Andrew Burford
Ethan DeTurk
Jiawei Qian

Phase 4 Test Report
The Posquatters

Test Generator Runs

Enumeration limits : The enumeration limit consists of the parameters **P,C,L,E** that we specify in each config_file.

Enumeration order: The enumeration order is controlled by the parameters **Random_partition**, **Random_leaders**, **Random_configurations** as they will randomize the enumeration order.

Name of the output file: The output files have the same name as the config files, but with a ".json" extension

1.Config File: deterministic_config.py

- Parameters:
  - Tests: 10000
  - R: 5
  - P: 3
  - K:10
  - C: 10000
  - L: 7
  - E: 1
  - N: 7
  - F: 2
  - Random_partition: False
  - Random_leaders: False
  - Random_configurations: False
  - Allow_non_faulty_leaders: False
  - Allow_quorumless_partitions: False
- Running Time: 0.5 seconds

2.Config File: random_leader_config.py

- Parameters:
    - Tests: 10000
    - R: 5
    - P: 3
    - K:10
    - C: 10000
    - L: 7
    - E: 1
    - N: 7
    - F: 2
    - Random_partition: False
    - Random_leaders: **True**
    - Random_configurations: False
    - Allow_non_faulty_leaders: **True**
    - Allow_quorumless_partitions: False
- Running Time: 0.57 seconds

3.Config File: random_partition_config.py

- Parameters:
    - Tests: 10000
    - R: 5
    - P: 3
    - K:10
    - C: 10000
    - L: 7
    - E: 1
    - N: 7
    - F: 2
    - Random_partition: **True**
    - Random_leaders: False
    - Random_configurations: False
    - Allow_non_faulty_leaders: False
    - Allow_quorumless_partitions: False
- Running Time: 0.5 seconds

4.<u>Config File</u>: random_config_config.py

- Parameters:
    - Tests: 10000
    - R: 5
    - P: 3
    - K:10
    - C: 10000
    - L: 7
    - E: 1
    - N: 7
    - F: 2
    - Random_partition: False
    - Random_leaders: False
    - Random_configurations: **True**
    - Allow_non_faulty_leaders: False
    - Allow_quorumless_partitions: False
- Running Time: 0.5 seconds

5.<u>Config File</u>: all_random.py

- Parameters:
    - Tests: 10000
    - R: 5
    - P: 3
    - K:10
    - C: 10000
    - L: 7
    - E: 1
    - N: 7
    - F: 2
    - Random_partition: **True**
    - Random_leaders: **True**
    - Random_configurations: **True**
    - Allow_non_faulty_leaders: **True**
    - Allow_quorumless_partitions: False
- Running Time: 0.36 seconds

6. Config File: allow_quorumless_partition_config.py

- Parameters:
  - Tests: 10000
  - R: 5
  - P: 3
  - K:10
  - C: 10000
  - L: 7
  - E: 1
  - N: 7
  - F: 2
  - Random_partition: False
  - Random_leaders: False
  - Random_configurations: False
  - Allow_non_faulty_leaders: False
  - Allow_quorumless_partitions: **True**
- Running Time: 0.53 seconds

6. Config File: intrapartition_drops.py

- Parameters:
  - Tests: 1000
  - R: 5
  - P: 1
  - K: 3
  - C: 100
  - L: 1
  - E: 20
  - N: 7
  - F: 2
  - Random_partition: **True**
  - Random_leaders: **True**
  - Random_configurations: **True**
  - Allow_non_faulty_leaders: **True**
  - Allow_quorumless_partitions: False
- Running Time: 0.5 seconds

Test Executor Runs

Config File: executor_configs/rotating_quorums.py

Scenarios: 2

Config name: no_timeouts

Parameters:

- replicas: a, b, c, d
- twins: d'
- rounds: 15
- Transmission delay: 0.8 seconds
- seed: 0
- timeout: 30 seconds
- first GST round: 8
- No bugs injected in DiemBFT

Outcome: This test case swaps out the members of a bucket containing a minimal quorum. This means replicas must sync up before they can vote if they sat out the previous round. All properties are upheld.

Running Time: 1.7 seconds

Logs: log/rotation_quorums/no_timeouts*.log

Config name: timeouts

Parameters:

- replicas: a, b, c, d
- twins: d'
- rounds: 15
- Transmission delay: 0.8 seconds
- seed: 0
- timeout: 30 seconds
- first GST round: 8
- No bugs injected in DiemBFT

Outcome: This test case swaps out the members of a bucket containing a minimal quorum. This means replicas must sync up before they can vote if they sat out the previous round. However, the leader is set to the node that sat out the previous round so this results in lots of timeouts. Fewer blocks are committed, but all properties remain upheld.

