Evaluation of Alastria DLT

Riccardo Cecco
Universidad Politécnica de Cataluña
Email: riccardo.cecco@estudiantat.upc.edu

Oriol Martinez Acón
Universidad Politécnica de Cataluña
Email: oriol.martinez.acon@estudiantat.upc.edu

***Abstract*—In this article, we are going to do a technical evaluation of Alastria trying to compare every aspect of it with the most well-known private and non-private DLT platforms. For the testing part, we increase step by step the complexity of the tests. Initially, we will run stress tests in our local environment using a smart contract, thanks to the use of Ganache and Truffle. Then we move on deploying a smart contract on Alastria. At the end we do a comparison of the various metrics we have obtained from our local environment and Alastria. Finally, we make a comparison of the key points of the well-know alternatives regarding permissioned Blockchain platforms.**

1. Introduction

1.1. Alastria

Alastria is a non-profit association that promotes the digital economy through the development of decentralized ledger technologies/Blockchain. Alastria partners have two operational networks (Network T and Network B) on which nodes can be deployed (either regular nodes or critical: validators and boot nodes).

1.1.1. Alastria Network T. Alastria Red T is based on Quorum. Quorum is based on Ethereum and is a fork of go-Ethereum. Unlike Ethereum which is open and permissionless, Quorum is a permissioned (i.e. private) network. In Quorum blockchain, there is only a single member owning all the nodes, or, a consortium blockchain network, where multiple members each own a portion of the network. Quorum uses either raft-based consensus or Istanbul BFT or Clique Proof-of-Authority (PoA).

1.1.2. Alastria Network B. Alastria Red B is based on Hyperledger Besu. Hyperledger Besu is an Ethereum client designed to be enterprise-friendly for both public and private permissioned network use cases.

From this point on, when we talk about Alastria we will refer to the T network based on GoQuorum.

2. Local Deployment

We have used Ganache and Truffle tools.
Ganache is a personal blockchain system for Ethereum development which you can use to deploy contracts, develop your applications, and run tests.
Truffle is a development environment utilizing the EVM (Ethereum Virtual Machine) as a basis. One of its purposes is to make the development of smart contracts more straightforward and more accessible.

3. Alastria Deployment

We have used REMIX IDE and MetaMask tools.
MetaMask is a cryptocurrency wallet that enables users to access the Web3 ecosystem of decentralized applications in order to execute different transactions.
Remix IDE is used for the entire journey of contract development with Solidity language. We have deployed our smart contract through the UPC's node inside the Alastria Blockchain (IP: 84.88.85.36).

4. Evaluation

4.1. GoQuorum

GoQuorum is an Ethereum-based distributed ledger protocol with transaction/contract privacy and different consensus mechanisms. Benefits of using GoQuorum:

- **Privacy** - GoQuorum supports private transactions and private contracts through public/private state separation, and utilizes peer-to-peer encrypted message exchanges for directed transfer of private data to network participants.
- **Alternative Consensus Mechanisms**: GoQuorum instead offers multiple consensus mechanisms that are more appropriate for consortium chains.
- **Peer Permissioning** - node/peer permissioning, ensuring only known parties can join the network.
- **Account Management** - GoQuorum introduced account plugins, which allows GoQuorum or clef to be extended with alternative methods of managing accounts including external vaults.
- **Higher Performance**: GoQuorum provides higher transaction speed since generally private contracts are used and private contracts work better than public ones. It has been tested in different papers that the Raft consensus algorithm performs better than the Istanbul BFT.

4.2. Consensus

Alastria uses **IBFT**. Istanbul BFT (IBFT) is a Byzantine fault-tolerant (BFT) consensus algorithm that is used for implementing state-machine replication in the Quorum blockchain.
Istanbul BFT is the best consensus algorithm for permissioned Blockchain because it provides the following benefits:

- **Immediate block finality**: There's only 1 block proposed at a given chain height. The single-chain thus removes forking, uncle blocks, and the risk that a transaction may be "undone" once on the chain at a later time.
- **Reduced time between blocks**: The effort needed to construct and validate blocks is significantly reduced (in particular with respect to PoW), greatly increasing the throughput of the chain.
- **High data integrity and fault tolerance**: IBFT uses a group of validators to ensure the integrity of each block being proposed. A super-majority 66 % of these validators are required to sign the block prior to insertion into the chain, making block forgery very difficult. The 'leadership' of the group also rotates over time—ensuring a faulty node cannot exert long-term influence over the chain.
- **Operationally flexible**: The group of validators can be modified in time, ensuring the group contains only full-trusted nodes.

