A Quality of Service Compliance System Empowered by Smart Contracts and Oracles

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 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 in contracts between two parts, which can be, as an example, between service providers or between customers and service providers. SLA clauses represent key aspects in the relationship between them. In order to enforce service reliability and SLA compliance, a continuous monitoring of Quality of Service metrics is required. Since policy breach is usually subject to expensive penalties, using external entities acting as trustful references is a common practice in order to avoid frequent legal disputes, external entities are usually required to verify it. However, coordination between different actors to agree in one transaction is difficult, because it can open up possibilities for multiple fraud attempts. We argue that this can be done more efficiently using Smart Contracts, programs that are executed in a blockchain. Such data is stored in a distributed structure without the need of an external entity to ensure data integrity and reliability. Smart contracts can also makes easy the charging or possibly compensating SLA violations. In this paper we propose a solution empowered by smart contracts in order to simplify, and to automate the process of SLA validation, i.e, verify if the contract is being respected. We perform some experiments and evaluate a proof of concept using data from a real network infrastructure in Brazil.**

***Index Terms*—Blockchain, Service Level Agreement, Smart Contracts, Quality of Service.**

I. Introduction

Service Level Agreements (SLAs) are formal contracts between custumers and service providers that outline the parameters of some IT services to be supplied [1]. Besides the nature of the service itself and its expected performance level, a SLA also specifies the procedures for monitoring and reporting problems, time limits for problem resolution and the consequences for both clients and service providers when clauses are violated.

In the network domain, a SLA can be established between costumers and network operator, or between two operators. Network SLAs are typically conceived in terms of network performance between exchange points where the service is provided. They can cover physical network aspects, such as bandwidth, availability, latency, jitter, packet loss or error rate [1]. However, there are SLAs that relate not only to network performance, but also to different types of IT outsourcing in clouds (such as computing units, data storage, hosting, etc).

Monitoring and reporting the Quality of Service (QoS) allow clients and service providers to check how well is the service complying with the SLA. If the specified performance parameters are violated during the course of operation of the system, the SLA constitutes a legal foundation that eventually serves as a means of judgment in case of dispute between service providers and costumers.

The root of the problem lies in the way transactions are controlled between partners. Since both parties maintain records with a centralized, isolated approach, there is little transparency and this may lead to lack of trust. Since traditional centralized databases are generally not encrypted, there is always a risk of an operator's sensitive data about a partner's customers being lost, exposed, or manipulated. In this way, SLA disputes are always problematic [2].

As different parties are involved in a SLA, to guarantee that the agreed service has been provided, a reliable record of the monitored parameters is required. Based on this record, SLA compliance can be dynamically verified and the necessary actions can be taken, but this is not trivial, and different works try to address this problem [3]. 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?"

A hypothesis is that blockchains and smart contracts can help to implement such secure record by providing an immutable data storage and a distributed consensus among members without the need of a trusted third party to validate it. As smart contracts are programs that are directly executed in a blockchain, their integrity and reliability are ensured, and they can be used to express, in a tamper-proof manner, SLAs containing quantitative terms, such as QoS metrics, without the need for any third party to arbitrate the agreement. When both parties sign the contract, they agree on the monitoring system and the algorithms to verify the SLA compliance.

In this context, the main goal of this paper is to propose a solution empowered by smart contracts aiming to simplify and automate the process of tracking SLA compliance levels. A second goal is to show the interaction of this system with a decentralized file system (IPFS - Interplanetary File

System) that will store the generated reports in a distributed and transparent way. A third goal is to show how the presented proposal works in a Brazilian academic network (Point of Presence-ES/RNP) that was built as a proof of concept to validate the proposal. In our deployment, SLAs are verified in less than 15 seconds in most of the cases.

