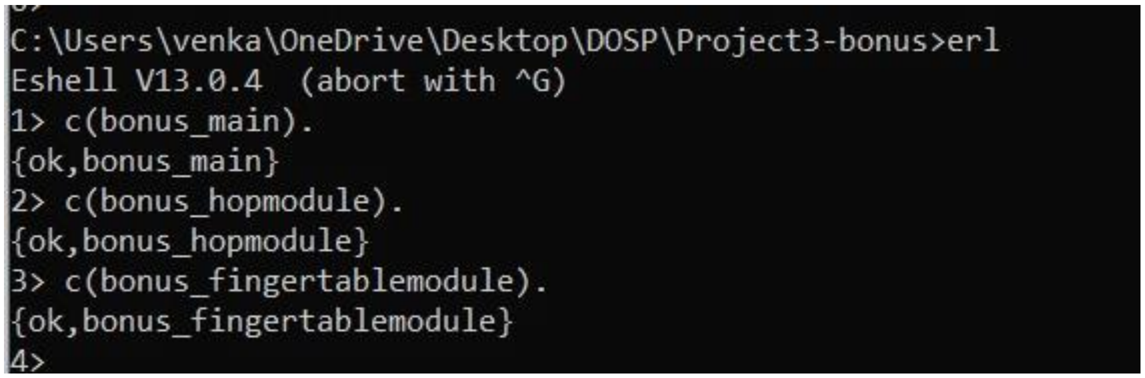
COP-5615 Project 3 Bonus README : Chord Protocol Simulation

Group members:

Venkata Ramana Pamarthy (UFID: 76176980)
Sreeram Aditya Potukuchi (UFID: 31395861)

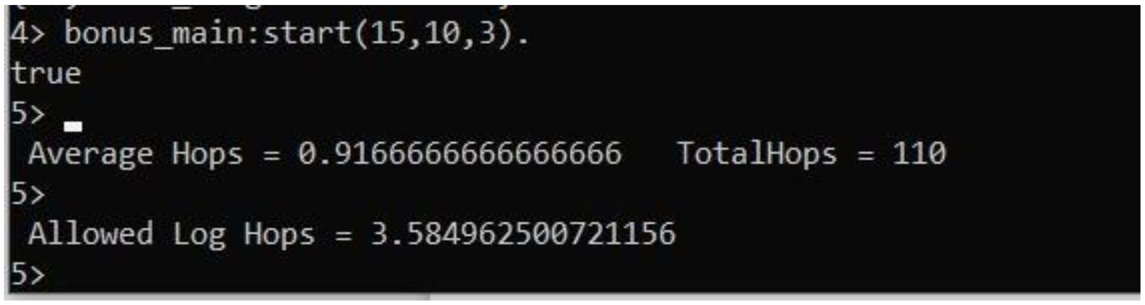
Instructions to Run:

- To compile the program, use the following command:
  c(bonus_main).
  c(bonus_hopmodule).
  c(bonus_fingertablemodule).



```
C:\Users\venka\OneDrive\Desktop\DOSP\Project3-bonus>erl
Eshell V13.0.4  (abort with ^G)
1> c(bonus_main).
{ok,bonus_main}
2> c(bonus_hopmodule).
{ok,bonus_hopmodule}
3> c(bonus_fingertablemodule).
{ok,bonus_fingertablemodule}
4>
```

- To execute the program, use the following command:
  bonus_main:start(NumberOfNodes, RequestLimit,FailureCount).
  where NumberOfNodes = Number of Nodes in the Chord Ring
  RequestLimit = Number of requests per peer
  FailureCount= Amount of failure induces (number of nodes to be killed)



```
4> bonus_main:start(15,10,3).
true
5>
 Average Hops = 0.9166666666666666   TotalHops = 110
5>
 Allowed Log Hops = 3.584962500721156
5>
```

<u>What is working?</u>

- In the normal project, we have implemented Chord with zero failures. But what if there is a failure in the network? This bonus implementation talks about dealing with the failure of nodes and its observations.
- We have taken the failure count as an input during the program execution and successfully executed node failures in the project.
- There is a visible change in the Avg. num of hops before and after the failure.