4.2.1. Comparing consensus protocols. GoQuorum implements the QBFT, IBFT, Raft, and Clique proof of authority (PoA) consensus protocols.

- **Immediate finality**: QBFT, IBFT, and Raft have immediate finality. Clique doesn't have immediate finality.
- **Minimum number of validators**: To be Byzantine fault-tolerant, QBFT and IBFT require a minimum of four validators. Raft and Clique can operate with a single validator but operating with a single validator offers no redundancy if the validator fails.
- **Liveness**: Raft and Clique are more fault-tolerant than QBFT and IBFT. Raft and Clique tolerate up to half of the validators failing.
- **Speed**: Reaching consensus and adding blocks is fastest in Raft networks, then in Clique networks. For QBFT and IBFT, the time to add new blocks increases as the number of validators increases.

4.2.2. Analysis IBFT vs RAFT. In many implementations, Quorum is used with RAFT, hence we want to understand why for the implementation of Quorum using Golang for the Alastria network they opted for IBFT. In the paper [4], in section 3.6.3, the authors did a comparison between IBFT and RAFT in terms of transaction throughput and transaction latency. The result is that RAFT and IBFT are comparable in terms of throughput except that RAFT performs slightly better at higher input transaction rate and IBFT slightly overperforms RAFT for lower transaction rates.
Fault tolerance evaluation: Raft can tolerate an f-number of faulty nodes in n=2f+1, where n is the total number of nodes. IBFT can tolerate an f-number of faulty nodes in n=3f+1, where n is the total number of nodes. IBFT is more expensive to run as it requires more nodes to stay fault-tolerant because it's not byzantine fault-tolerant but in different scenarios like permissionated blockchains where nodes are holded by different companies, it's important to have some BFT properties in order to be sure that everyone is beheaving correctly.

4.3. Cost

Cost outlines the associated cost incurred, if any, for any transaction to process or store data in the ledger. A system in which it is rather expensive to process and store any data might face the risk of being omitted in favor of others. Cost is often referred to as transaction fee in the blockchain domain. **Alastria hasn't any gas fee.**

5. Testing Our Local Environment

In this section we have tested our local environment to see if the smart contract, pet shop system, can tackle multiple transactions in the blockchain without leading into inoperative state. Then, the goal of the test is to show how many transactions are needed to make the system collapse and also to see the time spent per each transaction done. This local environment has running Ganache as the blockchain network system where all the transactions will be transmitted and, Truffle as the tool to generate all the transactions to test the smart contract. It is also important to highlight that the deployment of the smart contract is by the Truffle software. Both Ganache and Truffle tests are running on the same machine. The resources of the machine used in the local environment to make all the test are:

- **CPU**: Intel Core i7-7700HQ CPU @ 2.80GHz max 3.80 GHz, with 4 Cores.
- **RAM**: 16.0 GB DDR4.

The idea is to have different participants who are trying to adopt pets and then simulate a real scenario where people try to compete to obtain one of the items/objects in the blockchain system, in this case a pet. The script to recreate this use of case have been written in Bash, and it sends a lot of transactions per user using the Adopting function of our smart contract. The script creates X accounts and X pets. For each account, our script generates a random number Y between 1 and X, and it will be the Pet who that account wants to adopt. Each account follows the steps1 shown below in the diagram:

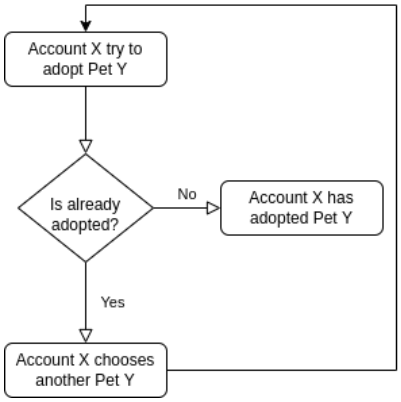


Figure 1. Steps

The script allows several transactions to be sent simultaneously. We are taking into account these statistics:

- **Number of transactions**
- **Time spent for each transaction**: we capture the instant when a transaction is sent and the instant when the response comes back for measuring the response time.
- **Performance**

All resources are available in our Github's repository.

5.1. Results

We have done different test depends on the number X and we have checked the resources of our computers by installing a node explorer on our OS. By using Prometheus and Grafana, we have generated these plots. We are able to test only for X=55, hence 55 accounts and 55 pets, because the computation to complete the transactions uses a lot of resources, and our laptops have crashed with more accounts and animals. As we see from the image 2 in the Host's tag, the IP is 172.17.0.1 because we are working in our local environment.

Figure 2. Local environment

In the figure 3, we can see that the computation of this test has almost saturated the entire computational capacity of our computer.

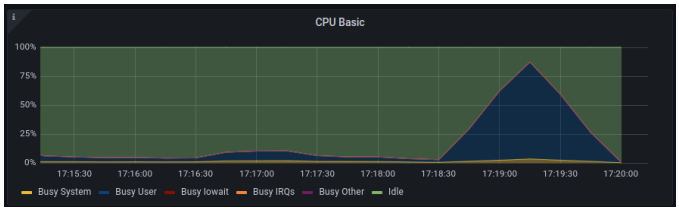


Figure 3. CPU USAGE

In the figure 4, we can see that the computation of this test has almost saturated also the RAM of our computer.

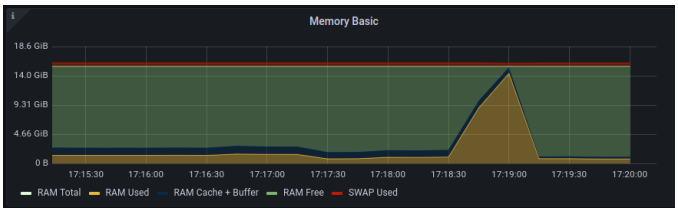


Figure 4. MEMORY USAGE

The images 3 and 4 show how computationally difficult it is and how many resources are used to compute the hash of each block. We have done an evaluation about the time spent for each transaction by taking into account the instant when a transaction is sent and the instant when the response comes back. Our results:

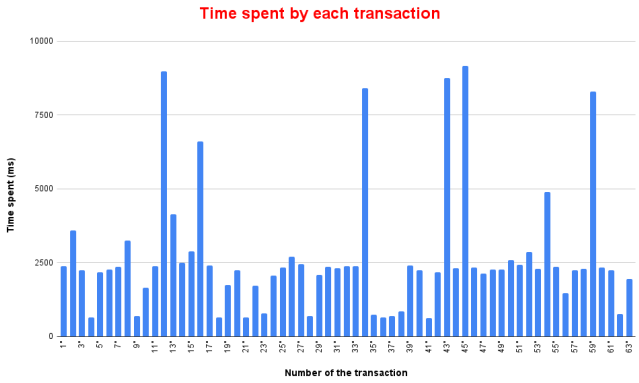


Figure 5. Overall analysis

The average time spent for a transaction is **2659.62 ms**. The tutorial and an execution example is available in the description of our repository.

6. Testing Alastria Node

6.1. Deploying the smart contract

The deployment of the smart contract in the Alastria network has been made through Remix IDE and Metamask. Is important to remind that the deployment of the smart contract and all the transactions done in the Alastria network has no gass fee. The node that has been used is a node from UPC that is connected to the Alastria network T. UPC's node is hosted on IP: 84.88.85.36. The figure 6 depicts how the smart contract has been deployed using the tools mentioned.