II. Background

A. The PoP-ES

The Brazilian National Education and Research Network operates a backbone service to serve the academic and research communities by providing access to the Internet through its regional Point of Presence(PoP). This network is used to basic access to the Internet by workstations, but also to devices with sensors that access the Internet and exchange information. In this Internet of Things case, the QoS requirements are even more necessary due to the large volume of information exchanged. Among the 27 Brazilian PoPs, which make up the national backbone (Ipê Network), there is the PoP Espirito Santo (PoP-ES), located in Vitória, Espirito Santo, Brazil.

The PoP-ES is in charge of maintaining, operating and coordinating actions on the academic Internet, serving as an access point for users to the Ipê Network backbone [4]. In addition, PoP-ES offers a variety of services related to the maintenance, management, planning and development of advanced networks.

PoP-ES arbitrates SLAs between network providers (link operators) and the academic institutions connected to the Ipê Network backbone. PoP-ES is the manager of each SLA and has the responsibility of verifying their compliance levels. Each SLA is defined qualitatively based on customer's required reliability and availability. Besides, the SLA specifies some metrics regarding maximum response time on open tickets (tickets are opened via a help desk system) and minimum time before requesting maintenance windows.

PoP-ES is responsible to periodically verify real-world compliance with the metrics established in the SLA of each customer. In the negative case, a report must be generated showing which metrics were violated and when. There are several network monitoring tools that monitors the network status and health, one of these tools is Viaipe(https://viaipe.rnp.br/), used in the Ipê Network.

Once using Viaipe is detected that a link is down, an unavailability event is registered in the help desk system. An alert is then generated to the network provider, the customer (academic institution), and the PoP-ES team in order to have the technical issues fixed. Once the link becomes available again, an availability event is registered in the help desk system too. Every month, each event must be manually evaluated by the PoP-ES team and each problem related to the event 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 of 99.6% agreed via SLAs with the served institutions (the clients). Also, according to SLAs, maintenance windows can be created by network providers. As there are three interacting entities (academic institutions, network providers and PoP-ES) to provide the agreed service, a reliable record of downtime is required to resolve disputes and verify that agreed levels are being met.

B. Blockchain

Despite its initial financial original application [5], blockchain technology has grown to a multiplicity of different applications, where there is the constant need for unchanging and distributed recording of operations performed on a system. The blockchain is a distributed ledger, where each participant has a copy of the database with all the validated information. Besides, a consensus protocol is implemented among the participants in order to allow them to agree about the global state of the blockchain.

In a blockchain, each block is a set of transactions chained through hash addresses. Each block includes, among other information, a timestamp, its hash, and the hash of the previous block, so that once the block is created, it cannot be tampered under the penalty of the stored hash not matching 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, validation of transactions and blocks is often based on the Proof of Work (PoW) consensus protocol. In PoW, a cryptic challenge is proposed in order to create a valid block, once solved, the block is propagated over the network. Only after a transaction has been validated (included in a valid block), it is actually performed, which might change the blockchain state.

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, other validation strategies has been proposed, as Proof of Stake (PoS) [6] and Proof of Authority (PoA) [6], both based more in age and reputation and less in computation power. At this point, it is important to differ between two basic types of blockchains: public and permissioned.

In public blockchains, any computer can join the network and have full access to it. Because of this anonymous character of computers, measures to mitigate attacks must be adopted which results in performance degradation, as the Proof of Work by example. Bitcoin and Ethereum [7] are examples of public blockchains. Permissioned blockchains are mainly used in corporate environments. This means that a user must have a certain level of access to interact on this network, read transactions, and participate in the consensus process. Hyperledger Fabric [8] is an example of a permissioned blockchain.

In addition to the applications already mentioned, blockchain technology can be used in a wide variety of fields,

such as distributed computing [9], Internet of Things [10], file storage [11], prediction [12], among many others.

C. Ethereum and Smart Contracts

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

Smart contracts capture and translate traditional legal contract clauses into a series of computational rules which are executed automatically and, once validated, don't require additional legal instruments [13].

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 a contract expires, the contract finish its execution.

