Leveraging Smart Contracts for Automatic SLA Compensation – The Case of NFV Environments

Eder John Scheid and Burkhard Stiller

Communication Systems Group (CSG), Department of Informatics (IFI)
University of Zürich (UZH), Binzmühlestrasse 14, CH 8050 Zürich, Switzerland
{scheid,stiller}@ifi.uzh.ch

1 Introduction and Motivation

Service Level Agreements (SLA) describe the requirements that Service Providers (SP) must meet when delivering a specific service to clients [3]. For example, in cloud computing, an SLA can describe that a database server must have 99.99% of availability. Moreover, for the same database server, an SLA can state that the server throughput must be greater than or equal to 1Mbps; otherwise, the client receives 10% of its payment back. The interaction between a client and an SP when specifying SLAs is performed within Operational Support Systems (OSS) and Business Support Systems (BSS). However, even with the employment of such systems, some tasks, *e.g.,* SLA specification, and compensation, still require manual effort and interaction to be accomplished. Such manual interaction hinders service agility and is prone to errors.

If an SLA is violated (*e.g.,* throughput less than 1Mbps) during the contract duration, then the client is entitled to compensation (*e.g.,* 10% payment back). The compensation process is costly, bureaucratic, and involve dedicated personnel in case of a dispute [6].discussed Moreover, this process requires trust between the involved parties. On the one hand, the client must provide evidence (*i.e.,* data) that the SP is not delivering what was agreed, and thus have to allocate resources for monitoring the contracted service. On the other hand, the SP must also monitor its services to provide evidence that they are being properly provided. Therefore, reducing the number of third-parties in SLA compensation process, while providing trust in the data to involved parties, is important to reduce costs and minimize unnecessary resource allocation.

Blockchains and Smart Contracts (SC) remove trusted third parties by decentralizing and replicating the data throughout all participants in the network. A blockchain is an append-only ledger where the data can only be inserted, but cannot be removed or changed. A SC is an executable code that runs on a given blockchain to facilitate, execute, and enforce agreements between untrusted parties [2]. Moreover, both technologies (blockchains and SCs) rely on cryptography, and consensus mechanisms to secure the data against tampering.

Blockchain-based SCs may be an option for the management of SLAs by automating compensation-related tasks while providing trust. The employment of blockchain-based SCs addresses two crucial aspects: the guarantee of contract

enforcement, and the immutability of the data. In the SLA compensation context, the former aspect assures the subscriber that it will receive compensation in case of an SLA violation. In contrast, this aspect also assures the SP that the client will pay the subscription fee. Moreover, the latter aspect assures to both parties, in the case of a trusted monitoring solution, that the monitored data cannot be altered after being appended to the blockchain. Thus, both aspects can aid to simplify the compensation in the case of an SLA violation.

2 Problem Description

Currently, the process of managing SLAs is cumbersome and involves different stakeholders, such as system administrators, incident response teams, managers, and end-users. The time that it takes between SLA violation detection and payment of monetary compensations impacts directly on the operational expenses of the involved organizations.

Taking the Amazon Compute SLA [6] as an example, the compensation process (*i.e.,* requesting credits) for the violation of the uptime SLA involves: (*i*) opening a formal case in the AWS Support Center, (*ii*) submitting a claim with "SLA Credit Request" in the subject line, (*iii*) informing dates and times of each unavailability incident, and the affected services (instances or volumes), and (*iv*) sending request logs (replacing or removing sensitive information) that document the errors and that corroborate the claimed outage. If the request is confirmed by the responsible team, then the service credit will be paid (within one billing cycle following the month in which the request was confirmed). If one fails to provide any necessary information or the team does not acknowledge the claim, then the credit will not be paid. This process is inefficient and prone to errors because of the lack of trust between both parties, the manual interaction required by the subscriber, and its complexity.

2.1 Research Questions

The present work aims to explore the use of blockchain-based SCs to automate SLA compensation process, which currently is manually performed and cumbersome. Within this context, three major aspects are to be investigated during the development of the Ph.D.: (*i*) feasibility to automate the process, (*ii*) complexity of the process, and (*iii*) integrity of the data involved in the process. Thus, it is expected to answer the following research questions over the Ph.D course:

(*i*) How long does it take to detect an SLA violation and pay the defined compensation to the subscriber using SCs?
(*ii*) How complex, regarding human-computer interaction, is the proposed approach compared to existing solutions?
(*iii*) How to monitor the terms of the SLA and resources while providing trust in the monitored data to involved parties?

