Bonus Report

Chord is an algorithm for implementing a distributed hash table. A distributed hash table stores key-value pairs into nodes which are spread across the network. To achieve this, we first calculate the hash of each node and then truncate it to m bits, where m is a system variable. The value of m should satisfy the following inequality: numNodes < 2^m. The identifiers of files are also calculated in the same way.

To achieve fault tolerance we implemented the stabilization algorithm, as described in the original paper, in the following gen-server call:
def handle_info(:fix_finger_table, current_map)
The above function is called periodically and recursively for each node in the network and it updates the finger table entries of the node. Therefore addition and deletions of nodes do not affect the file lookup procedure.

Further to achieve fault tolerance and load balance. While storing the files, the file is also stored in the predecessor and one successor of the node. This ensures that if the node holding the file fails, the file still remains in the system with the predecessor and the successor. Moreover, since the file now exists in at least 3 different nodes there is some load balancing where the queries for the file are handled by all three nodes.

In addition to that, we have added **r** successors to each node where **r = 2 * $\log_2$(n)**. This is because having r successors suffices to maintain lookup correctness with high probability.

Demonstration:
To demonstrate fault-tolerance we performed the file lookup operation twice: once in case of no failure and the other in case of a node failure. Each node queries all the files in the network and then the average number of hops required to search the file is printed.

Interesting points of observation:
* An interesting point of observation is that by having 2 * $\log_2$(n) successors for each node, the files are found in near logarithmic time even when 50% of the nodes have failed.
* Another interesting point of observation is that the average number of hops is always ~O($\log_2$(numNodes)).