EvoPriority: Evaluating Fitness Functions in Priority-Based Evolutionary Testing for the XRP Ledger Consensus Protocol

Author: Călin Ciocănea (c.ciocanea@student.tudelft.nl)
Responsible Professor: Burcu Kulahcioglu Ozkan
Supervisors: Annibale Panichella, Mitchell Olsthoorn

1. Background & Motivation

$116.216.217.291

Current Market Cap of XRP Cryptocurrency [1]

The **XRP Ledger** manages transactions of the **XRP cryptocurrency.**

It does this with the help of the **XRP Ledger Consensus Protocol.** This protocol allows **distributed nodes worldwide** to **reach consensus** on what transactions to apply to the ledger, even when **some nodes may be faulty or malicious.**

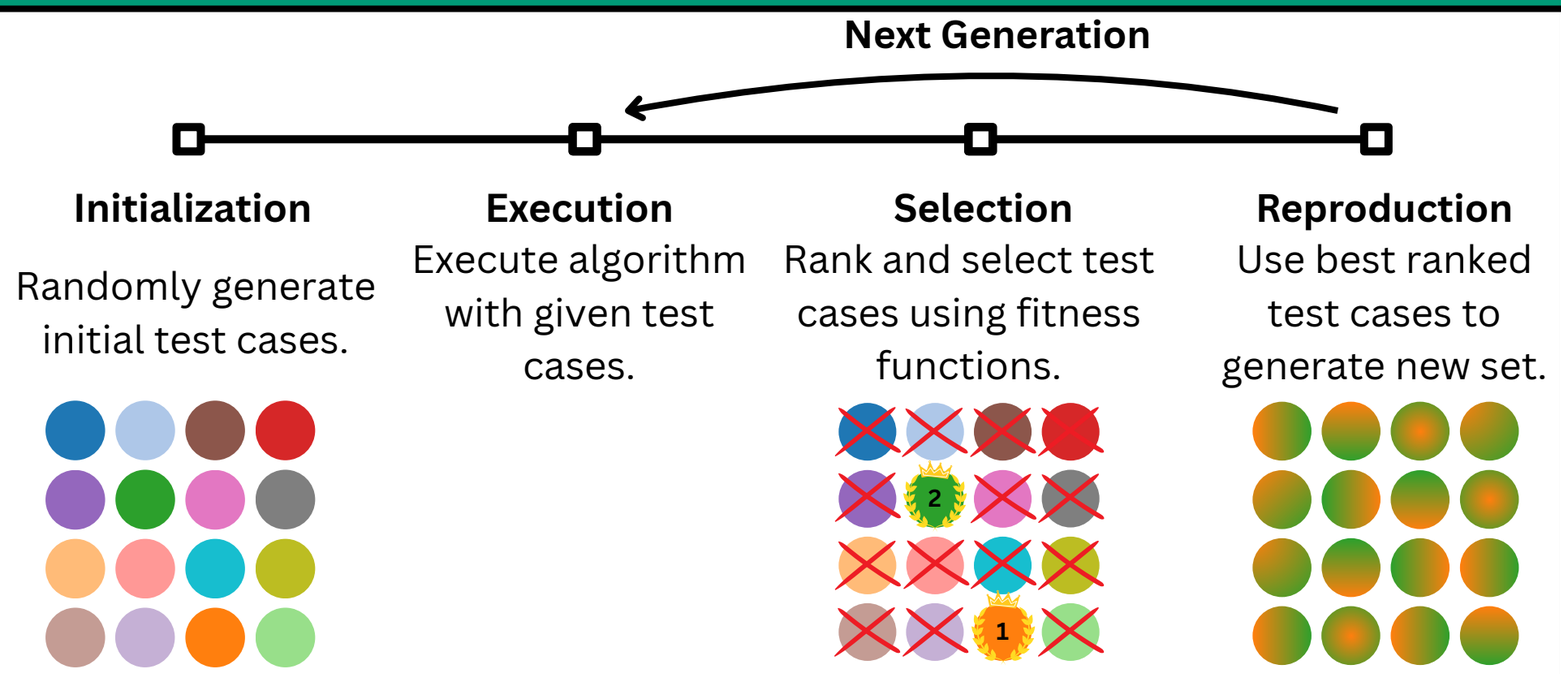
Although **proven to be correct** on paper, implementation-level **bugs are still possible. [2]**

To **keep funds flowing,** thoroughly and efficiently **testing the protocol is crucial.**

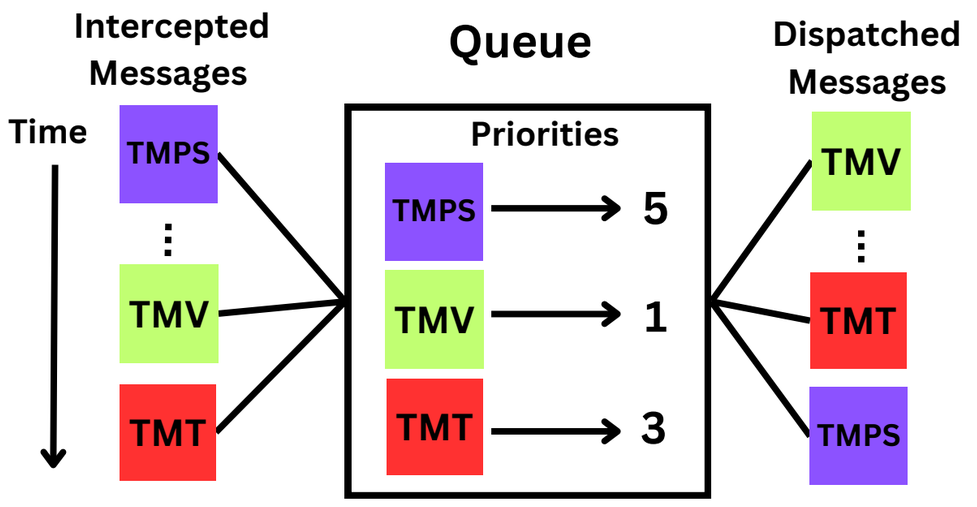
Many techniques have been proposed for **testing distributed systems,** including **systematic and randomized testing.** Some approaches target consensus protocols specifically, such as **ByzzFuzz [3],** which mutates messages to simulate Byzantine faults, or **evolutionary testing** applied to delay-based strategies **[4]**.

