A Co-Simulation Approach of Blockchain Connected Last-Mile Delivery Using Autonomous Robots

Oussema Gharsallaoui*, Damien Nicolas*, Marie-Laure Watrinet*, and Ion Turcanu*
*Luxembourg Institute of Science and Technology (LIST), Luxembourg
{ oussema.gharsallaoui, damien.nicolas, marie-laure.watrinet, ion.turcanu }@list.lu

***Abstract*—The integration of autonomous robots in last-mile delivery systems raises technological challenges such as communication, navigation, and cybersecurity. Blockchain offers a secure and transparent solution, but its scalability is a concern. In this paper, we present a co-simulation framework that combines the SUMO mobility simulator with the Ethereum Proof of Stake (POS) blockchain. We evaluate the impact of increasing the number of autonomous robots on blockchain performance, using Transaction Execution Time (TET) and Transaction Finality Time (TFT) as metrics. Our results demonstrate the feasibility and challenges of using blockchain in autonomous delivery systems.**

I. INTRODUCTION

The growing trend of using autonomous robots in last-mile delivery raises technological concerns such as communication and connectivity, navigation and localization, and cybersecurity [1]. This encourages the adoption of novel technologies, such as blockchain, due to its characteristics. Indeed, blockchain provides a secure, reliable, and transparent system for sharing data among multiple parties. However, blockchain is also known to still have some scalability challenges [2].

In this context, evaluating the scalability limits of blockchain integration with autonomous robots for last-mile delivery use cases is critical. However, conducting experiments in real-world urban mobility scenarios is associated with high costs and safety risks. Therefore, a popular approach among researchers is to first evaluate proposed solutions through simulations before implementing them in real-world situations [3]. To the best of our knowledge, existing urban mobility simulators, such as SUMO [4] and PTV [5], do not incorporate blockchain technology into their simulation framework. Therefore, it is necessary to implement the blockchain simulation component separately. This paper fills this gap by introducing a user-friendly simulator that allows users to run an integrated SUMO simulation of autonomous robots for last-mile delivery, which is bilaterally connected to a blockchain.

In summary, the contribution of this paper is twofold. First, we propose a co-simulation approach that couples the open-source mobility simulator SUMO with the Ethereum Proof of Stake (POS) Blokchain. This enables realistic, large-scale simulations of blockchain-enabled mobility solutions. Second, we demonstrate the potential of the proposed solution by examining the impact of increasing the number of autonomous robots in last-mile delivery on the performance of the Ethereum POS Blokchain. To achieve this goal, we use two evaluation metrics: (i) Transaction Execution Time (TET), which is defined as the time required for a transaction to be accepted and stored on the blockchain, and (ii) Transaction Finality Time (TFT), which refers to the point after which it is impossible to tamper with the transaction.

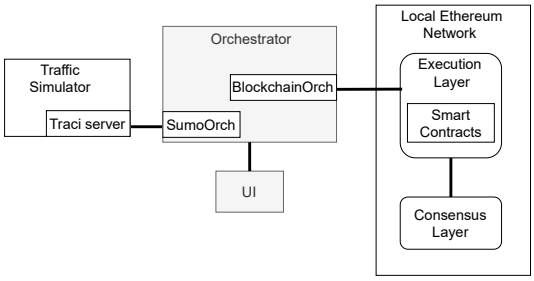


Figure 1. The architecture of the proposed co-simulation approach.

II. SYSTEM DESIGN

This section presents the system design of the proposed co-simulation framework. As shown in Figure 1, the platform consists of three main components: (i) the traffic simulator, (ii) the local Ethereum network, and (iii) the orchestrator with its associated User Interface (UI). The contribution of this paper includes the orchestrator and the UI (gray blocks in Figure 1). An illustration of the UI is shown in Figure 2.

The role of the traffic simulator is to run the urban mobility part of the simulation using the Traffic Control Interface (TraCI) server as defined in [6]. TraCI allows to retrieve real-time data from the simulated autonomous robots and to modify their behavior in real time.

