Noname manuscript No.
(will be inserted by the editor)

Smart Contracts for Service-Level Agreements in Edge-to-Cloud Computing

Petar Kochovski · Vlado Stankovski · Sandi Gec · Francescomaria Faticanti · Marco Savi · Domenico Siracusa · Seungwoo Kum

Received: date / Accepted: date

Abstract The management of Service-Level Agreements (SLAs) in Edge-to-Cloud computing is a complex task due to the great heterogeneity of computing infrastructures and networks and their varying runtime conditions, which influences the resulting Quality of Service (QoS). SLA-management should be supported by formal assurances, ranking and verification of various microservice deployment options. This work introduces a novel Smart Contract (SC) based architecture that provides for SLA management among relevant entities and actors in a decentralised computing environment: Virtual Machines (VMs), Cloud service consumers and Cloud providers. Its key components are especially designed SC functions, a trustless Smart Oracle (Chainlink) and a probabilistic Markov Decision Process. The novel architecture is implemented on Ethereum ledger (testnet). The results show its feasibility for SLA management including low costs operation within dynamic and decentralised Edge-to-Cloud federations.

P. Kochovski · V. Stankovski (corresponding author) · S. Gec
Faculty of Computer and Information Science
University of Ljubljana, Ljubljana, Slovenia
E-mail: vlado.stankovski@fri.uni-lj.si

M. Savi
University of Milano-Bicocca
Milan, Italy

F. Faticanti · M. Savi · D. Siracusa
Fondazione Bruno Kessler
Trento, Italy

F. Faticanti
University of Trento
Trento, Italy

S. Kum
Korea Electronics Technology Institute
Seongnam-si, Republic of Korea

Keywords SLA Management · Edge · Fog · Cloud · Blockchain · Smart Contract · Smart Oracle

1 Introduction

Today, smart applications are being developed for many domains including smart homes, cities and communities, robotics, industry 4.0, construction and similar [23, 29, 30]. Various Artificial Intelligence (AI) methods are used in order to provide intelligent features to such smart applications. Due to the greatly varying requirements that may dynamically change, the technology basis for smart applications is increasingly represented by the Internet of Things (IoT), the Cloud technology and blockchain. There are many expected benefits of using these technologies, such as improved Quality of Service (QoS) in conditions of dynamic operation and heterogeneity, higher utilisation of resources, and lower operational costs. In order to address various non-functional requirements of smart applications, Cloud computing today is stretched to the Edge of the network, and Fog generally refers to heterogeneous, geographically distributed Cloud computing offer.

Moreover, the IoT, AI and Cloud technologies alone are not enough to support dynamic application scenarios across broad geographic areas, for instance, when equipment, robots, cars or smartphones move from one place to another. In many scenarios, the applications are time-critical and it is therefore paramount to achieve high QoS operation of the applications. Therefore, stipulating Service Level Agreements (SLAs) between application users and infrastructure providers would provide certain level of guarantee that the required QoS of their AI application will be maintained above threshold at all times. This study addresses the problem of achiev-

ing high QoS by designing and implementing a new SLA management architecture, which operates promptly and transparently and is particularly suitable for smart multi-component applications deployed across the Edge-to-Cloud computing continuum.

Currently, the key problems when deploying compute- and network-intensive AI applications in the Edge-to-Cloud continuum are the great heterogeneity of possible application deployment options coupled with dynamically changing operational conditions [8]. Few possibilities for achieving high QoS in the Edge-Fog context is by increasing the redundancy, for example, by federating several computing resources [24], and then, by providing an orchestrator and a load balancing capability to dynamically (re)deploy AI containers from one computing resource to another in view of maintaining high QoS [22].

The goal of the present study is to design a new SLA management architecture that achieves high QoS operation through federation and orchestration of Edge-Fog computing offer, and is particularly suitable for the deployment of specific AI components in Edge-to-Cloud environments. This study is motivated by the emergence of blockchain, in particular its constructs and mechanisms such as Smart Contracts (SCs) and trustless Smart Oracles [16], and can be used to implement mechanisms for the dynamic federation of computing resources (e.g. VMs) coupled with transparent and traceable orchestration.

In addition to the above, our novel architecture utilises resource provisioning and application placement methods, which are based on Markov Decision Process (MDP) to model both internal and external (contextual information) metrics that affect the resulting QoS of the individual AI applications [15]. Since the runtime conditions can dynamically change, an automaton – a model derived by using an MDP – can be used in order to obtain QoS assurances, ranking and verification of available deployment options. All these three aspects can be used for transparent and trusted SLA management. Our hypothesis is that implementing such probabilistic assurances on blockchain can contribute to achieving trust relationships among the participating entities. The research value of the present work is the integration of the MDP method with Edge-to-Cloud resource federation and orchestration mechanisms implemented through the use of SCs and Smart Oracles. This study aims thus at showing that this novel approach can be used for SLA management in dynamic Edge-to-Cloud environments.

In summary, the contributions of this study are as follows:

- a novel blockchain-based SLA management system for federated Edge-to-Cloud computing environments;
- an unique combination of Smart Contracts and Smart Oracles, which substitutes third party trust authorities for transparent and traceable SLA management;
- the integration of a stochastic MDP method for QoS-aware orchestration of resources, which is designed as a plug-and-play software component;
- the implementation of a proof-of-concept SLA management scenario for the DECENTER Fog and Brokerage Platform [24], involving registration of trusted deployment options (i.e. trusted Edge-to-Cloud computing clusters), deployment, redeployment of applications and trusted and transparent management of the service payment process.

The rest of the paper is organised as follows. Section 2 identifies the gap addressed by the present study. Section 3 describes the motivation for developing SLA management architecture that introduces implementation of probabilistic decision-making methods and blockchain technologies for reliable SLA Management. Section 4 provides an architecture overview of the proposed solution alongside with proof-of-concept application (re)deployment scenarios implemented within the study. Section 5 discusses the results of the experimental evaluation. Section 6 concludes the study and presents future improvements of the proposed architecture.

2 Background

This study builds on recent developments in three areas, starting from computing opportunities in the Edge-to-Cloud continuum based on virtualisation, trusted SLA management and blockchain technologies. These developments are discussed briefly in the following three subsections.

2.1 Computing in the Edge-to-Cloud continuum

The broad diffusion of IoT devices has significantly influenced multiple domains, therefore introducing a plethora of automation opportunities [1, 14, 33]. Such environments usually produce large quantities of data (e.g. sensor measurements, images or video streams) that should be processed and responded upon in a timely manner. However, performing various operations over large quantities of data that continuously move from the field sensor devices to the Cloud computing infrastructures, can lead to low QoS that can be due to connectivity performance issues between the field devices

and the Cloud infrastructures. As a result, more distributed computing paradigms that are in close proximity to data sources, such as Edge and Fog computing, have emerged as a means to address such requirements. Edge computing is a highly distributed approach that performs computational operations on multiprocessor devices (e.g. Raspberry Pi, BeagleBoard) that operate in proximity of the sensor devices [26]. Fog computing is similar to Cloud computing, only that it lies somewhere between the IoT devices and the Cloud computing data centres [2]. However, organisations that are adopting the new IoT concepts and migrating their computing power to the Edge-to-Cloud computing continuum are still facing difficulties in situations when they need to guarantee the high QoS of their applications, which are often deployed across multiple computing infrastructures from various providers.

2.2 Blockchain essentials

Blockchain (BC) technology is a distributed ledger technology that can be used to address several requirements of distributed and decentralised systems. These include autonomy, data management, privacy, security, transparency and traceability. The blockchain technology has the role to replace third party trusted and centralised entities (i.e. certified middleman) by distributing trust in a decentralised network. In order to fully exploit its potential, system architects must carefully define the requirements of the distributed and decentralised system. In particular, they must focus on the agreement policy among the sub-systems and select the adequate blockchain topology ecosystem that offers additional advanced functionalities, such as SCs. Thus, in order to ensure trust, an agreement policy (i.e. consensus protocol) has to be satisfied. For example, the majority of blockchain network participants have to agree, through the voting system, on the modification of the blockchain ledger properties or rules.