We explore a **priority-based** message scheduling strategy, paired with **evolutionary testing.** This approach is called **EvoPriority** and can be applied with different **fitness functions.** We asses the performance of **EvoPriority** and the **impact different fitness functions have** on it. **Priority-based testing** offers **exploration of message interleavings,** simulating real-word message handling scenarios.

2. Evolutionary Testing



3. Priority-Based Testing



Messages sent over the network are **intercepted.** They are **assigned a priority** value and later **dispatched** to their intended location in order of that priority.

4. EvoPriority

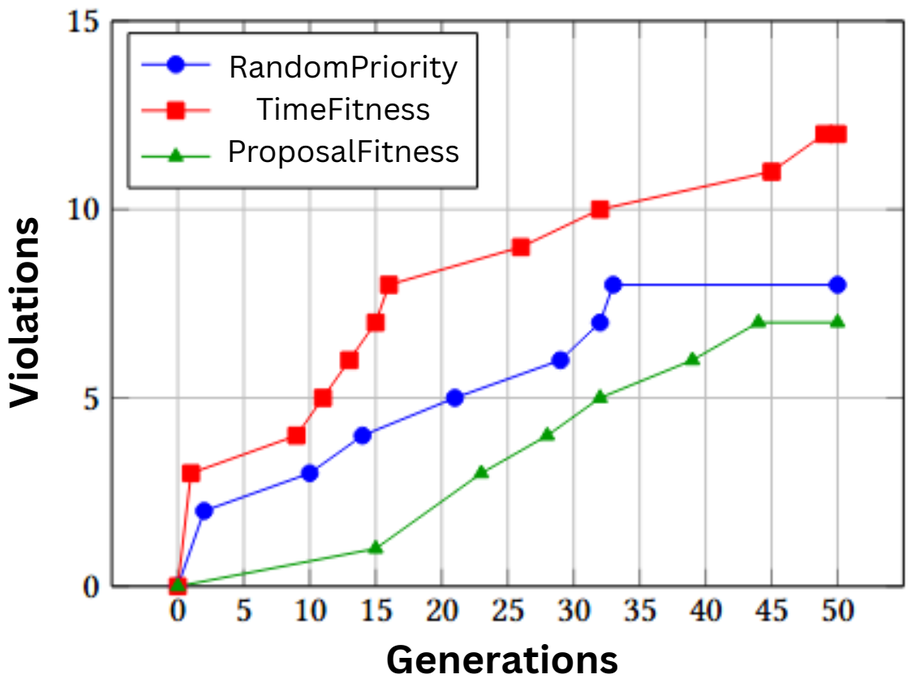
EvoPriority is a concurrency testing strategy that combines the **priority-based** approach with an **evolutionary algorithm.** The aim is to **guide** the generation of priority schedules towards **buggy executions.**

5. Fitness Functions

EvoPriority can be paired with different **fitness functions** for the selection step. We used:

- **TimeFitness:** guides towards executions which last longer, aiming for executions where the network stops making progress.
- **ProposalFitness:** guides towards executions in which more messages are sent over the network, suggesting consensus is difficult to reach.

6. Results & Conclusions



Testing Configuration	Number of Violations
RandomPriority (S)	*9*
TimeFitness (S)	*12*
ProposalFitness (S)	*7*
RandomPriority (O)	*0*
TimeFitness (O)	*0*
ProposalFitness (O)	*0*

Over **50 generations** of **10 test cases** each, the following conclusions were made:

- While capable of finding bugs on a bug-seeded version of the XRPL CP, EvoPriority was not able to find anything on the original version.
- Compared to the RandomPriority baseline, EvoPriority does not show any statistically significant improvement in bug finding capablities.
- Even though EvoPriority using TimeFitness found more bugs than with ProposalFitness, the difference might be due to a random variation, rather than a performance increase.

(S) - run on bug-seeded version of XRPL CP
(O) - run on original version of XRPL CP

[1] CoinStats. 2025. XRP (Ripple) Live Price, Market Cap and Charts. https://coinstats.app/coins/ripple/. Accessed: 2025-06-03.
[2] David Schwartz, Noah Youngs, and Arthur Britto. 2014. The Ripple Protocol Consensus Algorithm. Technical Report. Ripple Labs Inc.
[3]Lars Winter, Florian Buşe, Daniël de Graaf, Kai von Gleissenthall, and Burcu Kulahcioglu Ozkan. 2023. Randomized testing of byzantine fault tolerant algorithms, 757–788.
[4]M. van Meerten, B. K. Ozkan, and A. Panichella. 2023. Evolutionary Approach for Concurrency Testing of Ripple Blockchain Consensus Algorithm, 36–47.