Project 3 Bonus (Chord P2P System with Fail Nodes)

Group Members:	**UF ID:**
Kalpana Sathya Ponnada	5246-1920
Tharun Bhupathi	8328-9089

Bonus Project Goal:

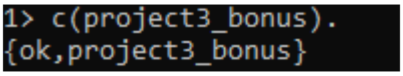
- In the main project, we implemented Chord P2P system without any failure nodes.
- To effectively analyze the robustness of the chord network, we introduce some faulty nodes in the network (number of faulty nodes taken as input from the user).
- When a node dies, the connection dies permanently.
- How these faulty nodes impact the average hops taken to find the key is the main goal of this project.
- Here, we take the number of failure nodes as an argument from the user and those many nodes are randomly chosen from the node list and made faulty.
- We determine the average number of hops taken to find the key.

Input:

Number of Nodes, Number of Requests, Number of Fail nodes

Steps to compile:

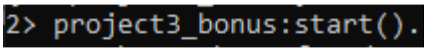
- Compile the following files in erlang shell
  project3_bonus.erl



```
1> c(project3_bonus).
{ok,project3_bonus}
```

Steps to Run:

- project3_bonus:start()

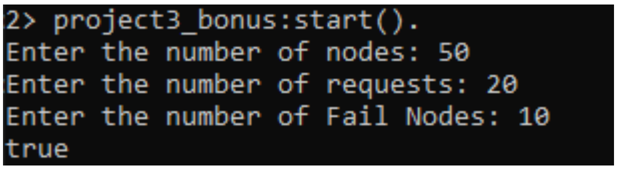


```
2> project3_bonus:start().
```

Parameters:

We take following as the command line input from the user:

1) Number of Nodes
2) Number of Requests
3) Number of Fail Nodes



```
2> project3_bonus:start().
Enter the number of nodes: 50
Enter the number of requests: 20
Enter the number of Fail Nodes: 10
true
```

System Configuration:

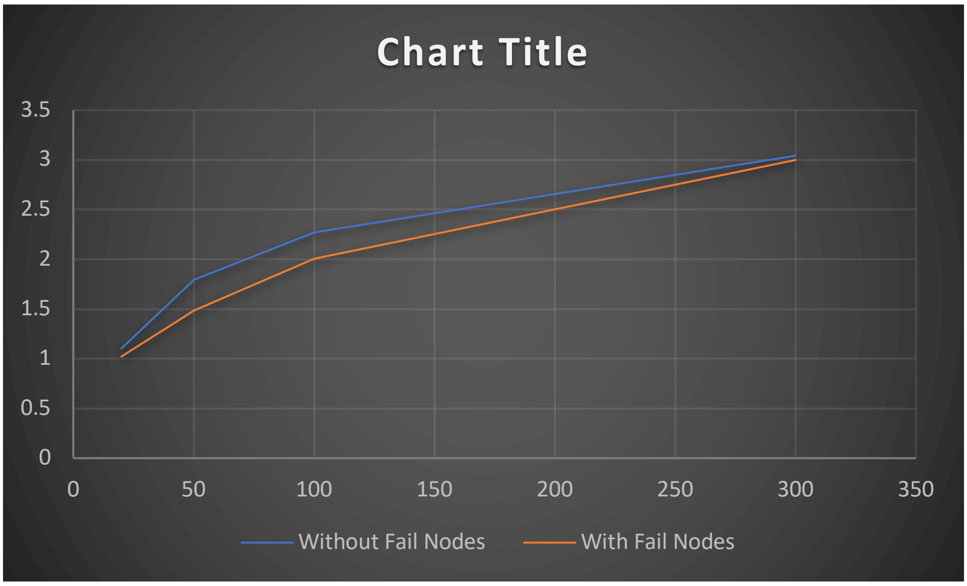
MacBook Pro (13-inch, 2017, Two Thunderbolt 3 ports)
2.3 GHz Dual-Core Intel Core i5 processor

Observations:

- **Lookup (Figure) Table after the introduction of Fail Nodes:**
  a) Even when the faulty nodes are introduced, the system can perform as expected because, we have a notify method which informs the predecessor and successor of the node about its exit from the chord ring. This way, the system is still stable
  b) After these nodes left the ring, the system again becomes stable with stabilize and fix_finger methods and hence the average hop time is still within the limits of $\log_2 N$.(Where N is the total number of nodes of the chord ring which is given as user input).
- **Node handling the situation of Finger Table with at least one failed node:**
  a) Let's assume that the ith index in Finger Table of a node contains a failed node. We will get to know that it is failed one (i.e., not alive) with the help of check_predecessor method.
  b) When this situation happens, we return the node present in the i-1th index of Finger Table which eventually helps us reaching the node which is successor.

- **Stabilize method updating the successor:**
  a) This process is enhanced which stabilizes each node's successor list.
  b) This is handled for each node N with the help of successor list for the node's successor. The last entry of the list is removed and adding the successor to the modified successor list.
- It can be seen that the chord ring's correctness is still intact even after the n=introduction of failure nodes. The below graph shows the variation of average hops vs Number of nodes in both without failure and after failure introduction cases.
- After the chord achieved stability after deleting the nodes, the average number of hops taken are still within the limit of $\log_2 N$.