3 Approach

To address the issues related to the management of SLAs, an approach to automatically manage the payment and compensation of SLAs using SCs running on a blockchain is presented herein. To provide a straightforward proof-of-concept, the approach is described using NFV-related SLAs as an example. Nevertheless, the presented approach can be applied to different types of contexts.

Blockchain-based SCs can be programmed to execute immutable agreements between two parties that do not necessarily trust each other, such as a subscriber and an SP. Thus, it is logical that SLAs can be translated into SCs. The proposed approach considers the use of a blockchain capable of running a Turing-complete language, *i.e.,* it is possible to implement any computational program using the provided language. Therefore, this Turing-complete SC language increases the spectrum of functions of the compensation process that can be implemented in an SC, such as the implementing a formula to calculate the exact reimbursement amount based on a variable number of inputs. One example of such a Turing-complete language can be found in [6]. Figure 1 depicts an overview of the architecture of the proposed approach.

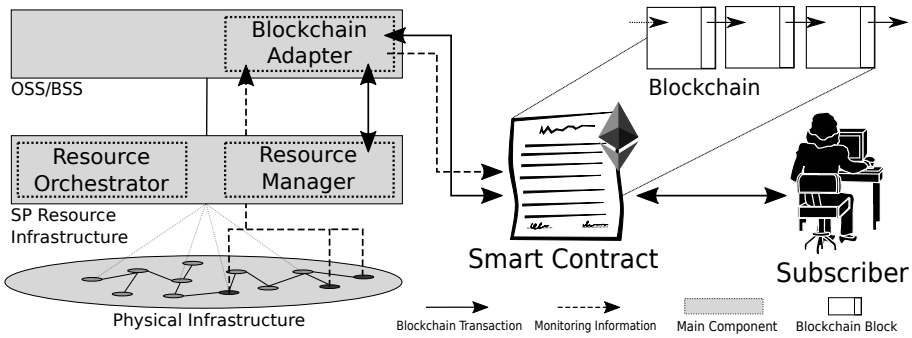

Fig. 1. Overview of the proposed general architecture

In the proposed approach, a subscriber can request the desired resources to an SP and specify the SLAs that must be complied. The SP, in turn, have to deploy an SC in the blockchain (using the *Blockchain Adapter*) that contains, amongst other information: (*i*) the translated SLAs in the form of code, (*ii*) the details of the requested resources, and (*iii*) the duration and price of the services. This SC will only be enforced when the subscriber transfers the necessary funds to the SC address; this means that it is accepting the terms (*i.e.,* SLAs) of the SC and subscribing to the service. Once the deposit transaction is accepted by the network, included in the blockchain, and acknowledged by the SP, the SP dispatches an *event* to the *Resource Manager* to deploy the requested resources in the *Physical Infrastructure*. When resources are deployed, the *Resource Manager* can update the SC, so that the subscriber is aware that the deployment

process is completed and that it can start to use resources. This interaction between the *Resource Manager* and the SC occurs using the *Blockchain Adapter* component, which implements an Application Programming Interface (API) to send transactions to the blockchain. Moreover, the *Blockchain Adapter* is placed within the OSS/BSS to support different *Resource Managers* and *Resource Orchestrators*. Thus, this component is agnostic to the technology used by the *SP Resource Infrastructure* to deploy resources in the *Physical Infrastructure*. One open issue of this approach is the monitoring of resources to ensure that the agreed SLAs are not being violated, this issue is discussed in [6].

In the Network Function Virtualization (NFV) [1] context, which proposes to virtualize and host physical middleboxes on standard hardware, SLAs can specify that Virtualized Network Functions (VNF) or a chain of such VNFs must provide a specific throughput to achieve a required Quality of Service (QoS). For instance, a subscriber of a security-related service chain (*e.g.,* a firewall, and a Deep Packet Inspection (DPI)) can define an SLA stating that this chain should process x packets per second. Otherwise, if the condition is not met, a financial compensation must be paid.

