COP5615 - Project3 - Bonus

Neel Rami, UFID: 7712-3151

Ma Haodi, UFID:7719-2198

Email: nrami@ufl.edu , ma.haodi@ufl.edu

Oct 1, 2018

1. Introduction

The goal of this project is to implement in Elixir using the actor model the Chord protocol and a simple object access service to prove its usefulness.

In this project we implement the network join and routing as described in the Chord paper (Section 4) and encode the simple application that associates a key (same as the ids used in Chord) with a string. We implement it using a similar API to the one described in the paper.

In this bonus part we implement node and failure models which include a node dies or a connection dies temporarily or permanently. To deal with these failures we use the methods described in Section 5 of the paper, for example, giving every node r successors instead of one and store them in the finger table. The result and findings will be shown in this report.

Assumptions made:

- We have built a simple application of doing a search for a key. This key is randomly generated and we are not associating this key with a string for simplicity.
- We are not storing keys in a node but we assume that all keys from predecessor of a certain node to that node are stored in a node. For example, if there is a node with ID 85 and its predecessor is 51, then all keys from 52 to 85 are assumed to be in node with ID 85.
- In rare cases during network Join, the process of stabilization doesn't get completed due to some reasons which can lead the program to go into a infinite loop. So in those cases, please run the program again.
- The value of r which we have used is 2log(n).
- The code runs only when number of requests is 1.

2. Implementation details

a) File explanation:

1) **GenServerMethod.ex**:

   This file contains methods which are related to GenServer and it contains all methods which are essential for Chord Protocol.

2) **proj3.ex:**

   This file serves as entry point of the project.

3) **utilityFunctions.ex:**

   This file contains some utility functions which are used frequently.

b) Approach used:

1) **Calculation of Node IDs:**
   - We have used the concept of consistent hashing to calculate the identifier for every node
   - We have used the :crypto.hash(:sha "nodeID")function to calculate SHA1
   - We calculate the hash of Node Index(i.e. 1,2,3…) to calculate the hash. Then we encode it to BASE 16 and truncate it to m bits. And then we convert the truncated result to integer and this integer serves as the node identifier for a node
2) **Calculation of Table size(m):**
   - We dynamically calculate the table size based on the number of nodes
   - We calculate $\log_2$(numNodes) to calculate the least number of bits required to represent a node and then we convert the logarithmic result to the nearest multiple of 4.
3) **Network Join and Routing:**
   - We have implemented the functionality of Network Join by creating a Chord Ring of n-1 nodes and then join the remaining node.
   - The functionalities of Network Join and Routing are implemented based on the explanation given in the research paper.
4) **Resiliency Testing of Chord Protocol:**
   - We test the resiliency by first killing number of nodes and then run the Chord Protocol.

c) Instructions for running the code:

1) For Ubuntu based systems:
   1. Go the project directory
   2. Type the command in the terminal: mix escript.build (Optional)
   3. Type the command in the terminal: ./proj3bonus 100 1 10
   4. Here the first command line argument is the number of nodes
   5. Here the second command line argument is the number of requests

6. Here the third command line argument is the number of fail nodes
7. General command: ./proj3bonus <node-num> <numRequests> <fail-nodes>