The first practical implementation of blockchain is Bitcoin [21], whose simplicity of exchanging digital assets encouraged many researchers and blockchain enthusiasts to develop their blockchain cryptocurrencies. Vitalik et al. [3] complemented the exciting blockchain concept by introducing Turing complete SCs that are similar to general (notary) contracts with limited, but at the same time sufficient functionalities to cover a wide range of use-cases.

Using blockchain and SCs within existing Cloud architectures has much potential. Carminati et al. [5] investigated blockchain as a platform for secure inter-organisational business processes management. Zhang et al. [34] presented *TOWN CRIER* (TC) aiming to provide trustworthy (trustful) data to SCs through a middleman service (TC Server). Furthermore, Smart Oracles are useful means that reduce the necessity of costly operations on a blockchain, such as storing and using data within SCs. In particular, external data that is provided by Smart Oracles can be used within an SC in order to decide, if a deployment option can satisfy the requested QoS, and consequently used to deploy an AI container on the deployment option automatically [16]. Advanced Smart Oracle solutions, such as Provable[1] (rebranded Oraclize[2]) provide Smart Contract templates, which ensure Oracle correct data flow. Another Smart Oracle solution is the Ethereum based Chainlink network[3] that provides reliable tamper-proof inputs and outputs for SCs on any blockchain and at the same time overcomes the limitation of the Provable transparency of the Smart Oracle running instance(s). To be more concrete, Chainlink enables running one's own Smart Oracle dockerized instances with custom cost policy (e.g. the request price determined in LINK tokens may be set). These few useful studies form the basis for the SLA management with SC in the present work.

2.3 Related works on SLA Management

SLAs play vital role in the management of service delivery among different parties. In particular, SLAs are essential in mediation of applications between systems. In the context of Cloud computing, an SLA is often considered as a set of constraints that ensure costumer's benefit when the agreement is violated by the infrastructure provider. For instance, SLAs are very important when defining requirements, such as availability and reliability. A suitable definition of SLA was presented by Buyya et al. [4]: *SLA is an officially exchanged document that describes (or tries to express) in measurable (and maybe qualitative) terms the service being presented to a customer. Any metrics involved in an SLA should be capable of being controlled on a systematic basis and the SLA should record by whom.*

A typical SLA lifecycle is composed of several phases, such as: service use, modelling, SLA management, SLA enforcement and SLA conclusion. According to a systematic survey [19] on SLA in IoT, the SLA management, though, is an important phase that covers multiple aspects (i.e. SLA definition, SLA modelling, SLA negotiation) that has received less research attention. A relevant challenge in SLA management is the trust

[1] https://provable.xyz/
[2] http://www.oraclize.it/
[3] https://chain.link/

Table 1: Positioning of the present approach among related works

Work	Objective	Trust Evaluation Method	Use of Smart Oracles	Computing paradigm
Gill et al. [10]	SLA-Management	Self-Management	N/A	Cloud
Labidi et al. [17]	SLA-Management	Third-Party Service	N/A	Cloud
Li et al. [18]	Resource Matchmaking	Third-Party Service	N/A	Cloud
Zhang et al. [35]	SLA-Management	Third-Party Service	N/A	Cloud
Zhou et al. [36]	SLA-Enforcement	Blockchain	✗	Cloud
Zhou et al. [37]	SLA-Enforcement	Blockchain & SC	✗	Cloud
Proposed Approach	SLA-Management	Blockchain & SC	✓	Edge-to-Cloud

between engaged parties, particularly the authority that detects SLA violation. In commercial solutions, often the provider handles the monitoring and violation detection (e.g. Amazon CloudWatch[4]). Since each service provider has its own monitoring and violation detection policies, proving an SLA violation can be a very complex operation for service costumers. Thus, establishing a trustful relationship between the service providers and consumers is very important. Gill et al. [10] presented a framework for self-management of cloud resources, whose software components are described in few consecutive studies [27], [28] and [11]. Their framework provisions and schedules cloud resources, whilst maintaining SLAs and reducing SLA violation rate. In their studies, the SLA management resides on the side of the service providers. On the other hand, to improve the trust that the consumers have in the service providers, some studies provide tools that assist them in monitoring the SLA. For instance, Muller et al. [20] proposed a third party SLA management platform that generates comprehensive SLA violation's explanations in order to aid the user to renegotiate SLAs. In addition, Labidi et al. [17] described a monitoring approach complemented with semantic SLA modeling, which provides the consumers with comprehensible models of SLA documents.

To avoid biased SLA management, some studies propose approaches that guarantee trustful SLA management by including a third party to handle the SLA management. For instance, authors of [18] designed a service operator trust scheme for resource matchmaking across multiple Clouds and proposed a trust evaluation method based on information entropy. Similar approach was also proposed by Zhang et al. [35], whose SLA management framework utilises a third-party auditor to ensure the benefit of consumers. In particular, their auditor is designed to verify and resolve the violations that can appear between a service provider and its consumers. However, in such schemata, all participants are obliged to trust a centralised third-party broker. Since trusting a third-party SLA management system can be difficult in reality, a blockchain-based SLA management system can aid to overcome that problem [36, 37], where the blockchain offers complete transparency and SLAs are precisely declared in the SC. In addition, blockchain integration into SLA management systems can lead towards a automated, simplified and less expensive sharing of infrastructures using SLA [13] and compensation between providers and consumers in case of SLA violation [25].

Our work complements and improves on the above efforts by delivering a new architecture for efficient SLA management in the Edge-to-Cloud where third party trust authorities are substituted with SC functions that automate the process and enhance trust between service providers and consumers in a loosely coupled, dynamic system. In contrast to previous studies, the proposed architecture integrates the triumvirate of technologies: blockchain, multi-objective decision making and multi-tier monitoring to assure automated and transparent interactions among humans and artificial agents across the complete Edge-to-Cloud continuum. Furthermore, this architecture addresses the limitations of existing blockchain-based SLA management solutions, particularly, it works based on less costly off-chain data which is facilitated through the use of Smart Oracles. Finally, a proof of concept and a feasibility evaluation of the proposed SLA management architecture is presented by investigating the trade-off between execution time and cost of the overall process.

3 Motivation

The motivation for this study is derived from the latest trends in software engineering, which are based on building flexible and reusable AI applications by implementing a multi-tier application architecture. However, splitting AI methods into several computing tiers, such as into Front and Rear parts of Deep Learning Neural Networks, may lead to greater privacy and security of the information when it is being executed in the Fog [6].

In this study, we focus on a two-tier AI application, which is designed for video surveillance and is necessary

[4] https://aws.amazon.com/cloudwatch/

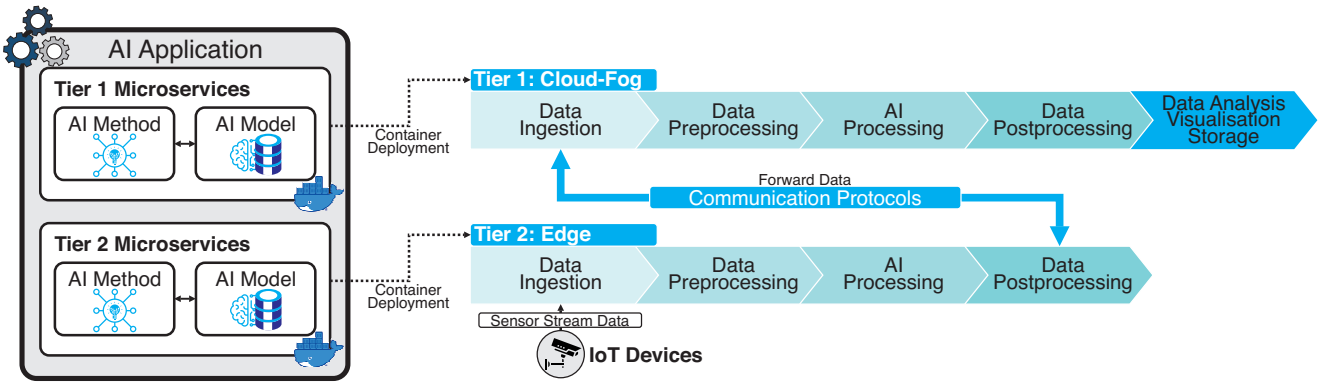

Figure 1: Video surveillance application implemented in a two-tier architecture.

