PROJECT 3

TEAM MEMBERS:

NEHA SINGH
RUCHI DESAI

WHAT IS WORKING:

1. Master creates actors which will serve as pastry nodes.
2. Master sends data necessary for forming the initial pastry network to the first 32 nodes (in the order in which they were created).
3. Once initialization is complete, the rest of the nodes join this network (one by one) as per the algorithm mentioned in the paper.
4. Once all the remaining nodes have successfully joined the pastry network, master asks all the nodes to start routing.
5. Each actor generates a random key after gap of 1 second and routes it as per the algorithm mentioned in the paper.
6. When all the nodes have finished routing the number of requests given as argument, the program displays the average number of hops and terminates.
7. The index of a node specifies the order in which nodes were created and added to the list of nodes. Node Id is generated by calculating the MD5 hash of this index. Since MD5 hash of a string is completely random, we have ensured that nodes close to each other by index do not have similar Node Ids
8. Proximity metric is the above mentioned index of the node.
9. b = 4, therefore Node Ids are hexadecimal.
10. Size of leafset = 16, size of neighborhood set = 32

LARGEST NETWORK THAT WE MANAGED TO DEAL WITH

numNodes = 3000, numRequest = 10

Output:

```
Initializing...
Nodes have started joining
Join process completed. Begin Routing
Routing complete
Total no of hops = 71630
Total no of requests = 30000
Average no of hops =
2.3876666666666666
Shutting system down...
```

numNodes = 1000, numRequests = 300

Output:

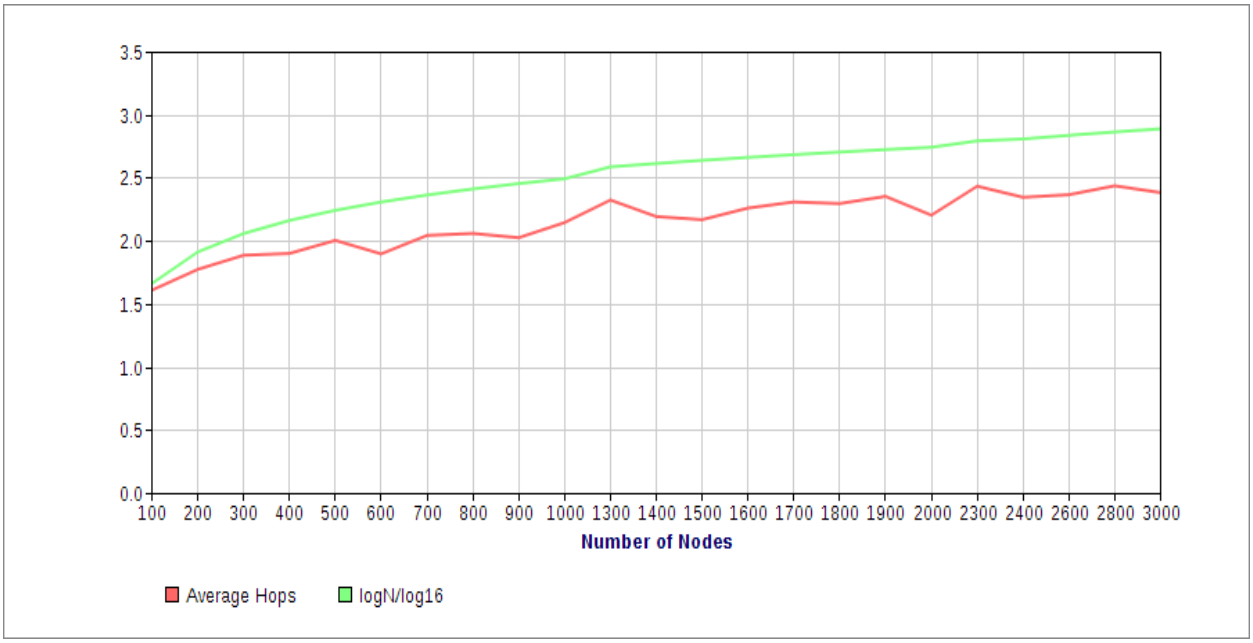
```
Initializing...
Nodes have started joining
Join process completed. Begin Routing
Routing complete
Total no of hops = 650252
Total no of requests = 300000
Average no of hops =
2.1675066666666667
Shutting system down...
```

