**Meeting notes 30-11-2021**

**Progress update**

1. Implemented the test harness. Defined in a simple text file. Top number is number of validated ledgers to wait for, below are at what ms to send transactions. Will change this to only include the transaction times and determine that the test case has stopped after all transactions are in a validated ledger.
2. Implemented node state. current round, current consensus\_phase and last validated ledger. Will extend with transactions validated / proposed

**Recap GA**

* Guassian or polynomial mutation. better for real-valued chromosomes.

**Chomosome design**

Recap. Same delays might result in different fitness function. Is this a problem? not really, a choice. Evaluation of one individual takes roughly 13 seconds, run in parallel in cluster.

* Also distinguish proposeSeq messages, perhaps drop messages with proposeseq not in chromosome. Makes chromosome length variable. Take cutpoints as percentage of length of parents.

**Fitness function ideas**

1. Establish -> open and open -> open count. Consensus failed in some way, retry.

2. Time till end of test case, where test case is done when transactions are validated

3. Rounds / validated ledgers till end test case.

4. Number of messages till end test case?

5. Instrument code to target branches. Look for log warnings/errors/fatals to target.

- Building canonical transaction setSee RCLConsensus.cpp [500]

- SHAMapMissingNode RCLConsensus.cpp [906]

- Node sees no consensus but UNL does Consensus.cpp [158] Consensus.h [1594]

- Node receives txSet but no peers are proposing it Consensus.h [876]

- Stale peer proposal Consensus.h [1381]

* Try every fitness function individually and see if they correlate. Analyze a population of fitness functions functions. Don't run GA,instead run the same test cases for every fitness function.

**Start writing thesis**

1. What is my research question and how should I formulate it  
   - How effective is applying evolutionary search to asynchrony in blockchain-based consensus algorithms at uncovering concurrency and consistency bugs?
2. How in depth should I go for each the following topics

- Blockchain in general

- Ripple in particular

- Consensus algorithms

- Distributed systems and event interleavings (combine in ripple explanation)

- Evolutionary algorithms (explain ingredients used and why)

- Search-based software testing

3. How long should my thesis be roughly (look at examples

* Basic background. What is the SOTA and what does my thesis add.
* Related work, asynchronous concurrency testing interleavings.
* We need a baseline, random search.
* We can inject a bug to see if the EA can find it if it doesn't find one itself.

Examples:

<https://repository.tudelft.nl/islandora/object/uuid%3A6d8a1835-9054-4e4a-a85f-99ac592978da?collection=education>

<https://repository.tudelft.nl/islandora/object/uuid%3A26da088e-25e1-4de4-bfc2-6935e32646ab?collection=education>

- Next steps

Implement test harness until transactions validated

Create structure of thesis

Implement Guassian or polynomial mutation

Implement proposeSeq in chromosome

Impement shutdown on GA completion

Compare fitness functions individually

Run on mac and record steps

See if the same test case reproduces the same message reordering