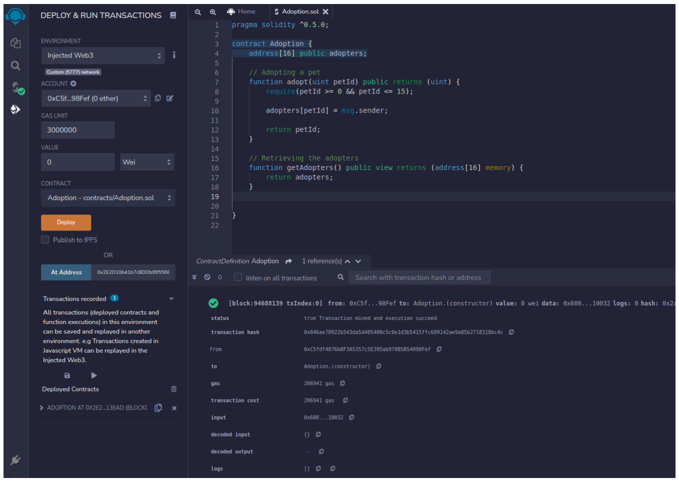


Figure 6. Deployment of the smart contract in Alastria Red T with Remix IDE and Metamask.

The changes are also visible in the Alastria's official monitoring tool, blkexplorer. This tool monitorize all the transactions done in Alastria red T.

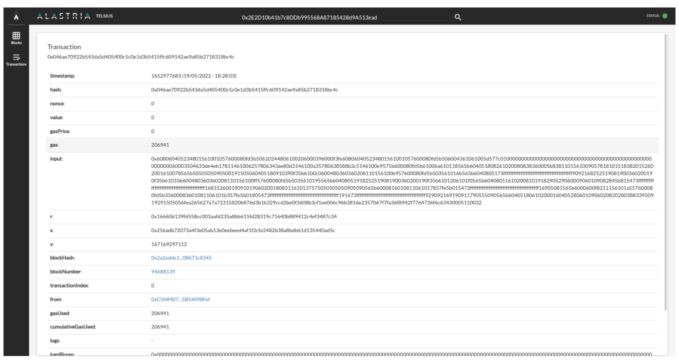


Figure 7. Transaction created in Alastria red T to deploy the pet-shop smart contract.

Finally we made a test of the smart contract deployed by adopting one of the pets, pet number 0, without cost to do the transaction.

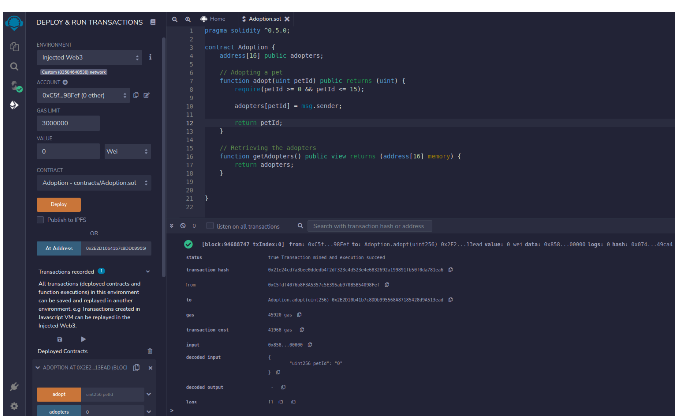


Figure 8. Adopting the pet ID 0 in the smart contract.

In the figure 9 we can see that the pet with the ID 0 has been adopted with the same account address that with the smart contract deployment.

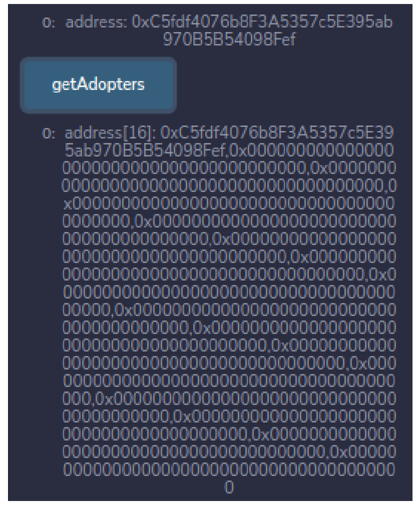


Figure 9. List of all the adopters.