Following table shows the values of average hops obtained for various values of number of nodes:

no of nodes	average hops*	log n (base 16)
100	1.608175	1.660964047
200	1.772916667	1.910964047
300	1.885	2.057204673
400	1.89925	2.160964047
500	2.002725	2.241446071
600	1.895845833	2.307204673
700	2.042333333	2.362802778
800	2.057875	2.410964047
900	2.024185185	2.453445298
1000	2.143976667	2.491446071
1300	2.322846154	2.586073977
1400	2.191857143	2.612802778
1500	2.1672	2.637686696
1600	2.2591875	2.660964047
1700	2.307588235	2.682829758
1800	2.294944444	2.703445298
1900	2.351842105	2.722945926
2000	2.202891667	2.741446071
2300	2.432565217	2.791854536
2400	2.34475	2.807204673
2600	2.365230769	2.836073977
2800	2.435821429	2.862802778
3000	2.3818	2.887686696

* average hops in the table above has been calculated by taking the aggregate of average hops obtained for numRequests = 2, 10, 20

GRAPH OF NUMBER OF NODES VS AVERAGE HOP COUNT ALONG WITH $\log_{16}N$



BONUS IMPLEMENTATION

We have been able to implement the failure of a single node and test the resilience of the network in that case.

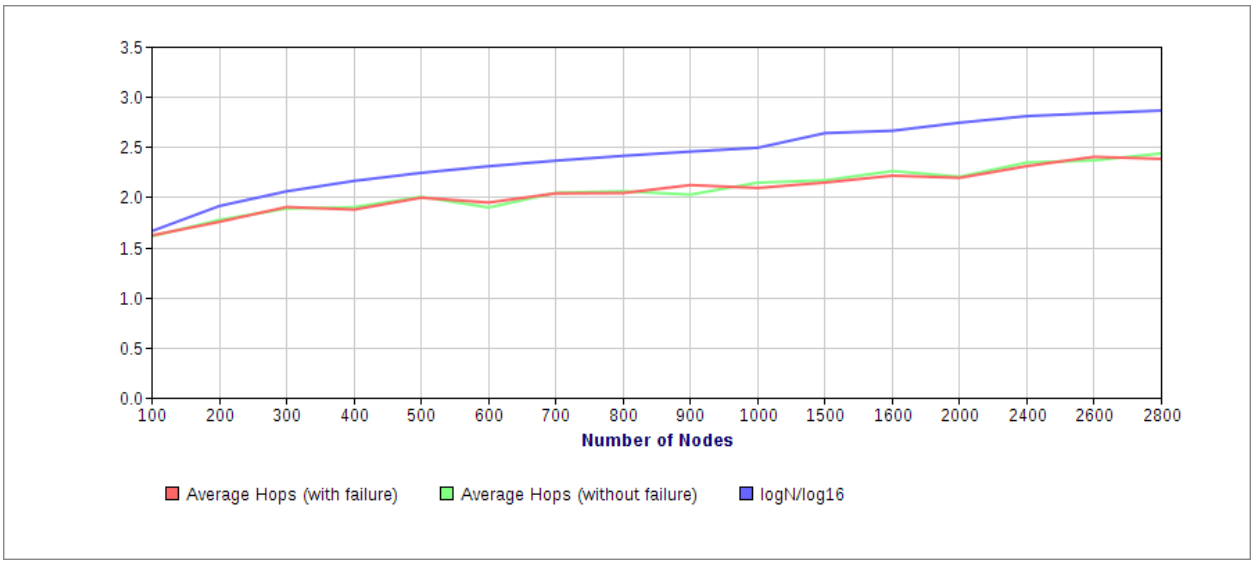
Implementation details: The node that is to fail is pre-decided and it assumed to fail after join is complete (i.e. during actual routing of messages). The other nodes referring to this failed node update their leaf sets, routing tables and neighborhood sets as mentioned in the paper.

LARGEST NETWORKS HANDLED WITH SINGLE NODE FAILURE

```
numNodes = 2800, numRequests = 2

Output:

Initializing...
Nodes have started joining
Join process completed. Begin
Routing
Routing complete
Total no of hops = 134258
Total no of requests = 56000
Average no of hops =
2.397464285714286
Shutting system down...
```

```
numNodes = 1000, numRequests = 300

Output:

Initializing...
Nodes have started joining
Join process completed. Begin
Routing
Routing complete
Total no of hops = 635983
Total no of requests = 300000
Average no of hops =
2.1199433333333335
Shutting system down...
```

As we can see in the graph below, there is very little or no effect of single node failure on routing of requests using pastry overlay network.



STEPS TO EXECUTE

Executing from the command prompt:

- Go to the path where the program is located
  e.g.: cd C:\Users\neha\Desktop\Semester related\semester2\DOS\project\project3
- Then type scala project3.scala 500 10 in the command prompt.

SOFTWARE VERSIONS

Scala version: 2.11.2.
Akka version: 2.3.6