**Meeting Notes 08-03-2022**

Progress update:

* Code guide in the readme
* Mu-lambda GA implementation [SERG-Delft/ConsensusTesting: Pull Request 6](https://github.com/SERG-Delft/ConsensusTesting/pull/6)
* Started writing, nothing sent yet because I’m not satisfied with the writing. Started writing a bit of everything. Shifted focus to just the background. Hope to have the first draft of this chapter somewhere this week.
* Message dropping. How fine-grained? If we just want network partitions, perhaps extend the problem encoding? Will need to resolve the partitions of more than 80% split or the network will not make progress. Think about mutation as well. If we want to use delay threshold, the algorithm will automatically start message dropping too much, penalize appropriately will require a lot of finetuning.
* Bug injection. Historical bug injection is rather difficult to find. No big consensus bugs found in release notes from 2016 till now. I suspect they are disclosed and fixed confidentially. Alternative could be to use the inherent flaws of ripple’s consensus algorithm. Try to create a ledger fork (from this paper: https://link.springer.com/chapter/10.1007/978-3-319-22846-4\_10) through network partitions? Message delays are not required for exposing this ‘bug’.

**Honors students’ liveness bug**

The paper states that the protocol will not progress under the following circumstances:

* All nodes in one UNL
* One node is byzantine and disseminates two different transactions to the first and the second half of the UNL
* The nodes see over 50% in agreement, so they keep the disputed transaction in their proposal
* Liveness is now compromised

This is not true, due to the avalanche protocol in the establish phase of the protocol. In the establish phase, the nodes need to see over 80% of their UNL agreeing on candidate transaction (CT) set to declare consensus. Given the previous scenario, the nodes will eventually postpone **both** transactions to the next round, due to the increasing threshold for keeping a transaction in the CT set. At the first phase this threshold is 50% but this increases to 65% after half the time of the previous consensus round has elapsed. At this point, when just over 50% of the nodes in their UNL propose both transactions, all correct nodes in the UNL will remove the transaction for their CT set and send this as a proposal reaching consensus shortly after. Furthermore, the transactions will then be gossiped to the other nodes, meaning just one or none of the transactions will be applied in the next consensus round.

Network partitioning is different subject from message dropping.

Try to recreate the bug in the paper before trying to change the algorithm to be able to handle it.

With a high delay from node A to B delay all messages originating from A instead of just the one to simulate proper network link failure.