CoralChain

Author: **Xinjian Zhang** (MSc Student)
Supervisor: **Mubashar Iqbal, PhD**

CoralChain – A Visual Gadget Collection for Exploring Blockchain Fundamentals

INTRODUCTION

CoralChain is a lightweight, web-based blockchain network simulator built with Ruby and Sinatra.

"It features interactive visuals that illustrate blockchain structure, synchronization, and consensus mechanisms, helping make these concepts more intuitive and accessible."

MAIN TECH

Ruby + Sinatra + Chart.js

METHODS

BLOCKCHAIN STRUCTURE

- Block contains: index, timestamp, data, previous_hash, nonce, and SHA-256 hash
- Hash = SHA256(index + timestamp + data + previous_hash + nonce)
- Valid Chain: contiguous hashes + deterministic rehash
- Block Acceptance: consensus-specific hash rules

MULTI-NODE ARCHITECTURE

- Each node = isolated chain + independent state
- Sync via **Longest Valid Chain** policy
- Forks resolved by comparing indexed blocks
- Byzantine nodes trigger chain-wide corruption

CONSENSUS MECHANISM

- **Proof of Work (PoW)**: Increment nonce to find a hash matching an N-bit zero prefix
- **Proof of Authority (PoA)**: Simulate validator rounds with randomized hashing
- **Proof of Stake (PoS)**: Stake-weighted nonce values shape final hash