Running Time: 31.6 seconds

Logs: log/rotation_quorums/timeouts*.log

Config File: executor_configs/syncmanager_bug.py

Scenarios: 2

Config name: with_bug

Parameters:

- replicas: a, b, c, d
- twins: d'
- rounds: 11
- Transmission delay: 0.8 seconds
- seed: 0
- timeout: 30 seconds
- first GST round: 4
- Sync response commit chain contains an unnecessary block at the beginning

Outcome: This is a bug we found in our code from phase 2. Replica c syncs up after receiving a's proposal in round 4. In the sync response that it receives, there is an additional block at the beginning on the commit chain. The syncmanager calls ledger.speculate with this block, however, this block has already been speculated by the ledger and the parent of this block has already been committed so it is no longer a pending block. The ledger code is not expecting this to happen, so it ends up overwriting the existing ledger state for this block but computing the ledger state id incorrectly. Since the ledger state referenced by prev_block_id is no longer stored as a pending ledger state, the ledger state id is only a hash of the transactions in this block, it is missing the hash of the previous block, essentially breaking the chain of hashes. This bug becomes evident by also writing the commit_state_id of each block to the ledger once it is committed since the property checking doesn't detect any issues.

Running Time: 1.9 seconds

Logs: log/syncmanager_bug/with_bug*.log

Config name: without_bug

Parameters:

- replicas: a, b, c, d
- twins: d'
- rounds: 11
- Transmission delay: 0.8 seconds
- seed: 0
- timeout: 30 seconds
- first GST round: 4
- No bugs injected

Outcome: The ledgers match up as expected.

Running Time: 1.4 seconds
Logs: log/syncmanager_bug/without_bug*.log

Config File: executor_configs/small_quorum_bug.py

Scenarios: 4

Config name: false_progress_without_bug

Parameters:

- replicas: a, b, c, d
- twins: d'
- rounds: 8
- Transmission delay: 0.8 seconds
- seed: 0
- timeout: 30 seconds
- first GST round: 1
- No bugs injected

Outcome: In the first round (round 0), the replicas are partitioned such that no bucket has a quorum of replicas. As expected, the nodes attempt to send each other timeout messages but no replicas ever progress past round 0. The test executor detects that all nodes have sent messages in round 0 and knows that they can't progress past this so it waits the transmission delay bound * 4 * 7 seconds for any possible progress to be made. As expected, no progress is made and the safety properties are upheld.

Running Time: 22.9 seconds

Logs: log/small_quorum_bug/false_progress_without_bug*.log

Config name: false_progress_with_bug
Parameters:
- replicas: a, b, c, d
- twins: d'
- rounds: 8
- Transmission delay: 0.8 seconds
- seed: 0
- timeout: 30 seconds
- first GST round: 1
- Quorum size changed to 2f instead of 2f+1

Outcome: The leader in round 0, replica a, is in a bucket with replica b, but this is enough for it to form a QC for its proposal in round 0. Replica a is also the leader in the first round of GST so it makes another proposal in round 1. Replicas c and d that missed the proposal from round 0 begin syncing up. The test executor sees that a is progressing past round 0, however it still waits for all replicas to reach the last reachable round (which is round 0). Replicas c and d don't send any messages until they receive a's proposal from round 1 and send a syncup message attempting to sync up to round 1. At this point, the test executor sees that all replicas have reached at least round 0 and waits for any replica to progress past round 0. This has already happened, so it reports that the Quorumless progress property has been violated. Safety is still upheld since a was the only leader and it is honest.
Running Time: 0.5 seconds
Logs: log/small_quorum_bug/false_progress_with_bug*.log

Config name: equivocating_without_bug
Parameters:
- replicas: a, b, c, d
- twins: d'
- rounds: 11
- Transmission delay: 0.8 seconds
- seed: 0
- timeout: 30 seconds
- first GST round: 5
- No bugs injected

Outcome: In the first round, d is the leader and also a twin. Each twin is in a different bucket, so they each make equivocating proposals. However, only one bucket has a quorum so only one of the twins makes progress. This continues for rounds 0, 1, and 2. In round 3, a is made the leader but it is still in round 0 so it doesn't make a proposal. Round 3 times out. When replica a receives a timeout

message for round 3, it sends a sync request. However, it also advances to round 4 where it is also the leader so it immediately sends a proposal before it has synced up. This proposal is for a genesis block so the other replicas do not vote for it. That means that round 4 also times out. After this, there are 7 rounds of GST and a has synced up so these rounds proceed as normal. All properties are upheld.
Running Time: 10.9 seconds
Logs: log/small_quorum_bug/equivocating_without_bug*.log

