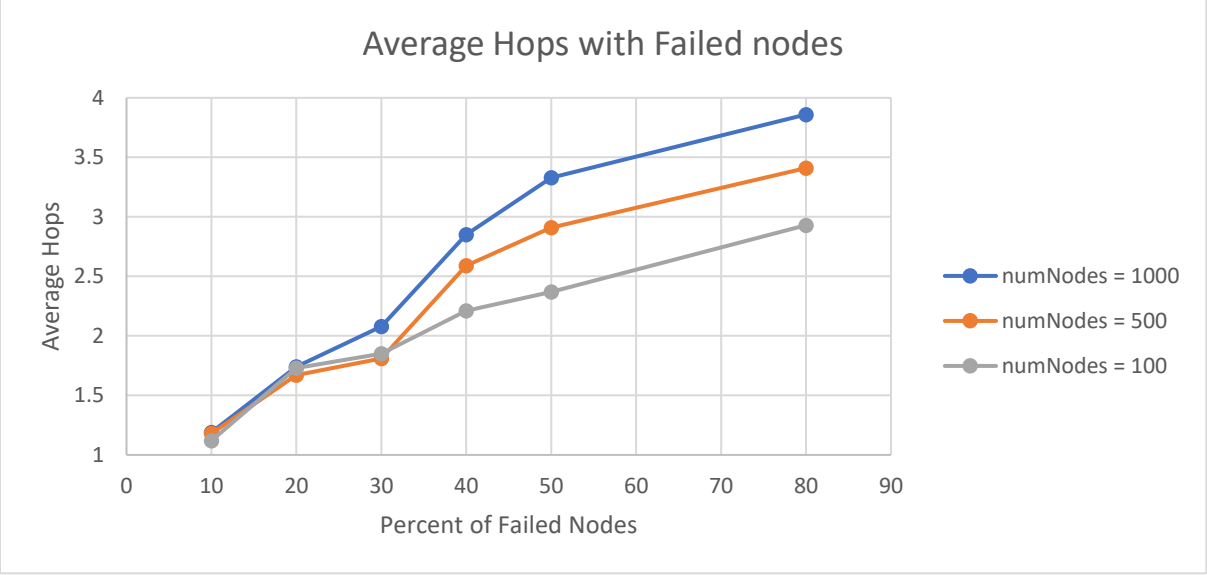
Project 3: Chord Protocol implementation using Elixir - BONUS

Group info: Priyam Saikia (9414-5292), Noopur R K (1980-9834)

Chord Protocol was implemented using Actor Model in Elixir utilizing the genServer/Supervisor-worker feature with functions based on the MIT paper on Chord Protocol. We have implemented the failure model for the Chord Protocol as mentioned on the Section 5 of the given paper. We check the resiliency of our protocol after implementation of this failure model.

Highlights:

1. The number of failed nodes is determined by the input that the user provides in the commandline. As per the input, we kill off that percentage of random nodes. We utilize the state of the processes to check if a node has failed and consequently maintaining the chord protocol without the failed nodes. This is done using the finger table and the successor list.
2. For every node, a successor list is maintained as mentioned in the paper. Whenever a node dies, the next node in the successor list is used to make the successor-predecessor relation. This is done using the join function in the chord protocol and the finger table is configured accordingly.
3. As the number of failed nodes (which is input in percent) increases, the average hop increases too. Basically, it takes more time to reach the destination as the failed nodes causes unsuccessful searches due to outdated entries in the finger table. This has been confirmed by our tests on various number of nodes as input.



4. We see that the overall time taken is still logarithmic even though the message delivery takes more time. This can be seen by checking the values of average hops for lower number of nodes vs the higher number of nodes.