CS606-P2 (Bully Election)

In this repository, I have implemented the bully election algorithm using dynamic configurations. The files and folders structures are as follow. First, the main implementation exists in `bullyElection.py` file and the configurations of the file exists in `constCS.py` file. Second, sample outputs have been uploaded to the `output` folder as a proof of correct implementation and that the program is running successfully.

Focusing on `bullyElection.py` file, first the program takes in the main (lines 117-133) the number of nodes and the random value `k`. Then, it creates `n` nodes with a random election ids in lines 123-127. To be specific, each node is setup as a server to listen to incoming requests from other nodes in lines 126-127. After that, the program waits for one second to make sure that all the nodes have been setuped and the listener functions have been initialized successfully (line 129). Then, a random node is selected to start the election process (lines 131-133). This node sends a message to all the nodes that it knows that they have higher election id. To be specific, it is implemented using threads for efficiency, as each thread will be blocked for some time. Each thread executed the function `send_election_message` (lines 87-114). If it received `OK` message, it means that a node with a higher ID exists. If the timeout happens, it means that this node is not responding anymore or it does not have a higher ID that it. Each node receives the election message (acts as a server in this case) in the lines (29-34). For each election message it receives, it creates a new thread that executes the function `server_request_handler` to handle this message in lines (37-51). The node compares its id with the ID indicated in the sent message. If it has a higher ID or the ID is the same with a similar node, then I use the node character name to decide. For example, if nodes `A` and `B` has the same election ID `3`. Node `B` will win the election because alphabetically, it is greater than node `A` (line 48).

Furthermore, many examples have been shown in the folder `output` to prove the correctness of the implementation. For example, `output/CS-606-p2-running-output1.png` file shows the output under the following configurations: `N=3, k=3`. This example shows that all the three nodes have the same election ID = 2. However, node C is selected as the coordinator at the end. Moreover, `output/CS-606-p2-running-output3.txt` shows the output for 5 nodes to prove that my scheme is generic. It is in a txt file format, because the output is too long to fit into one image.