Emulating an Ethereum Blockchain Network Using the LST

Andrei C. Azevedo, Eder J. Scheid, Muriel F. Franco, Lisandro Z. Granville

Informatics Institute (INF) – Federal University of Rio Grande do Sul (UFRGS)
Porto Alegre – RS – Brazil

{acazevedo,ejscheid,mffranco,granville}@inf.ufrgs.br

Abstract. *Besides the main blockchain use-case of exchanging cryptocurrencies, Distributed Applications (DApps) can also be developed on top of such a technology. However, due to the size of popular blockchains and price, testing these DApps in a real-world environment becomes challenging. Thus, blockchain emulators were proposed to address such as issue. This paper presents the experience of emulating an Ethereum network using a Docker-based lightweight testbed developed for Software Defined Networks (SDN).*

1. Introduction

Blockchain is a technology based on the concept of distributed ledgers that stores information in a decentralized and distributed network [Scheid et al. 2021]. It was first proposed in 2008, as the database and means to solve the double-spending problem for the Bitcoin cryptocurrency [Nakamoto 2009], and since then, its adoption has expanded to several other areas such as finances, cybersecurity and Internet of Things (IoT) [Monrat et al. 2019], as well as to other emerging cryptocurrencies, such as Ethereum [Buterin 2014].

Since the architecture of blockchains is complex, comprising several different layers (*e.g.*, network layer, data model layer, execution layer and application layer) [Belotti et al. 2019], it is difficult to create and maintain an actual blockchain environment for evaluation purposes. Further, the size of blockchains, such as Bitcoin and Ethereum, is over 400 GB, requiring dedicated hardware [Sanka and Cheung 2021]. Fortunately, it is possible to resort to blockchain emulation and simulators to avoid such difficulties. Simulation aims to reproduce a blockchain system model, enabling its evaluation in a parameterized way, without the need to implement the entire system [Paulavičius et al. 2021] whereas emulation aims to replicate the behavior of a system as closely as possible [Gill et al. 2021].

Thus, in this work, we rely on the Lightweight SDN Testbed (LST) [Kaihara et al. 2022] to emulate an Ethereum-based blockchain network containing different nodes. We defined a scenario with one signer creating and attesting blocks and two regular nodes sending transactions. We show that it is possible to emulate an Ethereum network in an flexible and reproducible manner while being possible to send transactions among the nodes and to synchronize the blockchain, enabling a complete evaluation of the system relying on real-world blockchain software.

The remainder of this paper is organized as follows. Section 2 presents an overview of LST and the Ethereum blockchain. Section 3 discusses related work on Ethereum blockchain simulator and emulators available in the literature. Section 4 describes the scenario for running the experiment, while Section 5 elaborates on its results. Finally, Section 6 concludes this paper and indicates future work on the topic.

2. Background

This section describes the two main technologies employed in this paper, LST and the Ethereum blockchain.

2.1. Lightweight SDN Testbed (LST)

LST [Kaihara et al. 2022] is a lightweight and easy-to-use tool for SDN and security studies. It allows reproducing scenarios in the context of SDN through the virtualization of physical infrastructures, by taking advantage of Docker containers. Through the virtualization offered by Docker containers, it is possible to achieve a highly configurable and isolated environment.

The tool relies on the Ryu Controller, a SDN framework for network management and control applications that supports several protocols, such as OpenFlow. Thus, it allows to perform different actions on the virtual switches *e.g.*, network analysis, changing network route policies and identifying malicious attacks. Further, as it relies on Docker containers, such a controller can be replaced and modified in a flexible manner.

2.2. Ethereum

Ethereum was proposed in 2015 as a blockchain able to support the creation of versatile applications [Buterin 2014]. It offers, within its blockchain ecosystem, a way for developers to create Decentralized Applications (DApps) using a Turing-complete programming language. With such a language, developers define immutable blockchain-based smart contracts that contain specific rules for the enforcement of transactions used by DApps. Thus, by providing this language, Ethereum became the *De Facto* standard for DApps.

Unlike Bitcoin, where the blockchain state is defined by Unspent Transaction Output (UTXO) transactions, Ethereum uses account-based states, where each account is defined by a set of fields (*e.g.*, balance), and state transitions are transactions of information or value between different accounts. Ethereum relies on the Proof-of-Stake (PoS) consensus method, where a node must stake capital (*i.e.*, Ethereum coins - ETH), into a smart contract. Only nodes with stake in the smart contract can propose and validate blocks; if the node does not follow a set of rules or propose invalid blocks, its stake is reduced, decreasing its chance of proposing blocks.

