

RESEARCH

Open Access

AWESOME: an auction and witness enhanced SLA model for decentralized cloud marketplaces



Zeshun Shi¹, Veno Ivankovic¹, Siamak Farshidi¹, Jayachander Surbiryala², Huan Zhou^{3*} and Zhiming Zhao^{1*} 

Abstract

In recent decades, the world has witnessed cloud computing as an essential technology that changes the traditional application Development and Operation (DevOps) lifecycle. However, current cloud software DevOps and Service Level Agreement (SLA) management often face challenges of 1) selecting the best fitting service providers, customizing services and planning capacities for large-scale distributed applications; 2) guaranteeing high-quality and trustworthy SLAs among multiple service providers; 3) enhancing the interoperability of cloud services across different providers; and 4) designing effective incentive models among stakeholders. This paper proposes a novel framework called Auction and Witness Enhanced trustworthy SLA for Open, decentralized service Marketplaces (AWESOME) to build a trustworthy cloud marketplace and address the above challenges. The proposed framework contains four subsystems: a customizable graphical user interface, an auction-based service selection model, a witness committee management mechanism, and a smart contract factory orchestration. We developed a prototype AWESOME decentralized application (DApp) based on the Ethereum blockchain. Extensive experiments are designed to evaluate the latency and cost of our model. The experimental results demonstrate that our model is economical and feasible.

Keywords: Decentralization, Cloud marketplace, Auction, Service level agreement

Introduction

The cloud computing paradigm provides flexible services based on pay-as-you-go business models [1]. In the current cloud marketplace, several well-known service providers maintain the traditional cloud marketplace, and the share of these top providers is continuously growing. According to a report, as of October 2020, AWS, Azure, Google, and Alibaba control 63% of the entire cloud marketplace, whereas all other providers only share 37%¹. Since product migration is complex, consumers become locked in a particular provider's ecosystem. In the future,

however, we can expect a more open, fair, and trustworthy cloud resource trading marketplace for all service providers and customers.

There are typically two cloud transaction models: centralized and decentralized [2]. In a centralized cloud environment, all service trading and trust-related issues rely on trusted third parties (TTPs), e.g., some well-known cloud service providers with good reputations and track records. However, those providers are not always trustworthy in practice and can be biased or conspire with any party. On the other hand, in a decentralized trading environment, all sellers or buyers perform transaction management and operations, avoiding the concentration of power and making the transactions more trustworthy. In this case, all trust assurance comes from a decentralized platform (e.g., blockchain), which needs to be appropriately designed, implemented, deployed, and monitored.

*Correspondence: huanzhou@nudt.edu.cn; zshi2@uva.nl; zzhao@uva.nl

³School of Computer Science, National University of Defense Technology, 410073 Changsha, China

¹Informatics Institute, University of Amsterdam, Science Park 904, 1098 XH, Amsterdam, the Netherlands

Full list of author information is available at the end of the article

¹<https://www.canalys.com/newsroom/worldwide-cloud-market-q320>

Traditionally, a Service Level Agreement (SLA) is a business concept that defines the contractual financial agreements between the roles engaging in the business activity. In the context of a cloud marketplace, it is an agreement between the cloud customer and provider regarding the cloud service quality [3]. For instance, the IaaS (Infrastructure-as-a-Service) provider, Amazon Elastic Compute Cloud (Amazon EC2), claims that the availability of its data center is no less than 99%. If this is not achieved, it will pay back 30% credits to its customers as compensation. In practice, however, this agreement is hard to enforce fairly and transparently; it is usually performed manually and dominated by giant providers in the traditional SLA management process.

Blockchain technology can be used to support decentralized applications (DApps), bringing new hints of possible solutions to address these challenges [4, 5]. It inspires the emergence of a new decentralized cloud marketplace that encourages greater inclusivity and participation from different service parties. We can foresee that such a decentralized marketplace will provide more choices and opportunities for both providers and consumers. Besides, the smart contract makes it possible to manage and automate the SLA process on the blockchain in a fair and tamper-proof way [6]. However, reaching a consensus on events that occur outside the blockchain is another possible challenge. Cloud customers or providers can still violate the agreed SLA despite using the blockchain to complete cloud transactions. For example, the provider may not provide the QoS (Quality of Service) they promised, and the customer may refuse to pay for the claimed cloud resources. In the blockchain community, the bridge between on-chain and off-chain events is called “oracle” [7]. One of the solutions to build this bridge is to retrieve data from Oraclize², a third-party company that performs as a trusted data source for the blockchain. However, this solution suffers from a single point of failure and needs extra commission fees. In this case, a decentralized witness mechanism is promising to judge SLA violations that occur off-chain.

This paper expects to use blockchain to enhance the cloud marketplace and SLA management lifecycle by introducing a novel Auction and Witness Enhanced trustworthy SLA for Open, decentralized service Marketplaces (AWESOME) framework. Specifically, a new role called auction witness is involved in the entire cloud service trading process, as shown in Fig. 1. In our model, the decentralized blockchain users can join the SLA judgment and work as witnesses through an incentive mechanism that motivates them to make truthful judgments to win profits. This paper is based on our previous conference paper [8], and we have extended the framework, protocols, and

experiments extensively. In brief, the main contributions of this paper can be summarized as follows:

- A novel auction and witness enhanced SLA model called AWESOME for decentralized cloud marketplaces. The model can support interactions between service providers, customers, and witnesses to complete trustworthy transactions and SLA enforcement.
- A prototype DApp based on the AWESOME model is fully developed on the Ethereum blockchain³. It contains customizable graphical user interfaces (GUIs) and advanced smart contract protocols to support the SLA business process.
- Extensive experiments are designed to evaluate the execution latency and cost of the proposed model and DApp. The experimental results demonstrate that our model is economical and feasible to implement.

In the rest of this paper, we first demonstrate the related works in “[Related work](#)” section. Then, in “[The AWESOME framework](#)” section, we analyze the system requirements, objectives, actors, on-chain and off-chain interactions, and then present the overall AWESOME system architecture. Next, we introduce the design choices of our AWESOME DApp and show the details of how the DApp works with a business process model in “[The AWESOME DApp demonstration](#)” section. “[Experiments and validations](#)” section shows the experimental results to demonstrate the feasibility of the AWESOME framework and DApp. Finally, we discuss key considerations and future development plans in “[Discussion and future work](#)” section, and conclude the whole paper in “[Conclusion](#)” section.

Related work

This section provides an overview of the state-of-the-art technologies that are related to the AWESOME framework.

Smart contract-based SLA management

Smart contracts are computer programs designed to execute contracts automatically on a blockchain. In this regard, many studies have discussed the use of smart contracts and blockchain for SLA management. Uriate et al. [9] proposed a domain-specific language known as SLAC, which is designed for SLA specification in the cloud computing domain to avoid ambiguities that come with SLAs. The core purpose of SLAC is to describe the contract, specify the contract terms, and define the guarantees of those terms. Gomes et al. [10] developed a smart contract that makes use of web3.js to connect

²<http://www.oraclize.it/>

³The code repository is open-sourced in: <https://github.com/ZeshunShi/AWESOME>

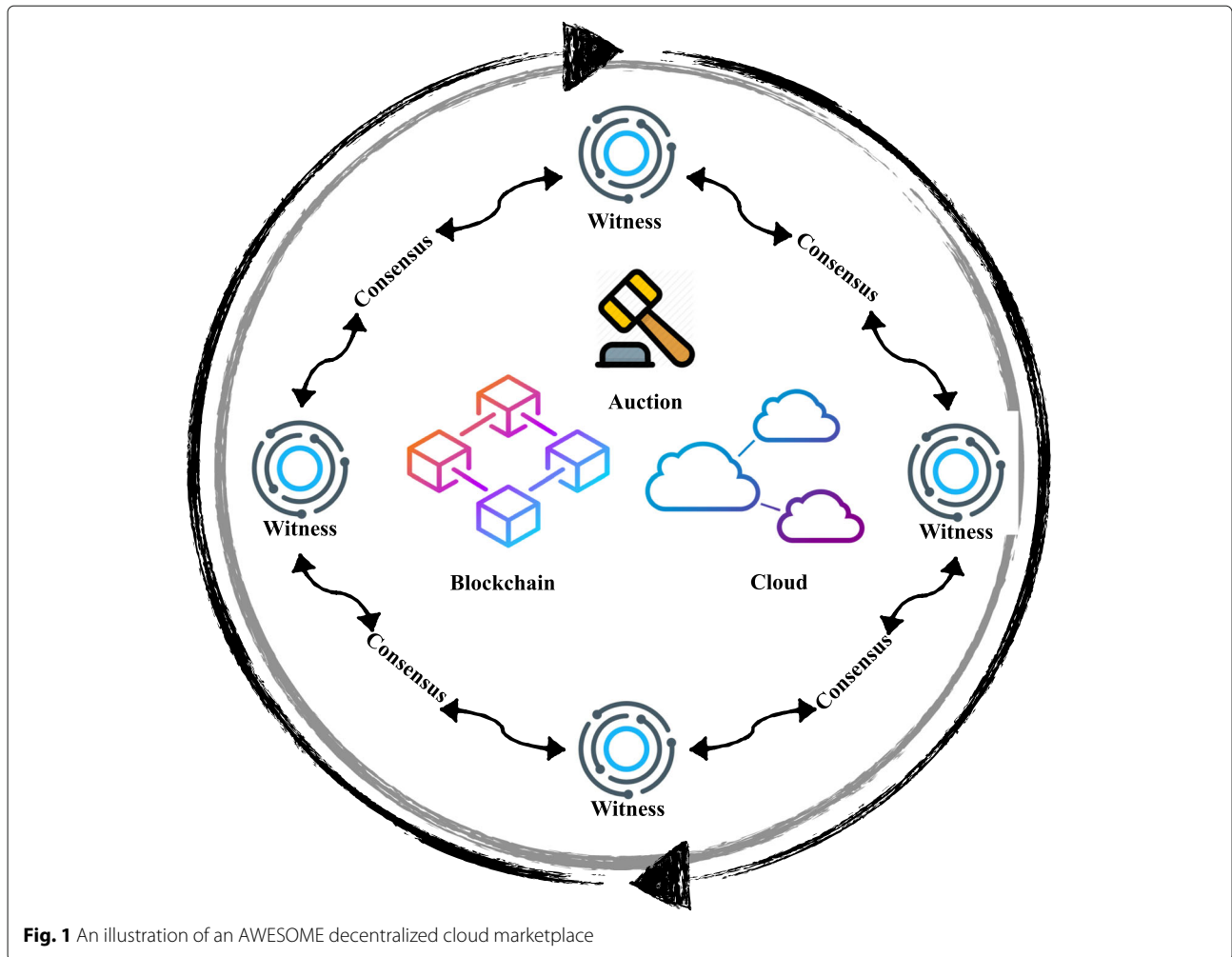


Fig. 1 An illustration of an AWESOME decentralized cloud marketplace

web applications to the blockchain. They use this smart contract to determine if an SLA violation has occurred and to enforce fines and compensation. Their proof of concept demonstrated that service operators could use DApps in cloud environments to simplify the process of SLA validation. The same research group also highlighted transparency and trust between different parties as a major issue in SLA validation [11]. They proposed a solution that uses a decentralized file system to store the generated agreements. An oracle, in this case, is used as a real-time data channel that inserts real-world data into the blockchain to determine if there is any loss of connectivity.

SLA monitoring solutions

There are various monitoring solutions in the SLA domain. For example, the Web Service Management Agent [12] implemented SLA monitoring of web services using agents from both server-side and customer-side. In addition, monitoring is done in practice by SLA decom-

position, where low-level system performance metrics are mapped to high-level SLA parameters [13]. From the consumer's perspective, SLAs should be understandable and avoid ambiguity to ensure QoS, so providers often write SLA documents in legal language. Besides, formal metrics need to be defined in order to facilitate successful monitoring [14, 15]. To ensure proper SLA modeling and monitoring, the authors in [15] suggested an SLA modeling process that defines the ontology to represent the SLA concepts. After that, monitoring can occur to measure different SLA parameters and detect violations. In [16], the authors stated that a monitoring engine should mainly consist of two different phases. The first phase is the runtime monitoring of SLA parameters, where the monitoring agents continuously record the application performance. In the second phase, an analysis engine verifies the values and compares them to predefined thresholds from the agreement's service-level objectives. Then, the engine identifies violations, and the provider is penalized in some cases.