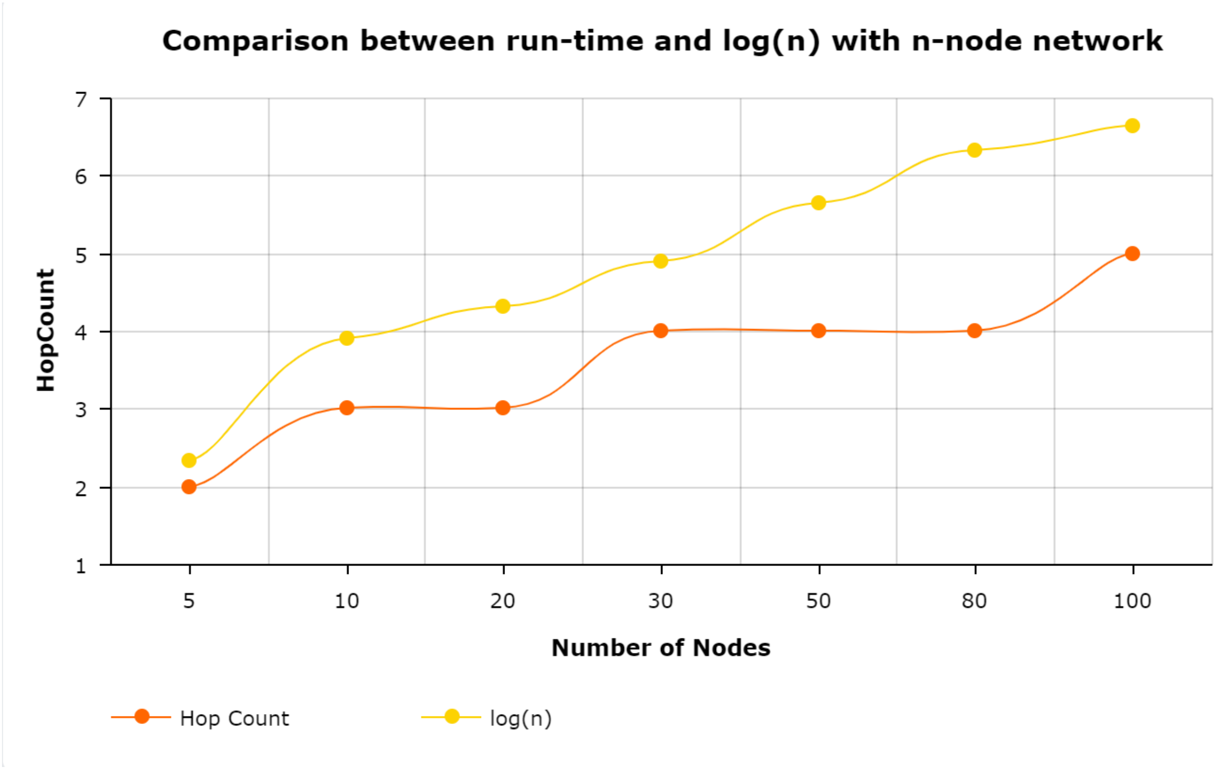
2) For Windows:
    1. Go the project directory
    2. Type the command in the cmd: mix escript.build (Optional)
    3. Type the command in the cmd: escript .\proj3bonus 100 1 10
    4. Here the first command line argument is the number of nodes
    5. Here the second command line argument is the number of requests
    6. Here the third command line argument is the number of fail nodes
    7. General command: escript .\proj3bonus <node-num> <numRequests> < Fail-nodes>

3. Result

Number of nodes	Hop Count
5	2*
10	3*
20	3*
30	4*
50	4*
80	4*
100	5*

Table 1:percentage of node failure = 10%

*: Rerun the program if the program doesn't display output



Number of nodes	Hop Count
5	2*
10	3*
20	3*
30	4*
50	4*
80	4*
100	5*

Table 2: percentage of node failure = 20%

*: Rerun the program if the program doesn’t display output.

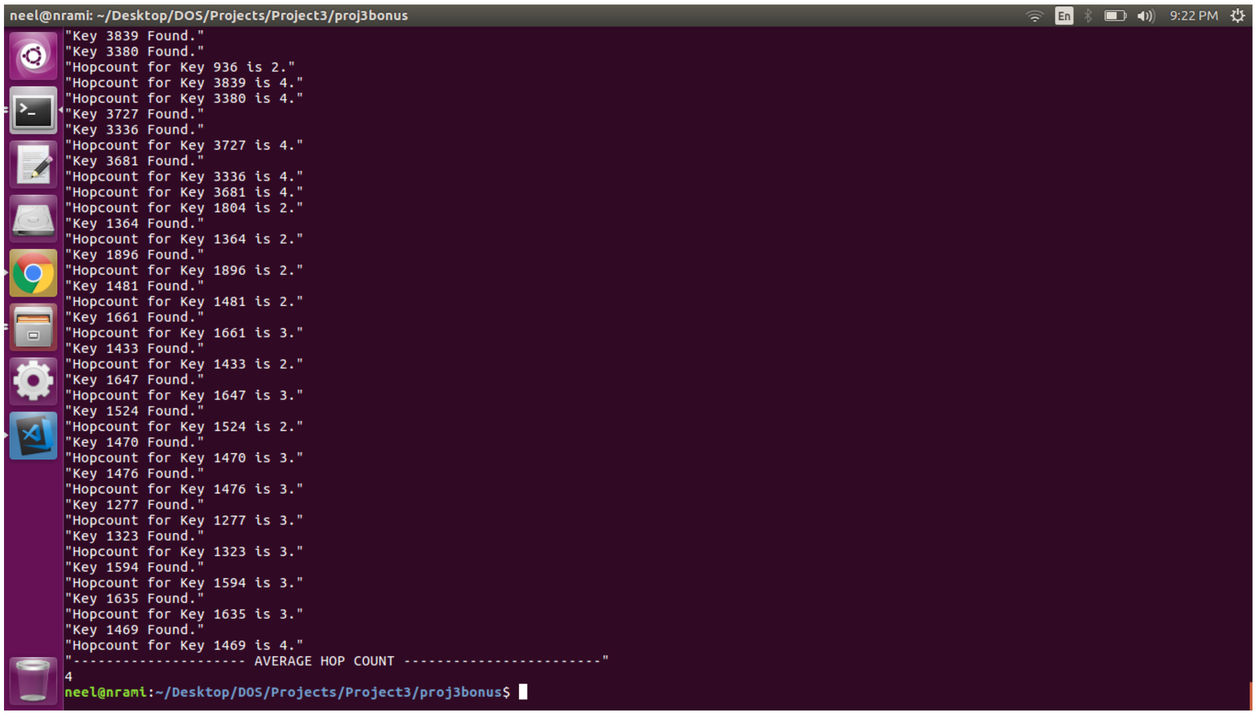
Number of nodes	Hop Count
5	**
10	**
20	**
30	**
50	**
80	**
100	**

