MultiPaxos

File Structure

- acceptor.py: implementation of acceptor class
- proposer.py: implementation of proposer class
- learner.py: implementation of learner class
- replica.py: implementation of replica, which interacts with acceptor, proposer and learner class
- client.py: implementation of client class
- common.py: containing common functions used by all classes
- driver.py: a script to spawn 2f+1 replicas given a configuration file
- manual_client.py: code for starting client manually
- manual_server.py: code for starting replica manually
- configs/config.txt: a template file for configuring replica

Script Mode

To run script mode, please use driver.py.

parameters:

```
--f: integer, number of tolerated failures

--config_path: string,  path of configuration file each line containing o
ne pair of ip and port
--log_dir: string, path of log directory for learner to write chat log to
--skip_slot (optional): integer, the array index that primary needs to sk
ip its proposal and propose the next index instead
--msg_loss (optional): integer, the probability of message loss (in perce
ntage) for simulating asynchronous network
--rand_seed (optional): integer, integer used to set random seed for mimi
cing real client behaviors
```

driver.py defines several test settings in TEST_TYPE variable to start the project.

```
'SINGLE CLIENT SINGLE REQ': single client sending single message
'SINGLE CLIENT MULTIPLE REQ': single client sending several messages
'MULTIPLE CLIENT SINGLE REQ': multiple clients sending single message
'MULTIPLE CLIENT MULTIPLE REQ': multiple client sending several messages
'PROPOSER 0 FAIL BEFORE PROPOSAL': proposer 0 fails before any proposal
'PROPOSER 0 AND 1 FAIL BEFORE PROPOSAL': proposer 0 and 1 fail before any
 proposal
'PROPOSER 0 FAIL AFTER PROPOSAL': proposer 0 fails after proposing one pr
oposal
'STRESS TEST 0': multiple clients sending 100 messages, each with random
time interval
```

Manual Mode:

Manual Replica

Start manual replica by running manual_replica.py.

Notice that replica will undergo a warm-up stage, where it has to receive all ready-up messages from all other 2*f replicas in order to proceed. This is to ensure that the setting is correct in the beginning.

parameters:

```
--f: integer, number of tolerated failures
--replica_id: integer, id for this replica
--config_path: string,  path of configuration file each line containing o
ne pair of ip and port
--log_dir: string, path of log directory for learner to write chat log to
--skip_slot (optional): integer, the array index that primary needs to sk
ip its proposal and propose the next index instead
--msg_loss (optional): integer, the probability of message loss (in perce
ntage) for simulating asynchronous network
```

Manual Client

Start manual client by running manual_client.py

As a manual client, you will be able to iteratively type in and send out message. The client will

wait until a request complete notification is received from any replica.