A Service Level Agreement Verification System using Blockchains

João Paulo de Brito Gonçalves, Roberta Lima Gomes, Rodolfo da Silva Villaca
Federal University of Espirito Santo (Ufes)
Vitoria–ES, Brazil
jpaulo@ifes.edu.br, rgomes@inf.ufes.br, rodolfo.villaca@ufes.br

Esteban Municio, Johann M. Marquez-Barja
IDLab – imec – University of Antwerp
Antwerp, Belgium
esteban.municio@uantwerpen.be
johann.marquez-barja@uantwerpen.be

***Abstract*—Service Level Agreements (SLAs) are used to establish a contract, an agreement between two parties, which can be between two operators, or between a customer and an operator. The SLAs methods are a key aspect between consumers and providers, which can continuously monitor the Quality of Service (QoS) attributes and enforce its reliability, but the SLA also needs an entity to manage it. Smarts contracts are programs that are executed in a blockchain and ensure integrity and reliability to data stored in the distributed structure. This work proposes a solution using smarts contracts and blockchains in order to simplify the process of SLA validation.**

***Index Terms*—Blockchain, Service Level Agreement, Smart Contracts, Network Monitoring**

I. Introduction

The Brazilian National Education and Research Network (RNP) operates a backbone service to serve the academic and research communities by providing access to the Internet through its regional Points of Presence (PoPs) which make up its national backbone. There are PoPs deployed along the all 27 brazilian federation units, and the PoP-ES is located in Vitória, Espírito Santo, Brazil.

The PoP-ES has the basic function of maintaining, operating and coordinating actions on the academic Internet, serving as point of access for users to the Ipê Network backbone [1]. In addition, PoP-ES offers a variety of services related to the maintenance, management, planning and development of advanced networks.

PoP-ES arbitrates Service Level Agreements (SLAs) between network providers (operators) and the academic institutions connected to the Ipê Network backbone. PoP-ES is the manager of each SLA and has the responsibility of verify if they are being respected. A SLA can be defined qualitatively for each customer based on their required resources, reliability and availability. A traditional SLA has the following performance metrics: bandwidth, availability, latency, packet loss and error rate [2]. There are also metrics related to response time to serve tickets opened in the a help desk application (call center) and minimum time before requesting maintenance windows [3].

PoP-ES is responsible to verify, periodically, if the SLA of each customer is being fulfilled per agreement. In the negative case, a report must be generated showing which metric was violated and when. In this case, every unavailability event is automatically generated and a ticket is opened in the help desk system of RNP. An alert is generated to the network provider, the customer (academic institution), and to the PoP-ES team. Now the problem must be addressed by those partners responsible to handle such problem. Once the link becomes available again, an availability event is generated and the ticket is closed. Every month, each ticket must be manually evaluated by the PoP-ES team and each problem related to the ticket must be classified regarding the responsibility: the network provider, the customer itself, or the PoP-ES. Only unavailability events generated by a problem related to the network provider can count on the SLA availability metric, as an example.

POP-ES maintains an availability level agreed via SLA with the institutions served of 99.6 and allows maintenance windows created by link operators. Since there are three interacting entities (academic institutions, network providers and POP-ES) in the SLA-based agreed service, a reliable record of downtime is required to resolve disputes and verify that agreed levels via SLA are being respected. In the face of this problem, the following research question is raised: ”how to ensure that the SLA is being fulfilled in a secure, certified and dynamic way?”

Blockchains and Smart Contracts can address the issues about certified the SLA in a secure, certified and dynamic way by providing an immutable data storage and a distributed consensus between members without the need of an external validation. In this sense, smart contracts executing in a blockchain can be used to express, in a tamper-proof manner, SLAs containing quantitative terms, such as QoS metrics [4], excluding the need for an entity to arbitrate the agreement. Besides that, the concept of Blockchain as a Service (BaaS) is characterized by the ability to create and manage the access to a blockchain network using a cloud computing environment.

This work proposes a solution empowered by smart contracts in order to simplify the process of SLA validation using a cloud environment.

