CSE535: Distributed Systems
Project: DiemBFT v4 Consensus Algorithm
Scott D. Stoller

phase 2 test cases: DiemBFT v4 consensus algorithm

version: 2021-10-18

Procedure

Please prepare configurations to run each of the following test cases (or variants of them). If your configuration mechanism does not provide the necessary parameters and flexibility for some of them, it is OK to make small edits to your code during the demo to set up these scenarios.

The demo will have two main parts.

Part 1. We will ask you to run each of the test cases below, in turn. If you know that your system fails a test case, you may want to run it during the demo anyway, if the behavior is close to correct. Instead or in addition, you may create and demo a variant of the test case, e.g., with fewer clients or fewer requests. If your system can't run the original test case or a modest variant, you can simply report this, and we will move on to the next test case (allowing more time for part 2).

Part 2. During the remaining time of the demo, you may run any test cases you created. Be prepared to describe them succinctly to the TA, and show the configurations to the TA, so the TA can confirm that the executed test case matches your description. Ideally, these test cases should be in your test report, but this is not mandatory.

Notes regarding these test cases:

- Each test case has **3 clients**, **each sending 4 requests**, except where noted otherwise. This means that each test case should contain at least 4 (and at most 12) rounds with non-empty blocks, plus a final round that proposes an empty block in order to commit the transactions in the previous round.
- All **failures are deterministic** (probability=1) except where noted otherwise.
- Each test case has **f=1** except where noted otherwise. **Always take N=3f+1**.
- "In the third round" means "in the third executed round" (e.g., corresponding to the third proposal message that is sent) . Depending on whether the round number is incremented before the first round (which is unclear from the paper), this could correspond to round=2 or round=3.
- "We assume replicas [i.e., validators] are numbered from 1 to N", as per phase2.docx section 2.5 (Fault Injection). If a team numbers validators starting from 0, adjust the numbers accordingly.
- For message delay failures, the delay should be long enough to cause the destination process to time-out before receiving the message.

Failures in a single round

Proposal Drop: in the third round, proposal messages to validators 1 and 2 are dropped.

Proposal Delay with f=2: in the third round, proposal messages to validators 1, 2, and 5 are delayed.

Proposal Drop + Timeout Delay: in the third round**,** all proposal messages are dropped, and all timeout messages are delayed.

Vote Drop: in the third round, vote messages from validators 3 and 4 are dropped or delayed.

SetAttr Failure: in round 1, a SetAttr failure with src=3, dest='_', type= MsgType.Vote, val=4, attr='current_round'. In other words, when validator 3 sends its vote in round 1, it sets Pacemaker.current_round to 4.

Failures in multiple rounds

Proposal Drop Twice: number of requests per client = 5. in the third and fourth rounds, proposal messages to validator 4 are dropped.

Vote Delay Twice: in the second and third rounds, vote messages to validators 1 and 3 are delayed.

Probabilistic Failures

Probabilistic Drop with f=2: number of requests per client = 20. seed=1234567 for the PRNG (for failure probabilities). drop failures in rounds 2, 3, 5, 6, 8, and 9 with src = '_', dest = '_', type = MsgType.Wildcard, probability=0.15.