6.2. Evaluate the smart contract

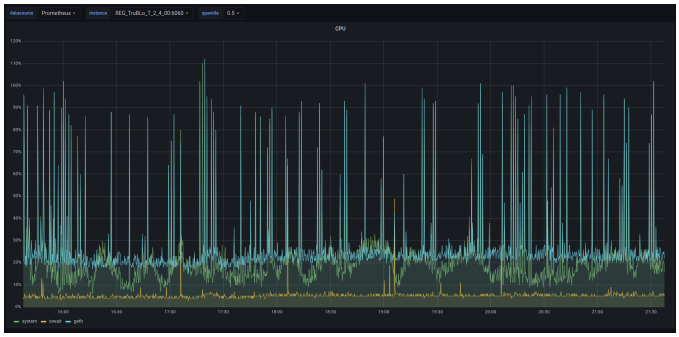
For the evaluation, we have used the data of the node called **REG_TruBLo_T_2_4_00** and the **UPC's node**.
We have decided to take into account 2 nodes because UPC's node is not able to offer blockchain specific metrics in Prometheus's format (only metrics such as CPU usage, Disk usage ...) while REG_TruBLo_T_2_4_00 has this implementation.
REG_TruBLo_T_2_4_00's hosted in an AWS EC2 instance with these characteristics: Cores:2, RAM:4GB, Storage: 96GB. IP of this machine: 185.180.8.152. For getting all the data and plotting them, we have used Prometheus combined with Grafana's dashboards.

The original idea was to try on the UPC's node the same script that we have used to test our local environment, but an error called "Transaction underprice" in recent days prevented us from doing this test. The metrics below are general for

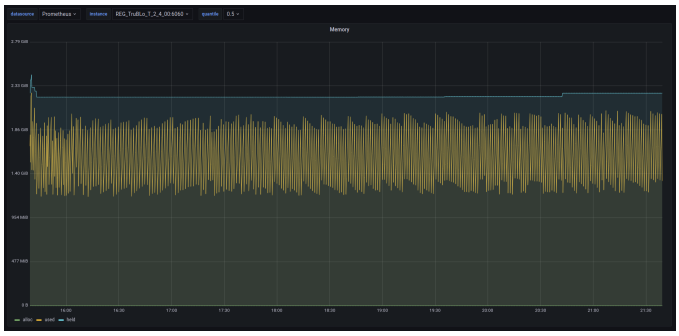
6.3. Metrics

The metrics below are for general node utilization since it was not possible to stress test the network. We can see that the workload follows approximately a regular pattern, therefore we could take decisions based on the analysis of the different components' computational load of our machine, by choosing the hardware who suits/perform better with the workload of our Alastria's node and, also we can save money and have less impact on the environment.

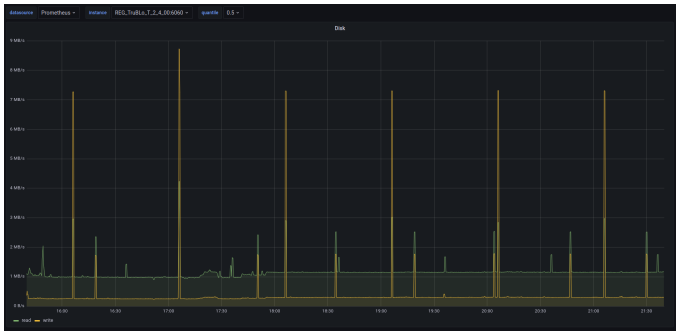
6.3.1. CPU.