to satisfy high QoS standards, thus respect SLAs. However, the overall approach is applicable to AI applications designed by following different design patterns. The European Union - Korea DECENTER project[5], in which we participate, currently implements several use case scenarios in which Deep Learning Neural Networks are applied to video streams. These scenarios include smart street crossings, smart construction, robotic vision and smart homes. For example, a two-tiered AI application can be used in the area of the smart construction sector in order to detect various safety violations, such as detecting if workers wear safety equipment. The execution of the Front part of the Deep Neural Network close to the Edge may contribute to greater privacy, while the execution of the Rear part in a more powerful Fog computing resource can be used to improve the QoS.

This generic two-tiered application design is depicted in Figure 1. The application is composed of several containerised components, which are organised for deployment onto two different tiers. The first tier is composed of Cloud computing resources that allow executing software components that require greater computing power. For instance, the first tier is capable of executing the Rear part of deep learning methods (e.g. TensorFlow) in order to analyse the incoming already preprocessed video frames. The second tier represents the Edge computing resources that execute the software component near the video surveillance data sources. Here the sensor data is preprocessed and an AI method is executed in order to perform time-critical data analysis.

The decision upon a deployment option for this application is a complex problem. There exist various internal and external metrics that could influence the resulting QoS of the application. For example, a definitive QoS metric for the application is considered to be its response time, that is, the time for a video frame to pass from the tier-two components up to the end of the tier-one components pipeline. This resulting QoS can be influenced by network related conditions (e.g. attributes like latency, throughput, packet loss and similar) and computing resource related operational conditions (e.g. CPU cores and available memory).

In our practical use case, the second tier is located in Ljubljana, Slovenia, and is composed of five deployment options, whereas the first tier is composed of 40 deployment options that are spread across Europe, Asia, Australia and North America. All deployment options are enlisted in Table 2 within Section 5.

In order to achieve high QoS in its operation, the smart AI application should be capable of immediate reaction in case of a high probability of an SLA violation. This means that the software engineer may set this probability in a way that a redeployment of the application component in the first tier will happen whenever the probability of not withholding the required QoS threshold is too high. In our study, the SLA management and the redeployment process is performed in a transparent, traceable and autonomous manner. These properties are achieved through the use of blockchain-based technologies, including SCs and Smart Oracles as elaborated in the following.

4 Blockchain-based SLA Management

In the following, we elaborate our novel SLA management architecture and system implementation that is designed to provide high QoS operation to DECENTER's smart applications. The goal of our proposed SLA management architecture is to facilitate an automated and transparent decision-making process for (re)deployment of two-tier applications in the Edge-to-Cloud computing continuum.

[5] https://www.decenter-project.eu/

The proposed architecture implements MDP methods that aid to automatically rank the available deployment options according to prior usage information, current monitoring data and QoS requirements that are precisely defined within the SLA.

In order to achieve these technical goals, the proposed architecture allows registering available deployment options by providers, definition of SLA user requirements and autonomous deployment and redeployment of applications among the available deployment options. The architecture can be observed through three scenarios, which are the following: (1) registering a certified deployment option, (2) automated deployment of applications and (3) automated redeployment of applications. These three scenarios are elaborated in the following subsections.

4.1 QoS-aware SLA assessment

In order to develop our proof-of-concept QoS models and provide autonomous SLA management, the proposed architecture utilises infrastructure- and application-level metrics. Infrastructure-level metrics show the current status of the deployment options; thus, they are necessary to successfully perform the deployment process. The application-level metrics are accumulated once a deployment is finished; thus, they are used as additional metrics for performing redeployment operations. The used attributes for the (re)deployment processes are as follows:

1. Infrastructure-level attributes:
   - Network throughput (Mb/s) – the rate at which data is transferred between two endpoints. At this level the endpoints are the source of data on one side and the deployment options on other side;
   - Network latency (ms) – the time required for a packet to be transferred between the source of data and the deployment options;
   - Cost ($/month) – cost for monthly use of a deployment option;
   - Amount of memory – amount of memory that a deployment option offers;
   - Amount of CPU cores – amount of CPU cores that a deployment option offers.
2. Application-level attributes:
   - Throughput (Mb/s) – the rate at which data is transferred between the two tiers of deployment options;
   - Latency (ms) – the time required for a packet to be transferred between the two tiers of deployment options.

The proposed architecture in this study is not limited to the specific set of QoS metrics and non functional requirements, because it supports any quantitative attribute that may be of interest to the software engineer.

4.2 Architecture overview

This section presents the high-level architecture design for SLA management that implements SCs to automate the deployment process which is depicted on Figure 2. The designed multi-level architecture follows the interoperability standards set by organisations such as the Cloud Native Computing Foundation (CNCF)[6], OpenFog Consortium[7] and Edge Computing Consortium Europe (ECCE)[8]. Each architecture level is described in the following.

1. *Application Layer* is an application with an intuitive graphical user interface (GUI). It is an entry point that the software engineer uses to define QoS requirements, which are incorporated into SLA. This layer is directly communicates with the Blockchain Layer, which is the Ethereum ecosystem. Thus, it allows the user to trigger a SC execution.
   In order for the Application Layer to communicate with the Ethereum (ETH) ecosystem, it implements ETH bridge called Metamask[9], which plays pivotal role in the process. Primarily, Metamask is an Ethereum wallet that allows users to: (1) create and switch accounts that can be used in various ETH networks (e.g. Main ETH network, Ropsten, Kovan or Rinkeby); (2) perform transactions between accounts. It facilitates the interaction with the Ethereum ecosystem by injecting a Javascript library called web3.js [9].
2. *Blockchain Layer* is necessary to automate the SLA management process and empower fairness between the involved parties (i.e. service providers and consumers) and executes traceable and transparent transactions on the Blockchain.
   The blockchain implementation is based on public Ethereum ledger as a public blockchain environment, because it is composed of two main components: SC templates and Smart Oracles. There are two main types of SC utilised in the system: SCs for registration of deployment options on the blockchain and SCs for automated (re)deployment of applications.

[6] https://www.cncf.io/
[7] https://www.openFogconsortium.org/
[8] https://ecconsortium.eu/
[9] https://metamask.io/