On the Ethereum platform, applications run on the Ethereum Virtual Machine (EVM), which executes smart contract instructions, allowing you to enter and query stored data. 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 [10]. Ethereum uses Proof of Work as its current consensus mechanism, but it is in a transition phase towards Ethereum 2.0 – this specification includes a switch to proof of stake as well as Sharding (each node having only a part of the data on the blockchain, and not all the information). These two main changes should enable the processing of up to 10000 transactions per second [14].

In Ethereum, the global state is made up of objects called accounts. Each account has an address in hexadecimal format (20 Bytes), and a transition state that are related to transfers and information between accounts. In the case of external accounts, there is no smart contract associated. These accounts can send messages creating and signing transactions. In the case of a smart contract account, whenever a message is received, the contract code is activated, allowing it to read or write to the blockchain internal storage, generating other messages or even a new contract. That is, while external accounts play an active role in the blockchain, contract accounts - and their codes - play a passive role.

The Ethereum platform is currently the largest general purpose public blockchain exponent on the Internet. It is a very flexible alternative to the development of dApps (Decentralized Applications) as it provides a complete programming language, different from the Bitcoin platform, which has a very limited scripting language and is used just to support basic and necessary network operations [10]. 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.

D. Oracles

Often smart contracts need information that is processed outside their computational logic and the availability of this information is crucial to the potential of smart contracts [15]. However, this is challenging since smart contracts can only access and write information that is stored on the blockchain, which is an enclosed network without direct interfaces to the real world. Oracles bridge the gap between the blockchain and the real-world by feeding data from outside the blockchain to smart contracts. they are usually application's APIs that produce data that can be consumed by smart contracts. They are used to report events and data after the smart contract has been programmed to allow the smart contract to react to future information. As is the case with regular applications, the usefulness largely depends on the available data and oracles enable smart contracts to query data similar to an API.

E. InterPlanetary File System

The InterPlanetary File System (IPFS) [16] is a peer-to-peer distributed file system that seeks 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. [17] presented the design and implementation of a smart contract that simplifies and automates the compensation process in SLA violations. The prototype was deployed in Ganache tool, that simulates an Ethereum-based blockchain to simplify dApps deployment and tests, but there is no deployment in a real environment.

Uriarte et al. [18] present a formal language (SLAC) to describe SLAs for cloud providers. SLAC is used in the context of blockchain and smart contracts. In [19], the same authors describe a prototype to support SLA management. They suggest an architecture with two different networks: sidechain and blockchain. The former is used for heavy computations such as discovery and negotiation of SLAs, the latter is used for smart contract execution, but without deployment in a real environment.

The work presented by Pascale et al. [20] proposes a smart contract to automate Small-Cell-as-a-Service (SCaaS) agreements between the small-cell owners and network operators. The smart contract code and implementation is available, but there are no evaluations or real deployments of the proposal.

In [21] a Blockchain Network Slice Broker is proposed to reduce the service creation time for dynamically slice acquisition and for verifiable charging and billing in service level

agreements. The proposal's implementation and performance analysis are not presented.

Zanzi et al. [22] proposed NSBchain, a novel Network Slicing Brokering (NSB) solution, which leverages the Blockchain technology to address the new business models needs beyond traditional network sharing agreements. They implemented NSBchain on top of Hyperledger Fabric and its benchmarking tool, namely Hyperledger Caliper, is used to evaluate the blockchain performance in network slicing scenarios, but the deployment is done on a private and local blockchain.

Also, as in the previous reference, in [23] is proposed a 5G Network Slice Brokering using Hyperledger Fabric and using Hyperledger Caliper as benchmark tool. The idea is to use a distributed process to replace the conventional centralized approach to slice brokering, where a single authority does not control the entire conduct of the market, but also without a deployment on a public blockchain.

Finally, Zhou et al. [24] introduces the concept of witness, based on the John Nash's Equilibrium Principle [25], to create a mechanism using Smart Contracts to report violations of compliance SLAs between providers and consumers of cloud computing service. This mechanism would dispense with the use of Oracles, commonly used to obtain reliable data external to the Blockchain. Although promising, this strategy was not used in the present work as is only a proposal, and not a standard in the Blockchain and Smart contract market and the scope of the problem was too small to justify the use of the technique.