The role of the local Ethereum network is to run Ethereum smart contracts. It consists of two layers: the execution layer and the consensus layer. Communication between these two layers is performed through a local Remote Procedure Call (RPC) connection. The orchestrator calls the execution layer to store the generated transactions. The execution layer first verifies the transactions and passes them to the consensus layer for final verification and validation. Once validated, the transactions are stored on the blockchain.

The main role of the orchestrator is to build Ethereum transactions containing real-time data collected by the traffic simulator and send them to the local Ethereum network. The orchestrator is a client of the TraCI server and is connected to the local Ethereum network via JSON RPC APIs.

In our particular case, the orchestrator manages a three-legged last-mile delivery scenario using autonomous robots. In

6th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS 2024)

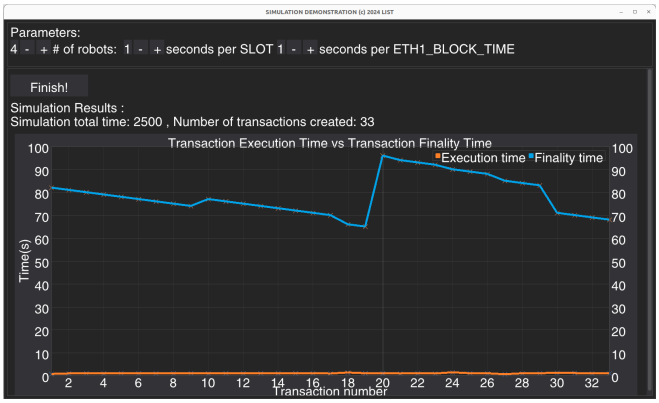


Figure 2. An illustration of the UI.

the first leg, a delivery truck transports N autonomous robots from the company's warehouse to a designated parking area called the *cluster hub*. In the second leg, each robot exits the truck and begins its journey to deliver packages to end customers. In the third leg, the truck waits for all N robots to return and then transports them back to the warehouse.

Figure 3 illustrates the second leg of this scenario, showing the interactions between stakeholders. The orchestrator generates transactions for the steps 2, 4, 6, 7, 8, 12 and 14. In the event of a conflict in step 9, the orchestrator generates a report explaining the details of the conflict based on the data collected from the transactions sent to the blockchain.

III. Implementation

The simulation environment runs on a virtual machine configured with 128 GByte of RAM, 96 CPUs, and 100 GByte of disk space. To run the local Ethereum network, we have installed the following binaries: Geth version 1.13.15, Lighthouse and its CLI tool version 5.1.3. In addition, Node.js version 18.20.3 is used to execute scripts to migrate the local Ethereum network from Proof of Authority (POA) to POS. Python version 3.8 and SUMO version 1.20.0 are installed to run the user interface, orchestrator Python scripts, and mobility simulation.

Four smart contracts are developed with Solidity v8.0 for the considered scenario: (i) The CUSTOMER smart contract is used to store the data related to customers on the Ethereum blockchain; (ii) The TRUCK DRIVER smart contract is used to store the data related to the truck driver; (iii) The ROBOT smart contract is used to store the data related to autonomous robots; (iv) The TRIP smart contract is used to store and share the data related to parties involved in the delivery process.

A. Orchestrator implementation

The orchestrator consists of a set of shell and Python scripts that are responsible for running the simulation. It initializes all the simulation models and smart contracts with the appropriate values, sets/gets their inputs/outputs, and coordinates their progress over the simulated time.