Blockchain-based auction models

Auction is a well-studied research topic due to its effective and fair transaction properties. In cloud computing, classic auction models, e.g., English auction, Dutch auction, and seal-bid auction, have been widely leveraged to optimize cloud resource allocation. For example, the authors in [17, 18] used different auction models to achieve optimal cloud customer/provider selection. While blockchain-based auction models have great potential in optimizing cloud transactions and marketplaces, most existing solutions focus only on the design of the auction models without considering the trustworthy enforcement of cloud SLAs [19].

DApp development challenges

The contribution of this paper is to provide DApp developers with a framework for a decentralized service marketplace in which users can customize to their own business needs and technologies. Therefore, the challenges in DApp development that currently exists should be identified. An initial challenge with AWESOME DApp development stems from the fact that it is new and unexplored. As a result, DApp requirements are not fully understood or require continuous change during production. This is why the use of Agile methods in DApp Development and Operation (DevOps) is often encouraged [20]. Unfortunately, few development frameworks can aid DApp developers in making service marketplaces. To the best of our knowledge, this paper is the first integrated model and DApp attempt to study generic auctions and witness mechanisms for decentralized cloud marketplaces.

The AWESOME framework

In this section, we first perform the requirement analysis using two industrial use cases and show the design objectives of the system. Then, we describe our AWESOME model in detail, including actor identification, on-chain vs. off-chain interactions, and system components & workflows.

Requirements analysis

In industrial innovations (e.g., crowd journalism and disaster early warning) and scientific applications (e.g., research data management), cloud services are playing an increasingly important role in real-time data processing (e.g., multimedia acquired by mobile devices), running simulations (e.g., for predicting possible disasters), and for enabling extensive scale collaborations (e.g., for running distributed scientific workflows). Therefore, it is necessary to employ multiple data centers or providers to handle decentralized collaboration between resource providers and customers in several industrial use cases.

1. Use case 1. Decentralized cloud marketplace for social media (taken from EU ARTICONF project⁴): crowd journalism for real-time news reporting during live sports, music events, or natural disasters. Individual citizen journalists make photos or videos of the news and trade them via the news platform. The system has to detect fake news from those crowdsourced contents by running real-time processing from multiple cloud providers or engaging human experts to review them.
2. Use case 2. Decentralized service marketplace for medical data management (taken from EU CLARIFY project⁵): sharing and utilizing pathology data provided by hospitals or individuals from different countries, where various medical data access constraints are often applied. When a machine learning application for studying breast cancer must use data from multiple hospitals, the application developer has to select cloud providers from a decentralized marketplace that meet the application needs (e.g., geolocation, capacity, and price).

We can therefore highlight the following requirements and challenges from those use cases:

- Provider selection, service customization, and capacity planning challenges. The developer has to select cloud services from different providers (very often multiple ones) due to distributed data locations (e.g., sensors or repositories), various data access constraints (e.g., for medical data), and performance constraints (e.g., for real-time decisions in early warning). The various price and reputation models make the selection time-consuming and challenging to be optimal.
- SLA interoperability and guarantee challenge. The time-critical application constraints, e.g., processing media contents during crowd news reporting and real-time decision-making, require the profound optimization of the application logic and the quality guarantee of the cloud services. However, the diverse SLA terms among providers and the uncertainties in the SLA guarantee make performance optimization difficult.
- Difficulties in setting up incentive models and customizing witness games in a decentralized marketplace. The business logic in a decentralized marketplace is often realized by smart contracts, which are supposed to be immutable after being deployed on blockchains. However, any careless design or mistake may cause unexpected loss.
- Virtual infrastructure automation challenge. When an application involves multiple providers or data

⁴<https://articonf.eu/>

⁵<http://www.clarify-project.eu/>

centers, the provisioning of the virtual infrastructure, deployment of the software platform and application components, monitoring, and adaptation of the application need to be ideally automated. However, the diverse Application Programming Interfaces (APIs) from different providers and the interoperability issues across those providers make automated provisioning and deployment a challenge. This leads to a high level of complexity in monitoring runtime infrastructure quality, detecting SLA violations, and adapting the infrastructure when violations happen.

To tackle these challenges in a decentralized cloud marketplace, we propose the AWESOME framework. The AWESOME software architecture consists of novel technologies in DApp DevOps, game theory, blockchain, and smart contracts.

System objectives

We aim to provide guidelines, tools, and templates that can aid developers in developing a DApp for a specific business that can benefit from a decentralized architecture. The system needs to provide service customers and providers a platform that can facilitate easy SLA generation and operation, and allow decentralized witnesses to monitor such SLAs. Furthermore, the system needs to be generic and modular. DApp developers who use the system could customize the model for their own business needs and adapt it to other blockchain use cases. Specifically, the objectives of the AWESOME model can be summarized as follows:

- Objective 1: Improve the customer/provider selection in a decentralized ecosystem by developing an automated service auction framework to enable dynamic business relations between a consumer(s) and providers and establish SLAs.
- Objective 2: Improve the service quality and SLA's trustworthiness between consumer(s) and providers by establishing a decentralized witness mechanism to monitor the SLA violations and automate the procedure for SLA compensation and payment.
- Objective 3: Improve the model usability by developing easy-to-use customizable DApp GUIs for general cloud users to interact with different smart contracts.
- Objective 4: Improve the continuous DevOps of DApps by providing an integrated contract factory to improve smart contracts' security and efficiency.

Actor identification

Actors which interact with the AWESOME model are human roles, external systems, or devices that exchange

information with the DApp. With this in mind, we identify the following actors:

- **Service Customer:** Service customers use the DApp to find providers that can offer them services. They should be able to make listings on the platform and sign an SLA with a service provider. They pay for these services but can receive compensation in case of SLA violations.
- **Service Provider:** Service providers use the DApp to list their available services on the platform for auction. They earn monetary rewards for these services but may be penalized in case of SLA violations.
- **Auction Witness:** Witnesses can use the DApp to monitor SLAs and receive monetary rewards for their efforts. The judgment from the witness committee can ensure that auctioned cloud services are delivered as agreed in SLAs.
- **AWESOME Operator:** An AWESOME operator could modify the developed framework for a specific business use case. They need to read the provided documentation, edit the user interfaces, blockchain APIs, and smart contract templates, and then deploy a custom decentralized service marketplace⁶.

In our model, customers are motivated to receive services at a low cost, and providers are incentivized to provide high-quality services in return for service fees. Two parties complete transactions and make profits by means of customized on-chain auctions. The payoffs of both parties are always positive; otherwise, an auction will never be settled. The incentive for witnesses, on the other hand, can include the following three parts: 1) a reward for their monitoring efforts; 2) a penalty if their reports about SLA violations fail to match the results of others (based on the majority rule); and 3) a blockchain transaction fee. We specify in our model that the monitoring reward is large enough so that the witness's payoff is always positive. In this way, witnesses are always motivated to participate in our model and earn their rewards.

On-chain vs. off-chain interactions

After identifying system objectives and actors, we can now design the system architecture as a whole. The ABCDE DApp development method [20] suggests first dividing the system into two subsystems. Namely, smart contracts running on top of the blockchain and external systems that interact with the blockchain. The authors also recommended formalizing what transactions and data should be placed on-chain and off-chain. In practice, the information that should be included in the blockchain is that with critical trust requirements. This is because on-chain

⁶For simplicity, customer, provider, witness, and operator are used in the following text.

information is immutable and enforces non-repudiation. In addition, not only “big data” is not suitable for the blockchain, but even “not-tiny” data should not be stored on the blockchain. This is mainly due to cost and scalability considerations; the computing power and data storage space available on the blockchain is limited. One of the typical solutions is to store raw data off-chain due to its size while storing small critical data and hashes on-chain. In terms of computation, blockchains are not the best for complex, intensive computations; however, they provide a benefit in their interoperability properties as many systems can access it [21]. The authors of the ABCDE development methodology suggest as a general guideline that data with transparent and immutable requirements for DApps should be managed on the blockchain system [20]. In this section, we follow this idea to demonstrate our design choices.

On-chain activities

The data and activities the system should keep on-chain include:

- **Auction:** To support transparent trade, the auction of service tasks should be conducted on the blockchain to maintain fairness and prevent fraud. Implementing decentralized auctions on the blockchain can also avoid the cheating auctioneers of traditional centralized auctions and save commission fees.
- **Witness reports:** In order to avoid tampering with the SLA reports, they should be placed on-chain. While this may lead to witnesses viewing each other’s reports and reporting accordingly, the reports should be transparent. One effective way to address this issue and protect privacy is to submit the hashes of the reports in a specified time window.
- **SLAs:** It is essential to include SLA details between the service provider, customer, and witnesses in the blockchain, as all these data have trust requirements. However, since SLAs may be larger textual files that could give a large load to the blockchain, a possible solution is to place SLA metadata that can unlock the SLA off-chain while keeping the cryptographic hash on-chain.
- **Payment enforcement:** In case of an SLA violation, a smart contract should be used to facilitate payments to providers, customers, and witnesses automatically. Blockchain cryptocurrencies can support the secure and fair enforcement of money payments.

Off-chain activities

The data and activities the system should keep off-chain include:

- **User interface:** Due to data loads, the way in which providers, customers, and witnesses interact with the

platform should mostly be off-chain.

- **Cloud Services.** Cloud services offered and used by providers and customers should be off-chain.
- **Pre-monitoring communication:** The platform should facilitate communications between providers and customers before entering an SLA agreement so that they can privately agree upon service and monitoring terms.

Overall system components

Based on the above analysis, we designed the system architecture of AWESOME. The AWESOME framework consists of four subsystems in response to the proposed objectives, as shown in Fig. 2:

1. **DApp Graphical User Interface (DGUI)** provides a flexible and customizable DApp interactive environment for different AWESOME users to connect on-chain and off-chain activities. It is designed to provide a bridge between customers, providers, and witnesses who do not have IT development knowledge and assist them in calling function interfaces between different smart contract agents. Furthermore, the usability of the AWESOME DApp is ensured through a customizable interface design for business needs. More specifically, *DGUI* is designed with interfaces regarding 1) auctions by customers and providers; 2) SLA monitoring activities by witnesses; and 3) DApp management and maintenance by operators.
2. **Auction-Based Service Selection (ABSS)** provides an auction-based service customer/provider selection solution. This subsystem will first diagnose the use case requirements and help the user select the most suitable auction model and algorithm to achieve desirable results. Then, the management of the auction process and the enforcement of the service fee payment (in the form of cryptocurrency) are executed on the blockchain, ensuring that the whole auction is open and trustworthy. Finally, *ABSS* also audits bidder candidates to prevent malicious actors from joining the auction.
3. **Decentralized Witness Committee Management (DWCM)** provides a trustworthy incentive framework to manage decentralized auction witnesses. First of all, an appropriate number of witness candidates will be selected in an unbiased way to perform off-chain monitoring of cloud SLAs. *DWCM* will then invoke a game theory incentive mechanism based on customized payoff functions to enable selected witnesses to make correct judgments to win more profits. The subsystem will also audit the witnesses’ reputations to reward/restrict their participation in future monitoring activities.