3. Related Work

There are several blockchain simulators available in the literature [Paulavičius et al. 2021]. However, in terms of blockchain emulation, there are few approaches in the literature. Thus, as the focus of this paper is on emulation, we only describe Ethereum-based emulators.

[Gill et al. 2021] emulates a real Ethereum blockchain by leveraging Containernet [Peuster et al. 2018], a Mininet fork that allows using Docker containers as hosts. It also provides network monitoring feature for retrieving statistics trough a Netflow collector, enabling traffic analysis. Although Mininet does support OpenFlow protocol, the proposed blockchain system emulator does not provide an interface for setting up a SDN Controller as does LST. Thus, although traffic can be analyzed with Netflow collected statistics, it is harder to perform any actions on incoming packets in the emulated system.

[Wang et al. 2019] presents an emulator that is highly scalable, but is designed to support only public blockchains that are based on the Proof-of-Work (PoW) consensus protocol. In contrast to LST, which uses Docker containers for each node, the proposed emulator is a Java application; thus, having a less isolated and flexible environment. In another work, [Polge et al. 2021] only emulates the networking layer of the blockchain, the remainder of the layers is simulated. Hence, it cannot be considered a full-fledged blockchain emulator but rather a hybrid emulator.

Based on such a review, it can be seen that, in the literature, one only approach focuses on using containers to emulate a blockchain network. However, SDN and lightweight emulation using Docker is not explored in this context. Hence, using LST as a flexible blockchain emulator with SDN to create a Ethereum network or any blockchain network presents an interesting opportunity.

4. Scenario

Figure 1 illustrates the blockchain network scenario emulated using LST. The host provides the required CPU, RAM, Disk for the virtualization of the elements of LST. Two blockchain nodes and a blockchain signer node are initialized as docker containers, and later connected to the virtual Switch container, creating the blockchain network topology. The Switch is connected to a custom Ryu Controller container, that enables us to record and later inspect the network flows. It also uses the network interface provided by the host, thus allowing the blockchain emulated system to have Internet connection.

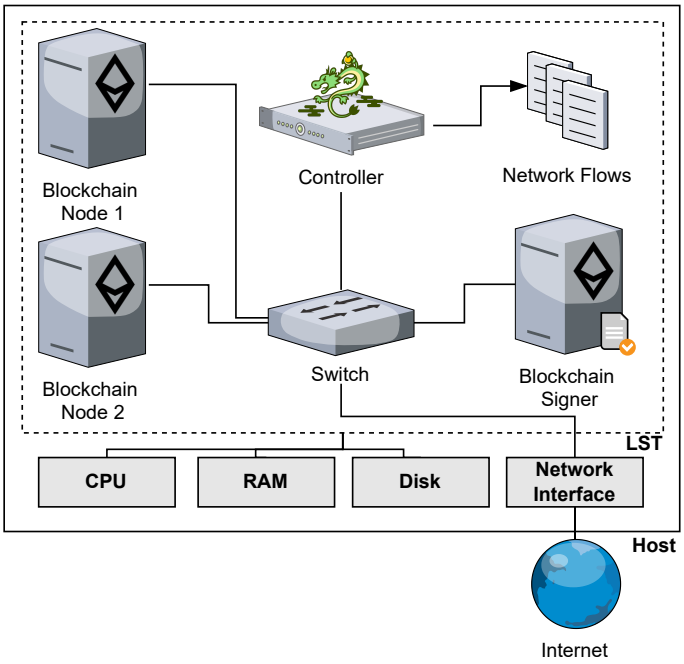

Figure 1. Emulated Scenario with LST

In this scenario, we have created a private Ethereum blockchain that relies on the Proof-of-Authority (PoA) consensus protocol, where one or more nodes are assigned as validators for the entire network. We define one of the participating nodes as the Ethereum Block Signer. This node contains the genesis file for our private Ethereum

blockchain system, which defines the initial data on the first block in the chain as well as other configurations (*e.g.*, difficulty, gas limit, initial account balances and minimum block time in seconds). Then, two other generic blockchain nodes, containing the same genesis file, are initialized and later connected to the virtual switch.

