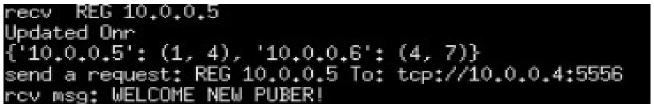
Report of Assignment 2

A. IMPLEMENTATION:

1. Publisher Registration:

Publisher register to any server in the DHT ring. This server (node A) will register this publisher (update ownership_strength table etc.) ,send message to its successor (node B) about the new publisher, node B will register this publisher , and forward this message to his successor node C, C will do the same thing until every server knows the existence of the new publisher. Publisher can join or leave anytime. Image below describes some server register a new publisher 10.0.0.5, and update the ownership table.



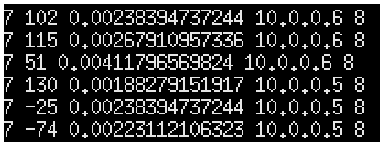
```
recv  REG 10.0.0.5
Updated Onr
{'10.0.0.5': (1, 4), '10.0.0.6': (4, 7)}
send a request: REG 10.0.0.5 To: tcp://10.0.0.4:5556
rcv msg: WELCOME NEW PUBER!
```

2. Publish information

Publishers use zmq.PUB publish information to all the DHT servers. We assign the ownership of topic based on the publisher's ip, thus each DHT node has the same ownership table. Suppose Node A received an event from publisher 1 with event key 5. A will first look at the ownership_strength table, if publisher 1 own this event, A will update the history of topic 5. Then A will check if A himself own this key. If A dose own key 5, he will publish this event to all subscribers, otherwise, A will discard this event. In this way, we can guarantee that each server has the same history, even some server failed, subscriber can still get the information they want from other server.

3. Failing Publisher

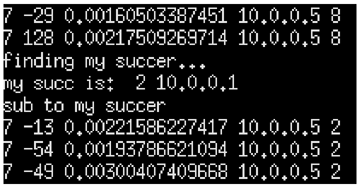
Each server will keep track of the connection with each publisher. When the server detect some publisher is not working (not receiving any new message from this publisher for some time), server will unregister this publisher and update the ownership table. Below is the received message of some subscriber, notice the original publisher is 10.0.0.6, then changed to 10.0.0.5



```
7 102 0.00238394737244 10.0.0.6 8
7 115 0.00267910957336 10.0.0.6 8
7 51 0.00411796569824 10.0.0.6 8
7 130 0.00188279151917 10.0.0.5 8
7 -25 0.00238394737244 10.0.0.5 8
7 -74 0.00223112106323 10.0.0.5 8
```

4. Subscriber

Subscriber will first register to any server in the DHT ring with the topic_Key he interested in. This node will find_successor of Key, tell the subscriber the ip of that successor. Then subscriber just SUB to that ip. If this successor failed, subscriber will register again, find the new successor, SUB to the new one. Subscribers can join or leave anytime, they can ask for history sample as in homework 1. Below is the received message of some subscriber, it used to receive message from node 8, after node 8 lost connection, it re-register, find new successor node 2.

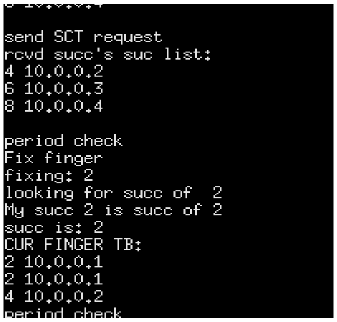


```
7 -29 0.00160503387451 10.0.0.5 8
7 128 0.00217509269714 10.0.0.5 8
finding my succer...
my succ is:  2 10.0.0.1
sub to my succer
7 -13 0.00221586227417 10.0.0.5 2
7 -54 0.00193786621094 10.0.0.5 2
7 -49 0.00300407409668 10.0.0.5 2
```

5. DHT stability (Failing Server):

Each node will maintain a finger table and a successor list (as described in CHORD paper). They will call function fix_finger(), check_successor() periodically, to make the network stable. For example, let's say node B suddenly failed, Node A can detect that his successor B failed by check_successor(), he will connect to the next successor in successor list: Node C. A will notify C, told C that I might be your predecessor. C will check it. After sometime, fix_finger() will fix the finger table, so that even though Node B failed, the DHT ring still work.

Below is an example of server check its successor list, fix finger.



```
8 10.0.0.4

send SCT request
rcvd succ's suc list:
4 10.0.0.2
6 10.0.0.3
8 10.0.0.4

period check
Fix finger
fixing: 2
looking for succ of  2
My succ 2 is succ of 2
succ is: 2
CUR FINGER TB:
2 10.0.0.1
2 10.0.0.1
4 10.0.0.2
period check
```

B. END-END MEASURE :

# Switches (Ranks)	# Publishers	# Subscribers	#Server (DHT Nodes)	Average End to End Time (seconds)
1	2	5	1	0.001357
1	2	5	4	0.002883

The first row is given from homework 1, we compare it with homework 2, where we use 4 DHT nodes. The DHT solution takes more time because it needs check the stability (fix_finger, check_successor etc.) periodically. We believe that DHT can handle larger scale than the one server solution.