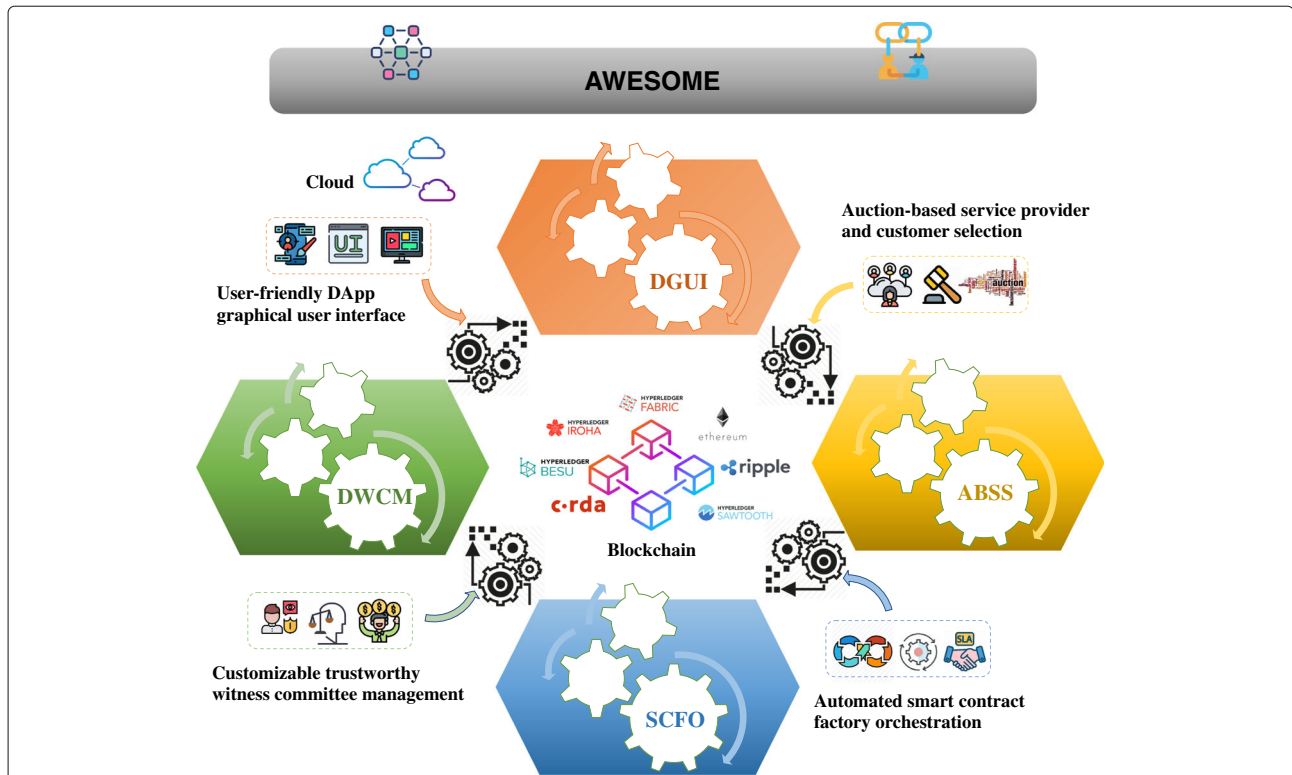


Fig. 2 The overall architecture of the AWESOME framework

4. **Smart Contract Factory Orchestration (SCFO)** provides tools and APIs for AWESOME operators to set the necessary smart contracts on the blockchain. More specifically, it is responsible for automating the process of planning, provisioning blockchain infrastructure, and deploying AWESOME business smart contracts. In addition, the *SCFO* subsystem also monitors and diagnoses smart contracts and the underlying blockchain infrastructure at runtime to provide effective adaptation solutions.

As shown in Fig. 3, the overall workflow of the AWESOME framework can be described as the following steps (using a reverse auction example). First of all, an AWESOME operator calls the *DGUI* subsystem to customize and generate customer, provider, and witness user interfaces (UIs) for the current use case. The AWESOME operator then calls the *SCFO* subsystem to initiate a contract factory. This contract factory automatically generates the required auction, witness, and SLA smart contracts to ensure trustworthy interaction between different participants. Meanwhile, it also invokes a runtime monitor for these contracts and returns the monitoring result to the AWESOME operator. Next, an AWESOME customer invokes the UI to transfer the specific business requirement to the *ABSS* subsystem. Based

on this requirement, an auction model is selected and configured to wait for providers to submit their bids. The decentralized service providers then start to register in the *ABSS* subsystems through their UIs. When there are enough bidder candidates, smart contracts automatically start the auction process to find qualified providers. Then, the witnesses invoke their UIs to register in the *DWCM* subsystem. The *DWCM* subsystem also performs game design and an unbiased witness screener to choose the appropriate witness to form the witness committee. Finally, the selected providers collaborate to provide cloud services, and selected witnesses monitor the SLAs to win profits. The service fee and witness fee will be paid and enforced with cryptocurrency using smart contracts when the cloud service ends.

In the entire AWESOME workflow, *DGUI* provides a customizable graphical interaction environment to support user-to-user interactions in business processes. *ABSS* selects candidate providers through an effective auction mechanism. *DWCM* ensures trustworthy SLA enforcement through truth-telling witness monitoring. Finally, *SCFO* provides automated smart contract support for the entire process. The four subsystems form a dynamic ecosystem that provides services to AWESOME users collaboratively.

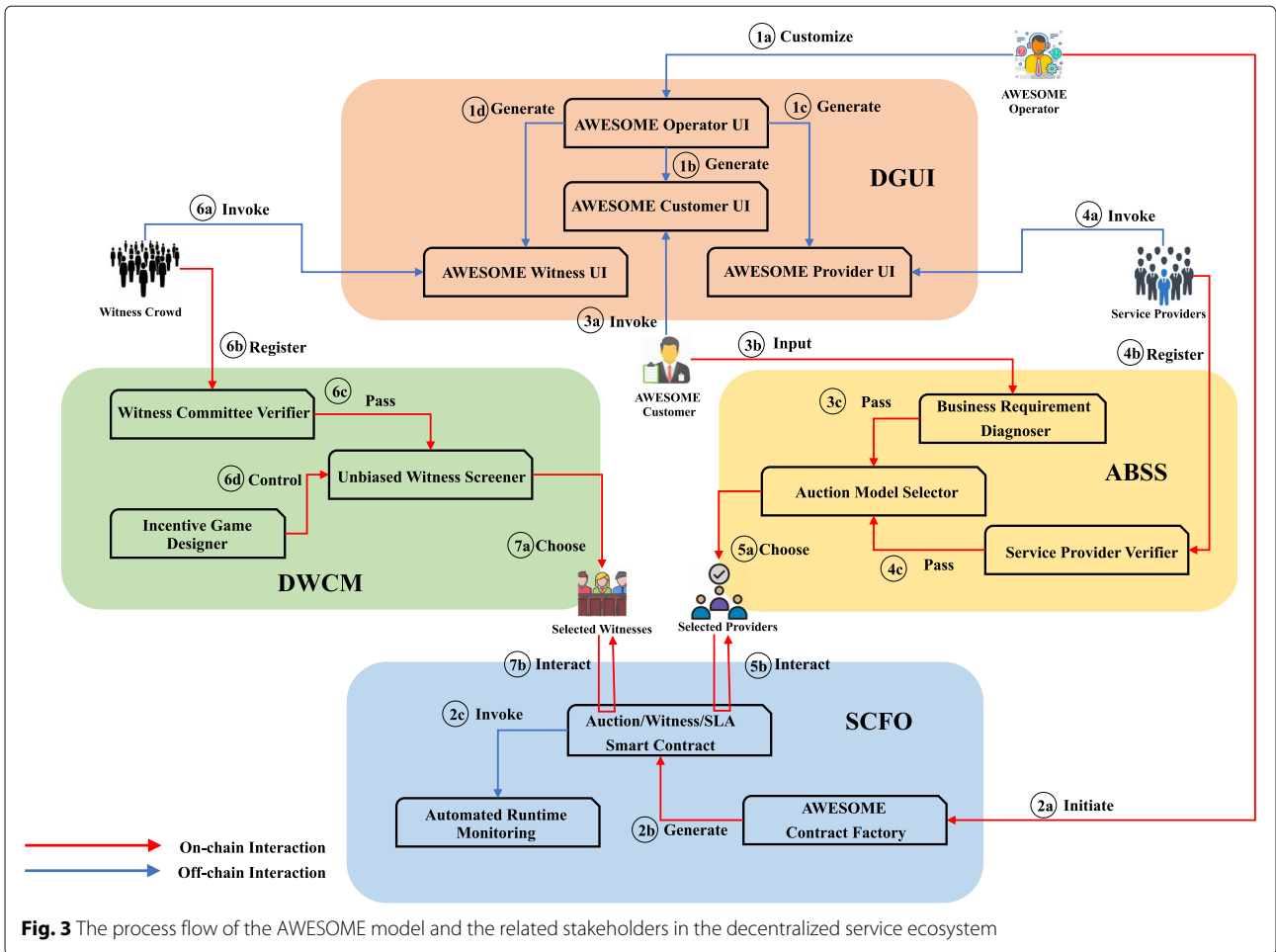


Fig. 3 The process flow of the AWESOME model and the related stakeholders in the decentralized service ecosystem

The AWESOME DApp demonstration

This section presents a prototype system based on our AWESOME framework. The prototype output is a DApp system that helps AWESOME operators and users trade cloud services in a decentralized service marketplace. This DApp will contain customizable auction and witness models for service provisioning and SLA violation monitoring. Specifically, we first examine different design choices and implementation options. Then we leverage a use case to demonstrate how the different roles would utilize the DApp.

Design choices

We discuss the design choices regarding the auction models, the blockchain infrastructure, and the smart contract protocols.

Auction models

In order to support dynamic business requirements, we should allow both customers and providers can be initiators and bidders. This produces a more customizable

system that aids in defining a broader range of auction possibilities. Therefore, at the highest level, the two types of auctioning that the system supports are: 1) forward auctions, where the provider is the initiator and the customers submit competing bids; 2) reverse auctions, where the customer is the initiator, and the providers submit competing bids. Specifically, AWESOME is designed to support the following eight auction models:

- **English Auction:** This is an ascending bid auction, where the price is gradually raised until only one final bidder remains, and that bidder wins the service at the finalized price.
- **Dutch Auction:** This is a descending bid auction, also known as a clock auction or open-outcry descending-price auction. The seller starts at a high price and lowers it until a bidder accepts the price.
- **First Price Sealed-Bid Auction:** In this auction, bidders submit sealed bids simultaneously to the seller, and the highest bidder wins and pays the value of their bid.

- **Second Price Sealed Bid Auction:** In this auction, bidders submit sealed bids and the highest bidder wins again, but the price they pay is the value of the second-highest bid. It is also known as the Vickrey auction, which encourages truthful bidding in terms of mechanism design [22].

We can also implement the symmetrical version of these four types for reverse auctions.

- **Reverse English Auction:** In this auction, the price is decremented by competing providers until no one bids at a lower price. The provider offering the lowest price wins the auction and provides the service at that price.
- **Reverse Dutch Auction:** This auction begins at a very low price for the service and gradually increases until a service provider accepts to provide the service at that specific price.
- **Reverse First Price Sealed-Bid Auction:** In this auction, providers submit sealed bids simultaneously to the customer. The one with the lowest bid wins and pays the value of their bid.
- **Reverse Second Price Sealed Bid Auction:** In this auction, providers again submit sealed bids, the lowest bid again wins, but the price he should pay is the value of the second-lowest bid.

Blockchain infrastructure

In general, blockchains can be permissionless or permissioned. Our modular AWESOME framework is not limited to the underlying blockchain infrastructure. AWESOME aims to build a decentralized cloud marketplace using both permissionless and permissioned blockchains. Some existing popular platforms may include:

- **Ethereum:** It is an open-source permissionless blockchain platform with smart contract and cryptocurrency functions. It provides a decentralized mechanism to handle peer-to-peer smart contract transactions through its proprietary Ethereum Virtual Machine. We are currently using the Ethereum blockchain to develop AWESOME smart contracts and DApps.
- **Hyperledger Fabric:** It is an open-source permissioned blockchain platform. The main reason for choosing a permissioned blockchain is the availability of a more efficient consensus mechanism, which increases scalability and reduces wasted resources.
- **Other promising blockchain platforms,** e.g., Hyperledger Sawtooth, Hyperledger Iroha, Hyperledger Besu, and IOTA, can also be implemented to support our decentralized cloud marketplace. These platforms have been investigated

in many research studies and commercial projects [23].

Smart contract protocols

To meet the requirements of the AWESOME model to build a decentralized cloud marketplace, we designed three smart contracts (i.e., auction contract, witness contract, and SLA contract) to support trustworthy and fair interactions between different stakeholders, as shown in Figs. 4, 5 and 6. Specifically, the auction contract is responsible for managing the auction process. The witness contract is responsible for the registration and management of auction witnesses, as well as the calculation of witness reporting results and corresponding witness fees. Furthermore, the SLA contract is used to build and manage a trustworthy SLA lifecycle. It should be noted that we leverage a contract factory to manage and generate subcontracts instead of developing different contracts separately in the AWESOME framework, as this is a more secure and efficient way [24].

The sequence diagram in Fig. 7 shows the interaction between the contract factory and different subcontracts. First, an AWESOME operator calls the contract factory to create a new auction contract. Next, an auction contract with a customized auction rule for business requirements is built to support a transparent and automated auction process. In this case, service providers can register and submit their bids for services on the blockchain. The auction contract then selects the winning providers based on the highest k bids and generates k SLA contracts for each provider. When the auction is settled (note that the services have not been delivered yet), the AWESOME operator calls the contract factory again to generate a witness contract that contains customized incentive mechanisms to encourage truth-telling witnesses. More details about a game theory-based witness payoff design are discussed in our previous research [25]. Then, different winner providers can start to deliver cloud services off-chain while the witnesses start to monitor all the services; if the QoS satisfies the requirements in the SLA contract, there is no violation; otherwise, there is a violation. The result of service monitoring is also returned to the auction contract to determine the status of the auction.