This paper is organized as follows: Section II provides a theoretical background on the topic and III the related work. In Section IV, the applications developed are presented. Section

V shows the tests performed and the section VI presents the final conclusions and some future work proposals.

II. Background

A. Blockchain

Despite his initial financial application, blockchain technology [5] has grown to a multiplicity of different applications where there is the constant necessity of unchanging and distributed recording of operations performed on a system such as distributed computing [6], Internet of Things [7], file storage [8], prediction [9], among many others. The blockchain is a distributed ledger, where each participant has a copy of the database with all the information already made and there is a consensus among the participants about validated transactions.

In the blockchain, each block is a set of transactions chained through hash addresses. Each block has recorded, among other information, the time it was validated, its hash and also the hash of the previous block, so that once the block is generated, it can not be tampered under the penalty of the stored hash not match to the hash of the modified block, thus evidencing the attempted fraud.

In public blockchains, i.e. where access is not controlled by a central authority, consensus and validation of transactions and blocks is preceded by a step called Proof of Work (PoW), where a cryptic challenge is proposed which, once solved, is propagated over the network. Only after the transaction has been validated, the pending transaction is actually performed. Invalidated transactions arriving at the network are stored in a transaction pool, where they await for validation.

PoW is used to discourage malicious users from creating fraudulent transactions on the network, but there is criticism regarding performance loss caused by their use in block creation. In the past few years, others validation strategies have been proposed, as Proof of Stake(PoS) [7] and Proof of Authority (PoA) [7], both based more in users' currency amount and reputation and less in computation power.

B. Ethereum

Several blockchain technologies have emerged in recent years and among the most popular are those based on the Ethereum platform. Ethereum is a platform for executing blockchain-based applications that are modeled as smart contracts [10] and has its own cryptocurrency, ether.

Another important Ethereum concept is gas. Gas is a way of decoupling the cost of transactions in the Ethereum from the floating exchange rate of the ether cryptocurrency, establishing a cost for each fluctuating computational job in the financial market. Gas is also a mechanism to prevent a smart contract with infinite loops from running indefinitely on the blockchain. Once the maximum amount of gas allocated to contract expires, the contract finish its execution.

On the Ethereum platform, smart contacts run on Ethereum Virtual Machine (EVM). It is completely isolated and the code that runs on it has no access to any external resources such as the network or the computer file system [11]. Ethereum uses Proof of Work as its current consensus mechanism.

The Ethereum platform is a very flexible alternative to the development of dApps (Decentralized Applications). A dApp is a decentralized application that uses a smart contract in the blockchain as backend and a web interface as frontend, allowing users to insert and receive data from the blockchain in a friendly way.

C. InterPlanetary File System

The InterPlanetary File System (IPFS) [12] is a peer-to-peer distributed file system that aims to connect all computing nodes in the same file system. In some ways, this is similar to the original aims of the World Wide Web, but IPFS is actually more similar to a single bit torrent exchanging objects. Due to its decentralized nature, IPFS is used to store files that would be too expensive to write in the blockchain.

IPFS is considered a promising solution for saving data for decentralized applications. Without IPFS, the blockchain would be reduced to any other regular storing mechanism with many limitations. In IPFS, the file storage address is the hash, providing a unique identification that is tightly linked to the file itself.

III. Related Work

Scheid et. al. [4] presented is presented the design and implementation of a blockchain-based smart contract that simplifies and automates the compensation process in case of an SLA violation. The prototype was deployed in Ganache tool, that simulates a Ethereum-based blockchain to simplify dApps deploy and tests.

Uriarte et. al [13] present a formal language(SLAC) to describe their SLAs for cloud providers. In this paper, they inform that are currently analysing the use of SLAC in the context of blockchain and smart contracts, but don't provide any implementation details. In [14] the same authors describe a prototype for the support of the SLA management.

Pascale et. al [15] proposes automatize Small-Cell-as-a-Service (SCaaS) agreements between the small-cell owners and network operators using smart contracts. The smart contract code is available, but the paper does not describe any performance test or deployment process.