The virtual switch container is instantiated with a Ryu Controller that saves incoming packets from network into a Packet Capture (PCAP) file. This enables later inspection of the emulated blockchain system network traffic. It is important to mention that the controller can be customized and exchanged if required; thus, showing that LST can serve as a testbed for experiments regarding the analysis of traffic in different scenarios.

Once all of the participating nodes are connected to the virtual switch, they are instantiated using Geth, the official Ethereum client built in Go. After the node discovery process ends and the nodes are synchronizing with blockchain network, we can then start sending transactions to the network, check account balances, and inspect network flows that are recorded through the Ryu Controller.

The following steps were required to perform this experiment: *(i)* create the docker containers, and *(ii)* execute the experiment Python script. In the first step, it is created the docker images for the nodes, signer and OpenFlow switch. The images for the nodes and the signer contains the Geth client, and the genesis file for the creation of the Ethereum blockchain network. It is possible to use the default docker images provided by LST and install the required software, for example Geth for the Ethereum node. In the second step, we execute the experiment Python script, which creates the blockchain topology presented above, performs Ethereum transactions and check the accounts balance.

5. Experiment and Results

This experiment was conducted on a host with an Ubuntu 22.04 LTS operating system, with 16 GB RAM and an AMD® Ryzen 5 3600 6-core processor. The code to reproduce the scenario is available at `https://github.com/andrei-azevedo/ERRC23-ETH-LST`. Figure 2 demonstrates an example of an emulated Ethereum blockchain system using LST. In the example, we successfully sent transactions from a node using the function `sendTransaction()` that calls to a given `node1`'s Geth to send the transaction and check the account balance after the transaction.

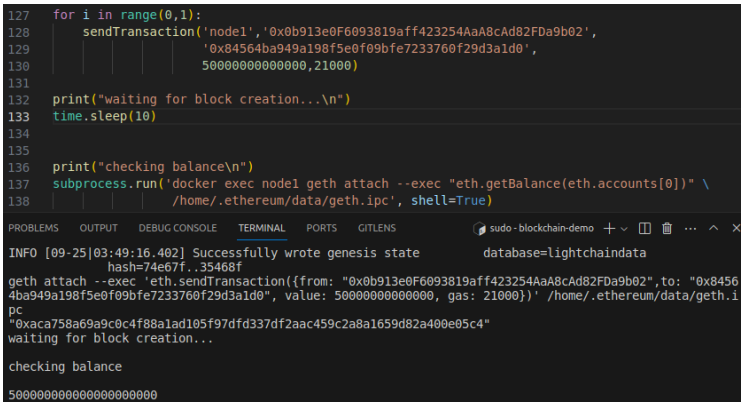

Figure 2. Sending a transaction and checking account balance in an Ethereum blockchain network emulated with LST

We can also see that LST is a very flexible tool that allows us to easily set up new nodes in the blockchain network and submit transactions with different ETH values. Figure 3 depicts an example of a different topology containing four common nodes (two more nodes than the original scenario), and randomly sending several transactions with different ETH values among them (see line 132). This allows us to verify the communication between several nodes in the network with the signer and verify if all the nodes are synchronizing with the blockchain. Further, with the use of LST and the experiment defined as a Python script, it can be repeated by other researches to achieve the same result, fostering reproducible research.

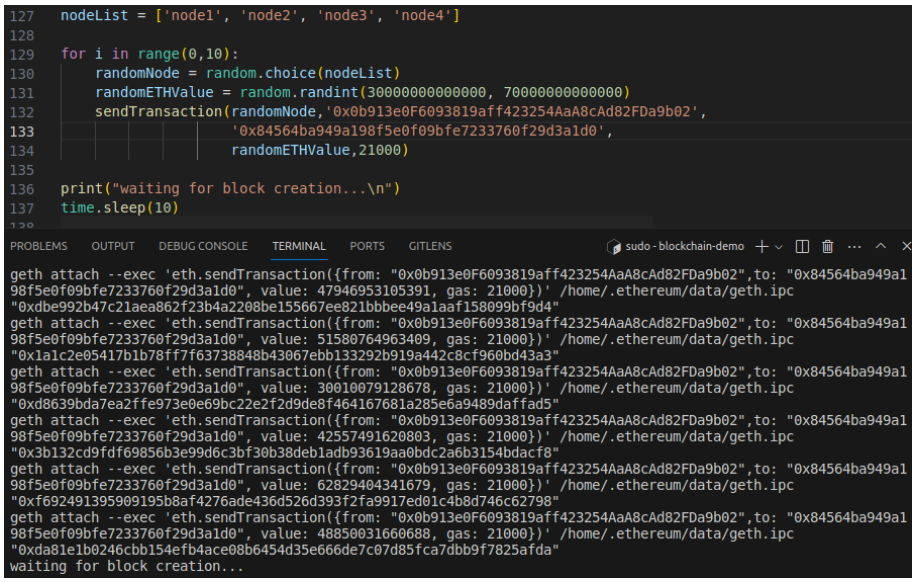

Figure 3. Sending several transactions from any of the four participating nodes with different ETH values.