Use case demonstration

We use the business process model in Fig. 8 to demonstrate how our AWESOME DApp works. There are three parties of stakeholders who interact with the AWESOME DApp to complete a reverse auction. In this auction, the customer acts as the purchaser of the cloud service and the initiator of the auction. The providers need to compete for bids to get the right to sell their services. The entire AWESOME business process is described as follows. When the AWESOME DApp is launched, it first

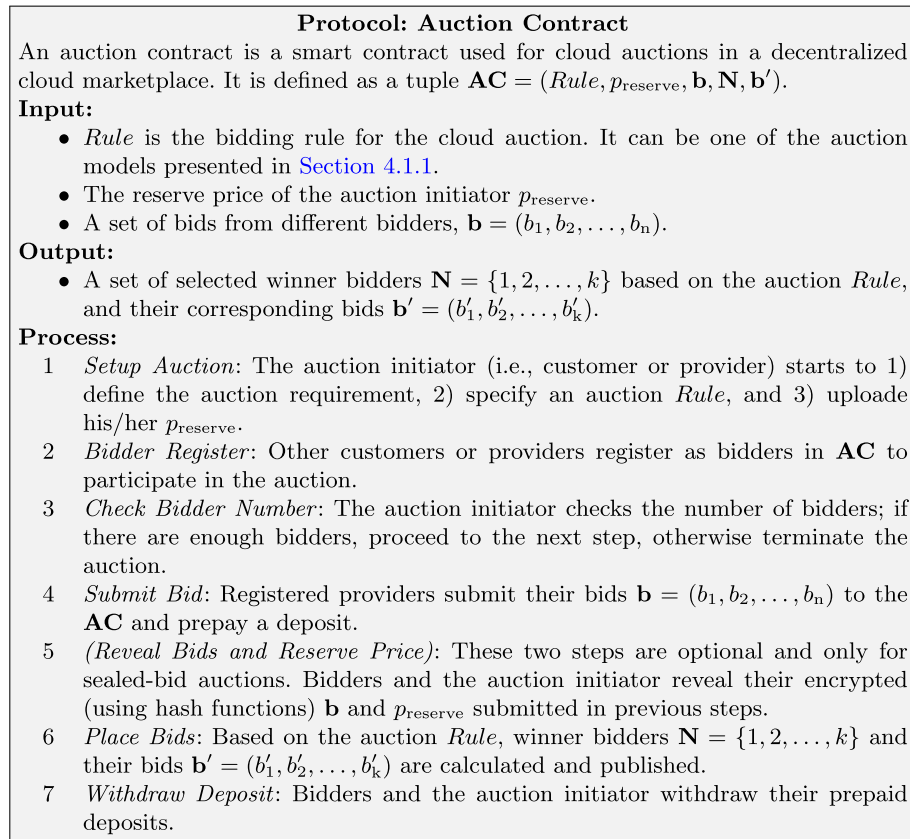


Fig. 4 Protocol: Auction Contract

invokes the AWESOME contract factory to generate the auction contract to support the auction process management. At this time, a customer can register, set up, and post an auction through the customer UI, as illustrated in Fig. 9a. An example output of the JSON data structure when setting an auction object is shown in Listing 1. This auction invitation is displayed on the AWESOME DApp to make it visible to providers. When noticing the auction invitations, different providers can register as bidders and submit their bids through the provider UI, as shown in Fig. 9b. When enough bids are received to meet the customer's requirement, the auction is settled, and the winning providers are selected.

After that, the AWESOME contract factory will generate SLA contracts to prepare for service delivery and monitoring. The customer and providers need to sign and confirm the SLA contracts in AWESOME DApp, respectively. At the same time, a witness contract is also generated. At this point, the auction initiator (i.e., the customer) needs to define the rules for witnesses, including the number of witnesses, the minimum consensus percentage required to confirm a violation, the time window for submitting reports, and the rewards and penalties for each witnesses' report. This is illustrated in Fig. 10a

and b. Then, witnesses can register and interact with the contract through the Witness UI. The cloud service is officially launched only when all sub-SLAs are confirmed, and there are enough witnesses for SLA monitoring. Next, the customer and providers perform off-chain cloud service provisioning and consumption. Witnesses perform continuous SLAs monitoring and report the monitoring results to the AWESOME DApp, as shown in Fig. 11. Based on the results reported by the witness committee, SLA violation is confirmed: when there is a violation, the service fee prepaid by the customer is refunded; while when the service is completed as agreed in the SLA, the providers can withdraw their corresponding service fees. Finally, witness fees are calculated and allocated based on the monitoring results of the witness game.

```

1 {
2   owner: 'provider1',
3   auctionState: 'active',
4   auctionID: 'auction647',
5   bids: [],
6   service: {
7     commodity: {
8       docType: 'vmInstance',
9       properties: [
10        {os: 'Ubuntu 18.04'}],

```

```

11      {storage_capacity: '32 GiB'
12      },
13      {memory: '4 GiB'}, signifies
14      {cpu_cores: '8'},
15    ],
16    pricing: {
17      pricingSubscription: '
18      cryptocurrency',
19      pricingCurrency: 'ether'
20    }
21    slaRulesAndFines: [
22      {
23        identifier: 'Availability',
24        action: 'The server will be
25        available.',
26        constraint: '99.95% of the
27        time.',
28        measurementPeriod: '12:00:0
29        0',
30        fine: '100',
31      },
32      {
33        identifier: 'Response Time'
34      },
35      {
36        action: 'The server
37        responds within 1
38        second.',
39        constraint: '85% of the
40        time.',
41        measurementPeriod: '01:00:0
42        0',
43        fine: '120',
44      },
45    ],
46    },
47    },
48    },
49  }

```

Listing 1 The JSON data structure when a customer defines an auction object

Experiments and validations

In this section, we present the evaluation and validation of the AWESOME framework. We identified two key metrics that affect the performance of our AWESOME model: latency and cost.

Latency analysis

To evaluate the performance of the AWESOME DApp and the feasibility of the model, we first simulated cloud

Protocol: Witness Contract

A witness contract is a smart contract used for monitoring the SLAs. It is defined as a tuple of three elements $\mathbf{WC} = (\mathbf{W}, F_{\text{witness}}, \varphi)$.

Input:

- A set of monitoring results $\mathbf{W} = (W^1, W^2, \dots, W^k)$ for k SLAs from m witnesses, where $W^k = \{w_1^k, w_2^k, \dots, w_m^k\}$ is the monitoring results reported for cloud SLA k .
- The payment for each witness F_{witness} , which is used to motivate witnesses to monitor the cloud services.
- The penalty function $\varphi(w)$ for witness i , which defines the penalty to be deducted from his/her F_{witness} . For a particular SLA, the closer the results reported by witness j are to those of others, the less penalty he receives.

Output:

- $\mathbf{SLA}_{\text{violate}} = (SLA_{\text{violate}}^1, SLA_{\text{violate}}^2, \dots, SLA_{\text{violate}}^k)$ is a set of SLA violation reports for k SLAs, where $SLA_{\text{violate}}^i = \{0, 1\}$ is used to denote the final result of whether SLA j is violated.

Process:

- 1 *Witness Register*: Normal blockchain users register as witnesses in \mathbf{WC} to participate in the SLA monitoring.
- 2 *Check Auction Settled*: Check the current auction state; if the auction is settled successfully, proceed to the next step, otherwise terminate the auction.
- 3 *Submit Reports*: Registered witnesses submit their encrypted (using hash functions) reports $\mathbf{W} = (W^1, W^2, \dots, W^k)$ to \mathbf{WC} and prepay a deposit.
- 4 *Reveal Reports*: Witnesses reveal their encrypted \mathbf{W} submitted in the previous step, using their private keys and the real messages.
- 5 *Calculate Witness Fee*: \mathbf{WC} calculates the $\mathbf{SLA}_{\text{violate}}$ for k SLAs and the payment of each witness based $\varphi(w)$ and F_{witness} .
- 6 *Withdraw Witness Fee*: Witnesses withdraw their corresponding witness fees.

Fig. 5 Protocol: Witness Contract

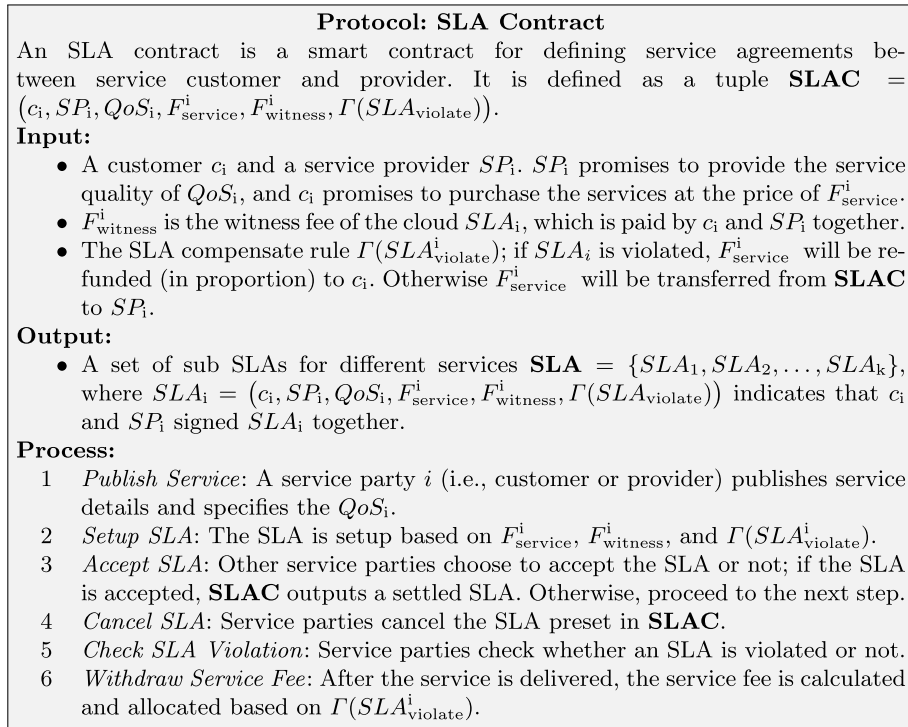


Fig. 6 Protocol: SLA Contract

auction and SLA management scenarios with different player numbers. Then, the latency is tested in the local Ethereum blockchain and calculated in two ways: 1) the response time of the AWESOME DApp; 2) the difference between the block timestamps [26]. The former reflects the latency of executing transactions in our DApp, and the latter demonstrates the transaction processing time of the underlying blockchain. In addition, we simulated two cases of blockchain mining network congestion. The “best” case implies that there are enough miners to process transactions promptly. In contrast, the “average” situation indicates seconds of delays for transaction processing due to network congestion.

Figure 12 consists of 20 plots representing the performance of 20 function interfaces in a “best” blockchain mining network. Overall, the response time of the AWESOME DApp API is a few seconds longer than the blockchain block time, which is in line with our expectations. It can be seen that the execution time of most of the function interfaces increases linearly with the number of players. One exception is *Place Bids*, which has an exponentially increasing trend. This is because this function requires on-chain calculations to place bids and therefore takes significantly more time when the number of players increases or the auction data becomes complex. Some other functions that take longer time include *Setup Auction*, *Calculate Witness Fee*, *Publish Service*, and *Setup*

SLA. Except for these functions, the latency of all others holds at a low level (10 to 15 seconds with 100 players).

Similarly, Fig. 13 shows the performance of 20 functional interfaces on an “average” mining network. It can be found that the DApp API response time and the blockchain block time have both increased significantly; the two latency values also tend to be close to each other because of the huge increase in the total time needed to process blockchain transactions. Nevertheless, all latency is maintained within a few minutes when the number of players increases. A strange observation is that the latency is high in the *Cancel SLA* function when the player number is 1. By analyzing our experimental workflow, we found that this is because he needs to process a large number of SLAs generated from the previous step. This also proves that the latency is influenced by the player numbers and the complexity of the data to be processed. In summary, we conclude that the AWESOME framework has a good performance in terms of execution latency. Some individual functions that require complex calculations have longer time delays, but they may not be time-critical in the auction process. We also conclude that the main factor affecting DApp latency is the performance of the underlying blockchain; namely, the latency is heavily dependent on the congestion of the blockchain mining network. A more detailed comparison is also presented in Table 1.