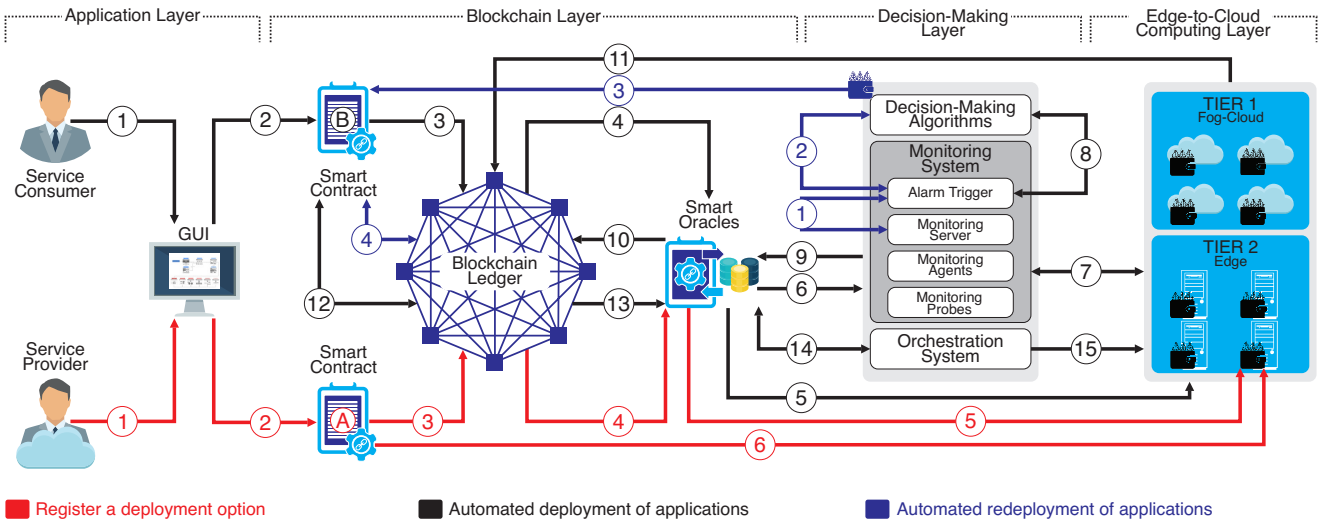

Figure 2: Architecture for SLA management and detailed design of loosely coupled system components.

The deployment of the SCs occurs on demand through the blockchain service which plays the role of a non-biased system that executes the SC and pays the service provider in case the SLA is not violated. However, in case there is an SLA violation, the SC terminates, pays the service provider for the service provided until the moment of the SLA violation, whilst compensating the service consumer for the remaining of time. A more detailed overview of the SCs work and the SLA management workflow is elaborated in the sections bellow.

However, SCs by default cannot act outside the blockchain, thus they are not capable to retrieve off-chain data. Since SCs in this SLA management system have to communicate with external services, such as computing nodes, QoS monitoring system or decision-making mechanisms, Smart Oracles had to be implemented. The Smart Oracles are trusted third-party services that provide means for SCs to communicate with registered APIs from the external services. This approach results in enhanced integrity of the functions that verify the correctness of the API queries by using unique API keys and thus avoid calls from potential malicious SCs.

3. *Decision-Making Layer* estimates the optimal deployment option for deployment of containerised software components and initiates the container deployment process. In order to estimate an optimal deployment option, this layer queries a Smart Oracle from the Blockchain Layer to retrieve monitoring data and prior usage knowledge only for deployment options, which are registered on the blockchain. This layer is composed of components that are products of our earlier research work. In particular, the Decision-Making Layer is composed of three systems: decision-making mechanism [15], monitoring system [31] and an orchestration system [22].

The implemented decision-making mechanism is based on the Markov Decision Process (MDP) that generates a probabilistic finite automaton that is built for each microservice. MDP utilises the automaton to derive utility value for each deployment option. These values are later used to produce a ranking list, where the first ranked deployment option is considered as an optimal deployment option and returned as an output result that satisfies the engineer's QoS requirements. The MDP automaton contains: a set of states (i.e. deployment options); a set of actions (i.e. deployment actions applicable to the set of states); a set of transition probabilities, which represent the transition probability between states due to the available deployment actions; and a set of rewards, which represent the expected reward for transitioning from one state to another. Transition probabilities, which are estimated from prior usage experience of the deployment options and state rewards, which are estimated from the monitoring metrics are essential when calculating the utility of each state. A detailed description of the algorithm including the calculation of rewards and transition probabilities is available elsewhere [15].

The monitoring system constantly gathers QoS data from the deployment options that are registered on the blockchain. Each of those deployment options run Monitoring Agents and Monitoring Probes, which accumulate the QoS metrics and forward them to the Monitoring Server. In addition, the monitoring system contains an Alarm Trigger, which is a rule-

based entity that continuously verifies the incoming monitoring data. If the Alarm Trigger experiences abnormal behaviour (i.e. SLA violation) it is responsible to initiate the redeployment process. The orchestration is performed by automatically with an orchestration system such as Kubernetes. Once the decision-making mechanism delivers an optimal deployment solution and a SC is successfully executed, the orchestrator receives deployment instructions (i.e. YAML script) that provide information on the deployment infrastructure, applications for deployment, backup and replication policies.

4. *Edge-to-Cloud Computing Layer* is composed of deployment options, which are registered on the blockchain by the service providers. The deployment options are used for deployment of the containerised two-tier applications, where each tier of deployment options play a different role. For instance, the Edge-based deployment options are responsible for running the software components that require less computing power, whereas the other components run on the Cloud/Fog-based deployment options.

4.3 Process workflow

The proposed system consists of two types of SC: (1) a SC for registering deployment options on the BC and (2) a SC for executing (re)deployment operations through the blockchain whilst following the SLA. The workflow of the proposed architecture can be presented with three correlated scenarios. A high level representation of the interactions between architecture components is presented in Figure 2. In addition, the detailed flow of interactions between the fundamental components in the presented system within the three scenarios are depicted in Figure 3.

The first scenario is registration of certified deployment options. It allows infrastructure providers to register via blockchain their available deployment options in the pool of certified deployment options. In order to do so, the infrastructure provider (1-2) invokes a SC to execute methods for (3) registration a deployment option on the BC. Because the deployment options are off-chain data, (4) BC communicates with them through a Smart Oracle and (5) assigns to the specific deployment option a public address (i.e. digital wallet). When the public address is assigned, the SC registers the new public address onto the BC. Finally, (6) the provider verifies the public address of the deployment option.

The second scenario is automated deployment of applications. It performs SLA assessment, preparation, negotiation, contracting and deployment of software components on the optimal deployment options. This scenario is executed in the following 15 consecutive steps: (1) the software engineer defines application QoS requirements and preferred usage-time for the deployment option; (2) the GUI through the *triggerSC()* method invokes the SC through the Metamask Ethereum bridge; (3) the SC initiates deployment process and (4) triggers the Smart Oracle through the *selectDO()* method; (5) the Smart Oracle gathers information on blockchain-certified deployment options and (6) triggers the decision-making mechanism; (7) the decision-making mechanism retrieves prior usage data and current monitoring metrics for the available deployment options from the Monitoring System, (8) estimates the optimal option and (9) returns the results to the Smart Oracle, which (10) triggers the SC to (11–13) verify the wallet address, executes the deployment process; (14) the Smart Oracle triggers the Kubernetes Orchestrator cluster to (15) deploy the two-tier application on the selected deployment options.

The third scenario extends the second scenario by adding the automated redeployment functionality. The monitoring system, which is a part of the Decision-Making Layer, contains an Alarm Trigger that constantly examines the monitoring data for threshold violations (1). Once a violation is detected it triggers the decision-making mechanism to reassess the probability of achieving high QoS (2). In case the probability for QoS violation is high (i.e. above certain threshold), the decision-making mechanism retrieves a new optimal deployment option. The decision-making mechanism then directly forwards the solution to the SC through its public address that is dedicated for the redeployment process (3-4). Once the SC is triggered, it initiated the compensation process through the *initiateRefund()* function that is represented within the alternative scenario (i.e. alt) from Figure 3. In particular, the SC responsible for the initial deployment estimates the amount of time the deployment option was utilised and pays the service provider, whilst the service consumer is reimbursed for the remaining time for which the deployment option remained unused through the SC event *FundsReleaseEvent()*. After the compensation is finished, the SC has the same workflow as for the deployment scenario, which is: price determination, reaching consensus, executing payment and deployment of the application.