Table 3: percentage of node failure = 90%

**: Chord Protocol failed for 90% failure model.

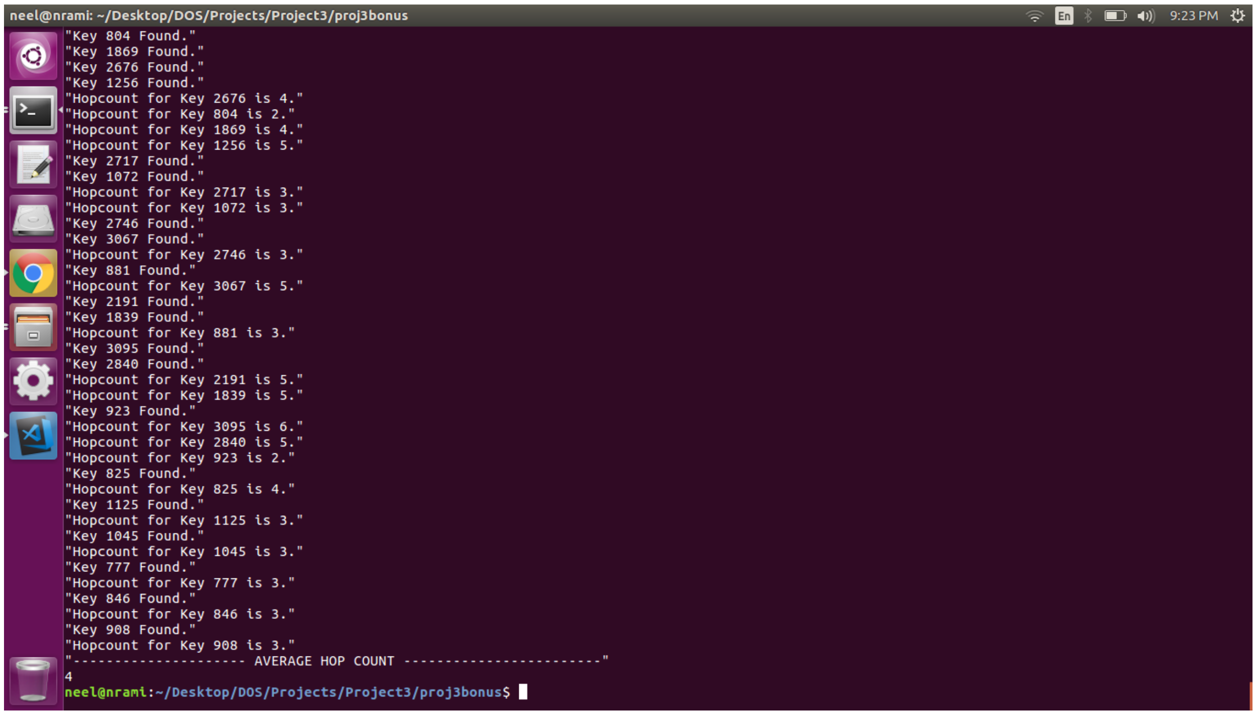
4. Output

10% Failure Model where numNodes=50



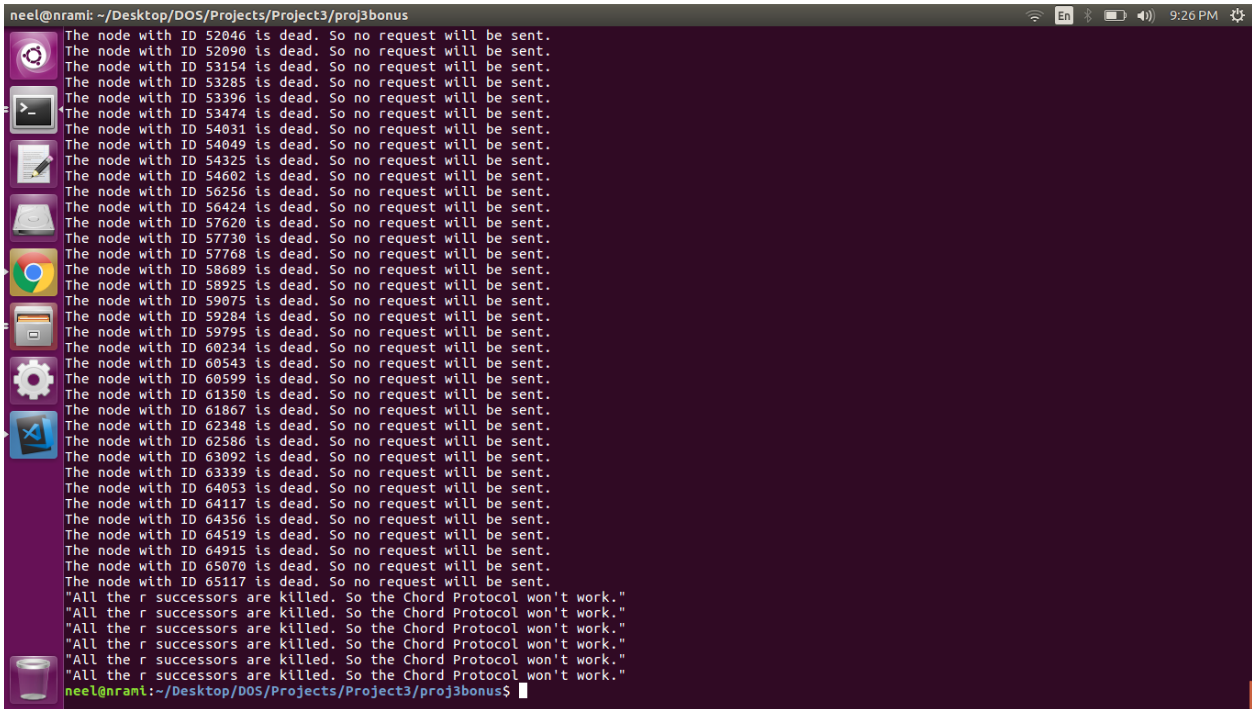
```
neel@nrami: ~/Desktop/DOS/Projects/Project3/proj3bonus
"Key 3839 Found."
"Key 3380 Found."
"Hopcount for Key 936 is 2."
"Hopcount for Key 3839 is 4."
"Hopcount for Key 3380 is 4."
"Key 3727 Found."
"Key 3336 Found."
"Hopcount for Key 3727 is 4."
"Key 3681 Found."
"Hopcount for Key 3336 is 4."
"Hopcount for Key 3681 is 4."
"Hopcount for Key 1804 is 2."
"Key 1364 Found."
"Hopcount for Key 1364 is 2."
"Key 1896 Found."
"Hopcount for Key 1896 is 2."
"Key 1481 Found."
"Hopcount for Key 1481 is 2."
"Key 1661 Found."
"Hopcount for Key 1661 is 3."
"Key 1433 Found."
"Hopcount for Key 1433 is 2."
"Key 1647 Found."
"Hopcount for Key 1647 is 3."
"Key 1524 Found."
"Hopcount for Key 1524 is 2."
"Key 1470 Found."
"Hopcount for Key 1470 is 3."
"Key 1476 Found."
"Hopcount for Key 1476 is 3."
"Key 1277 Found."
"Hopcount for Key 1277 is 3."
"Key 1323 Found."
"Hopcount for Key 1323 is 3."
"Key 1594 Found."
"Hopcount for Key 1594 is 3."
"Key 1635 Found."
"Hopcount for Key 1635 is 3."
"Key 1469 Found."
"Hopcount for Key 1469 is 4."
"---------------------- AVERAGE HOP COUNT -----------------------"
4
neel@nrami:~/Desktop/DOS/Projects/Project3/proj3bonus$
```