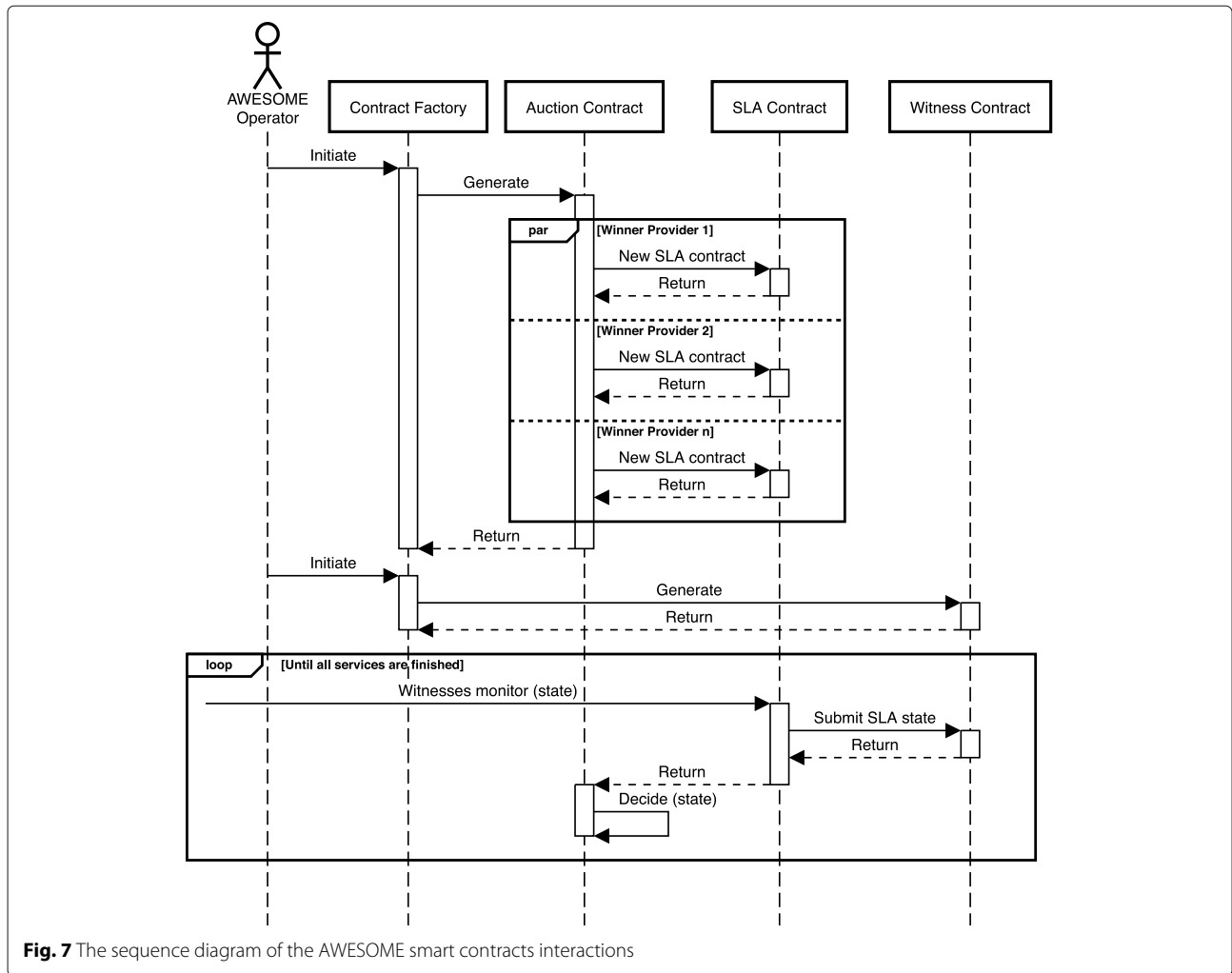


Fig. 7 The sequence diagram of the AWESOME smart contracts interactions

Cost analysis

Next, we measured the gas consumption and transaction fee (Ether) of each function interface in AWESOME smart contracts, as shown in Fig. 14 and Table 2. These functions have built-in access control mechanisms; only specific stakeholder groups can access and call them. Specifically, three transaction submission speed modes (i.e., low, average, and high) were tested⁷. By analyzing the testing results, we can find that the transaction fees of most function interfaces are maintained at a relatively low level (less than 0.01 ether), except for only three special cases, namely *Place Bids* (auction contract), *Calculate Witness Fee* (witness contract), and *Check SLA Violation* (SLA contract). These function interfaces require specific computational tasks on-chain and therefore need more transaction fees to pay for miners.

Table 3 further shows the transaction fee of each AWESOME user (converted into USD). Overall, the customer is the beneficiary and initiator of the service auction and should therefore bear more commission fees. Providers have lower transaction costs since they only join the model as bidders. Besides, the transaction fee per witness is economical (about \$15-20), which ensures that they have sufficient incentive to join the SLA monitoring activities. It is worth noting that although the customer pays over \$200 to initiate the auction, this fee is fixed and independent of the final service price. In contrast, some popular online auction platforms charge a percentage of the sale price as a commission (e.g., eBay charges 12.9% of the sale price). Therefore, our model has a price advantage, especially when the price of the auctioned cloud services becomes very expensive. On the other hand, if the commission is higher than the cost of the cloud service itself, our model will not be suitable for a public blockchain like Ethereum. A fee-free permissioned blockchain (e.g., Hyperledger Fabric), in this case, can be used as an alternative and the whole model is still valid.

⁷The estimated transaction confirmation duration for three modes are 16 minutes, 2 minutes and 19 seconds, and 30 seconds, respectively. Data was collected on April 30, 2021, at <https://etherscan.io/gastracker>

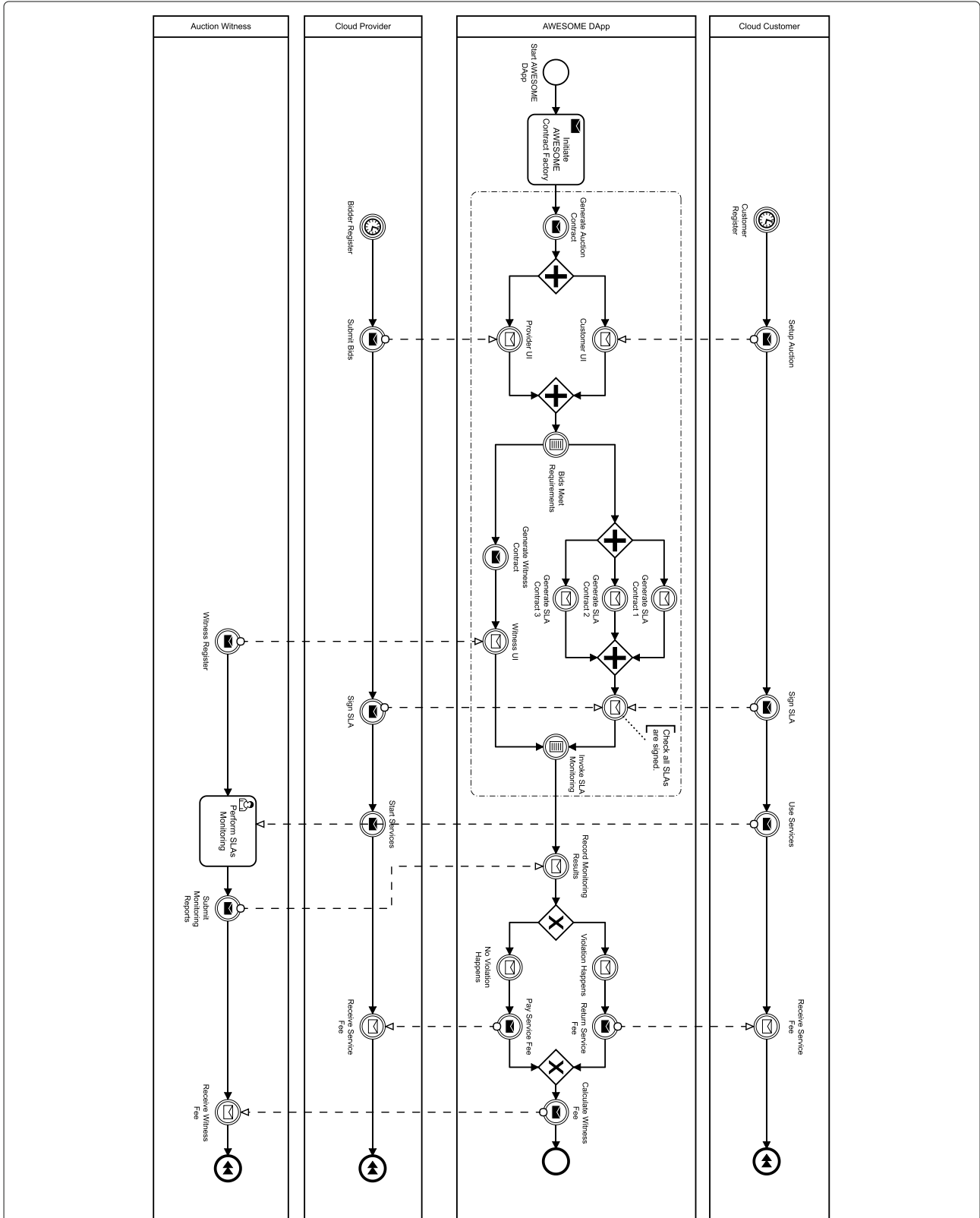


Fig. 8 The business processes in an AWESOME service trading activity

Use this form to define an auction game

Choose Bidding Type

English Auction

English Auction

Reserve Price

80 Flat Fee €

Starting Price

50 Flat Fee €

Bidding step

10 Flat Fee €

AWESOME Auction

Service 1

Use this form to define a commodity

Define commodity type

vmInstance

Add commodity property types and values

Add property

storage 60GIB Remove

Use this form to define the SLA rules and violation penalties

Add SLA rules and penalties

Add Rule

server down 120 € Remove

(a) Customer UI: Setup auction

auction458

Auction Bids

English Auction

Commodity type: vmInstance
Auction owner:
x509:./OU=org1/OU=client/OU=department1/CN=provider:/C=US/ST=North
Carolina/L=Durham/O=org1.example.com/CN=ca.org1.example.com

Service Properties

storage: 60GIB

SLA rules

server down: 120 €

Auction Pricing

Pricing subscription: Flat fee

Start Price: 80 €

Service Reserve price: 100 €

Bidding step: 5 €

Bid

Submit your bid amount for the service:

90 Flat Fee €

Submit Bid

(b) Provider UI: Submit bid

Fig. 9 The auction initiator (i.e., the customer in this auction) sets the auction rules, and bidders (i.e., providers in this auction) submit their bids

Setup witness game

Overall witness rules

Submit the number of witnesses you want monitoring your SLA:

Submit minimal witness consensus for a violation to occur:

 %

Define the length of a timed window for witnesses to submit their SLO votes:

 Hours
 Minutes
 Seconds

SLA witness rules

SLO 1

SLO 2

Submit witness monitoring rules

(a) Customer UI: Setup witness game

Define the length of a timed window for witnesses to submit their SLO votes:

 Hours
 Minutes
 Seconds

SLA witness rules

SLO 1

SLO 2

SLO 1:

Title: VM availability
Condition: It will be possible to ssh into the VM.
Violation fine amount: 150 €

Submit the reward if a witness says "no violation" and there was no violation

 €

Submit the reward if a witness says "violation" and there was a violation

 €

Submit the punishment if a witness says "no violation" but there was a violation:

 €

Submit the punishment if a witness says "violation" but there was no violation:

 €

SLO 2

Submit witness monitoring rules

(b) Customer UI: Define rewards and penalties

Fig. 10 The auction initiator (i.e., the customer in this auction) sets the monitoring rules for witnesses