6.3.2. Memory.

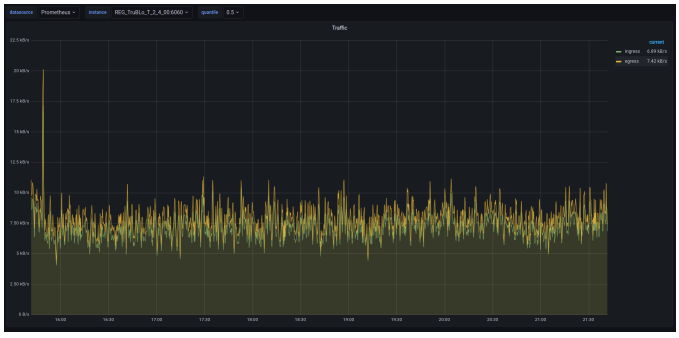


6.3.3. Disk.

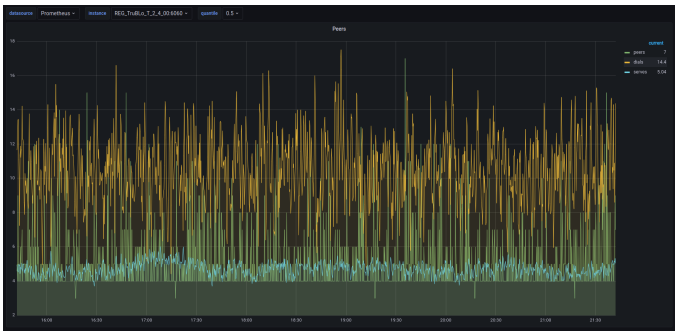


6.4. Network

6.4.1. Traffic.



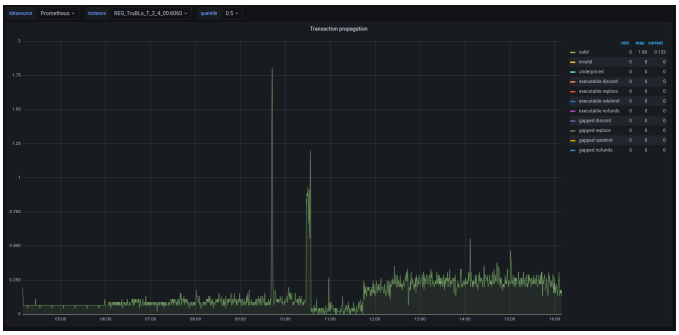
6.4.2. Peers.



6.5. Blockchain

6.5.1. Block Processing.

6.5.2. Transactions propagation.



7. Comparison with other DLTs

We saw that there are already different papers who analyzed the differences between the other alternatives for permissioned blockchain platforms. Below, we have collected data from these papers to show in a schematic view of the differences between these platforms. For more detailed information, we advice you to read the papers: [1], [2] ,and [5].

7.1. Permissioned Blockchain Platforms Comparison

In this section, we want to take in account only permissioned blockchain platforms that support smart contracts, therefore we will compare Quorum with Hyperledger Fabric, R3 Corda and MultiChain.

	Consensus	Smart Contract	Open Source	Support/Governance	Cryptocurrency	Smart contract language(s) supported
Hyperledger Fabric	Voting-based (Solo) or pluggable consensus	☑	☑	Linux Foundation	☐	Go, Node.js, Java
Ethereum	PoW or Cloque PoA	☑	☑	Ethereum developers	☑	Solidity, Vyper
Quorum	Clique PoA or RAFT-based or Istanbul BFT	☑	☑	Ethereum developers & JPMorgan Chase	☐	Solidity, Vyper
MultiChain	Round Robin validation	☐	☑	Coin sciences	☐	
R3 Corda	Single Notary, Raft, BFT-SMaRt	☑	☑	R3 Consortium	☐	Java, Kotlin

The image below was taken from the article [5]:

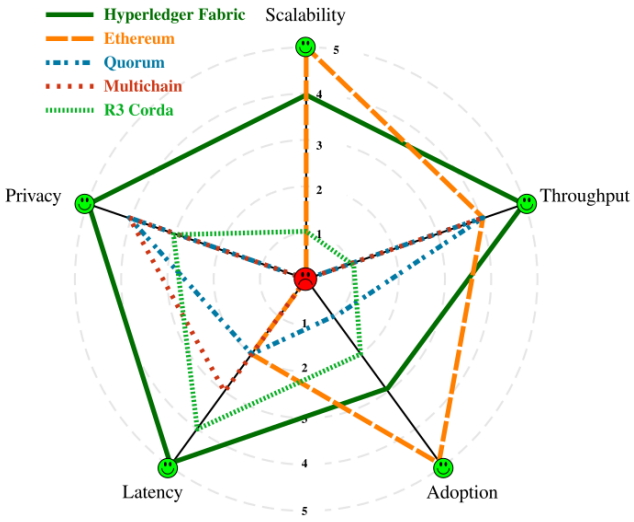


Figure 10. Overall analysis

Regarding the results presented in the article [5], Hyperledger Fabric is the most balanced.

