Pelinsu Saraç - 28820

CS 403 Distributed Systems PA3 Report

Program starts with parsing the arguments that were given at running the program paxos.py, via the argparse library of the Python. Validity of these arguments is checked. Then, Paxos algorithm starts by creating the Paxos nodes according to the arguments given. Main process then waits for Paxos nodes to terminate.

Each Paxos node starts with trying to understand whether she is the proposer node of this round or not, by round mod number of nodes formulation. Before that, each node initializes the necessary ZeroMQ sockets for pushing and pulling messages via pipeline.

If the node is proposer, it starts the current round by broadcasting a start message through the ZeroMQ push socket. broadcastFailure function is used to broadcast messages with a probability of failure, which is given at the beginning of the program. After the broadcast, proposer waits for node responses. Nodes can crash or they can join. If they can join, their maxVotedRound and maxVotedValue values are evaluated while receiving the messages to decide the possible proposal value in the next stage. If there are enough joining nodes to form a join quorum, proposer continues with the proposal stage, otherwise round is continuing with the next. For the proposal, if any of the nodes has recently voted, that value is proposed. If not, default value of the proposer, which was assigned at creation of the process, is proposed. Proposal is done over broadcasting with failure.

Finally, proposer waits for the node votes. Some nodes may crash. If they did not, they sent a vote message. If the number of votes is enough to form a vote quorum, the proposed value is set as the decision of the round. If not, nodes continue with next round.

Acceptors on the other hand, first waits for proposer to start the round. If they receive start, they try to join but they may crash with a probability. Nodes will wait for another message. This can be roundchange. If that is the case, they continue to next round. Otherwise, it can be proposal, then nodes try to vote but they can again fail.

Synchronization among the nodes was achieved via the Barrier object of multiprocessing library of Python. Nodes call wait method of barrier object at the end of phases to synchronize, as stated in the assignment paper.

Randomness of the failure was achieved in the way below:

- A random value is decided between 0 and 1
- If the value if smaller than the probability given at the start of the program, it means fail
- Otherwise, node is alive and can send message.