20% Failure Model where numNodes=50



```
neel@nrami: ~/Desktop/DOS/Projects/Project3/proj3bonus
"Key 804 Found."
"Key 1869 Found."
"Key 2676 Found."
"Key 1256 Found."
"Hopcount for Key 2676 is 4."
"Hopcount for Key 804 is 2."
"Hopcount for Key 1869 is 4."
"Hopcount for Key 1256 is 5."
"Key 2717 Found."
"Key 1072 Found."
"Hopcount for Key 2717 is 3."
"Hopcount for Key 1072 is 3."
"Key 2746 Found."
"Key 3067 Found."
"Hopcount for Key 2746 is 3."
"Key 881 Found."
"Hopcount for Key 3067 is 5."
"Key 2191 Found."
"Key 1839 Found."
"Hopcount for Key 881 is 3."
"Key 3095 Found."
"Key 2840 Found."
"Hopcount for Key 2191 is 5."
"Hopcount for Key 1839 is 5."
"Key 923 Found."
"Hopcount for Key 3095 is 6."
"Hopcount for Key 2840 is 5."
"Hopcount for Key 923 is 2."
"Key 825 Found."
"Hopcount for Key 825 is 4."
"Key 1125 Found."
"Hopcount for Key 1125 is 3."
"Key 1045 Found."
"Hopcount for Key 1045 is 3."
"Key 777 Found."
"Hopcount for Key 777 is 3."
"Key 846 Found."
"Hopcount for Key 846 is 3."
"Key 908 Found."
"Hopcount for Key 908 is 3."
"---------------------- AVERAGE HOP COUNT -----------------------"
4
neel@nrami:~/Desktop/DOS/Projects/Project3/proj3bonus$
```

90% Failure Model



```
neel@nrami: ~/Desktop/DOS/Projects/Project3/proj3bonus
The node with ID 52046 is dead. So no request will be sent.
The node with ID 52090 is dead. So no request will be sent.
The node with ID 53154 is dead. So no request will be sent.
The node with ID 53285 is dead. So no request will be sent.
The node with ID 53396 is dead. So no request will be sent.
The node with ID 53474 is dead. So no request will be sent.
The node with ID 54031 is dead. So no request will be sent.
The node with ID 54049 is dead. So no request will be sent.
The node with ID 54325 is dead. So no request will be sent.
The node with ID 54602 is dead. So no request will be sent.
The node with ID 56256 is dead. So no request will be sent.
The node with ID 56424 is dead. So no request will be sent.
The node with ID 57620 is dead. So no request will be sent.
The node with ID 57730 is dead. So no request will be sent.
The node with ID 57768 is dead. So no request will be sent.
The node with ID 58689 is dead. So no request will be sent.
The node with ID 58925 is dead. So no request will be sent.
The node with ID 59075 is dead. So no request will be sent.
The node with ID 59284 is dead. So no request will be sent.
The node with ID 59795 is dead. So no request will be sent.
The node with ID 60234 is dead. So no request will be sent.
The node with ID 60543 is dead. So no request will be sent.
The node with ID 60599 is dead. So no request will be sent.
The node with ID 61350 is dead. So no request will be sent.
The node with ID 61867 is dead. So no request will be sent.
The node with ID 62348 is dead. So no request will be sent.
The node with ID 62586 is dead. So no request will be sent.
The node with ID 63092 is dead. So no request will be sent.
The node with ID 63339 is dead. So no request will be sent.
The node with ID 64053 is dead. So no request will be sent.
The node with ID 64117 is dead. So no request will be sent.
The node with ID 64356 is dead. So no request will be sent.
The node with ID 64519 is dead. So no request will be sent.
The node with ID 64915 is dead. So no request will be sent.
The node with ID 65070 is dead. So no request will be sent.
The node with ID 65117 is dead. So no request will be sent.
"All the r successors are killed. So the Chord Protocol won't work."
"All the r successors are killed. So the Chord Protocol won't work."
"All the r successors are killed. So the Chord Protocol won't work."
"All the r successors are killed. So the Chord Protocol won't work."
"All the r successors are killed. So the Chord Protocol won't work."
"All the r successors are killed. So the Chord Protocol won't work."
neel@nrami:~/Desktop/DOS/Projects/Project3/proj3bonus$
```

5. Interesting finding

- When the fail nodes is greater than 2log(n), the Chord Protocol will fail.
- The Chord Protocol is resilient as number of hop counts is almost similar when there is no failure.