7.2. Comparison with general DLTs

Platforms	Type	Block Size	Block Time	Cost	Energy	Consensus
Bitcoin	Public	1 MB	10m	☑	High	PoW
Etherum	Public	Implicit restriction	15s	☑	High	PoW
Eos	Public	MB, dynamically configurable	0.5s	☑	N	DPos
Cardano	Private	Configurable	20s	☑	N	Ouroboros
Sawtooth	Private	Configurable	NA	☐	Very Low	Tangle
IOTA	Private	Configurable	Configurable	☐	Very Low	Distributed consensus among indentified validators

Figure 11. Overall analysis

Comparing Alastria with general DLTs, organizations that want to implement Alastria have the option of configuring many factors such as block time and block size. Alastria doesn't have a cost for each transaction, therefore the gas fees are 0. GoQuorum is based on IBFT, therefore the demand of energy is very low and it's more sustainable.

8. Future Works

8.1. Local environment

- Moving Ganache network to another node (machine). If Ganache is running in the same machine that creates all the transactions (Truffle test), there is a lot of unnecessary noise that do not let us to test only the smart contract running on the blockchain system.

8.2. Alastria environment

- Adapting the local script to send all the transactions to the Alastria node.
- Creating a continuous test platform to monitorize error in the net. We found an error called "Transaction underpriced" who didn't allow us to deploy and test the platform in a good way. It could be interesting to see how many times errors happened by implementing a console log on our node.
- Using more than one node to stress the deployed smart contract in the Alastria blockchain. Because if we are using only one node we are not simulating a real scenario where there are a lot of nodes trying to interact with the smart contract.

9. Conclusion

To sum up, it was very interesting and instructive to have to interact with a real blockchain like Alastria. We think that Alastria is a solid project with a well organized infrastructures, counting the continuous development of the application and the continuous integration with business realities that are part of different sectors.
For the evaluation of the technical part, we think that all the decision that Alastria made are consistent, first of all it implemented GoQuorum with Ibft instead of the others and it's the right decision to guarantee a byzantine fault-toleran in a permissioned blockchain holded by different companies. The project was complex as it was the first time for both of us to interface with many of the tools used. We are

very happy because we learned how to use many different tools that also allowed us to create our own local testing environment. In addition, we were able to get real, concrete results that allowed us to make an overall evaluation of our local environment. Due to the transaction underprice error, we haven't the possibility to do the stress test of the Alastria network . If Alastria wants to become more and more accessible to different businesses and sectors, it surely needs to make the network more robust to bugs and errors of all kinds.

References

[1] M. CHOWDHURY, MD. SADEK FERDOUS, K. BISWAS, N. CHOWDHURY, A. S. M. KAYES, M. ALAZAB, P. WATTERS, A Comparative Analysis of Distributed Ledger Technology Platforms

[2] S. FAN, H. KHAZAEI, S. GHAEMI, P. MUSILEK, A Performance Evaluation of Blockchain Systems: A Systematic Survey

[3] H. MONIZ, The Istanbul BFT Consensus Algorithm

[4] A. Baliga, I. Subhod, P. Kamat, S. Chatterjee, Performance Evaluation of the Quorum Blockchain Platform

[5] J. Polge, J. Robert, Y. Le Traon, Permissioned blockchain frameworks in the industry: A comparison

[6] M. Mazzoni, A. Corradi, V. Di Nicola, Performance evaluation of permissioned blockchains for financial applications: The ConsenSys Quorum case study