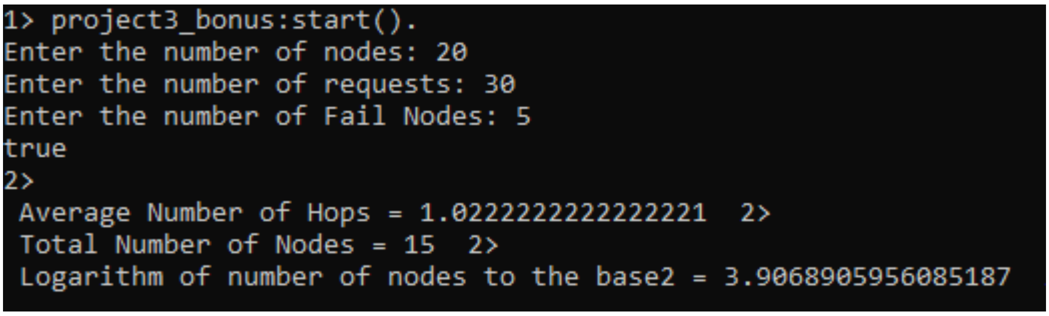
Variation of Avg Number of Hops with the Number of nodes for both i. Without Fail Nodes & ii. With Fail Nodes.



Output:

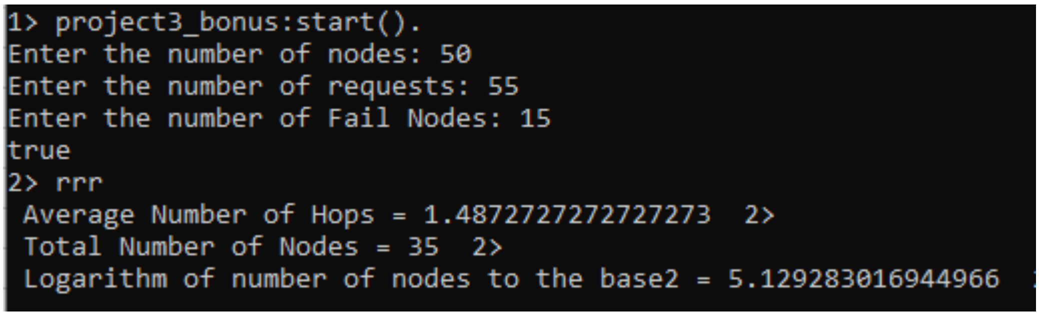
- We displayed the average number of hops for different total number of nodes with given number of fail nodes.
- Mean of the hops for the total number of peers along with the logarithm of number of peers.

a) Number of peers=20



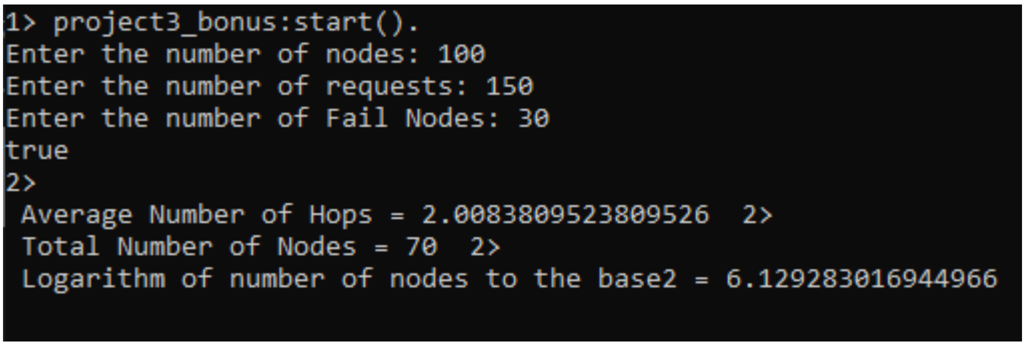
```
1> project3_bonus:start().
Enter the number of nodes: 20
Enter the number of requests: 30
Enter the number of Fail Nodes: 5
true
2>
 Average Number of Hops = 1.0222222222222221   2>
 Total Number of Nodes = 15   2>
 Logarithm of number of nodes to the base2 = 3.9068905956085187
```

b) Number of peers=50



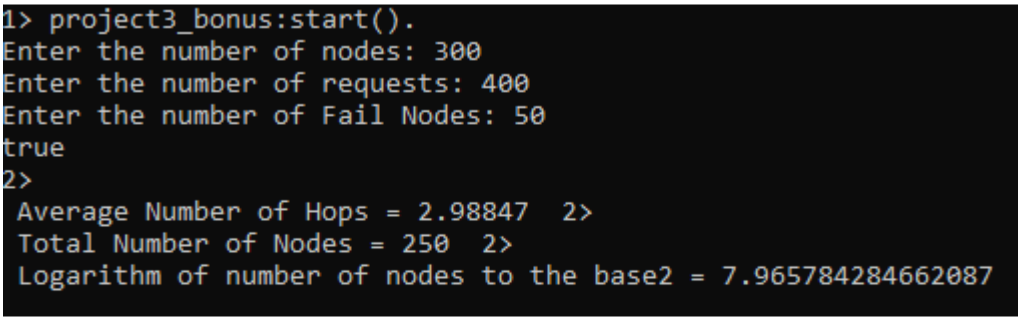
```
1> project3_bonus:start().
Enter the number of nodes: 50
Enter the number of requests: 55
Enter the number of Fail Nodes: 15
true
2> rrr
 Average Number of Hops = 1.4872727272727273   2>
 Total Number of Nodes = 35   2>
 Logarithm of number of nodes to the base2 = 5.129283016944966
```

c) Number of peers=100



```
1> project3_bonus:start().
Enter the number of nodes: 100
Enter the number of requests: 150
Enter the number of Fail Nodes: 30
true
2>
 Average Number of Hops = 2.0083809523809526  2>
 Total Number of Nodes = 70  2>
 Logarithm of number of nodes to the base2 = 6.129283016944966
```

d) Number of peers=300



```
1> project3_bonus:start().
Enter the number of nodes: 300
Enter the number of requests: 400
Enter the number of Fail Nodes: 50
true
2>
 Average Number of Hops = 2.98847  2>
 Total Number of Nodes = 250  2>
 Logarithm of number of nodes to the base2 = 7.965784284662087
```