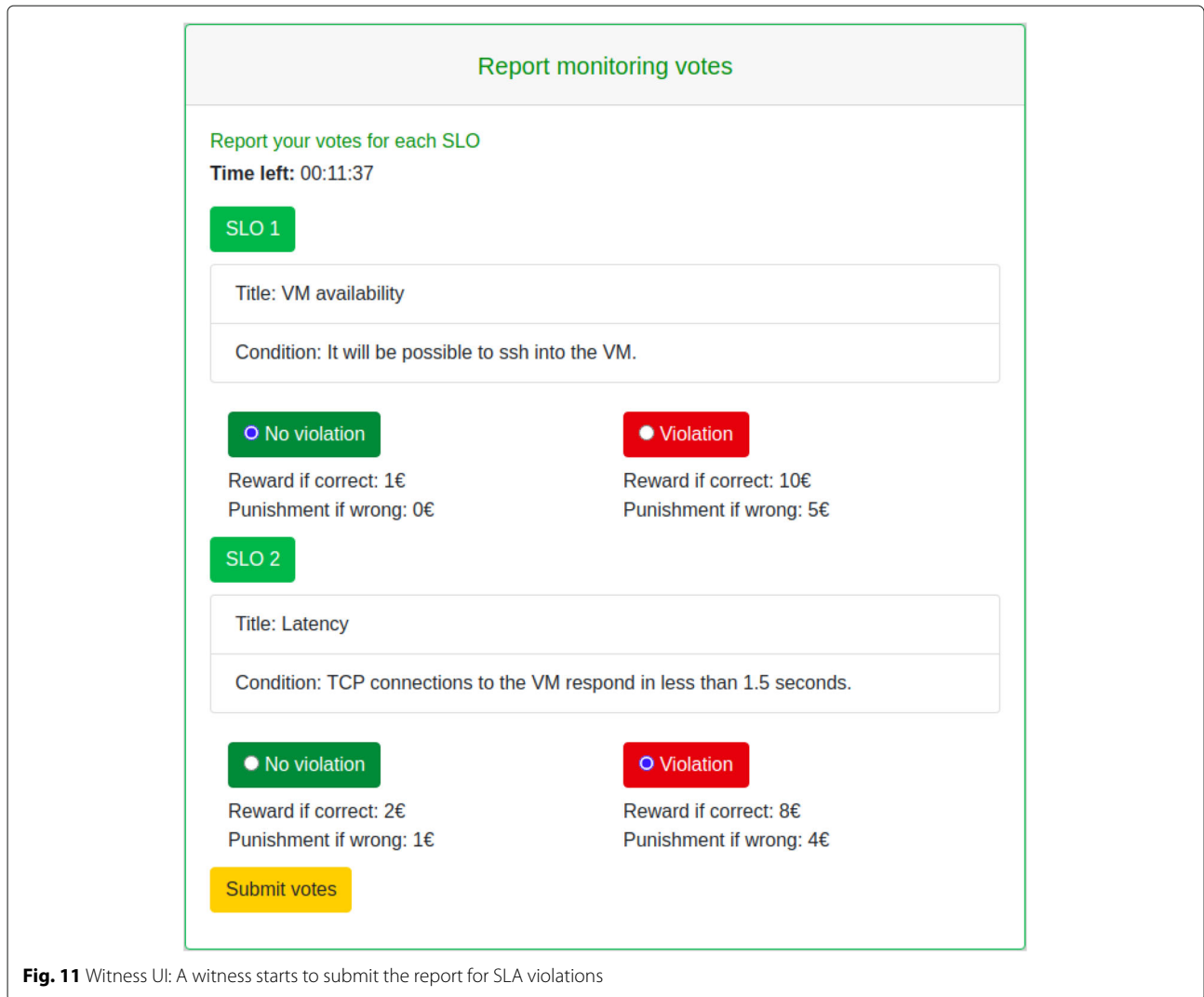


Fig. 11 Witness UI: A witness starts to submit the report for SLA violations

Discussion and future work

AWESOME aims to build a trustworthy cloud marketplace using blockchain technologies. In this section, we discuss the design choices and future development plans.

Blockchain technologies

The first consideration is the tradeoff between choosing permissionless and permissioned blockchain technologies. Permissioned blockchains can address the huge operational cost and low-performance issues of permissionless blockchains, but this is often considered at the cost of transaction security, especially in a low trust environment [27]. We used a static sealed-bid auction model to demonstrate and evaluate our AWESOME DApp in “[The AWESOME DApp demonstration](#)” and “[Experiments and validations](#)” sections. Such an auction does not require high performance and scalability of the blockchain since each bidder only needs to submit a bid once. However,

in a high-frequency and large-scale dynamic auction, the blockchain’s performance is a key factor; therefore, we believe that a permissioned blockchain is a better choice. Another consideration is to support auction payments using cryptocurrencies. Ethereum’s widely recognized token Ether can be commonly regarded as fiat money, which primarily motivated us to use the Ethereum blockchain to develop the current DApp. We leave the implementation of permissioned blockchain platforms (e.g., Hyperledger Fabric) in the AWESOME framework as our future work.

Auction models

In order to support dynamic business requirements, we design our AWESOME framework to allow both forward and reverse auctions. This gives users a more customizable system that aids in defining a broader range of auction possibilities. Introducing both forward and

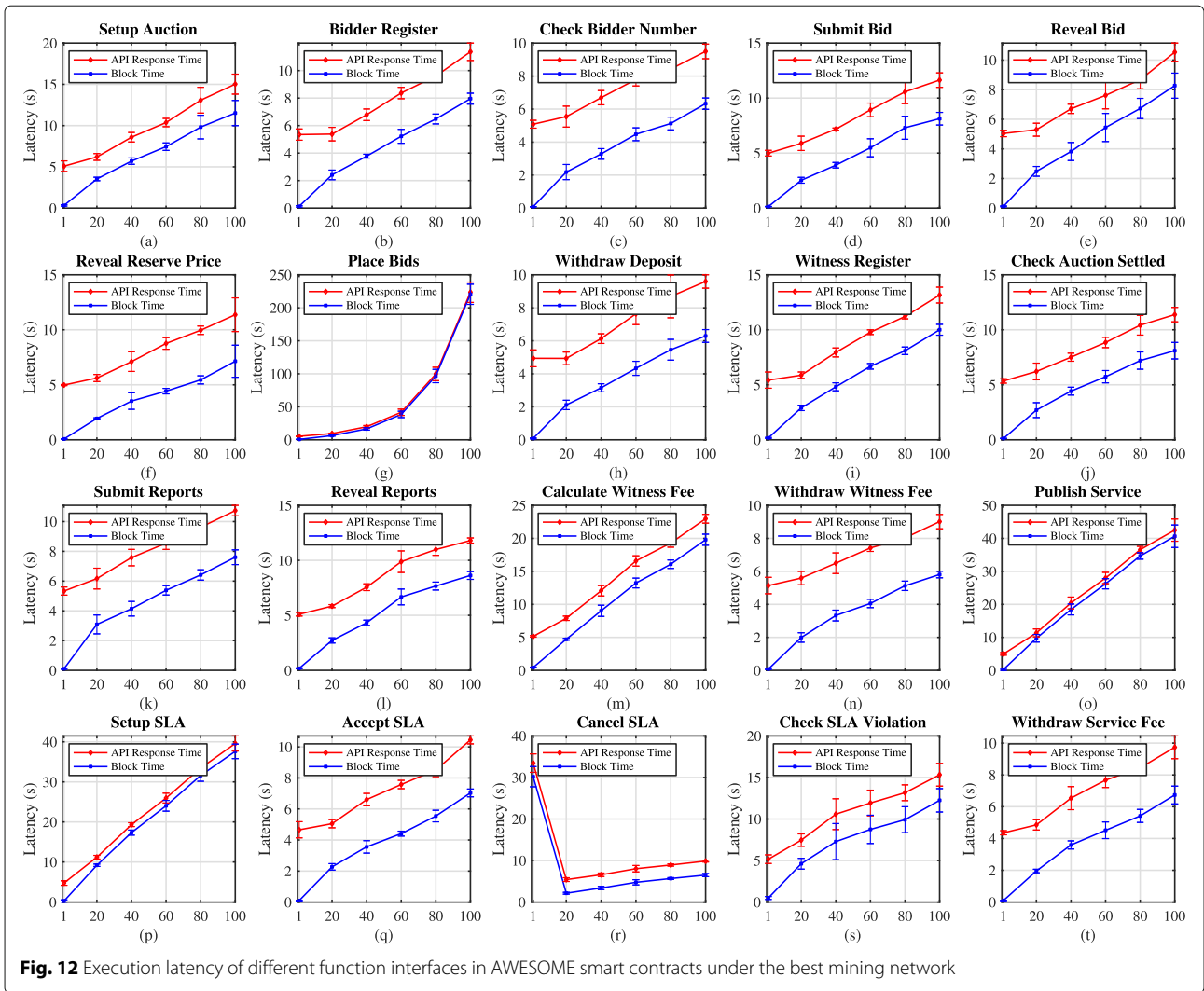


Fig. 12 Execution latency of different function interfaces in AWESOME smart contracts under the best mining network

reverse auctions also increases the possibilities for users on the DApp and promotes fairness between parties as both can use the DApp to serve their business needs in their own way. Besides, many other popular auction models, e.g., double auction, combinatorial auction, and VCG auction, have demonstrated their great potential for integration with blockchain and cloud computing [19]. We leave the implementation of these auctions as our future work. In addition to the eight auction models mentioned in “Auction models” section, AWESOME users can also customize various auction models to support different business needs. We expect a richer AWESOME ecosystem to be built with the contribution of more auction models from users.

Smart contract optimization

We do not let users deploy AWESOME smart contracts directly in the current model. Instead, users need to invoke contract templates through the DApp UI to gen-

erate auction, witness, and SLA contracts automatically. This design provides users an easy way to customize and deploy contracts while helping the DApp operator keep track of all deployed contracts. We believe such a design is reasonable and effective. Besides, the gas consumption of contracts is a huge challenge. In “Cost analysis” section we show that although the contract code has been optimized and the overall cost of the AWESOME DApp is economical, there are some functional interfaces such as *Place Bids* and *Calculate Witness Fee* may invoke a large amount of gas consumption. This will, to some extent, hinder the widespread utilization of the AWESOME DApp. To handle this issue, some off-chain solutions (e.g., state channels and trusted execution environments) can be leveraged to offload the on-chain computation tasks to off-chain networks. In addition, contract vulnerabilities should be carefully investigated in the future to prevent diverse security attacks from causing huge losses to users.

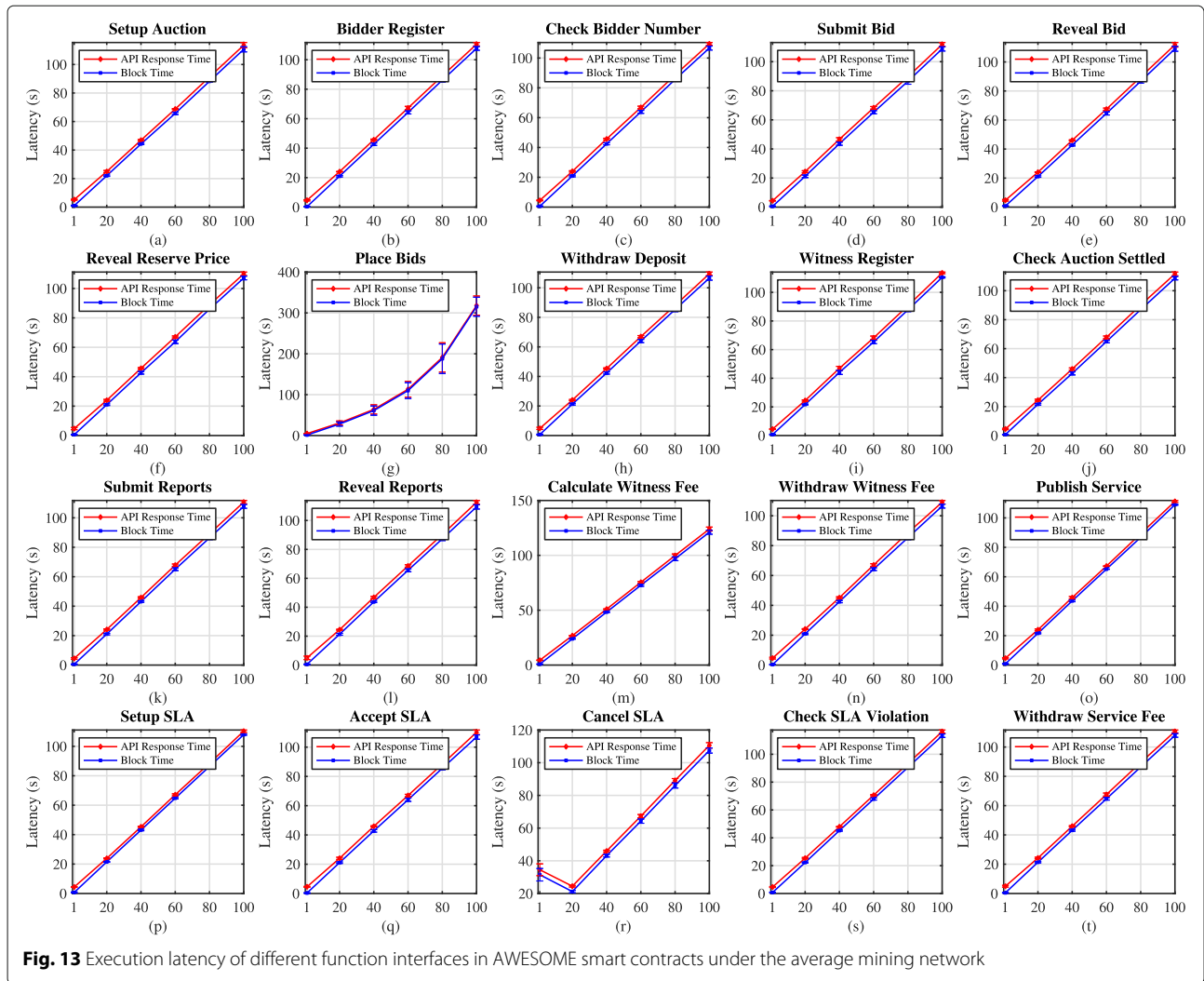


Fig. 13 Execution latency of different function interfaces in AWESOME smart contracts under the average mining network