A high-level proposal for the interaction steps of the approach using VNFs as a resource example is described in [6]. The first step is the request of desired VNFs by the subscriber to the SP, informing VNF Descriptors and SLAs. The SP in turn codes and deploys the SC in the blockchain containing the information received from the subscriber. After the SC is appended in the blockchain, the SP sends the address of the SC to the subscriber which deposits the funds in the SC, subscribing to the service and agreeing on the terms of the SLAs. Once the SP acknowledges the deposit in the SC, it requests for the deployment of VNFs by the NFV Infrastructure (NFVI). The NFVI deploys the VNFs and updates the SC with the information regarding the deployed VNFs (*e.g.,* IP address, port number, and status). With the information stored in the SC, the subscriber can redirect its network traffic through the VNFs. Trusted monitoring agents within the NFVI continuously monitor the VNFs checking for any SLA violation, if any violation is encountered, they send this information to the SC so that the subscriber can receive the defined compensation, if no violation is encountered and the contract is not expired they continue the monitoring. Otherwise, if the contract has expired, then the SP can receive the funds stored in the SC.

4 State-of-the-Art

Regarding the application of SCs to support the management of SLAs, [4] proposes a formal description of Web API alongside with the specifications of related SLAs. Moreover, the authors propose an SLA contract method built on top of Ethereum. However, the proposed solution focuses on Web API-related SLAs only. Moreover, [5] proposes an SC, also on top of Ethereum, to implement SLAs in the Small-Cell-as-a-Service context. The SLA implemented in the SC relates to an individual home or business user providing a service for mobile network

operators. Even though this solution implements SLA in SCs, the authors do not delve into the details of how the monitoring of the terms is performed.

The efforts of these two approaches point towards that it may be feasible to translate SLAs into SC and to automate some SLA management steps that are being currently performed manually, such as the compensation process previously described. However, some aspects still need to be researched, especially regarding the complexity of employing blockchain-based SCs to solve this issue, and the integrity of the monitored data utilized to verify whether the SLAs are being complied or not.

5 Discussion and Next Steps

The approach herein presented is based on SCs for an automated compensation of SLAs. It automatically enforces the payment of the defined subscription fee or the monetary compensation in case of an SLA violation. Supporting the proposal of the approach, it was presented a proof-of-concept describing the SLA compensation process between a subscriber and an NFV SP being performed through an SC. Moreover, different facets of SLA management can be incorporated in the approach, such as SLA negotiation, and contract definition, enhancing the features provided by current OSS/BSS. Thus, it is expected that with the development of this approach further aspects of SLA management can be improved, reducing the bureaucracy, costs, and involved parties.

Furthermore, the research on the employment of SCs to automate the compensation of SLAs and payment of subscriptions contributes not only to the SLA management area but the overall study on blockchains and related applications, which is still in its infancy. Next steps include, but are not limited, to implement the approach, conduct a qualitative and quantitative evaluation of the implementation, and seek to answer the listed research questions.

References

1. Network Functions Virtualisation (NFV). White Paper (October 2012), Available at https://portal.etsi.org/nfv/nfv_white_paper.pdf Accessed 08 January, 2018
2. Alharby, M., van Moorsel, A.: Blockchain-based Smart Contracts: A Systematic Mapping Study. CoRR abs/1710.06372 (2017)
3. Mubeen, S., Asadollah, S.A., Papadopoulos, A.V., Ashjaei, M., Pei-Breivold, H., Behnam, M.: Management of Service Level Agreements for Cloud Services in IoT: A Systematic Mapping Study. IEEE Access PP(99), 1–1 (2017)
4. Nakashima, H., Aoyama, M.: An Automation Method of SLA Contract of Web APIs and Its Platform Based on Blockchain Concept. In: 2017 IEEE International Conference on Cognitive Computing (ICCC). pp. 32–39 (June 2017)
5. Pascale, E.D., McMenamy, J., Macaluso, I., Doyle, L.: Smart Contract SLAs for Dense Small-Cell-as-a-Service. CoRR abs/1703.04502 (2017)
6. Scheid, E.J., Stiller, B.: Automatic SLA Compensation based on Smart Contracts. Technical Report IFI-2018.02 https://files.ifi.uzh.ch/CSG/staff/scheid/extern/publications/IFI-2018.02.pdf, Universität Zürich, Zürich, Switzerland (April 2018)