In summary, the main contribution of this paper is that, unless previous works, it is the first one in testing a full-powered smart contract based system to verify SLAs in a real production environment. This proposal advances the state of the art in SLA management by building a real application that runs on a cloud computing infrastructure and uses a public Ethereum-like blockchain to keep it on-line 24 hours per day, besides the integration of blockchain and IPFS technologies to store the generated reports. It is an unprecedented large scale deployment, running for a long period of time, with a large amount of data gathered. Beside this, the proof of concept was implemented using data from a real use case scenario (from PoP-ES) inserted into the system, acting like an Oracle.

IV. Design and Implementation

Due to Ethereum's flexibility and extensive developer community, it has been chosen to develop a proof of concept of the proposed SLA verification system based on smart contracts.

In order to validate the presented hypothesis, two dApps were created to store unavailability events, registered and validated by the PoP-ES about the service provided by a network provider to one of the academic institutions connected to the Ipê Network. The goal of the dApps is to verify the SLA compliance regarding availability of a link ($\geq$ 99.6%).

A smart contract was developed in Solidity language, which runs on the Ethereum Virtual Machine (EVM). This smart contract is connected to a web interface, composing a dApp. The dApp interface was created using React framework, a JavaScript library for building user interfaces.

The smart contract was deployed in the Ropsten network, a public Ethereum test network. This test network was chosen among all others because it is the only one that implements the PoW verification approach, being closer to the real Ethereum's current network, but with no real monetary expenses to execute transactions. In Ropsten, faucets are used to create ethers with no real value.

To visualize the transactions submitted to the blockchain, Etherscan [26] was used. Etherscan is a block explorer and analytics platform for Ethereum blockchain. In its dashboard, it is possible to see all the transactions details: status, block number, timestamp, gas used, gas price as well as the transaction content itself.

The information is inserted in the dApp and forwarded to the smart contract using the web3.js library, that provides a mechanism to connect web applications to the blockchain. The Metamask plugin was also used in the web browser. It handles users' Ethereum accounts, and enables the interaction with the blockchain without the need to locally run a full Ethereum node.

Streamr is a framework that enables live storing and sharing of data streams [27]. Streamr chooses to build on top of an existing chain, rather than building a specific blockchain, thus it is blockchain agnostic and runs off-chain. Currently, only the Streamr Engine, which is in charge of user authentication and payment, runs on Ethereum.

The Streamr Engine [27] was used in the proof of concept to capture data produced by Viaipe API, process and send it to the blockchain in an automatic way. To receive the packet loss of each connection directly from Viaipe API, we created a microservice which consumes and acts upon real-time data, using a canvas in Streamr. This real-time data channel will act as a Oracle, inserting real-world data in the blockchain and the data security before the insertion is guaranteed by the TLS connection.

IPFS is the storage system selected to store the reports generated by the solution proposed in this paper. To connect to IPFS, we used Infura, a scalable back-end infrastructure for building dApps, to connect to both blockchain network and decentralized storage.

A. System Dynamic

Both dApps use the same smart contract address. One dApp will be used by PoP-ES to register unavailability events and their timestamps. The events are: down and up of each link. This dApp also presents an option allowing PoP-ES to create and save in the IPFS a report of the unavailability periods registered so far. The hash of the report, which is also the IPFS address, will be stored in the blockchain, and be available for further compliance verification.

The other dApp will be used by the network provider (link operator) to schedule maintenance windows. Maintenance

windows duration will then be excluded from the total unavailability calculation, having no influence on the availability level agreed in the SLA.