Table 1 Execution latency of the AWESOME contracts with different player numbers (s)

Function Interface	No. of Players	API Response Time		Block Time	
		Best	Average	Best	Average
Setup Auction	1	5.081 ±0.641	5.229 ±0.319	0.326 ±0.083	1.001 ±0.345
	20	6.198 ±0.409	24.950 ±0.718	3.527 ±0.212	22.014 ±0.488
	40	8.590 ±0.572	46.920 ±0.498	5.693 ±0.385	44.281 ±0.456
	60	10.368 ±0.502	68.758 ±0.362	7.448 ±0.455	65.726 ±0.812
	80	13.077 ±1.564	91.226 ±0.421	9.821 ±1.433	88.224 ±0.703
	100	15.026 ±1.213	113.532 ±1.876	11.520 ±1.525	110.639 ±1.796
Bidder Register	1	5.346 ±0.410	4.651 ±0.431	0.128 ±0.053	0.302 ±0.170
	20	5.381 ±0.491	24.042 ±0.170	2.414 ±0.351	20.975 ±0.484
	40	6.786 ±0.420	45.686 ±0.500	3.777 ±0.140	42.668 ±0.806
	60	8.367 ±0.414	67.255 ±1.108	5.222 ±0.506	64.246 ±1.030
	80	9.603 ±0.361	89.016 ±1.284	6.480 ±0.359	86.135 ±1.408
	100	11.367 ±0.632	110.538 ±1.156	7.956 ±0.397	107.674 ±1.267
Check Bidder Number	1	5.083 ±0.238	4.511 ±0.256	0.082 ±0.009	0.564 ±0.248
	20	5.543 ±0.634	23.970 ±0.469	2.184 ±0.458	20.891 ±0.561
	40	6.690 ±0.431	45.679 ±0.541	3.282 ±0.321	42.436 ±0.643
	60	7.795 ±0.394	66.857 ±0.843	4.475 ±0.390	63.879 ±0.979
	80	8.427 ±0.228	88.656 ±1.033	5.128 ±0.387	85.632 ±1.322

Table 1 Execution latency of the AWESOME contracts with different player numbers (s) (Continued)

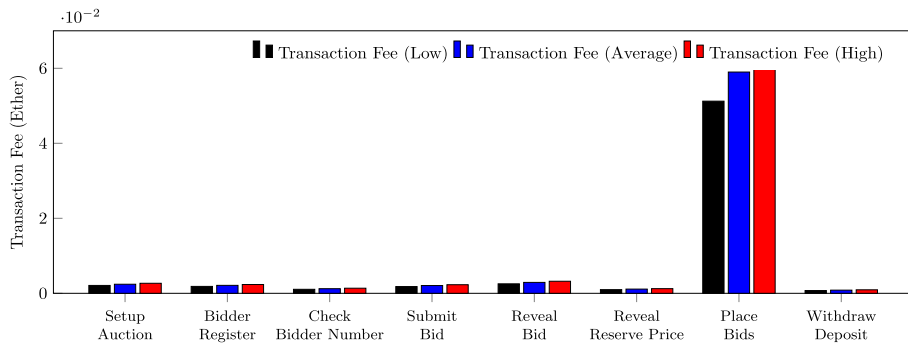
Function Interface	No. of Players	API Response Time		Block Time	
		Best	Average	Best	Average
Submit Bid	100	9.495 ±0.443	109.509 ±0.896	6.329 ±0.343	106.681 ±1.335
	1	4.993 ±0.257	4.381 ±0.171	0.136 ±0.018	0.550 ±0.276
	20	5.888 ±0.651	24.423 ±0.643	2.529 ±0.264	21.183 ±0.862
	40	7.177 ±0.104	46.639 ±1.107	3.899 ±0.249	43.606 ±1.080
	60	8.928 ±0.621	68.299 ±0.885	5.485 ±0.819	65.229 ±1.021
	80	10.562 ±1.064	89.638 ±1.507	7.304 ±1.045	86.534 ±2.021
	100	11.635 ±0.658	111.857 ±1.342	8.126 ±0.574	108.806 ±1.460
Reveal Bid	1	4.976 ±0.215	4.734 ±0.552	0.119 ±0.030	0.796 ±0.197
	20	5.617 ±0.436	24.039 ±0.252	2.483 ±0.330	21.071 ±0.513
	40	7.105 ±0.312	46.029 ±0.343	3.823 ±0.610	42.764 ±0.625
	60	8.758 ±0.909	67.481 ±0.750	5.447 ±0.950	64.534 ±1.071
	80	9.958 ±0.618	89.659 ±1.057	6.730 ±0.683	86.685 ±1.230
	100	11.368 ±0.615	111.816 ±1.576	8.270 ±0.848	109.56 ±1.695
	Reveal Reserve Price	1	5.041 ±0.050	4.723 ±0.751	0.075 ±0.006
20		5.297 ±0.323	23.947 ±0.606	1.942 ±0.058	20.987 ±0.551
40		6.707 ±0.890	45.704 ±0.533	3.519 ±0.750	42.649 ±0.875
60		7.615 ±0.528	67.066 ±0.654	4.422 ±0.251	63.721 ±1.194
80		8.674 ±0.381	88.870 ±0.785	5.449 ±0.372	85.926 ±1.080
100		10.533 ±1.529	109.980 ±1.194	7.143 ±1.455	107.125 ±1.223
Place Bids		1	5.304 ±0.540	4.753 ±0.499	0.601 ±0.278
	20	9.526 ±0.654	30.767 ±5.002	6.361 ±0.842	28.077 ±5.088
	40	19.790 ±1.887	63.894 ±11.285	16.490 ±1.792	60.981 ±11.291
	60	41.688 ±4.794	113.183 ±19.302	38.475 ±4.832	110.20 ±19.428
	80	99.762 ±10.176	191.103 ±35.886	96.615 ±10.118	187.842 ±36.20
	100	223.549 ±15.436	317.983 ±23.430	220.420 ±15.282	315.121 ±23.237
	Withdraw Deposit	1	4.939 ±0.511	4.849 ±0.752	0.085 ±0.031
20		4.934 ±0.384	24.024 ±0.400	2.112 ±0.284	21.250 ±0.582
40		6.133 ±0.294	45.342 ±0.451	3.151 ±0.246	42.295 ±0.726
60		7.639 ±0.659	66.946 ±0.730	4.333 ±0.425	63.979 ±1.063
80		8.690 ±1.298	88.097 ±1.156	5.458 ±0.629	85.291 ±1.446
100		9.597 ±0.405	109.507 ±1.151	6.293 ±0.386	106.567 ±1.403
Witness Register		1	5.425 ±0.737	4.483 ±0.263	0.171 ±0.047
	20	5.874 ±0.292	24.440 ±0.466	2.901 ±0.247	21.651 ±0.290
	40	7.941 ±0.408	46.923 ±1.383	4.818 ±0.363	44.057 ±1.394
	60	9.782 ±0.212	68.228 ±1.230	6.681 ±0.255	65.324 ±1.166
	80	11.204 ±0.212	90.606 ±0.754	8.092 ±0.342	87.740 ±1.206
	100	13.157 ±0.720	113.572 ±0.583	10.001 ±0.491	110.452 ±0.421
	Check Auction Settled	1	5.349 ±0.198	4.513 ±0.459	0.125 ±0.015
20		6.211 ±0.759	24.427 ±0.617	2.698 ±0.676	21.563 ±0.551
40		7.509 ±0.367	45.950 ±0.755	4.422 ±0.358	42.907 ±1.060
60		8.847 ±0.470	67.829 ±0.909	5.736 ±0.559	64.888 ±0.858
80		10.415 ±0.897	90.098 ±0.717	7.200 ±0.785	86.940 ±1.002
100		11.380 ±0.654	111.617 ±1.275	8.101 ±0.750	108.645 ±1.159
Submit Reports		1	5.330 ±0.273	4.519 ±0.537	0.112 ±0.037
	20	6.168 ±0.698	24.137 ±0.420	3.087 ±0.634	21.056 ±0.537
	40	7.578 ±0.552	45.851 ±0.545	4.140 ±0.490	43.093 ±0.492
	60	8.548 ±0.419	67.868 ±0.765	5.379 ±0.313	65.090 ±0.900
	80	9.619 ±0.307	89.155 ±0.863	6.414 ±0.348	86.343 ±1.143
	100	10.738 ±0.352	110.656 ±1.126	7.604 ±0.494	107.751 ±1.273

Table 1 Execution latency of the AWESOME contracts with different player numbers (s) (Continued)

