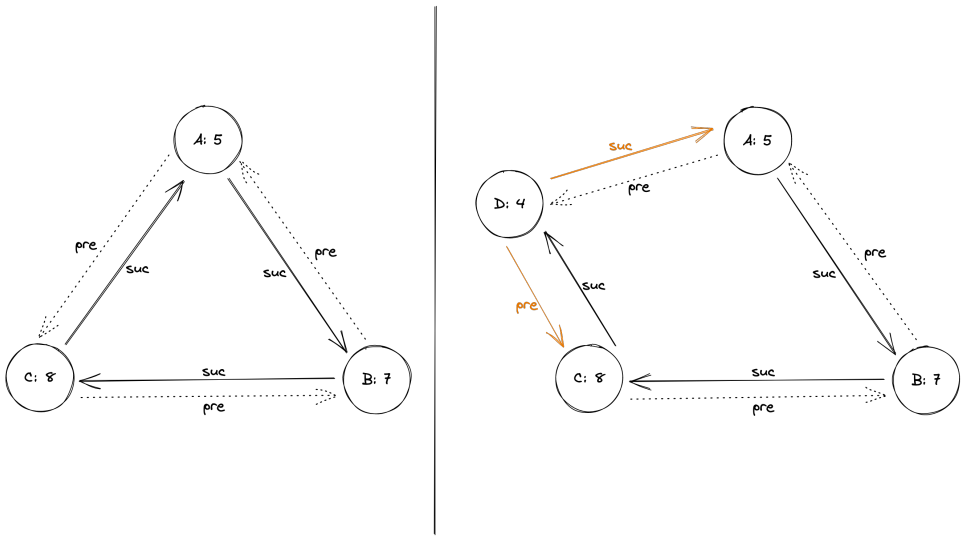

Figure 1: Add a new node to the ring.

```
Eshell V12.0.3  (abort with ^G)
1> A = test:start(node1).
<0.82.0>
2> B = test:start(node1, A).
<0.84.0>
3> C = test:start(node1, A).
<0.86.0>
4> D = test:start(node1, A).
<0.88.0>
5> A ! probe.
Node(s): [461528,885454,726054,580460].
 Time (per round): 40.
probe
```

Figure 2: Implementation for a basic ring

2.2 Store Key, Value In The Ring

When we successfully build the ring, we can add key-value pairs into it. The key-value pairs are stored as the `list` in each node, and each node has the responsibility to store the keys between $(node.predecessor, node]$. When a new node is added, we need to try to handle some responsibilities if necessary. Specifically, we split the `Store` list and `Keep` the own parts and `Handover` the rest.

```
Eshell V12.0.3  (abort with ^G)
1> A = test:start(node2).
Id: 2101
<0.82.0>
2> B = test:start(node2, A).
Id: 291
<0.84.0>
3> C = test:start(node2, A).
Id: 5809
<0.86.0>
4> A ! probe.
Node(s): [291,5809,2101].
 Time (per round): 35.
probe
5> test:add(1957, hej, B).
ok
6> test:add(5612, hello, C).
ok
7> test:add(5610, nihao, C).
ok
8> test:add(6928, hi, C).
ok
9> test:add(6930, bonjour, A).
ok
10> test:add(1955, hallo, B).
ok
```

Figure 3: Add nodes and key-value pairs

```
11> test:lookup(1957, A).
{1957,hej}
12> test:lookup(5612, C).
{5612,hello}
13> test:lookup(5610, B).
{5610,nihao}
14> test:lookup(6928, A).
{6928,hi}
15> test:lookup(6930, B).
{6930,bonjour}
16> test:lookup(1955, C).
{1955,hallo}
17> D = test:start(node2, A).
Id: 4697
<0.101.0>
18> test:lookup(1957, A).
{1957,hej}
19> test:lookup(1957, D).
{1957,hej}
20> test:lookup(5610, D).
{5610,nihao}
21> test:lookup(6930, C).
{6930,bonjour}
22> test:lookup(1955, D).
{1955,hallo}
23> test:lookup(6928, A).
{6928,hi}
24> test:lookup(5612, D).
{5612,hello}
```

Figure 4: Lookup for key-value pairs and add a new node to the ring

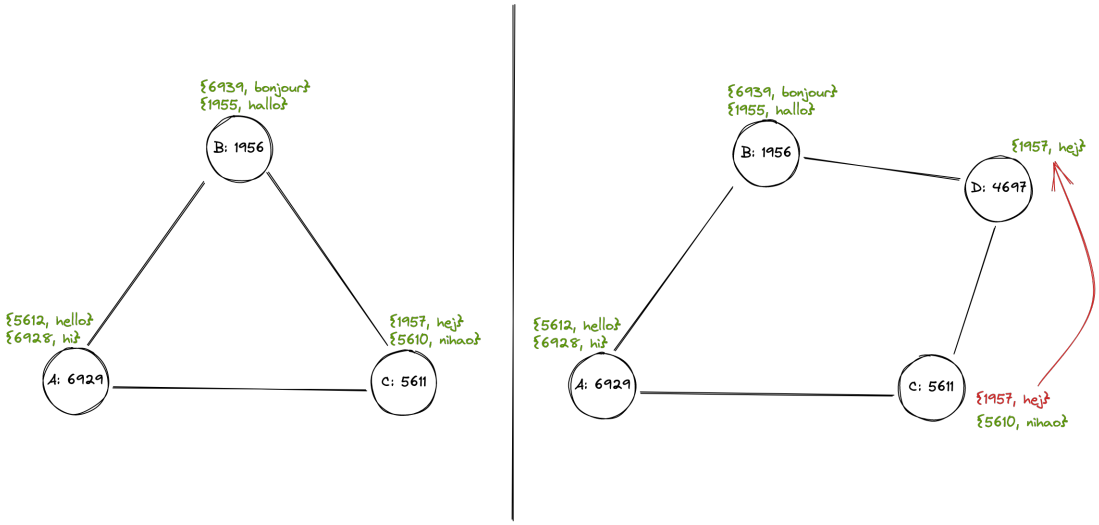

Figure 5: Key-Value store process

3 Evaluation

The evaluation process are shown above in the section 2 Main Problems And Solutions, and also will show during the zoom meeting room.

4 Conclusions

In this assignment, I implemented a distributed hash table, or DHS, and acquire the knowledge of how to add the new node to the existed ring, and how to stabilize the system. I also learned when the DHS system wants to store several key-value pairs, each node has the responsibility to keep the keys in its range, i.e. $key \in (node.predecessor, node]$. When a new node is added into the ring, key-value pairs could be split and handover several responsibilities between two nodes.