For each unavailability period (down and up events) inserted in the dApp, the unavailability (time between down and up events) 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 used by the smart contract for calculating service availability given the events recorded in the blockchain is described in Figure 1. When the costumer and provider agree with the Smart Contract, they also agree with this algorithm. In the smart contract implementation there are two arrays, one to store downtime events and the other to store up events. For each i-th up event, its timestamp is subtracted from the corresponding i-th downtime timestamp, and the *counter* variable is incremented with this unavailability period. After that, the *Availability* is calculated as showed in Figure 1, where the constant 2592000 is the total number of seconds in a month (30 days).

Algorithm 1: Downtime Accounting

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

```
1  downtime[i] ← down.timestamp;
2  uptime[i] ← up.timestamp;
3  if (uptime[i] ≥ downtime[i]) then
4  |   period ← uptime[i] - downtime[i];
5  |   counter ← counter + period;
6  |   i ← i + 1;
7  else
8  |   return InvalidInput;
9  end
10 Availability ← ((2592000 - counter) * 100)/2592000;
11 return Availability;
```

Fig. 1. Availability Algorithm

Nevertheless, as each new call to the smart contract updates the blockchain state, it is possible to see, in a dynamic way, the compliance of its respective SLA (if the availability is greater or equal to 99.6%, as an example).

Through the network provider interface, maintenance windows can be scheduled at least 48 hours in advance, in order to respect the SLA restrictions. If the operator tries to insert a maintenance window less than 48 hours in advance, it will not be validated by the smart contract. Validated maintenance windows are subtracted from the unavailability counter in the smart contract, after checking if it intersects with at least one of the registered downtime events. The systems dynamics is illustrated in Figure 2.

Using Streamr Engine, the data incoming from Viaipe monitoring tool are collected by the *HTTPRequest* component, which through a pulling mechanism (every 10 seconds) checks if there are new data available. This component receives as input: i) the API endpoint; ii) the HTTP method to be used; iii) the data type to be received; and finally, iv) the time interval to access the endpoint.

The collected data is filtered to obtain the packet loss of each customer and this information is sent to the smart contract running on the Ropsten blockchain by using the *EthereumCall* component. This component receives as input: i) the smart contract address; ii) the Ethernet account; and, finally iii) the function to be executed in the smart contract.

The packet loss is also a metric that influences the quality of the service provided, so it must also be part of the specification of the service level agreement. In addition, when an unavailability event occurs, packet loss will automatically occur, so these metrics are interrelated.

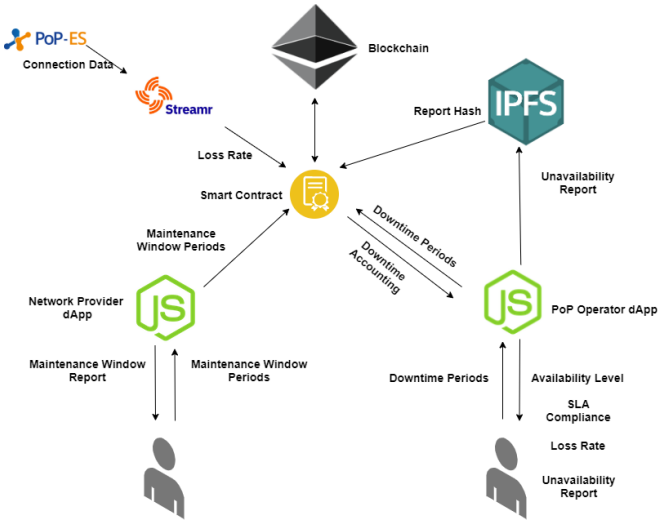

Fig. 2. System Dynamics

V. Deployment and Evaluation

Our proof of concept is running on a virtual machine created in an OpenStack [28] cloud (Pike version), distributed on four servers with the following configuration:

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

The dApps were deployed and installed on a virtual machine (8 vCPUs and 16 GB memory) in a Ubuntu 18.04 operating system hosted in the OpenStack cloud. We monitored

PoP-ES in about 30 days and got 90 unavailability events, that were created during the dApps utilization.

As the analysis was focused on blockchain behavior, the results were focused on measuring blockchain-related metrics. The main differences of blockchains compared to traditional programs and databases, are the occurring monetary cost of transactions as well as the high latency of processing due to the creation time of blocks.