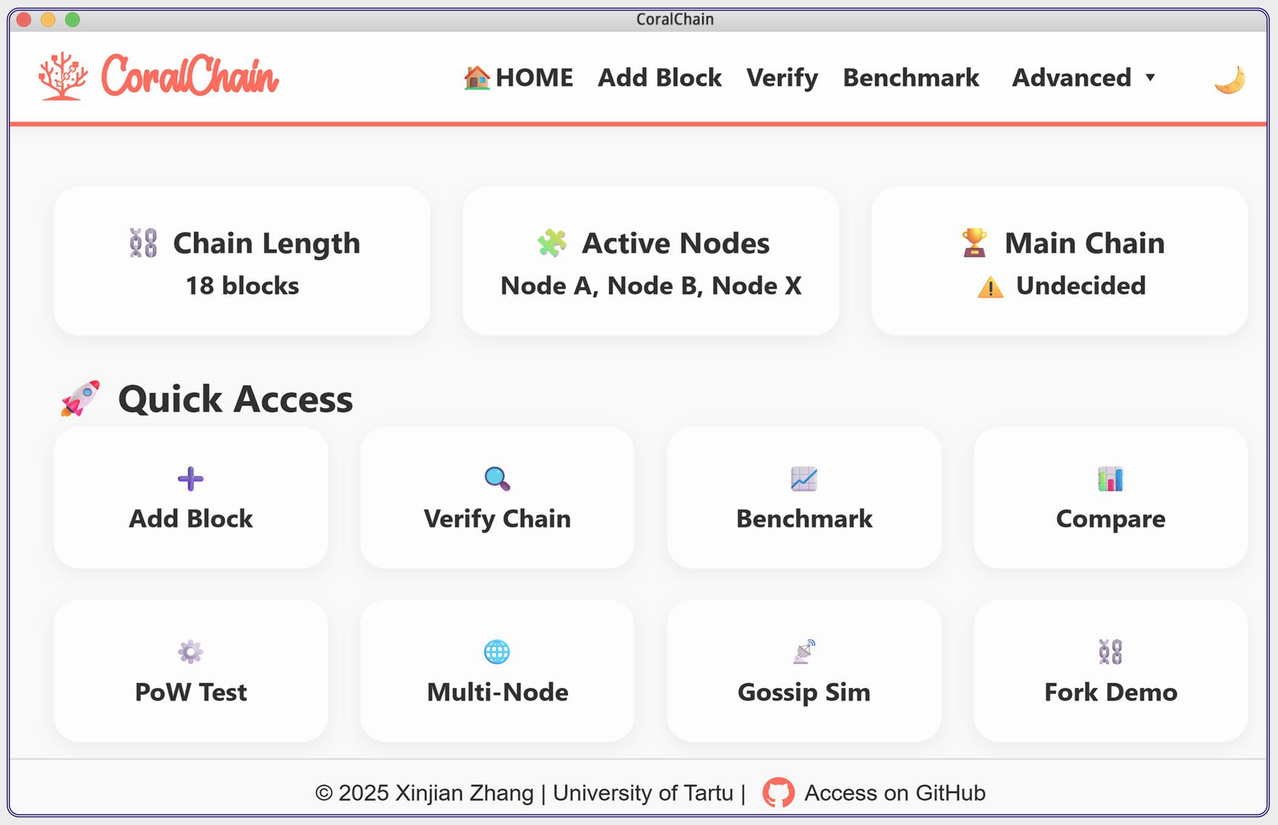

Fig 1. Main Interface of CoralChain

KEY FEATURES

1 Block Operations & Verification

- **Create blocks with input data and chosen consensus method**
- **Hash and verify chains using SHA-256 linkage**
- **Tamper blocks to trigger chain-wide invalidation**
- **Show mismatched hashes and real-time verification results**

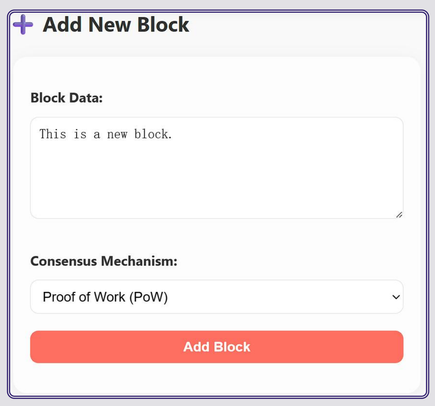

Fig 2. Block Creation

Fig 3. Blockchain Integrity Check

2 Multi-Node Interaction & Synchronization

Fig 4. Chain Synchronization Between Nodes A & B

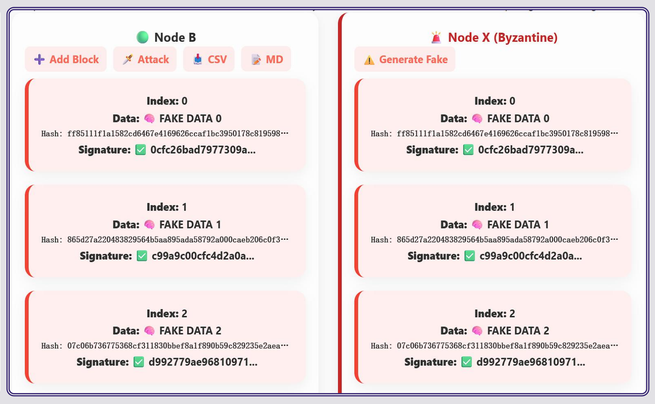

Fig 5. Byzantine Fake Chain Injection (Node X -> B)

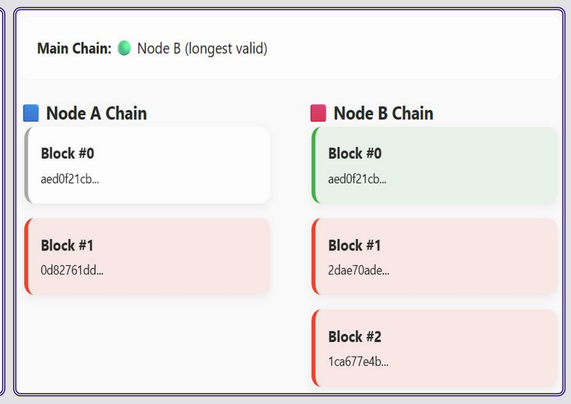

Fig 6. Chain Fork Between Nodes A & B

- **Run nodes (A, B, X) with isolated chain states**
- **Compare chains block-by-block to detect divergence**
- **Resolve forks using longest-valid-chain replacement**
- **Simulate Byzantine nodes, conflict injection scenarios**

3 Consensus & Network Simulation

- **Switch between PoW, PoS, and PoA during runtime**
- **Simulate gossip-based message spread with delay and faults**
- **Track block time and performance across algorithms**
- **Export chain data and results as CSV**