In a first step, the orchestrator starts the simulation by launching a local Ethereum POS network using a shell script. The local Ethereum POS network is a customized version

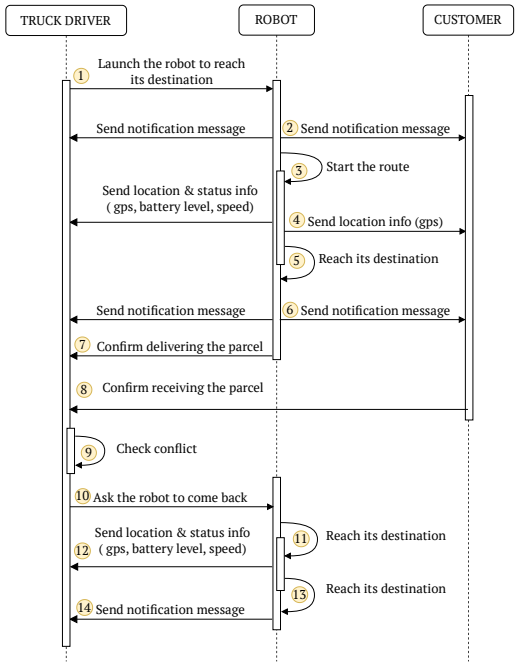


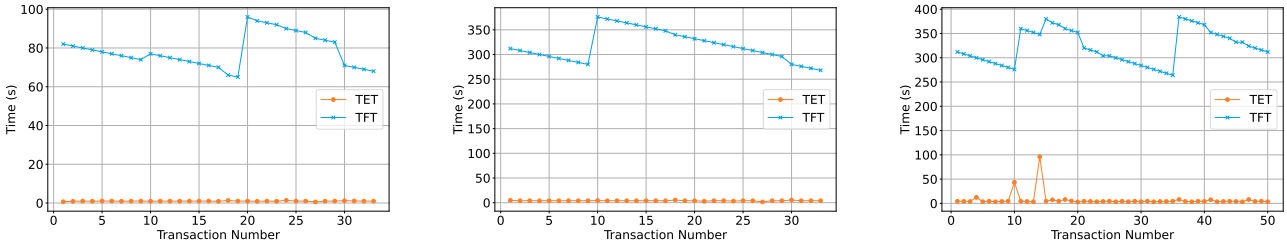
Figure 3. The UML sequence diagram of last-mile delivery scenario.

of the open source project in [7]. It takes as input the number of nodes N, the number of seconds per slot S (block period of the consensus layer), and the number of seconds per ETH1_Block_Time T (block period of the execution layer). The orchestrator starts: the execution layer consisting of N Geth nodes, and the consensus layer consisting of N Lighthouse beacon chain nodes with their associated N Lighthouse validator clients. Each validator client is connected to a beacon chain node to validate the blocks on the beacon chain. Each beacon chain node is connected to a Geth node to stay updated on transactions executed in the execution layer.

In a second step, the orchestrator deploys the smart contracts to the local Ethereum network and retrieves the addresses of the smart contracts from the Ethereum blockchain as proof of successful deployment. In a third step, the orchestrator runs SUMO and its TraCI server. It uses two Python modules, *SumoOrch* and *BlockchainOrch*, to couple SUMO and the local Ethereum network.

The SumoOrch module provides reusable functions to dynamically add new robots and trucks to SUMO in real time and assign them to their routes based on data provided by the orchestrator. It allows to obtain real-time data from the robots and trucks, such as GPS location and battery status. It is developed based on the TraCI API and is also responsible for starting a TraCI server and connecting to the SUMO simulator.

BlockchainOrch provides reusable functions to interact in real time with the smart contracts deployed on the local Ethereum network. This module is developed based on Web3 API. It checks if the local Ethereum network is already running and available for connection. In case of a connection issue to the local Ethereum network, the module shows an error log.



(a) 4 robots, 1 s (b) 4 robots, 4 s (c) 10 robots, 4 s

Figure 4. TET and TFT when running varying number of robots and seconds per slot.

Finally, the orchestrator calculates the TET and TFT of each transaction generated by the simulation. TET is defined as the time difference between the moment the transaction is sent to the execution layer and the moment we receive the confirmation back from the execution layer. TFT is the time that elapses from the sending of a transaction to the point where the transaction can be considered to be settled and therefore cannot be altered, reversed, or canceled.