In this paper we propose a solution based on the real operation of PoP-ES, as well as the deployment of a proof-of-concept on a cloud structure and on a real and public Ethereum-like blockchain to keep it on-line 24 hours per day, besides the integration of blockchain and IPFS technologies.

IV. System Design

In order to validate our proposal, we created two dApps to store the periods of unavailability in the service provided between PoP-ES and institutions served and to verify the SLA conformity.

We developed the smart contract in Solidity language. This smart contract is connected to a web interface, composing a dApp. We created the dApp interface using React framework, a JavaScript library for building user interfaces.

We deployed the smart contract in the Ropsten network, a test network based in Ethereum blockchain. We choosed this network among all the Ethereum-like test networks because it is the only one that implements the PoW, being closer to the real Ethereum's current network but with no monetary expenses to execute transactions. In the Ropsten, are used faucets, ethers without real value that can be request and used.

To visualize the transactions submitted to the blockchain, we use Etherscan [16], a block explorer and analytics platform for Ethereum blockchain. In its dashboard, wen can see all the transactions details: status, block, timestamp, gas used, gas price as well as the transaction content itself.

As showed in figure 1, the information is inserted in the dApp and forwarded to the smart contract using the web3.js library, that connects web applications to the blockchain. We use Metamask plugin in the browser to allow our interaction with the blockchain without the need to run a full Ethereum node. In Metamask, the Ethereum's account used to perform transactions is configured. The users' accounts authorized to use the applications are hard coded in the source code, so only these users will be able to perform operations, even if other accounts are used in Metamask.

To connect to IPFS, it was used Infura, a scalable back-end infrastructure for building dApps, to connect to both blockchain network and decentralised storage.

A. System Dynamic

There are two dApps that use the same smart contract address as we show in figure 1. One dApp will be used by the PoP operator to register downtime periods. The other dApp will be used by the link operator to insert maintenance windows that will be disregarded from the total unavailability count and have no influence in the availability level accounting.

There is an option in the PoP operator interface to create a report of the unavailability period registered that will be saved in IPFS servers. The file hash which is also the IPFS address will be stored in the blockchain, and is showed in the graphical interface for further compliance verification.

For each downtime period inserted in the PoP operator interface, the downtime time accounting is stored in the blockchain for further verification. As result, it is possible to verify the actual availability level and the compliance or not with the SLA in the dApp.

The algorithm for calculating service availability to the client is described in Algorithm 1. There are two arrays, one that receive the downtime moments and the other receives the uptime moments. After that, we compare if the uptime moment is smaller that the downtime moment, if so, it is a inconsistence. If not, the *counter* variable its incremented with this new unavailability period. After that, the *Availability* is calculated using the total of minutes in a month subtracted the *counter*, multiplied by 100 and divided for the total of minutes in a month. After every availability level update, is possible to see the compliance with the SLA or not in the prototype.

Algorithm 1: Downtime Accounting

Input : Downtime periods
Output: Availability level
Result: Calculates the Availability

```
downtime[i] ← down.timestamp;
uptime[i] ← up.timestamp;
if (uptime[i] ≥ downtime[i]) then
    period ← uptime[i] - downtime[i];
    counter ← counter + period;
    i ← i + 1;
else
    return InvalidInput;
end
Availability ← ((43200 - counter) * 100)/43200;
return Availability;
```

In the link operator interface, the maintenance windows are inserted 48 hours in advance to respect the SLA clauses. If the operator tries to insert a maintenance window less than 48 hours earlier, it can not be allowed by the prototype.

The period inserted is decremented of the unavailability accountability in the smart contract, after checking if the maintenance window intersects with one of the registered downtime events.

The system dynamic is presented in Figure 1.