4.4 Smart Contract implementation details for SLA Management

This subsection will provide implementation details about the SCs that were designed for the SLA management scenario. For the purposes of this study, two SC were

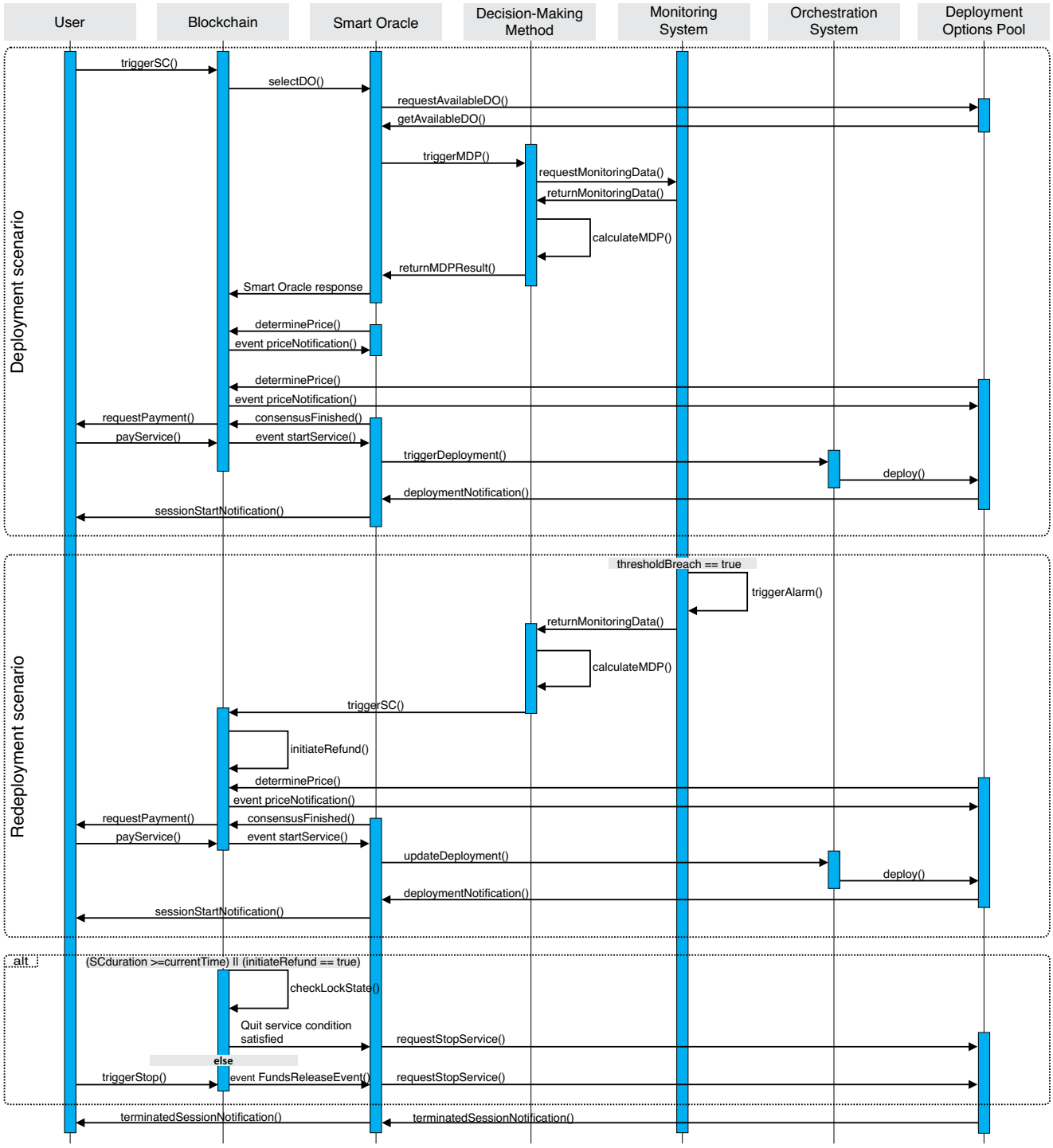

Figure 3: Sequence diagram of the deployment and redeployment scenarios.

designed: SC "A" that is necessary to register deployment options on the BC and SC "B" that is responsible for the (re)deployment scenarios. Both SCs follow a design pattern that allows one version of the SCs to be deployed and used multiple times until destroying them. The design of the SCs follows the Oracle pattern proposed by Wöhrer et al. [32] and best practices presented in the OpenZeppelin framework[10]. Contracts are available in the following public repository[11].

[10] https://openzeppelin.org/

[11] https://bitbucket.org/friljubljanaslovenia/smart-contracts-for-sla/

The developed SC "A" allows service providers to register their deployment options on the Blockchain. This will allow to have transparent information for each deployment option. In particular, only deployment options registered on BC through this SC will be used during the decision-making process. SC "A" is composed of two pillar functions *registerDO()* and *deleteDO()* where the first function is used by the service provider to register a deployment option on BC with requested requirements passed as an input to the SC function. The second function *deleteDO()* is use to remove the existing deployment options from the set of available ones, ordinary due to the significant change of the requirements or the deprecation of the deployment option.

The developed SC "B" deals with the (re)deployment scenario. The SC contains functions for managing the deployment process, based on the Chainlink Smart Oracle that accesses information from the decision-making mechanism. In addition, the SC "B" is responsible to compensate the service consumers if the SLA is violated, by implementing dynamic price business model. In particular, the service consumer knows the total cost that is needed for the deployment option utilisation and once an agreement is reached, consumer's funds are locked by the SC. If the maximum utilisation time limit is reached then the funds are unlocked and the service provider is fully paid. However, if there is an SLA violation, the service provider receives funds only for the time their deployment option was used and the rest of the funds are refunded to the service consumer. This SC refund approach make it possible to automate and facilitate the monetising process among the service provider and the service consumer.

In the context of the deployment, the Smart Oracle plays the crucial role of managing the data flow between the Smart Contract consisting of on-chain data and the dedicated services (e.g. orchestration service) consisting of off-chain data. By doing that we preserve high level of traceability and at the same time the condition fulfilment checks are increased. To fully exploit the Smart Oracle capabilities it is recommended to run all Smart Oracle triggered services on a dedicated hardware to preserve full auditing and data verification of all the triggered data through the services.

5 Experimental evaluation

The goal of the experimental evaluation is to show that the proposed architecture for SLA management with smart contracts is fully functional and can be used to perform (re)deployment operations in order to maintain high QoS at all times. The experimental evaluation consists of evaluating the probabilistic decision-making method and the blockchain performance. The proposed solution in this study was tested by a user (software engineer) that is based in Ljubljana, Slovenia, who used the system to deploy an AI application on an optimal deployment option. Due to the variability of the monitored metrics (i.e. networking, infrastructure and application metrics), an optimal deployment option at one instance in time may not be an optimal solution in another instance in time.

The engineer could deploy his application on one out of 42 available deployment options within Tier 1 (Fog-Cloud) and on one out of 5 deployment options within Tier 2 (Edge). The Tier 2 deployment options were hosted on the network edge, near to the sources of data, and were not subject to experimentation. On the other hand, the available deployment options from Tier 1 were hosted on Google Cloud Platform[12], Amazon AWS EC2[13] and ARNES[14], at 6 different locations: Ljubljana, Frankfurt, London, Tokyo, Sydney and Oregon. The current properties of the utilized deployment options are listed in Table 2. The network performance of the available deployment options is depicted in Figure 4 and Figure 5, where it can be seen that geolocation plays important role in the network performance. In particular, all deployment options that are located geographically closer to Tier 2 show significantly better network performance regarding the latency threshold. Therefore, the (re)deployment in this scenario will also consider parameters such as the network performance between the two tiers.

Tier 2 deployment options have normally attached on-field sensors and cameras that produces the data for the AI application. Therefore, the deployment of software components on Tier 2 deployment options in this scenario mainly depends on previously determined hard constraints by the software engineer (e.g. sensor type of data, sensor location, data context and similar). Therefore, our evaluation related to (re)deployment techniques is focused on Tier 1 deployment options.

5.1 Probabilistic decision-making evaluation

The probabilistic model for the deployment process was generated by using multi-level monitoring and usage data that was collected in a period of one month, prior to the experimental evaluation.

