University of California, Santa Cruz
CMPE 264 Computer Network Project

Weiting Zhan

December 15, 2018

Contents

1 Chord Implementation **1**
1.1 Initialize the Chord 1
1.2 Finger table 1
1.3 Key migration 3
1.4 Look-up Keys 4
1.5 Node leave 4

2 Space Shuffle Implementation **5**
2.1 Topology Construction 5
2.2 Greediest Routing 6

List of Figures

2.1 Space 5

List of Tables

1.1 Summary of Finger Table 1
1.2 Finger Table for Node 22 2
1.3 Finger Table for Node 88 2
1.4 Finger Table for Node 123 2
1.5 Finger Table for Node 168 3
1.6 Finger Table for Node 215 3

2.1 Nodes coordinates in space 1 and space 2 5
2.2 The distance of node0 to neighbors in space 1 and space 2 6

Chapter 1

Chord Implementation

The report is based on one possible result of implementation. The code generate random network can be found https://github.com/WaitingZhan/Chord-and-Shaffer-Space.

1.1 Initialize the Chord

Chord Stoica, *et al.* [2001] is a protocol and algorithm for a peer-to-peer distributed hash table. In my implementation, the key space of chord resides between 0 and $2^8 - 1 = 255$, inclusive. Randomly initial the chord with 5 nodes. Each node maintain a finger table which has 7 elements.

1.2 Finger table

Randomly initiate the Chord with Node20, Node88, Node123, Node168, Node215.
Generate key [33,45,66,189,244]
Key migration rule:
If the key's value less than the node value, then store the key in the node.
If the key's value larger than the node value, find the next node whose value less than key by using the finger table of the previous node.
The finger table of each Node are shown below:

i	0	1	2	3	4	5	6
Node20	21	22	24	28	36	52	84
Node88	89	90	92	96	104	120	152
Node123	124	125	127	131	139	155	187
Node168	169	170	172	176	184	200	232
Node215	216	217	219	223	231	247	279

Table 1.1: Summary of Finger Table

The finger table of Node 22. Since only one node in the network, store all the key in the node. Node 22 has key [33,45,66,189,244]

Entry	The closest ID	Contact
1	21	N88
2	22	N88
3	24	N88
4	28	N88
5	36	N88
6	52	N88
7	84	N88

Table 1.2: Finger Table for Node 22

Joint the Node 88. The network has 2 nodes.
Node 22 has key []
Node 88 has key [33,45,66,189,244]

Entry	The closest ID	Contact
1	89	N123
2	90	N123
3	92	N123
4	96	N123
5	104	N123
6	120	N123
7	152	N168

Table 1.3: Finger Table for Node 88

Joint the Node 123. The network has 3 nodes.
Node 22 has key []
Node 88 has key [33,45,66]
Node 123 has key [189,244]

Entry	The closest ID	Contact
1	124	N168
2	125	N168
3	127	N168
4	131	N168
5	139	N168
6	155	N168
7	187	N215

Table 1.4: Finger Table for Node 123

Joint the Node 168. The network has 4 nodes.
Node 22 has key []
Node 88 has key [33,45,66]

Node 123 has key []
Node 168 has key [189,244]

Entry	The closest ID	Contact
1	169	N215
2	170	N215
3	172	N215
4	176	N215
5	184	N215
6	200	N215
7	232	N20

Table 1.5: Finger Table for Node 168

Joint the Node 168. The network has 4 nodes.
Node 22 has key []
Node 88 has key [33,45,66]
Node 123 has key []
Node 168 has key []
Node 215 has key [189,244]

Entry	The closest ID	Contact
1	216	N20
2	217	N20
3	219	N20
4	223	N20
5	231	N20
6	247	N20
7	279-255=24	N88

Table 1.6: Finger Table for Node 215

1.3 Key migration

Insert a Node90.
Visit Node 22. Didn't find the ID
Visit Node 88, find the ID.
So insert the Node after Node 88.

Joint the Node 168. The network has 4 nodes.
Node 22 has key []
Node 88 has key [33,45,66]
Node 90 has key []
Node 123 has key []
Node 168 has key []
Node 215 has key [189,244]

1.4 Look-up Keys

look for key 189.
visit node22, didn't find it, needs to forwarding.According to node22 finger table, forward to Node88.
visit node88, didn't find the key 189, Needs to forwarding. According to node88 finger table, forward to Node168.
visit Node168,didn't find the day 189. Needs to forwarding. According to node88 finger table, forward to Node215.
visit Node215. find the key at Node 215.

1.5 Node leave

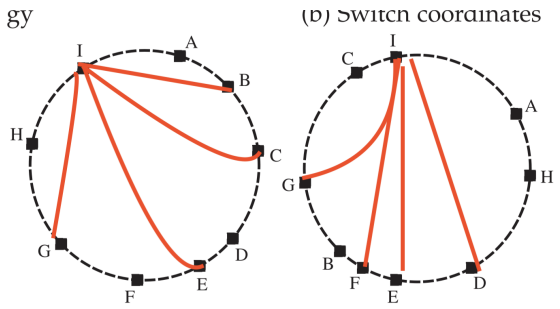
Node 88 leave the network. The network has 4 nodes.
Node 22 has key []
Node 90 has key [33,45,66]
Node 123 has key []
Node 168 has key []
Node 215 has key [189,244]

Chapter 2

Space Shuffle Implementation

2.1 Topology Construction

This section i implement Space shuffle Yu dan Qian [2016].
Initialize 2 space, called space 1 , space 2, and 9 nodes purely random into space 1, space 2 and space 3., then each server has 6 neighbors. Neighbors are chosen randomly. And the space 1 and space 2 may have different neighbors.As shown in figure.



(c) Space 1 (d) Space 2

Figure 2.1: Space

Node	0	1	2	3	4	5	6	7	8
space 1	1	222	46	101	38	52	84	89	77
space 2	33	12	193	196	32	120	152	67	4

Table 2.1: Nodes coordinates in space 1 and space 2

Node 0 Randomly generate 5 neighbors.

$$neighbor = [3, 1, 2, 5, 6] \tag{2.1}$$

2.2 Greediest Routing

The distance between 2 neighbors in each space is the difference between 2 node's value.So for each node, has 6 distance in 2 space. Then for each node has 12 distance.

$$distance = \min(|node1 - node2|, 255 - |node1 - node2|) \tag{2.2}$$

$$shortestpath = \min(distanceinspace1, distanceinspace2) \tag{2.3}$$

When node0 s receives a packet whose destination is node7, it first checks whether X t is its own coordinates. If so, node0 forwards the packet to the server whose identifier is ID. Otherwise, s selects a neighbor v such that v minimizes the circle distance; X to the destination, among all neighbors.

Routing goal: node 0 find the shortest path to coordinate 200.
Greediest Routing:
step 1 : Initialize 9 nodes and 6 neighbors in space 1 and space 2.
step 2: one node as start node.
step 3 : calculate the distance of the node to all the neighbors in space 1 and space 2.
step 4: choose the least distance in space 1 and space 2.

Node	space 1 distance	space 2	min space1 and space2
1	34	21	21
2	45	95	45
3	100	92	92
5	51	87	51
6	83	119	83

Table 2.2: The distance of node0 to neighbors in space 1 and space 2

Bibliography

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F. and Balakrishnan, H. (2001). "Chord: A scalable peer-to-peer lookup service for internet applications." *ACM SIGCOMM Computer Communication Review* **31**. 149–160

Yu, Y. and Qian, C. (2016). "Space shuffle: A scalable, flexible, and high-performance data center network." *IEEE Transactions on Parallel and Distributed Systems* **27**. 3351–3365