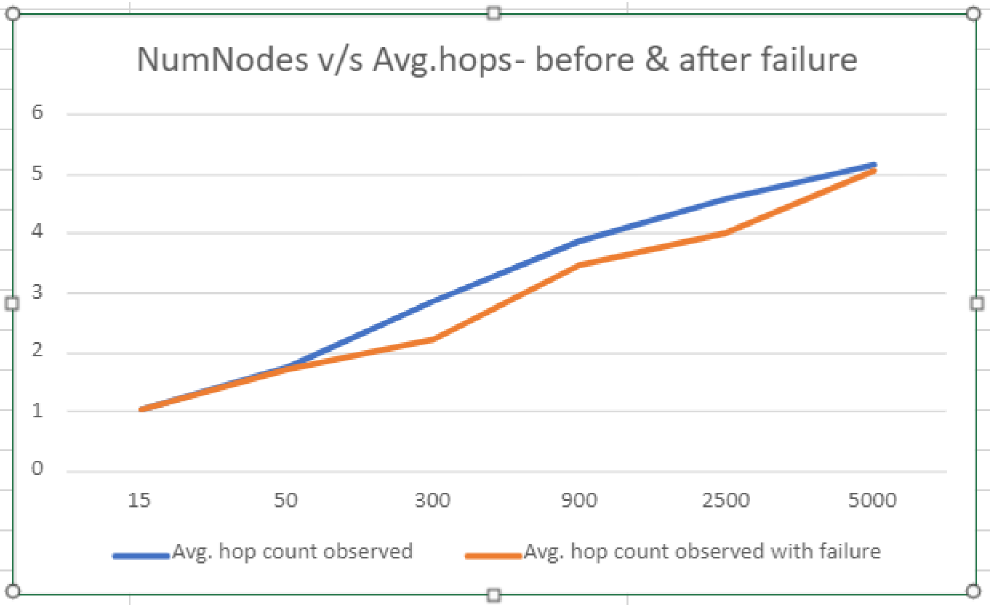
<u>How it works:</u>

- In this bonus implementation, we have added a third parameter to the input function which is the "FailureCount". This is the amount of failure, in terms of the number of nodes that we want to induce into our program.
- This input is used to then create failures., i.e., kill/destroy the given number of nodes from the network to check its performance.
- Now, when the program executes, we will see a clear change in the output. Since there are fewer nodes now, convergence is reached quicker than the normal case.
- Observations on the same are listed below.

<u>Observations:</u>

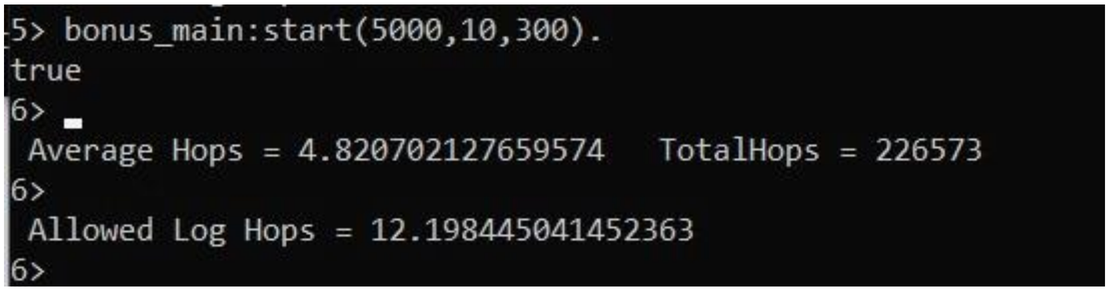
- When we induce "FailureCount" number of failures, we observe that the time taken to converge is lesser that usual(when there was no failure).
- This is probably because, as the number of nodes leave the chord, the lookups become faster and hence the time taken to converge also decreases.
- This proves that Chord is quite stable even when the nodes are joining/leaving the network on a regular/irregular basis.

Num of Peers	**Number of Messages**	**Avg. hop count observed**	**Avg. hop count observed with failure**
15	10	1.04	1.033
50	10	1.756	1.729
300	10	2.870	2.212
900	10	3.882	3.482
2500	10	4.578	4.010
5000	10	5.157	5.083



- Clearly from the above graph, we see that the failure case converges much quicker than the normal case.
- These were the observations we had from our implementation.

We managed to create a chord network and create the simulation with 5000 nodes, 10 messages per node and 300 failure nodes.



```
5> bonus_main:start(5000,10,300).
true
6> ▬
 Average Hops = 4.820702127659574   TotalHops = 226573
6>
 Allowed Log Hops = 12.198445041452363
6>
```