May the Delays Be Ever in Your Favor

Genetic Operators in Delay-Based Testing of the XRPL Consensus Algorithm

Wishaal Kanhai | Author
Annibale Panichella, Mitchell Olsthoorn, Burcu Kulahcioglu Ozkan | Supervisors

1

XRP LEDGER A blockchain to facilitate efficient cross-border transactions

$460.831.398.702
XRP Trading volume (Dec 2024)

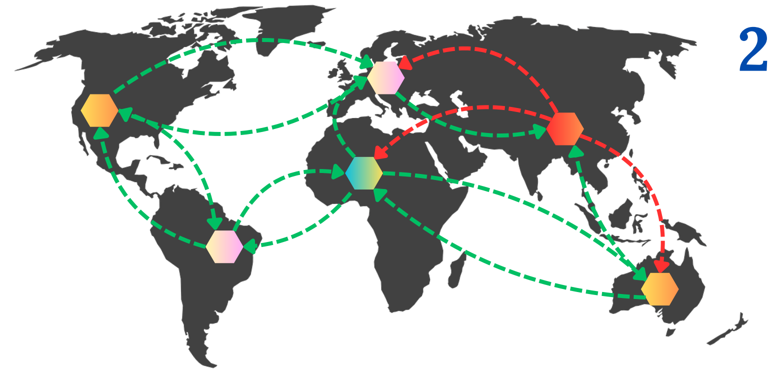
Potential bugs in XRPL could result in frozen funds or the validation of invalid transactions

2

The XRPL Consensus Algorithm

XRPL uses **trusted validators** to agree on a correct set of transactions, even if some validators behave maliciously. Every correct validator follows the steps defined by the consensus algorithm.

It provides **strong gaurantees** in theory, in practice it might contain **implementation mistakes**.



3

Testing requires **live network simulations**, despite some success, it remains **underexplored**
We address this using **evolutionary testing**:

Initial generation
Generate random test cases and execute them

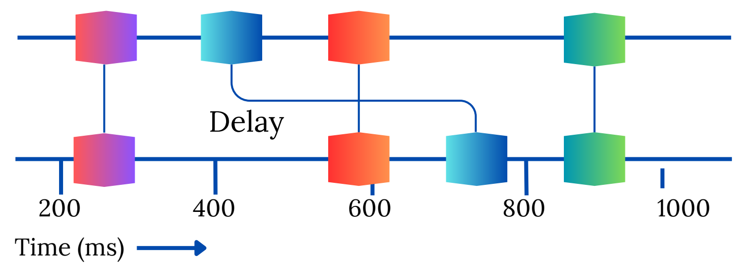
Select fittest individuals
Based on a fitness function, select best individuals to generate new test cases

New generation
Replace the worst test cases while keeping the best, repeat the process until satisfied

Evolutionary approaches have been shown to be **more effective** than systematic testing in distributed systems

4

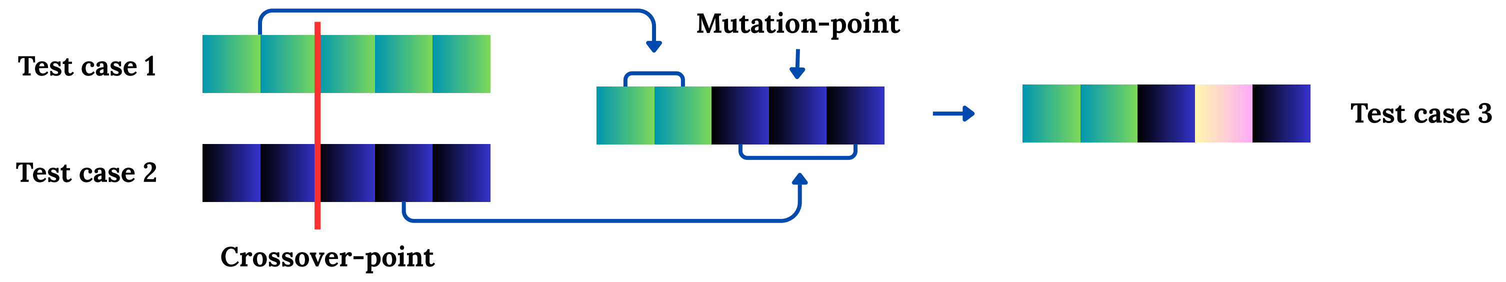
Reordering message arrivals using delays



These reorderings can cause concurrency issues
Halted progress, network forks, etc.
Our test cases consist of delays to apply

5

Evolutionary testing approaches use **genetic operators** (mutation & crossover) to generate new test cases:



RQ "How does the selection of genetic operators affect the performance of an evolutionary approach for delay-based testing of the XRPL consensus algorithm?"

6

We inject a consensus bug and try to discover it!

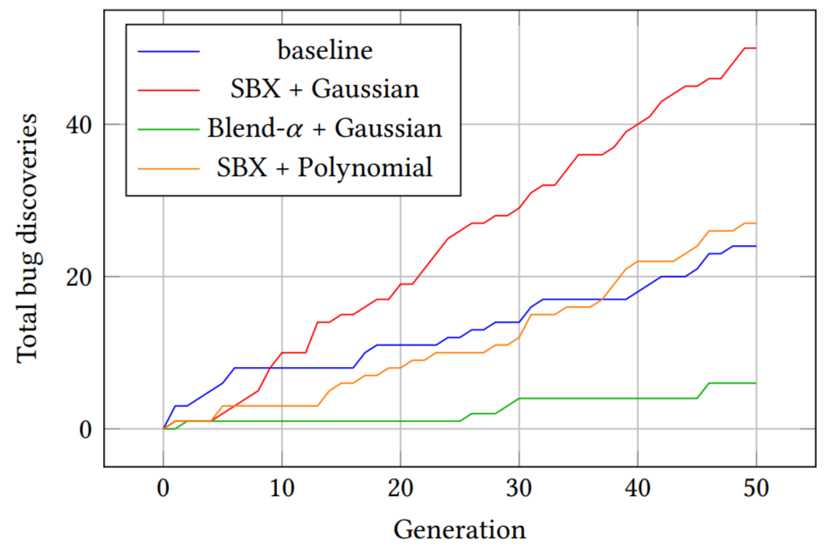
Genetic operators differ in their <u>degrees of exploration and exploitation</u>, we compare the following:

Exploitative:
- SBX Crossover
- Polynomial Mutation

Explorative:
- Blend-α Crossover
- Gaussian Mutation

Additionally, we experiment with an unguided baseline which does not use any operators

7



Trajectory of bug discoveries across generations

Best performance: exploitation with subtle exploration

Worst performance: heavily explorative configurations

Baseline (random testing): strong performance with high input diversity

Balanced strategies **outperform** extreme ones. **Diversity** is valuable, but guided refinement helps uncover clusters of bug-prone inputs more effectively.