For the needs of the experimental evaluation, the software engineer had an AI application that had to process high-resolution video surveillance data, there-

[12] https://cloud.google.com/
[13] https://aws.amazon.com/ec2/
[14] https://arnes.splet.arnes.si/

Table 2: Experimental testbed Edge-to-Cloud infrastructures used for the deployment process

Deployment option	CPU cores	vRAM	Location	Cost [$/month]	id
arnes-0	1	4	Ljubljana	0	0
g1-small	1	1.7	Frankfurt, London, Tokyo, Sydney, Oregon	16.56; 16.56; 16.45; 18.24; 13.13	1-5
n1-standard 1	1	3.75		31.27; 31.27; 31.17; 34.45; 24.27	6-10
n1-standard 2	2	7		62.55; 62.55; 62.34; 68.9; 48.55	11-15
n1-standard 4	4	15		125.09; 125.09; 124.68; 137.8; 97.09	16-20
n1-standard 8	8	30		250.19; 250.19; 249.37; 275.6; 194.18	21-24
a1.medium	1	2	Frankfurt, Tokyo, Sydney, Oregon	21.31; 23.50; 24.38; 18.67	25-29
a1.large	2	4		42.61; 47.0; 48.76; 37.34	30-33
a1.xlarge	4	8		85.21; 93.99; 97.51; 74.67	34-37
a1.2xlarge	8	16		170.41; 187.98; 195.01; 149.33	38-41

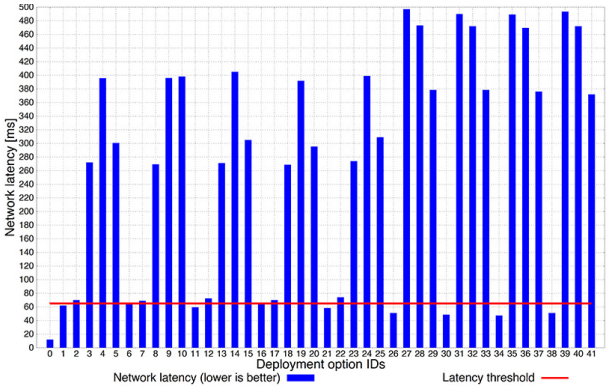

Figure 4: Latency between the source of data and the deployment options.

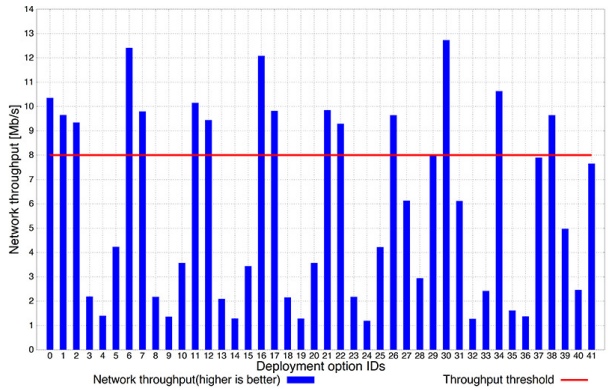

Figure 5: Throughput between the source of data and the deployment options.

fore the following QoS requirements were selected: CPU cores more than or equal to 4, vRAM more than or equal to 4GB, cost less than or equal to 100$/month, application level latency less than or equal to 65 ms and throughput more than or equal to 8 Mb/s. The implemented MDP considers the QoS requirements as soft constraints, thus the MDP solution may not satisfy all constraints, but will tend to choose the option with the lowest amount of threshold violations. For the purposes of the SLA Management, the software engineer also selected the following Service Level Objectives (SLOs): application level latency less than 100 ms, application level throughput higher than 6 Mb/s, deployment option's memory utilisation less than 80% and deployment option's CPU utilisation less than 90%.

Once a deployment decision takes place, the deployment option was monitored using the Prometheus Monitoring System[15] against the defined SLOs. Then, artificial network latency (see Figure 6) on the chosen deployment option was gradually applied, using the network emulator NetEm[16], which forced the alarm trigger to react. When the alarm trigger [31] detects a violation, it notifies the probabilistic model to perform probabilistic evaluation and estimate the redeployment confidence level in order to initiate redeployment process. Each time the MDP can provide a new deployment result. The results, presented in Table 3 show the scores of the top five ranked deployment options for deployment and redeployment of the AI applications. In both scenarios, deployment and redeployment the top ranking deployment options were the ones that satisfied the highest amount of QoS requirements and were located

[15] https://prometheus.io/
[16] http://man7.org/linux/man-pages/man8/tc-netem.8.html

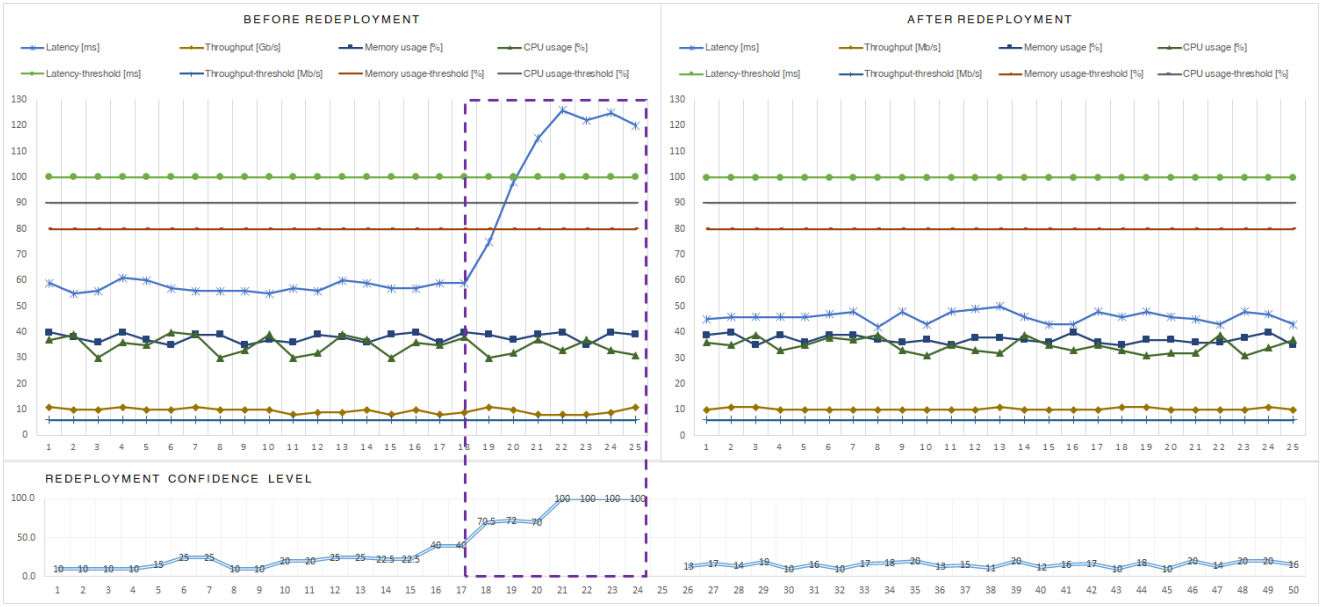

Figure 6: Monitoring data comparison before and after the redeployment process. The MDP method's probabilistic assurances are provided as confidence levels and are used to make a decision on redeployment. When such decision is taken the MDP decision-making method is used to choose a new deployment option based on the available pool of resources.

geographically closer to Tier 1. In this case, for both scenarios deployment an redeployment, the mechanism chose deployment options that satisfied all QoS requirements. Although the deployment option with id=0 was in closes proximity to Tier 1, during the evaluation process it did not met the requirement for CPU cores amount, thus with one threshold violation was ranked fifth during the redeployment scenario.

