CSE535: Distributed Systems

Project: DiemBFT v4 Consensus Algorithm Phase 4

Team Name: PreeManiSai

Manikanta Sathwik Yeluri,
Preetham Kumar Reddy Katta,
Sai Bhavana Ambati

Test Report

All the test cases follow the same format for the config file; ledger files for validators; and log files for all validators as well as clients mentioned in the following format.

Config File:

‘*config/config.da’* contains configurations of Diem BFT to be executed.

Test Case:

‘*tests/test_case_*.json’* contains configurations of test cases to be executed.

Ledger File:

For each test case in each configuration file, each validator creates its own ledger file under ‘*ledgers/config*/test_case_*/ validator_*.ledger’*

Log File:

For each test case in each configuration each validator and client creates its own log file under ‘logs/config/test_case_*’/ validator_*.log’ , ../client_*.log*
Executor/play ground log files is at ‘logs/config/test_case_*/twins_executor.log*

Explanation of all the variable names mentioned in the config file:

```
'nvalidators': Number of Validators/Replicas,
'nTwins': Number of Twin Validators,
'nclients': Number of Clients,
'nclientops': Number of operations each client performs,
'sleeptime': Delay between consecutive operations for the same client in
seconds,
'clienttimeout': Amount of time the client waits in seconds to receive
the response. If no response is received, it retransmits that request
'delta': Amount of time in seconds used to decide the pacemaker timer
timeout time,
'window_size': Window size used for Leader Election,
'exclude_size': Exclude size used for Leader Election,
'delay': Delay time when a fail type occurs
'quorum_bug': Change the quorum from 2f+1 to 2f when set to true
'accept_conflicting_votes': Vote for multiple proposal messages in a round
'liveness_bound': test case run duration bound
'n_test_cases': number of test cases to run for a particular config
```

For each round the Failure Configuration is added in the test_case.json, the parameters are

src: The source of failure injection for the given message Example(replica_0 for 0th validator, replica_0f for its twin, This config is used for all validators. Same for dest parameter),
dest: The destination of failure injection for the given message,
message_type: Type of the message Example(0 -Proposal Msg, 1-Vote Msg, 2- Timeout Msg),
fail_type: Type of failure Example(0- FailType.MsgLoss, 1 -FailType.Delay, 2 -No Failure),

Test Case 0
Scenario: Partitions with twins as leaders and multiple overlapping partitions with different message types and fault types as drop and delay for 5 rounds

Outcome: The system follows a “Happy Path” with 5 Validators and 1 twin. There are no Safety violations and Liveness violations. All the ledger files have consistent transactions written by their corresponding validator. The clients and validators terminate gracefully after all the commands by clients have been executed in the ledger.

```
'n_replicas' : 5,
        'n_twins' : 1,
        'n_rounds' : 5,
        'n_partitions' : 2,
        'is_leader_faulty' : True,
        'partition_num_limit' : 100,
        'n_test_cases' : 5,
        'leader_partitions_num_limit' : 100,
        'random_seed' : 1234567,
        'is_deterministic' : False
```

```
        'nclients': 3,
        'nclientops': 2,
        'sleeptime': 1,
        'clienttimeout': 4,
        'delta': 0.25,
        'window_size': 5,
        'exclude_size': 0,
        'delay': 1,
        'quorum_bug': False,
        'accept_conflicting_votes':False,
        'liveness_bound' : 10,
```

Test Case 1
Scenario: Injecting quorum bug

Outcome: Injected quorum bug which makes quorum to create a QC and TC 2f during the 3rd round. Both the twins in different partitions generate their own proposal messages and are committed by the replicas in the respective partitions violating safety but not liveness.

```
"3": {
           "leader": "replica_0",
           "partitions": [
               [
                   "replica_1",
                   "replica_4",
                   "replica_3",
                   "replica_0"
               ],
               [
                   "replica_2",
                   "replica_4",
                   "replica_0f"
               ]
           ],
```

```
        'nclients': 3,
        'nclientops': 2,
        'sleeptime': 1,
        'clienttimeout': 4,
        'delta': 0.25,
        'window_size': 5,
        'exclude_size': 0,
        'delay': 1,
        'quorum_bug': False,
        'accept_conflicting_votes':False,
        'liveness_bound' : 10,
```

Test Case 2
Scenario: Injecting conflicting votes bug.
Outcome: During the round 1 with conflicting votes bug where a replica can vote for multiple proposal messages for a round is introduced. Safety violation is detected where two proposals are committed for the same round.

```
1": {
            "leader": "replica_1",
            "partitions": [
                [
                    "replica_1",
                    "replica_4",
                    "replica_2",
                    "replica_0",
                    "replica_0f",
                    "replica_3"
                ],
                [
                    "replica_2",
                    "replica_0f"
                ]
            ],
```

```
'nvalidators': 5,
        'nTwins': 1,
        'nclients': 3,
        'nclientops': 2,
        'sleeptime': 1,
        'clienttimeout': 4,
        'delta': 0.25,
        'window_size': 5,
        'exclude_size': 0,
        'delay': 1,
        'quorum_bug': False, #quorum = 2f
        'accept_conflicting_votes': True,
        'liveness_bound' : 10,
        'n_test_cases' : 5
```