CSE 535: Distributed Systems - Test Report

Team- Name: Evil Geniuses

1) Nihal Goalla (113276929)
2) Rohith Vaddavalli (113261811)
3) Manoj Kumar Ravuri (113262634)

Test case - 1

Filename - configNoFault.txt
Ledger files - libra_blockchain/test_logs/configNoFault_N=4/data
Log Files - libra_blockchain/test_logs/configNoFault_N=4/logs
Number of Replicas - 4
Number of Clients - 2
Number of Client Requests - 4
Request Gap -1

In this test case we have included 2 clients each sending 4 requests with a request gap of 1 second to 4 Validators(N=4).

All the Ledgers at each node are synchronous and follow the same order.

Test case - 2

Filename - configNoFault.txt
Ledger files - libra_blockchain/test_logs/configNoFault_N=100/data
Log Files - libra_blockchain/test_logs/configNoFault_N=100/logs
Number of Replicas - 4
Number of Clients - 100
Number of Client Requests - 4
Request Gap - 1

In this test case We have executed a stress test with 100 clients each sending 4 requests with a request gap of 1 second to 4 Validators.

All the Ledgers are at each node are synchronous and follow the same order.

Test case - 3

Filename - configNoFault.txt
Ledger files - libra_blockchain/test_logs/configNoFault_N=4_RG=5/data
Log Files - libra_blockchain/test_logs/configNoFault_N=4_RG=5/logs
Number of Replicas - 4
Number of Clients - 4
Number of Client Requests - 4
Request Gap -5

In this test case we have included 4 clients each sending 4 requests with a request gap of 5 second to 4 Validators.

All the Ledgers at each node are synchronous and follow the same order.

Test case - 4

Filename - configTimeoutCommit.txt
Ledger files - libra_blockchain/test_logs/configTimeoutCommit/data
Log Files - libra_blockchain/test_logs/configTimeoutCommit/logs
Number of Replicas - 4
Number of Clients - 2
Number of Client Requests - 4
Number of faulty nodes - 1

In this test case, we sent each request with a gap of 5 seconds. The other replicas are almost on the verge of timeout, but one of the fault replicas will always timeout.

The failure scenario in this is as follows:

We have one faulty node which will always timeout
But a quorum always forms with 3 of the responses, so the round always advances.

We see that in the outcome all the Ledger files, including that of the faulty node is the same. We can confirm that the protocol is working with ‘f’ faulty nodes.

Test case - 5

Filename - configTimeoutCertificate.txt
Ledger files - libra_blockchain/test_logs/configTimeoutCertificate/data
Log Files - libra_blockchain/test_logs/configTimeoutCertificate/logs
Number of Replicas - 4
Number of Clients - 2
Number of Client Requests - 4
Number of faulty nodes - 2

In this test case, we sent each request with a gap of 1 seconds.

The failure scenario in this is as follows:

1. We have two faulty nodes which always timeout
2. So, there is never a quorum formed, and the honest validators try to process the requests, but the faulty nodes won’t participate in the process

We see that all the ledger files are empty. The honest validators try to process the requests, but due to no quorum, nothing gets committed.

Test case - 6

Filename - configClientFailure.txt
Ledger files - libra_blockchain/test_logs/configClientFailure/data
Log Files - libra_blockchain/test_logs/configClientFailure/logs
Number of Replicas - 4
Number of Clients - 2
Number of Client Requests - 4
Number of faulty nodes - 1
Number of faulty clients - 1

In this we have one faulty client which will have a bad signature, so none of the replicas will accept any request from the faulty client.

We verify the signature and then send back “Bad Signature” to the client. So, the client retransmits the request again with the correct signature later.

We also included duplicate requests from one of the clients, and the replicas will drop the request.

We have a faulty client which becomes honest after some time. The ledgers are the same across all nodes because every client is honest and a quorum forms always.

Test case - 7

Filename - configMessageDelay.txt
Ledger files - libra_blockchain/test_logs/messageDelay/data
Log Files - libra_blockchain/test_logs/messageDelay/logs
Number of Replicas - 4
Number of Clients - 2
Number of Client Requests - 10
Number of message delay nodes - 1

In this scenario, we implement a message delay by delaying the vote messages from the message delay replicas.

Even if the votes are stopped from one client. The system is resilient to f faulty message losses.

The failure scenario in this is as follows:

> We have one faulty node which won't send any vote messages.
> But a quorum always forms with 3 of the responses, so the round always advances.
> But the messages are delayed by one second which would stop the system from terminating till all the messages delayed are sent back.

Test case - 8

Filename - configMessageLoss.txt
Ledger files - libra_blockchain/test_logs/configMessageLoss/data
Log Files - libra_blockchain/test_logs/configMessageLoss/logs
Number of Replicas - 4
Number of Clients - 2
Number of Client Requests - 10
Number of faulty nodes - 1
Number of faulty clients - 1

In this, the faulty node will not send any messages from itself. So, when it is called, it will act as a Byzantine failure.

All the Ledgers at each node are synchronous and follow the same order.