Table 3: Deployment and redeployment ranking results of top five deployment options

	Deployment			Redeployment		
Rank	**Deployment option**	**ID**	**Score**	**Deployment option**	**ID**	**Score**
I	a1.xlarge	34	167.96	a1.large	30	161.57
II	a1.2xlarge	38	159.22	n1-standard-8	21	157.31
III	a1.large	30	158.21	n1-standard-4	16	157.09
IV	n1-standard-8	21	158.03	n1-standard-2	11	151.85
V	n1-standard-4	16	152.84	arnes-0	0	150.98

Figure 6 depicts the difference in the monitoring values before and after the redeployment took place during the experimental evaluation. The moment when the monitoring values of a1.xlarge with id=34 were close to violate the SLA agreement, the system had a high level of redeployment confidence that led towards initiating the redeployment process and selection of a1.large with id=30 as optimal deployment option.

5.2 Blockchain performance evaluation

SCs have an important role in the system architecture, therefore the individual functions including the SC deployment have to be efficient in terms of cost and execution performance. In comparison to the traditional SCs interacting fully with on-chain data, our SCs "A" and "B" presented in Section 4.4 include Smart Oracle mechanisms that increase the data interaction on an off-chain level but at the same time slightly increase the overall cost of the Smart Oracle enabled functions.

In our experimental environment we used the Ethereum testing environment Rinkeby and we presented the averaged results of 10 executions, running own Chainlink nodes with the determined Smart Oracle cost per interaction of 0.001 LINK tokens, which is equivalent to the main Ethereum environment of approximate 0.037 USD. The evaluated SCs ("A" and "B") from the gas consumption metric are depicted in Figure 7. The results indicate that the most expensive functions are constructors and following the Smart Oracle enabling functions. Since the deployment of the SC "A" is performed only once and the deployment of the SC "B" once per service consumer, the SCs are costly feasible. Moreover, the majority of the functions is triggered by the *Service Provider* stakeholders (e.g. Service owner, Cloud providers, etc.), while only 4 functions (triggerSC, payService, checkLockState and triggerStop) are executed from *Service Consumer* stakeholder. The Service Consumer

SC functions are relatively unexpensive and thus make the SCs overall costly acceptable.

It is known that the execution of SCs is possible within one block or multiple ones. We focused on the main three functionalities registration, deployment and redeployment. The results shown in Table 4 indicate that the fastest operation is registration, due to the simplicity of the SC functions, but the deployment and redeployment execution time varies from 1 to 2 blocks depends on Ethereum network load and used transaction fee. In the case of low execution time it is used high transaction fee of $20 \cdot 10^{-9}$ ETH, otherwise in the case of high execution time it is used low transaction fee of $1 \cdot 10^{-9}$ ETH [7]. Even though the performance of the SC functionalities are consuming and it is not possible to reduce the execution below one block (that is approximate 15 seconds), the trust benefits among the involved entities prevail.

Table 4: Performance analysis for registration, deployment and redeployment operations.

Operation	**Low Execution Time [sec]**	**High Execution Time [sec]**
Registration	12	15
Deployment	14	24
Redeployment	13	18

5.3 Critical Analysis

The proposed SLA management system was fully tested and evaluated. Our evaluation addressed the performance of the probabilistic decision-making and the implemented blockchain solutions. The MDP-based decision-making method derived the ranking of the deployment options based on current monitoring data and prior usage data that was aggregated within the duration of one month. In both cases, that is, deployment and redeployment, the method provides correct output by ranking first the deployment options that satisfy the highest number of soft QoS requirements. Hence, the decision-making method proves useful for the design of an SLA management system. Furthermore, the blockchain implementation was evaluated in terms of cost and execution performance, because these criteria have an essential role, if we intend to achieve business value. Our results shown that the blockchain operations take at average no more than 24 seconds to execute, thus allowing the system to timely address any SLA violations, while at the same time satisfying high security and trust related standards that are offered by this technology. On the other hand, the transaction fees for the operations depend on the amount of traffic that the blockchain is experiencing and the required execution time for each operation. In other words, the faster the operations need to be executed, the more expensive it will be. Although, as described in Section 5.2, the transaction fees are significantly low during the evaluation process, there is still space to further optimise the performance of our blockchain implementation. In summary, the proposed SLA management approach is able to deliver QoS-aware deployment operations, whilst at the same time maintaining trust relationships among the involved entities.

6 Conclusion and future scope

The motivation of the present study is to address the requirements of computationally, memory and network intensive microservices. With Edge-to-Cloud computing offer it is necessary to find new and dynamic means for the management of SLAs in view of achieving high QoS operation of the microservices. This study explores the possibility of using Blockchain and Smart Contracts and presents a new architecture that is, to the best of our knowledge, the first of its kind.

Based on our implementation and presented experimentation, we may conclude that our proposed Smart Contract-based architecture and system achieves federation of resources and automated deployment and redeployment of applications in a dynamic and decentralised way. It is particularly suitable for two- (or even multi-) tier AI applications that are engineered as groups of containers (pods), for example, front and rare parts of Deep Neural Networks, including pre- and post-processing methods.

The MDP method can be used for automated decision-making based on the probability of achieving high QoS. Based on various QoS metrics and prior usage data, the MDP method is used to build an automaton (probabilistic model). Its states represent all available deployment options, and its transitions contain probabilities for maintaining an idle state or (re)deployment of the software components from one deployment option to another. This happens when a QoS metric is violated. Then, the automaton is used to: (1) assess the probability that the QoS will be maintained above the threshold (assurances), (2) rank the available Cloud deployment options, and (3) verify the results. It was shown that the developed blockchain-based architecture and system can be used (1) to federate computing resources close to each other, thus increasing the available deployment options, and (2) to manage the SLAs in an automated and stochastic way.

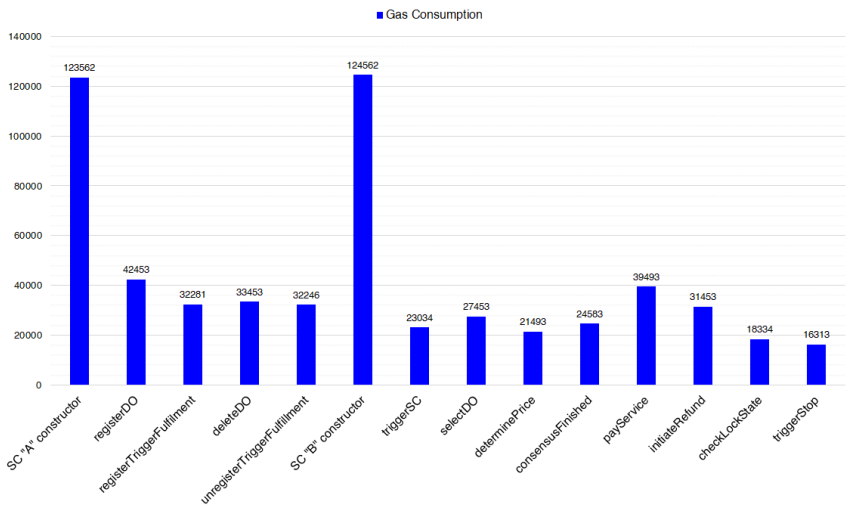

Figure 7: Gas consumption of all SC functions in SC "A" and SC "B".

The experimental evaluation was performed using 43 deployment options from the Fog/Cloud (Tier 1), 5 deployment options from the Edge (Tier 2), and 7 QoS-related metrics. The results from the experimental evaluation show that the proposed architecture can be used to rapidly resolve QoS threshold violations, thus offering solution that will satisfy SLAs and compensate the service consumer when necessary. This design study forms an integral part of the design of the DECENTER's Fog Computing Platform.

With the emergence of new paradigms and technologies, such as IoT, Blockchain and AI, the Edge-to-Cloud computing transforms towards intelligent computing and becomes more complex overall. A recent study in this context has analysed the transformative effects of related technologies and paradigms on cloud computing, and revealed new trends and challenges for energy management, resource management, fault tolerance, security, privacy and many more topics that will be addressed in future research [12].