Config name: equivocating_with_bug
Parameters:

- replicas: a, b, c, d
- twins: d'
- rounds: 10
- Transmission delay: 0.8 seconds
- seed: 0
- timeout: 60 seconds
- first GST round: 3
- Quorum size changed to 2f instead of 2f+1

Outcome: For rounds 0, 1, and 2, each bucket has a quorum of size 2f and d and d' send equivocating proposals to their own buckets for these rounds. Both buckets make progress, committing two separate genesis blocks. In round 3, GST begins and a becomes leader. Since a was in the bucket with d (not d'), it extends the chain started by d. The replicas in the bucket with d' attempt to sync up, but they have different genesis blocks so this causes a sync error. They are never able to sync up and therefore do not vote for any of a's proposals. But since d does not need to sync up, a continues making progress using the quorum of a and d. After these two replicas progress through all rounds of GST, the test executor waits for b, c, and d' to reach the final round. This occurs once the round past GST times out and b, c, d' attempt to sync up to this timeout message. The test executor sees these sync up messages and determines that all nodes have reached the final round. Property 1 is violated because 2 different blocks are certified in round 0. This also happens in round 1 and 2 but DistAlgo quantification quits after detecting the first violation. Property 2 is upheld because it says that for any two globally direct-committed blocks, one must extend the other. However, the definition of global direct-commit is that a block receives f+1 votes from non-Byzantine replicas. All the blocks proposed by a and d only have 1 vote from a so are not globally direct committed. Only the block proposed by d' in round 0 is globally direct-committed so property 2 is upheld. Property 3 and 4 are both violated because b and c never commit blocks during GST. In the ledger

files, you can see that different genesis blocks are committed by the replicas in each bucket in round 0.
Running Time: 4.1 seconds
Logs: log/small_quorum_bug/equivocating_with_bug*.log

Config File: executor_configs/conflicting_votes_bug.py

Scenarios: 2

Config name: bug

Parameters:

- replicas: a, b, c, d
- twins: d'
- rounds: 8
- Transmission delay: 0.8 seconds
- seed: 0
- timeout: 30 seconds
- first GST round: 1
- Inject bug so that replicas can vote twice in one round

Outcome: In round 0, the faulty leader d equivocates and sends two proposals (one as d and the other as d'). Both proposals receive votes from all replicas, violating Property 1 (safety). In round 1, a sends two proposals, but this is *not* an example of equivocation. a is not faulty, and it sends a proposal twice because process_new_round_event is called once for each QC that is formed. Normally, this would be impossible because only one QC is formed per round.

Running Time: 0.5 seconds

Logs: log/conflicting_votes_bug/bug*.log

Config name: no_bug

Parameters:

- replicas: a, b, c, d
- twins: d'
- rounds: 8
- Transmission delay: 0.8 seconds
- seed: 0
- timeout: 30 seconds
- first GST round: 1
- No bugs injected

Outcome: In round 0, the faulty leader d equivocates and sends two proposals (one as d and the other as d'). This time, a only receives one vote message from each replica and only one proposed block from round 0 is certified. 7 rounds of GST proceed as normal, and all properties are upheld.

Running Time: 0.5 seconds

Logs: log/conflicting_votes_bug/no_bug*.log

Config File: executor_configs/vote_leader_current_round.py

Scenarios: 2

Config name: bug

Parameters:

- replicas: a, b, c, d
- twins: d'
- rounds: 13
- Transmission delay: 0.8 seconds
- seed: 0
- timeout: 30 seconds
- first GST round: 6
- Inject bug so that replicas can vote twice in one round

Outcome: Starting in round 2, every other round ends in a timeout despite full network partitions. This violates Property 4 (liveness)

Running Time: 19.3 seconds

Logs: log/conflicting_votes_bug/bug*.log

Config name: no_bug

Parameters:

- replicas: a, b, c, d
- twins: d'
- rounds: 13
- Transmission delay: 0.8 seconds
- seed: 0
- timeout: 30 seconds
- first GST round: 6
- No bugs injected

Outcome: A block is committed in every round and all of the properties are upheld.

Running Time: 0.5 seconds

Logs: log/conflicting_votes_bug/no_bug*.log