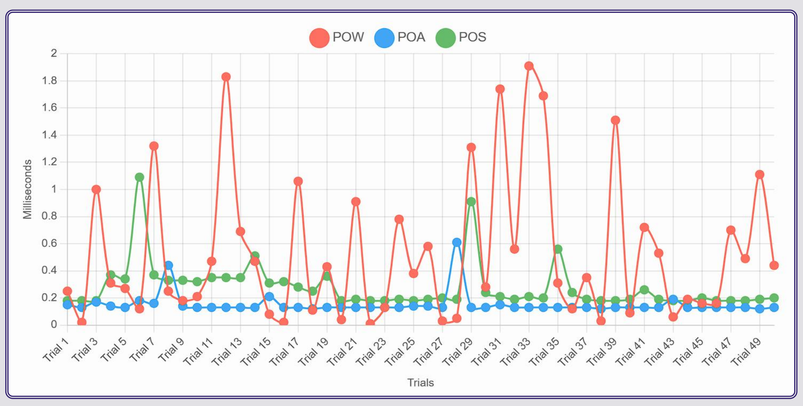

Fig 7. Consensus Benchmark Results (PoW, PoA, PoS, 50 Trials Each)

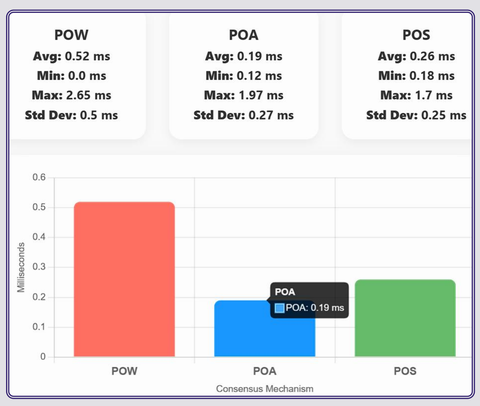

Fig 8. Consensus Performance Comp. (50 Trials Each)

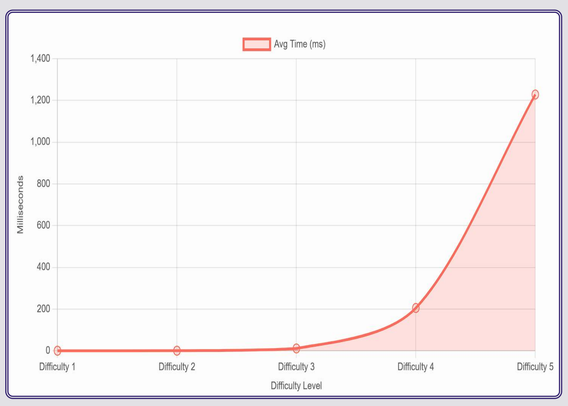

Fig 9. PoW Difficulty Experiment (5 Blocks, Levels 1–5)

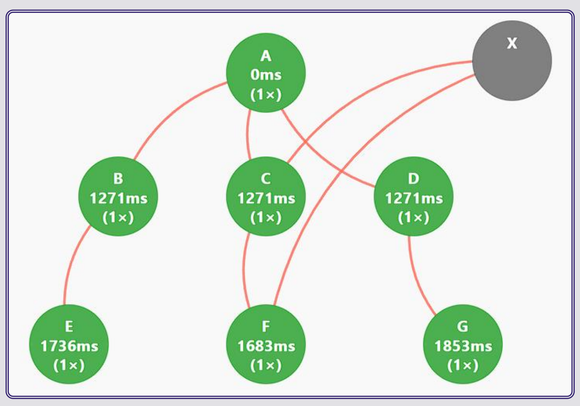

Fig 10. Gossip Propagation Sim. (RandDelay, 7 Nodes)

INSIGHTS

This project originated from a spontaneous idea, without a predefined goal in mind.
Though its functionality is still limited, I hope it might offer a small spark of interest for anyone curious about blockchain. ✨

SCAN HERE

Website

GitHub Repo

UNIVERSITY OF TARTU
Institute of Computer Science
1632

CHESS
Cyber-security Excellence Hub in Estonia and South Moravia
https://chess-eu.cs.ut.ee/

Visit the Website for More Details
Available in English and Estonian