Our future work priorities go into the following directions. First, we need to improve the MDP method and to consider various blockchain-based architectures. The MDP method will be further evaluated and improved for various redeployment scenarios in multi-tier architectures, such as consolidating pairs of Edge and Fog deployment options as pairs within the states of the automaton. The Blockchain Layer, on the other hand can be furthermore improved by introducing more Smart Contracts model, interledger support, which can allow transactions between different types of ledgers and additionally reduce the cost over the blockchain operations. Finally, our new Horizon 2020 Research and Innovation Action ONTOCHAIN (Trusted, traceable and transparent ontological knowledge on blockchain) aims at achieving trust in highly decentralised environments by means of integration between Semantic Web and Blockchain technologies and its novel set of protocols will be directly useful to achieve trusted SLAs across the computing continuum.

Acknowledgements The research and development reported in this paper have received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no. 815141 (DECENTER: Decentralised technologies for orchestrated Cloud-to-Edge intelligence) and grant agreement no. 957338 (ONTOCHAIN: Trusted, traceable and transparent ontological knowledge on blockchain).

References

1. Al-Fuqaha A, Guizani M, Mohammadi M, Aledhari M, Ayyash M (2015) Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys & Tutorials 17(4):2347–2376
2. Bonomi F, Milito R, Zhu J, Addepalli S (2012) Fog computing and its role in the internet of things. In: Proceedings of the first edition of the MCC workshop on Mobile cloud computing, ACM, pp 13–16
3. Buterin V (2013) Ethereum white paper. GitHub repository, https://github.com/ethereum/wiki/wiki/White-Paper
4. Buyya R, Garg SK, Calheiros RN (2011) Sla-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. In: 2011 in-

ternational conference on cloud and service computing, IEEE, pp 1–10
5. Carminati B, Ferrari E, Rondanini C (2018) Blockchain as a platform for secure inter-organizational business processes. 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC) pp 122–129
6. Castillo EA, Ahmadinia A (2017) Distributed deep convolutional neural network for smart camera image recognition. In: Proceedings of the 11th International Conference on Distributed Smart Cameras, pp 169–173
7. Chen S, Choo KR, Fu X, Lou W, Mohaisen A (eds) (2019) Security and Privacy in Communication Networks - 15th EAI International Conference, SecureComm 2019, Orlando, FL, USA, October 23-25, 2019, Proceedings, Part I, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 304, Springer
8. Dastjerdi A, Gupta H, Calheiros R, Ghosh S (2016) Chapter 4—fog computing: Principles, architectures, and applications. ininternet of things: Principles and paradigms, ed. r. buyya, and av dastjerdi, 61–75
9. Ethereum core team (2016) web3.js-ethereum javascript api v.1.2.7. https://web3js.readthedocs.io/en/v1.2.7/
10. Gill SS, Buyya R (2019) Resource provisioning based scheduling framework for execution of heterogeneous and clustered workloads in clouds: from fundamental to autonomic offering. Journal of Grid Computing 17(3):385–417
11. Gill SS, Chana I, Singh M, Buyya R (2018) Chopper: an intelligent qos-aware autonomic resource management approach for cloud computing. Cluster Computing 21(2):1203–1241
12. Gill SS, Tuli S, Xu M, Singh I, Singh KV, Lindsay D, Tuli S, Smirnova D, Singh M, Jain U, et al. (2019) Transformative effects of iot, blockchain and artificial intelligence on cloud computing: Evolution, vision, trends and open challenges. Internet of Things 8:100118
13. Hang L, Kim DH (2019) Sla-based sharing economy service with smart contract for resource integrity in the internet of things. Applied Sciences 9(17):3602
14. Kochovski P, Stankovski V (2018) Supporting smart construction with dependable edge computing infrastructures and applications. Automation in Construction 85:182–192
15. Kochovski P, Drobintsev PD, Stankovski V (2019) Formal quality of service assurances, ranking and verification of cloud deployment options with a probabilistic model checking method. Information and Software Technology DOI https://doi.org/10.1016/j.infsof.2019.01.003
16. Kochovski P, Gec S, Stankovski V, Bajec M, Drobintsev PD (2019) Trust management in a blockchain based fog computing platform with trustless smart oracles. Future Generation Computer Systems 101:747–759
17. Labidi T, Mtibaa A, Gaaloul W, Tata S, Gargouri F (2017) Cloud sla modeling and monitoring. In: 2017 IEEE International Conference on Services Computing (SCC), IEEE, pp 338–345
18. Li X, Ma H, Zhou F, Gui X (2014) Service operator-aware trust scheme for resource matchmaking across multiple clouds. IEEE transactions on parallel and distributed systems 26(5):1419–1429
19. Mubeen S, Asadollah SA, Papadopoulos AV, Ashjaei M, Pei-Breivold H, Behnam M (2017) Management of service level agreements for cloud services in iot: A systematic mapping study. IEEE Access 6:30184–30207
20. Müller C, Oriol M, Franch X, Marco J, Resinas M, Ruiz-Cortés A, Rodríguez M (2013) Comprehensive explanation of sla violations at runtime. IEEE Transactions on Services Computing 7(2):168–183
21. Nakamoto S (2019) Bitcoin: A peer-to-peer electronic cash system. Tech. rep., Manubot
22. Paščinski U, Trnkoczy J, Stankovski V, Cigale M, Gec S (2018) Qos-aware orchestration of network intensive software utilities within software defined data centres. Journal of Grid Computing 16(1):85–112
23. Rawat DB, Brecher C, Song H, Jeschke S (2017) Industrial Internet of Things: Cybermanufacturing Systems. Springer
24. Savi M, Santoro D, Di Meo K, Pizzolli D, Pincheira M, Giaffreda R, Cretti S, Kum Sw, Siracusa D (2020) A blockchain-based brokerage platform for fog computing resource federation. In: Conference on Innovation in Clouds, Internet and Networks
25. Scheid EJ, Rodrigues BB, Granville LZ, Stiller B (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
26. Shi W, Cao J, Zhang Q, Li Y, Xu L (2016) Edge computing: Vision and challenges. IEEE Internet of Things Journal 3(5):637–646
27. Singh S, Chana I (2016) Resource provisioning and scheduling in clouds: Qos perspective. The Journal of Supercomputing 72(3):926–960

28. Singh S, Chana I, Buyya R (2017) Star: Sla-aware autonomic management of cloud resources. IEEE Transactions on Cloud Computing
29. Song H, Rawat DB, Jeschke S, Brecher C (2016) Cyber-physical systems: foundations, principles and applications. Morgan Kaufmann
30. Song H, Fink GA, Jeschke S (2017) Security and Privacy in Cyber-Physical Systems. Wiley Online Library
31. Taherizadeh S, Stankovski V (2019) Dynamic multi-level auto-scaling rules for containerized applications. The Computer Journal 62(2):174–197
32. Wöhrer M, Zdun U (2018) Design patterns for smart contracts in the ethereum ecosystem. In: 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp 1513–1520
33. Zanella A, Bui N, Castellani A, Vangelista L, Zorzi M (2014) Internet of things for smart cities. IEEE Internet of Things journal 1(1):22–32
34. Zhang F, Cecchetti E, Croman K, Juels A, Shi E (2016) Town crier: An authenticated data feed for smart contracts. Cryptology ePrint Archive, Report 2016/168, https://eprint.iacr.org/2016/168
35. Zhang H, Ye L, Shi J, Du X, Guizani M (2014) Verifying cloud service-level agreement by a third-party auditor. Security and Communication Networks 7(3):492–502
36. Zhou H, Ouyang X, Ren Z, Su J, de Laat C, Zhao Z (2019) A blockchain based witness model for trustworthy cloud service level agreement enforcement. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, pp 1567–1575
37. Zhou H, Ouyang X, Su J, de Laat C, Zhao Z (2019) Enforcing trustworthy cloud sla with witnesses: A game theory–based model using smart contracts. Concurrency and Computation: Practice and Experience p e5511