B. Results

For simplicity, we consider the number of seconds per slot to be equal to the number of seconds per ETH1_Block_Time. Figure 4a shows the results of the simulation using four autonomous robots, with the time slot fixed at 1 s. We can see that the total TET is stable at 1 s, while the TFT varies between 64–95 s, i.e. the receiver has to wait about 1.5 min to make sure that the transaction is immutable.

If we increase the time slot to 4 s (see Figure 4b), i.e., to give nodes more time to register more transactions per block, we see that the TET remains stable around 4 s, but the TFT increases to 256–380 s. This means that our blockchain is still stable with four robots, but the receiver has to wait up to 6 min to make sure the transaction is immutable.

Finally, Figure 4c shows the results of the simulation when we increase the number of autonomous robots to 10 and keep 4 s as the time slot. We see a perturbation in the TET, as some of the transactions took longer than expected (one transaction execution took 50 s, while another transaction execution took 100 s). This is most likely due to synchronization issues between the blockchain nodes. However, increasing the number of robots does not affect the TFT, which is still between 256–380 s, similar to Figure 4b.

These results confirm that there is a trade-off between the efficiency of the blockchain, in terms of registered transactions per block, and the time a user would have to wait before a transaction becomes irreversible. In particular, while increasing the number of seconds per slot is beneficial in terms of blockchain storage efficiency, it has a negative impact on TFT. On the other hand, increasing the number of robots has very limited impact on the TFT, but it has a negative impact on the blockchain stability.

IV. Conclusion

In this paper, we presented a co-simulation framework that combines the SUMO mobility simulator with the Ethereum POS blockchain. The proposed solution is applied in a last-mile delivery system and used to evaluate the blockchain performance when increasing the number of autonomous robots. The orchestrator allows to highlight the TET and TFT through a user interface, and to run multiple simulations by changing the number of robots and the number of seconds per slot.

In addition to providing valuable insights into the practical application of the blockchain technology for the considered use case, our solution enables rapid testing and development of innovative decentralized mobility-as-a-service platforms for urban mobility systems. As future work, we plan to generalize the orchestrator to enable the execution of different user-defined scenarios. We also intend to include network simulation into the framework to allow for more accurate performance evaluation based on real-time network status.

V. Acknowledgement

Supported by the IN2CCAM project, funded by European Union's Horizon Europe research and innovation program under grant agreement No 101076791.

References

[1] E. Ayyildiz and M. Erdogan, "Addressing the challenges of using autonomous robots for last-mile delivery," *Computers & Industrial Engineering*, vol. 190, p. 110 096, 2024.

[2] G. Bendiab, A. Hameurlaine, G. Germanos, N. Kolokotronis, and S. Shiaeles, "Autonomous vehicles security: Challenges and solutions using blockchain and artificial intelligence," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 3614–3637, 2023.

[3] T. Hardes, I. Turcanu, and C. Sommer, "Poster: A Case for Heterogenous Co-Simulation of Cooperative and Autonomous Driving," in *14th IEEE Vehicular Networking Conference (VNC 2023)*, Istanbul, Türkiye: IEEE, Apr. 2023.

[4] P. A. Lopez et al., "Microscopic Traffic Simulation using SUMO," in *21st international conference on intelligent transportation systems (ITSC)*, IEEE, 2018, pp. 2575–2582.

[5] T. Kučera and J. Chocholáč, "Design of the city logistics simulation model using PTV VISSIM software," *Transportation Research Procedia*, vol. 53, pp. 258–265, 2021.

[6] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux, "TraCI: an interface for coupling road traffic and network simulators," in *Proceedings of the 11th communications and networking simulation symposium*, 2008, pp. 155–163.

[7] P. Chunhapanya, *local-tesnet*, https://github.com/ethereum/local-testnet, 2023.