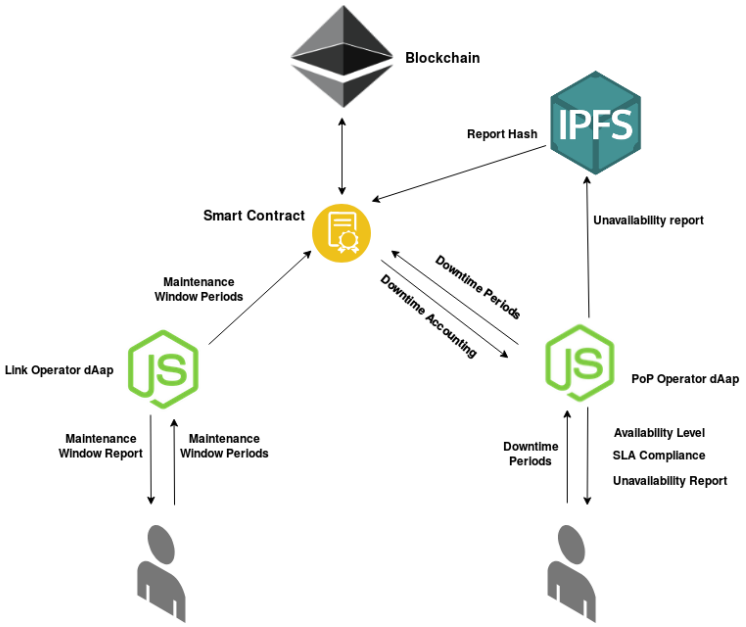

Fig. 1. System Dynamics

V. Deployment and Tests

The proof of concept was deployed on a virtual machine in a Pike version OpenStack cloud, distributed on four servers with the following configuration:

• Controller node with 32GB RAM, Intel® Xeon® processor E3-1240 3.70GHz, Ubuntu 16.04 Operating System;

• 2 computing nodes with 32GB RAM, Intel® Xeon® processor 3.70GHz E3-1240, Ubuntu 16.04 Operating System;
• 1 computing node with 128GB RAM, Intel® Xeon® processor E5-2650 2.20GHz, Ubuntu 16.04 Operating System.

Ninety transactions were made in the two dApps and the system behavior was analysed, as well the interaction between the two systems.

The deployment and interactions with the smart contract that alters its state require a payment of a certain gas fee. Using Etherscan we can see a transaction and block history, providing information such as the amount of gas consumed per transaction and per block. The transactions were executed at a fixed gas price of around 0.000000001 Ether per unit and transaction fees that oscillate between 0.0000841 to 0.00042832 ether. It is noticeable that the creation of the smart contract is the most expensive function, since it is the operation that writes the most amount of data on the blockchain. This emerges from the way transaction costs are composed in Ethereum, namely a fixed and a variable part. The EVM demands a fixed cost of 32,000 gas for a contract creation in addition to the 21,000 gas that every transaction costs. The remainder is the variable part which depends on the size of the contract code. Each byte of code consumes 200 gas units, the more code a contract has the more expensive its creation is [17].

The other operations consist of timestamps writes and numerical operations, these are composed of the fixed gas fee of 21,000 plus a certain fee for each operation.

In a scenario without IPFS integration, the reports storage would be the most costly operation, but after the integration, only the file addresses are written via smart contract.

It was also noticed that although the transactions are inserted into blocks after the Proof of Work, the time to update contract variables did not exceed 15 seconds, which makes the response time acceptable for applications as the one we propose in this work, where the recording of downtime and maintenance windows is made sporadically.

VI. Conclusions and Future Work

This work proposes a solution empowered by smart contracts in order to simplify the process of SLA validation. As proof of concept we developed and deployed two decentralized applications in a cloud infrastructure in order to be used by PoP-ES and link provider operators.

In the proof of concept is possible to insert downtime periods and maintenance windows that are used to calculate the total availability and his compliance with the SLA. It is possible to verify that the cost in ethers to execute the basic operations of the smart contract was not too excessive and the response time of validated transactions even in a public blockchain is acceptable.

As future work, we foresee the deploy of the proof of concept on the official Ethereum blockchain, what would enable the inclusion of monetary refund for clients that suffered SLA violations, implemented using the cryptocurrency itself.

