Demo Abstract: eVIBES: Configurable and Interactive Ethereum Blockchain Simulation Framework

Aditya Deshpande
Technical University of Munich
aditya.deshpande@tum.de

Pezhman Nasirifard
Technical University of Munich
p.nasirifard@tum.de

Hans-Arno Jacobsen
Technical University of Munich

ABSTRACT

Cryptocurrencies and Distributed Ledger Technologies, such as Ethereum have received extensive attention over the past few years. With the increasing popularity of Ethereum, comprehensive understanding of its various properties plays a critical role in the widespread adaptation. However, due to the significant requirements for deploying a full Ethereum blockchain and high running costs, it is challenging to study the dynamic properties of the Ethereum. In this work, we propose eVIBES, a configurable simulation framework for gaining empirical insights into the dynamic properties of Ethereum.

CCS CONCEPTS

• **Computer systems organization → Peer-to-peer architectures**;

KEYWORDS

Blockchain, Ethereum, Simulation

1 INTRODUCTION

Comprehensive analysis of the dynamic properties of a real deployment of Ethereum is a difficult task due to the massive deployment efforts and high running costs. In this study, we propose eVIBES, an event-driven, concurrent, message-oriented, broadcast-based configurable Ethereum simulation for simulating large-scale Ethereum networks. eVIBES enables users to study the behavior of Ethereum-like blockchain systems by deriving empirical insights into the system by configuring the blockchain parameters. We designed eVIBES with two primary objectives in mind: scalability and configurability. Scalability enables the system to simulate a large number of nodes, without compromising the simulation speed or efficiency. Configurability offers the user to configure the Ethereum network both before the start of the simulation and during the simulation execution. eVIBES is inspired by VIBES [5], a configurable Bitcoin-like blockchain simulator capable of conducting large-scale peer to peer networks simulation.

Middleware '18, December 10–14, 2018, Rennes, France

ACM ISBN 978-1-4503-6109-5/18/12…$15.00
https://doi.org/10.1145/3284014.3284020

2 RELATED WORK

The Ethereum ecosystem is rich with tools such as Ganache[1], Hive[2] and other peer-to-peer network tools such as PeerSim [4], OPNET [3], but all have different objectives than ours when it comes to testing the Ethereum networks. Ganache is a tool for creating and testing private Ethereum blockchains. However, it does not capture how the nodes work with each other and how they reach a consensus. Hive is a practical tool for testing the clients in the Ethereum blockchain by creating docker images for the clients. However, simulating a large number of nodes is difficult with limited computational resources. None of these tools analyzes the behavior of the system as a whole and are limited to testing Ethereum smart contracts and decentralized applications. eVIBES simulates the whole Ethereum blockchain without creating full-fledged node instances and aims to enable the users to understand the whole blockchain network.

3 SYSTEM ARCHITECTURE

eVIBES architecture is based on the Actor Model and follows the reactive manifesto[3]. The primary components of the system are the Orchestrator and Reducer, as Figure 1 displays. The Nodes, which are marked in blue, operate similarly to an actual Ethereum client as described in the Ethereum yellow paper [1], excluding the Proof-of-Work (PoW) computations. The Nodes execute independently and perform the block verification and transaction execution.

The Orchestrator node is the primary component for controlling the whole simulation. It is responsible for receiving the simulation configuration from the users and setting up the simulation which includes the creation of all Ethereum nodes, accounts and periodical transactions generation. The user can start/stop and control the simulation parameters during its execution by communicating with the Orchestrator. The Reducer is responsible for generating the output of the simulation and presenting the data to the user. The visualization of the simulation outcome is inspired by the existing systems like ethstats[4] and etherchain[5].

The user can configure the following simulation parameters: the number of nodes, the number of accounts, the number of transactions, the rate of transaction generation, the range of gas limit for miner nodes, and the smart contract code (ability to upload smart contracts for execution during simulation). Besides, users can override default values for the genesis block before the execution starts. The simulation outputs the following metrics: the total execution time, the total number of transactions processed, the throughput (transactions per second), the block propagation delay, the cost

[1] http://truffleframework.com/ganache/
[2] https://github.com/karalabe/hive
[3] https://www.reactivemanifesto.org/
[4] https://ethstats.net/
[5] https://www.etherchain.org/