6. Conclusion and Future Work

In this paper, we have demonstrated that it is feasible to emulate an Ethereum blockchain using Docker containers and SDN with the use of LST, which is a versatile and highly configurable tool. Thus, allowing us to perform several analyses and actions on the emulated network. By leveraging Docker container technology, the tool creates an isolated environment that is ideal for testing purposes on blockchain systems without relying on a testnet, which might require prohibitive hardware resources and human effort for configuration of nodes.

Our experiments demonstrated that setting up new nodes and sending transactions with different values in the emulated Ethereum blockchain network using LST is a straightforward task. In addition, the experiments can repeated with predictable results by other researchers aiming to test novel blockchain-based approaches and DApps in a real-world-like blockchain network with minimal effort.

For future work, it could be investigated the use of the SDN controller to identify Ethereum packets in the network traffic so that cryptojacker traffic is mitigated and to create smart contract interactions simulations with several real-world Ethereum nodes and traffic conditions.

Acknowledgment

This work was supported by The São Paulo Research Foundation (FAPESP) under the grant number 2020/05152-7, the PROFISSA project.

References

Belotti, M., Božic, N., Pujolle, G., and Secci, S. (2019). A Vademecum on Blockchain Technologies:When, Which and How. *IEEE Access*, pages 3796–3838.

Buterin, V. (2014). Ethereum Whitepaper: A Next-Generation Smart Contract and Decentralized Application Platform. `https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf`.

Gill, S., Lee, B., and Qiao, Y. (2021). Containerchain: A Blockchain System Emulator based on Mininet and Containers. In *IEEE International Conference on Blockchain (Blockchain 2021)*, pages 1–7, Melbourne, Australia.

Kaihara, A., Bondan, L., Gondim, J., Rodrigues, G., Marotta, M., and Rodrigues, G. (2022). LST: Testbed Emulado Leve para Redes SDN Aplicado ao Contexto de Seguranca. In *Anais Estendidos do XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 41–48, Fortaleza/CE.

Monrat, A. A., Schelén, O., and Andersson, K. (2019). A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities. *IEEE Access*, 7:117134–117151.

Nakamoto, S. (2009). Bitcoin: A Peer-to-Peer Electronic Cash System. `https://bitcoin.org/bitcoin.pdf`.

Paulavičius, R., Grigaitis, S., and Filatovas, E. (2021). An Overview and Current Status of Blockchain Simulators. *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*.

Peuster, M., Kampmeyer, J., and Karl, H. (2018). Containernet 2.0: A Rapid Prototyping Platform for Hybrid Service Function Chains. In *IEEE Conference on Network Softwarization and Workshops (NetSoft 2018)*, pages 335–337, Montreal,Canada.

Polge, J., Ghatpande, S., Kubler, S., Robert, J., and Le Traon, Y. (2021). BlockPerf: A Hybrid Blockchain Emulator/Simulator Framework. *IEEE Access*, 9:107858–107872.

Sanka, A. I. and Cheung, R. C. (2021). A Systematic Review of Blockchain Scalability: Issues, Solutions, Analysis and Future Research. *Journal of Network and Computer Applications*, 195:103232.

Scheid, E. J., Rodrigues, B., Killer, C., Franco, M., Rafati, S., and Stiller, B. (2021). Blockchains and Distributed Ledgers Uncovered: Clarifications, Achievements, and Open Issues. In *Advancing Research in Information and Communication Technology*, volume 600 of *IFIP AICT Festschrifts*, pages 289–317. Springer, Cham, Switzerland.

Wang, X., Al-Mamun, A., Yan, F., and Zhao, D. (2019). Toward Accurate and Efficient Emulation of Public Blockchains in the Cloud. In *International Conference on Cloud Computing (CLOUD 2019)*, pages 67–82, San Diego, CA, USA. Springer International Publishing.

All links accessed on 11/09/2023