Another suggestion is the use of service level agreement verification in other use cases which involve contracts for the use of network resources between clients and providers as cloud computing and network slicing. About the latest, an implementation is already under development.

Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 and in part by the European Union's Horizon 2020 Research and innovation program, under grant agreement No. 826284 (ProTego). Authors would also like to thank FAPES, CNPq, PoP-ES/RNP and University of Antwerp and IMEC for supporting this research.

References

[1] L. Sampaio *et al.*, "Implementing and deploying network monitoring service oriented architectures: Brazilian national education and research network measurement experiments," in *2007 Latin American Network Operations and Management Symposium*. IEEE, 2007, pp. 28–37. [Online]. Available: https://doi.org/10.1109/LANOMS.2007.4362457

[2] Y. Nugraha and A. Martin, "Understanding trustworthy service level agreements: Open problems and existing solutions," 2017. [Online]. Available: https://hal.inria.fr/hal-01684231

[3] J. R. Amazonas, A. Akbari-Moghanjoughi, G. Santos-Boada, and J. S. Pareta, "Service level agreements for communication networks: A survey," *INFOCOMP Journal of Computer Science*, vol. 18, no. 1, pp. 32–56, 2019. [Online]. Available: http://infocomp.dcc.ufla.br/index.php/INFOCOMP/article/view/584

[4] E. J. Scheid, B. B. Rodrigues, L. Z. Granville, and B. Stiller, "Enabling dynamic sla compensation using blockchain-based smart contracts," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 53–61. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8717859

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." [Online]. Available: http://bitcoin.org/bitcoin.pdf

[6] J. B. Pollack and H. Lipson, "The golem project: Evolving hardware bodies and brains," in *Proceedings of the 2nd NASA/DoD Workshop on Evolvable Hardware*. IEEE, 2000, pp. 37–42. [Online]. Available: https://doi.org/10.1109/EH.2000.869340

[7] F. Saleh, "Blockchain without waste: Proof-of-stake," SSRN, Tech. Rep., 5 2019. [Online]. Available: https://dx.doi.org/10.2139/ssrn.3183935

[8] S. Wilkinson, T. Boshevski, J. Brandoff, and V. Buterin, "Storj a peer-to-peer cloud storage network," 2014. [Online]. Available: https://storj.io/storj2014.pdf

[9] J. Peterson and J. Krug, "Augur: a decentralized, open-source platform for prediction markets," *arXiv preprint arXiv:1501.01042*, 2015.

[10] N. Szabo, "Formalizing and securing relationships on public networks," *First Monday*, vol. 2, no. 9, 1997. [Online]. Available: https://doi.org/10.5210/fm.v2i9.548

[11] I. Bashir, *Mastering Blockchain*. Packt Publishing Ltd, 2017.

[12] Q. Zheng, Y. Li, P. Chen, and X. Dong, "An innovative ipfs-based storage model for blockchain," in *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. IEEE, 2018, pp. 704–708. [Online]. Available: https://doi.org/10.1109/WI.2018.000-8

[13] R. B. Uriarte, R. D. Nicola, V. Scoca, and F. Tiezzi, "Defining and guaranteeing dynamic service levels in clouds," *Future Generation Computer Systems*, vol. 99, pp. 17–40, 10 2019. [Online]. Available: https://doi.org/10.1016/j.future.2019.04.001

[14] R. B. Uriarte, R. de Nicola, and K. Kritikos, "Towards distributed sla management with smart contracts and blockchain," in *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Dec 2018, pp. 266–271. [Online]. Available: https://doi.org/10.1109/CloudCom2018.2018.00059

[15] E. D. Pascale, J. McMenamy, I. Macaluso, and L. Doyle, "Smart contract slas for dense small-cell-as-a-service," *CoRR*, vol. abs/1703.04502, 2017. [Online]. Available: http://arxiv.org/abs/1703.04502

[16] E. Team, "Etherscan: The ethereum block explorer," 2017. [Online]. Available: https://etherscan.io/

[17] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, 2014. [Online]. Available: http://gavwood.com/paper.pdf