The performance of a blockchain application/network can be measured using the following metrics:

- Transaction Throughput: measured in transactions per second (TPS) and represents the number of transactions that are processed by the blockchain and written on the ledger in a given second.
- Transaction Latency: The amount of time taken from the moment when a transaction is submitted until the moment when it is confirmed and available on the blockchain. This includes the propagation time and the processing time due to the consensus/ordering mechanism.
- Operation Costs: The amount of computing resources and monetary costs consumed by the blockchain throughout the operating time, including the processing power, memory, storage, I/O, network and transaction fees. This metric is of great importance as it could determine the cost efficiency of a blockchain application. Furthermore, besides the capital expenditure for providing the computing capacity, blockchain networks could require huge amounts of energy to operate. Therefore, the computing intensity would also affect the operation costs of the blockchain.

Regarding the operation costs, the higher the gas price the more expensive the transaction costs but it also leads to faster validation of the transaction. It is essential to find a balanced point between the cost and the speed for the proposal. The deployment and interactions with the smart contract that alters its state required the payment of a certain gas fee. Using Etherscan it is possible to see a sample transaction and its corresponding 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 0.00008410 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 greater amount of data on the blockchain. This emerges from the way how transaction costs are composed in Ethereum, namely by a fixed and a variable part. The EVM demands a fixed cost of 32,000 gas for the creation of a smart contract in addition to the 21,000 gas for each transaction. 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 [7]. The other operations are timestamp writing, storing and operations. These are composed of a fixed gas fee of 21,000 plus a certain fee for each operation.

In a scenario without IPFS integration, the storage of the availability reports would be the most costly operation. But with the IPFS integrated to the system, only reports hash addresses are written via the smart contract.

Since every transaction interacting with the smart contract needs to be included into a block in order to be validated, this leads to a blockchain-dependent latency. In practice, participants in a blockchain network are geographically distributed. Such distribution introduces additional latency. In the Ethereum blockchain, the transaction latency was approximately 15 seconds to verify the registration of an unavailability event, which is in the Ethereum blockchain the average block time [7].

This small latency to update variables in the smart contract is acceptable for the applications proposed in this paper, where the recording of downtime and maintenance windows are made sporadically and none of these periods lasts less than 15 seconds, so there are low probability of having inconsistencies due to the validation delay.

Regarding the metric number of transactions per second, the application is subject to the maximum throughput of the ethereum blockchain, that is, around 15 transactions per second [7].

VI. Conclusions and Future Work

This paper presented a solution for the problem of validating SLAs in a secure, certified and dynamic way. The proposed solution leverages blockchain and smart contracts in order to simplify, and securely automate, the process of tracking SLA compliance levels. A proof of concept using two dApps was designed, implemented and deployed in a cloud infrastructure (connected to a real Ethereum test network) to be used by PoP-ES and network providers.

In the proof of concept, it was possible to insert up/down events and maintenance windows, which were used to calculate the total availability of the link and its compliance with the SLA. Besides that, a real-time data analysis platform was used to consume and send to the blockchain the data about loss in the costumers' connections collected by the Viaipe monitoring tool, acting as a Oracle. An Oracle receives claims about the state of the world and uploads it to the blockchain, connecting the blockchain to the real world by providing it with relevant information.

The proof of concept evaluation shows that the costs, expressed 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 suggest to extend the proposed solution and include other SLA parameters besides packet loss, such as latency, bandwidth and jitter. In another scenario, specific parameters of IoT devices communication, such as battery consumption and energy efficiency can also be inserted and verified in the proposed system. Another important step is to allow the smart contract to implement monetary compensation in the case of SLA violations. When it is the case, once

the proposed solution is deployed on the official Ethereum blockchain, cryptocurrencies (ethers) can be automatically transferred from network providers to PoP-ES (for example).

Another future work proposal is to use some distributed oracle mechanism, increasing the reliability, such as Chainlink [29].