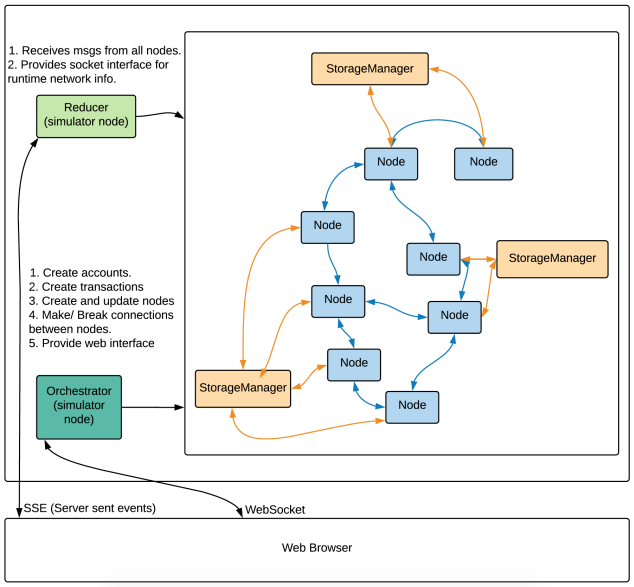

Figure 1: eVIBES system architecture.

per transaction, a log of all transactions. Other metrics such as the change in difficulty over time, the change in block duration over time, the change in gas limit over time are also presented at the end of the simulation. Output also includes the start and end states of all accounts. The system allows the inputs to be changed while the simulation is running. eVIBES allows manipulation of parameters like the transaction rate, the miner gas limit, and the network latency during the simulation execution. eVIBES models the valid block generations (PoW in Ethereum network), as a probability score. This feature leads to multiple nodes generating valid blocks around the same time for similar transactions, as is the case in Ethereum blockchain. eVIBES stores these orphan blocks as Uncle/Ommer blocks in the blockchain and rewards the miners of those blocks following the GHOST protocol [2].

We open-sourced the simulator, and the source code is publicly available[6].

4 SMART CONTRACT EXECUTION

Smart contracts are one of the essential features of Ethereum. We create additional entities to enable contract execution in eVIBES, as the Figure 2 shows. Initially, a user can upload contract code using the user interface. Execution of all the uploaded contracts is performed by the *SolidityExecutionEnv*, during the initialization of the simulation. Each contract gets associated with an account. We store the context of these executions in the system. Similar to Etheruem system, the contract's code executes when a transaction is sent to a contract account.

5 SOFTWARE DEMONSTRATION

eVIBES offers a web-based interface for managing the simulation process. A real-time connection to the Reducer enables the user

[6]https://github.com/i13-msrg/evibes

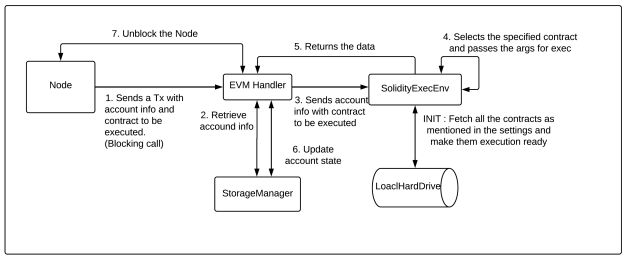

Figure 2: Smart contract execution workflow.

to view all the activities in the simulation. The simulation can simulate a large number of nodes to mimic the behavior of Ethereum Blockchain network.

6 CONCLUSIONS

We offered a design and development of eVIBES, an event-driven, concurrent Ethereum blockchain simulator for large-scale network simulations. The visualization of simulation outcome, the scalability and run-time reconfigurability of eVIBES make it a useful tool in analyzing the dynamic behavior of the blockchain.

ACKNOWLEDGMENTS

Alexander von Humboldt Foundation supported this project.

REFERENCES

[1] 2015. ETHEREUM: A secure decentralised generalised transaction ledger byzantium version adc4e61 - 2018-04-04. Retrieved August 10, 2018 from https://ethereum.github.io/yellowpaper/paper.pdf
[2] 2015. A Next-Generation Smart Contract and Decentralized Application Platform. Retrieved August 10, 2018 from https://github.com/ethereum/wiki/wiki/White-Paper
[3] JS. Banerjee, D. Goswami, and S. Nandi. 2014. OPNET: A New Paradigm for Simulation of Advanced Communication Systems. (2014), 319–328.
[4] A. Montresor and M. Jelasity. 2009. PeerSim: A scalable P2P simulator. , 99-100 pages.
[5] L. Stoykov, K. Zhang, and H. A. Jacobsen. 2017. VIBES: Fast Blockchain Simulations for Large-scale Peer-to-peer Networks: Demo. In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Posters and Demos*. ACM, 19–20.