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 IMEC - University of Antwerp and 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 IMEC for supporting this research.

REFERENCES

[1] D. C. Verma, "Service level agreements on ip networks," *Proceedings of the IEEE*, vol. 92, no. 9, pp. 1382–1388. [Online]. Available: https://doi.org/10.1109/JPROC.2004.832969

[2] S. Venkatasubramanian, "Itu and its dispute settlement mechanism," in *Dispute Settlement in the Area of Space Communication*. Nomos Verlagsgesellschaft mbH & Co. KG, 2015, pp. 21–32. [Online]. Available: https://doi.org/10.5771/9783845258584-21

[3] J. Ahmed, A. Johnsson, R. Yanggratoke, J. Ardelius, C. Flinta, and R. Stadler, "Predicting sla violations in real time using online machine learning," *arXiv preprint arXiv:1509.01386*, 2015.

[4] 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

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

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

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

[8] V. Dhillon, D. Metcalf, and M. Hooper, "The hyperledger project," in *Blockchain enabled applications*, pp. 139–149. [Online]. Available: https://doi.org/10.1007/978-1-4842-3081-7_10

[9] 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

[10] A. D. Dwivedi, G. Srivastava, S. Dhar, and R. Singh (2019), "A decentralized privacy-preserving healthcare blockchain for iot," *Sensors*, vol. 19, no. 2, p. 326. [Online]. Available: https://doi.org/10.3390/s19020326

[11] 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

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

[13] 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

[14] A. Skidanov and I. Polosukhin, "Nightshade: Near protocol sharding design," *URL: https://nearprotocol. com/downloads/Nightshade. pdf*, p. 39, 2019.

[15] Z. Hess, Y. Malahov, and J. Pettersson, "Æternity blockchain," *Online]. Available: https://aeternity. com/aeternity-blockchainwhitepaper. pdf*, 2017.

[16] 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

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

[18] 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

[19] 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

[20] 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

[21] J. Backman, S. Yrjölä, K. Valtanen, and O. Mämmelä, "Blockchain network slice broker in 5g: Slice leasing in factory of the future use case," in *2017 Internet of Things Business Models, Users, and Networks*. IEEE, 2017, pp. 1–8. [Online]. Available: https://doi.org/10.1109/CTTE.2017.8260929

[22] L. Zanzi, A. Albanese, V. Sciancalepore, and X. Costa-Pérez, "Ns-bchain: A secure blockchain framework for network slicing brokerage," *arXiv preprint arXiv:2003.07748*, 2020.

[23] N. Afraz and M. Ruffini (2020), "5g network slice brokering: A distributed blockchain-based market," in *2020 European Conference on Networks and Communications (EuCNC)*. IEEE, pp. 23–27. [Online]. Available: https://doi.org/10.1109/EuCNC48522.2020.9200915

[24] H. Zhou, X. Ouyang, Z. Ren, J. Su, C. de Laat, and Z. Zhao, "A blockchain based witness model for trustworthy cloud service level agreement enforcement," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1567–1575. [Online]. Available: https://doi.org/10.1109/INFOCOM.2019.8737580

[25] K. Binmore *et al.*, *Game theory: a very short introduction*, vol. 173. [Online]. Available: https://ssrn.com/abstract=1284255

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

[27] M. Spiekermann (2019), "Data marketplaces: Trends and monetisation of data goods," *Intereconomics*, vol. 54, no. 4, pp. 208–216. [Online]. Available: https://link.springer.com/article/10.1007/s10272-019-0826-z

[28] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "Openstack: toward an open-source solution for cloud computing," *International Journal of Computer Applications*, vol. 55, no. 3, pp. 38–42, 2012. [Online]. Available: https://doi.org/10.5120/8738-2991

[29] S. Ellis, A. Juels, and S. Nazarov, "Chainlink a decentralized oracle network," *Retrieved March*, vol. 11, p. 2018, 2017. [Online]. Available: https://link.smartcontract.com/whitepaper