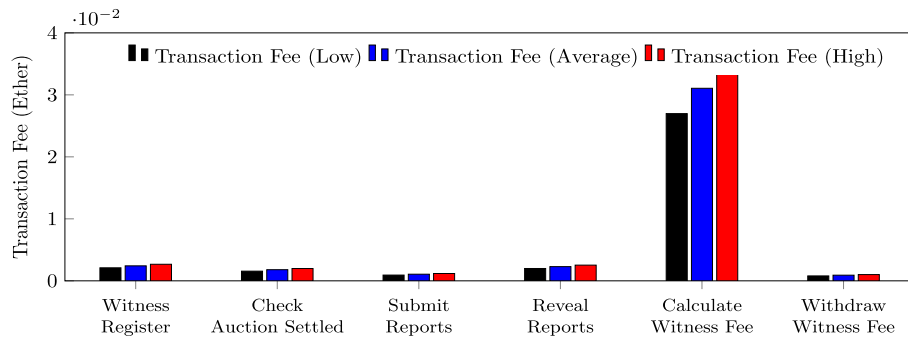
Function Interface	No. of Players	API Response Time		Block Time	
		Best	Average	Best	Average
Reveal Reports	1	5.098 ±0.163	4.983 ±1.190	0.184 ±0.046	0.514 ±0.249
	20	5.841 ±0.136	24.431 ±0.329	2.721 ±0.236	21.276 ±0.629
	40	7.559 ±0.308	46.788 ±0.794	4.324 ±0.239	43.683 ±0.485
	60	9.878 ±0.980	68.559 ±0.839	6.674 ±0.718	65.553 ±0.987
	80	10.984 ±0.525	90.275 ±1.068	7.664 ±0.360	87.288 ±1.188
	100	11.805 ±0.234	112.551 ±1.290	8.626 ±0.357	109.384 ±1.481
Calculate Witness Fee	1	5.153 ±0.119	4.424 ±0.178	0.441 ±0.065	0.847 ±0.200
	20	7.892 ±0.330	26.818 ±0.245	4.714 ±0.146	23.899 ±0.307
	40	12.065 ±0.786	50.975 ±0.244	9.023 ±0.848	48.278 ±0.493
	60	16.577 ±0.782	75.447 ±0.768	13.233 ±0.743	72.716 ±0.923
	80	19.295 ±0.630	99.818 ±1.442	16.081 ±0.637	96.850 ±1.706
	100	22.975 ±0.668	123.946 ±1.800	19.797 ±0.839	121.80 ±1.844
Withdraw Witness Fee	1	5.140 ±0.498	4.655 ±0.577	0.079 ±0.032	0.316 ±0.148
	20	5.592 ±0.405	24.114 ±0.292	1.992 ±0.288	20.910 ±0.488
	40	6.493 ±0.626	45.126 ±0.604	3.324 ±0.327	42.453 ±0.699
	60	7.419 ±0.211	67.014 ±0.862	4.055 ±0.254	64.226 ±0.876
	80	8.029 ±0.192	88.599 ±0.939	5.124 ±0.284	85.305 ±1.308
	100	9.015 ±0.435	109.059 ±1.189	5.808 ±0.199	106.464 ±1.299
Publish Service	1	5.001 ±0.405	4.657 ±0.444	0.297 ±0.280	0.841 ±0.268
	20	11.345 ±1.191	23.981 ±0.518	9.727 ±1.161	21.590 ±0.339
	40	20.429 ±1.788	45.851 ±0.784	18.449 ±1.648	43.599 ±0.568
	60	27.985 ±1.714	67.248 ±0.069	26.244 ±1.550	65.170 ±0.088
	80	36.563 ±1.033	89.267 ±0.466	34.661 ±0.990	86.791 ±0.114
	100	42.494 ±3.358	111.204 ±0.639	40.646 ±3.405	108.942 ±0.342
Setup SLA	1	4.754 ±0.556	4.411 ±0.102	0.277 ±0.278	0.719 ±0.368
	20	11.203 ±0.453	23.844 ±0.289	9.194 ±0.293	21.455 ±0.272
	40	19.328 ±0.474	45.302 ±0.347	17.334 ±0.634	43.113 ±0.500
	60	25.957 ±1.229	67.098 ±0.639	24.030 ±1.340	64.767 ±0.483
	80	33.449 ±1.503	88.470 ±0.292	31.595 ±1.440	86.324 ±0.330
	100	39.588 ±1.885	110.304 ±0.790	37.625 ±1.807	107.931 ±0.447
Accept SLA	1	4.658 ±0.521	4.556 ±0.313	0.097 ±0.027	0.324 ±0.231
	20	5.052 ±0.264	24.028 ±0.732	2.269 ±0.207	21.063 ±0.538
	40	6.604 ±0.398	45.886 ±0.388	3.557 ±0.401	42.759 ±0.830
	60	7.576 ±0.269	66.953 ±0.812	4.411 ±0.160	64.031 ±1.111
	80	8.513 ±0.426	88.380 ±0.767	5.543 ±0.373	85.537 ±0.995
	100	10.450 ±0.256	110.139 ±1.565	7.033 ±0.251	106.885 ±1.466
Cancel SLA	1	33.458 ±2.225	34.516 ±3.614	30.187 ±2.435	31.467 ±3.801
	20	5.412 ±0.451	24.460 ±0.507	2.133 ±0.219	21.189 ±0.712
	40	6.575 ±0.420	45.962 ±0.533	3.391 ±0.381	43.116 ±0.801
	60	8.039 ±0.767	67.295 ±1.002	4.765 ±0.621	64.242 ±1.275
	80	8.921 ±0.261	89.148 ±1.156	5.665 ±0.223	86.043 ±1.545
	100	9.862 ±0.215	110.630 ±1.646	6.492 ±0.377	107.441 ±1.575
Check SLA Violation	1	5.164 ±0.520	4.658 ±0.360	0.495 ±0.176	0.738 ±0.236
	20	7.458 ±0.747	25.358 ±0.532	4.616 ±0.647	22.242 ±0.362
	40	10.582 ±1.853	47.916 ±0.183	7.277 ±2.174	45.181 ±0.441
	60	11.931 ±1.544	70.610 ±0.416	8.752 ±1.703	67.853 ±0.727
	80	13.160 ±0.952	93.304 ±0.343	9.915 ±1.56	90.546 ±0.379
	100	15.327 ±1.369	116.324 ±1.104	12.248 ±1.406	113.332 ±1.241

Table 1 Execution latency of the AWESOME contracts with different player numbers (s) (Continued)

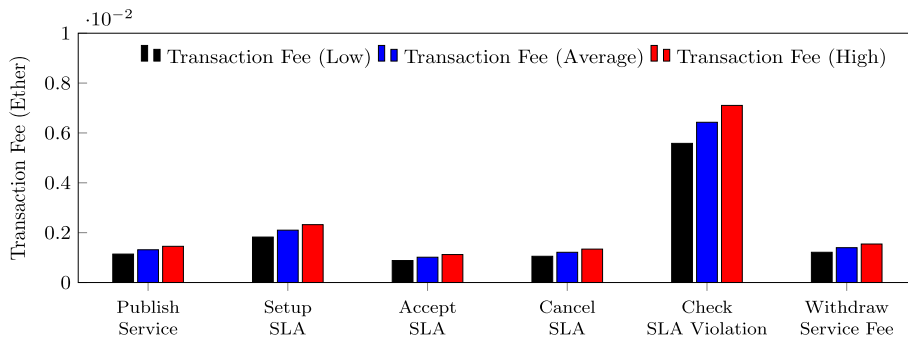
Function Interface	No. of Players	API Response Time		Block Time	
		Best	Average	Best	Average
Withdraw Service Fee	1	4.360 ±0.131	4.958 ±0.618	0.102 ±0.016	0.468 ±0.256
	20	4.857 ±0.323	24.229 ±0.515	1.957 ±0.110	21.327 ±0.422
	40	6.539 ±0.727	45.944 ±0.401	3.594 ±0.257	43.139 ±0.607
	60	7.671 ±0.464	67.570 ±1.034	4.517 ±0.524	64.640 ±0.844
	80	8.447 ±0.315	89.396 ±0.893	5.420 ±0.408	86.363 ±0.984
	100	9.737 ±0.715	110.668 ±0.963	6.735 ±0.559	107.909 ±1.227



(a) Auction Contract



(b) Witness Contract



(c) SLA Contract

Fig. 14 The transaction fee of each function interface in AWESOME smart contracts

Table 2 The gas consumption and access control mechanisms of function interfaces in AWESOME smart contracts

Smart Contract	Access Control	Function Interface	Gas Consumption	Transaction Fee (Ether)		
				Low	Average	High
Auction Contract	Customer	Setup Auction	63694	0.002101902	0.002420372	0.002675148
	Provider	Bidder Register	55835	0.001842555	0.00212173	0.00234507
	Customer	Check Bidder Number	32420	0.00106986	0.00123196	0.00136164
	Provider	Submit Bid	54555	0.001800315	0.00207309	0.00229131
	Provider	Reveal Bid	76605	0.002527965	0.00291099	0.00321741
	Customer	Reveal Reserve Price	29630	0.00097779	0.00112594	0.00124446
	Customer	Place Bids	1552698	0.051239034	0.059002524	0.065213316
	Customer & Provider	Withdraw Deposit	22242	0.000733986	0.000845196	0.000934164
Witness Contract	Witness	Witness Register	63630	0.00209979	0.00241794	0.00267246
	Customer	Check Auction Settled	47298	0.001560834	0.001797324	0.001986516
	Witness	Submit Reports	28159	0.000929247	0.001070042	0.001182678
	Witness	Reveal Reports	60330	0.00199089	0.00229254	0.00253386
	Customer	Calculate Witness Fee	817390	0.02697387	0.03106082	0.03433038
	Witness	Withdraw Witness Fee	24017	0.000792561	0.000912646	0.001008714
SLA Contract	Provider	Publish Service	34587	0.001141371	0.001314306	0.001452654
	Provider	Setup SLA	55275	0.001824075	0.00210045	0.00232155
	Customer	Accept SLA	26721	0.000881793	0.001015398	0.001122282
	Provider	Cancel SLA	31926	0.001053558	0.001213188	0.001340892
	Provider	Check SLA Violation	169151	0.005581983	0.006427738	0.007104342
	Customer & Provider	Withdraw Service Fee	36785	0.001213905	0.00139783	0.00154497

Witness mechanism

Regarding the witness mechanism of the SLA, a possible optimization solution is to perform a random and fair selection of the registered witnesses, which can effectively avoid Sybil attacks and malicious witnesses. In this regard, an unbiased screening algorithm proposed in our previous work [25] could be integrated into the current model to improve trustworthiness. Besides, a reputation system could also be designed to prevent the inclusion of disreputable witnesses. Furthermore, the current judgment of the SLA violation is implemented based on a static game model between different witnesses. In the future, we plan to design more complex and effective game models and incentive mechanisms to motivate witnesses to report trustworthy monitoring results.

Privacy concerns

All data stored on the blockchain must be public to all peer nodes to ensure traceability, verifiability, and immutability.

This conflicts with the privacy requirements of auction users, especially for those applications with critical business secrets [28]. Our AWESOME model will not be widely used if privacy and security are not adequately safeguarded. In general, blockchain-based auction models have two privacy concerns: identity privacy and transaction privacy. The former considers preventing transactions from being associated with specific auction users and their blockchain addresses. The latter covers the privacy of auction information regarding bids, contracts, payments, and other transaction details. Several privacy protection solutions are already available in this context, e.g., cryptographic techniques, mixing/tumbler service, differential privacy, trusted execution environment, and permissioned blockchain [19]. AWESOME does not restrict users from choosing auction models and application scenarios. Therefore, users should choose and implement appropriate privacy protection solutions for their specific needs in practice.

Table 3 The transaction fee of each AWESOME user in a specific auction event

User	Gas Consumption	Transaction Fee			USD		
		Low	Average	High	Low	Average	High
Customer	2628878	0.086752974	0.099897364	0.110412876	230.94335702592	265.93477475712	293.92790894208
Provider	536961	0.017719713	0.020404518	0.022552362	47.17129358304	54.31845927744	60.03619183296
Witness	176136	0.005812488	0.006693168	0.007397712	15.47330805504	17.81774866944	19.69330116096

Conclusion

This work proposes a novel AWESOME model for building a decentralized cloud marketplace, which is enhanced by auction models and witness mechanisms. The model can support interactions between service providers, customers, and witnesses to complete trustworthy auctions and SLA enforcement. Compared with classic cloud service models, our model uses blockchain and smart contracts for decentralized service auctions; transaction efficiency is ensured through customizable auction models, and auction trustworthiness is guaranteed by the monitoring of decentralized witnesses. We also developed a prototype AWESOME DApp based on the Ethereum blockchain. It contains customizable GUIs and several advanced smart contract protocols to support the SLA business process. Extensive experiments are designed to evaluate the latency and cost of our model and DApp. The experimental results demonstrate that our model is economical and feasible.

It is worth noticing that the AWESOME framework aims to develop generic software architecture for a decentralized cloud ecosystem. Some features, such as the integration with other blockchain platforms (e.g., Hyperledger Fabric) and auction models (e.g., VCG auction and double auction), are still under development. In the future, we will continue to test our framework and demonstrate its feasibility in two ongoing industrial projects (i.e., EU ARTICONF and CLARIFY).

Acknowledgements

Not applicable.

Funding

This work is funded by the European Union's Horizon 2020 research and innovation program under grant agreements 825134 (ARTICONF project), 824068 (ENVRI-FAIR project), and 862409 (BLUECLOUD project). The work is supported by the National Natural Science Foundation of China under grant No. 62102434 and No. 62002364. The work is also supported by the Chinese Scholarship Council and EU LifeWatch ERIC.

Authors' contributions

ZS, VI, and JS carried out the experiment and wrote the manuscript with support from HZ, SF, and ZZ. ZS and ZZ conceived the original idea. ZZ supervised the whole project. All authors read and approved the final manuscript.

Availability of data and materials

The DApp code and simulation data used in this paper are publically available on Github, and the links have been included in the paper.

Declarations

Competing interests

The authors declare that they have no competing interests.

Author details

¹Informatics Institute, University of Amsterdam, Science Park 904, 1098 XH, Amsterdam, the Netherlands. ²Department of Electrical Engineering and Computer Science, University of Stavanger, Rennebergstien 30, 4021 Stavanger, Norway. ³School of Computer Science, National University of Defense Technology, 410073 Changsha, China.

Received: 2 December 2021 Accepted: 20 June 2022

Published online: 19 August 2022

References

- Ibrahim S, He B, Jin H (2011) Towards pay-as-you-consume cloud computing. In: 2011 IEEE International Conference on Services Computing. IEEE, pp 370–377. <https://doi.org/10.1109/SCC.2011.38>
- Uriarte RB, Zhou H, Kritikos K, Shi Z, Zhao Z, De Nicola R (2020) Distributed service-level agreement management with smart contracts and blockchain. *Concurr Comput-Pract Exp*:e5800. <https://doi.org/10.1002/cpe.5800>
- Patel P, Ranabahu AH, Sheth AP (2009) Service level agreement in cloud computing. <https://coresholar.libraries.wright.edu/knoesis/78/>. Accessed 15 Nov 2021
- Wüst K, Gervais A (2018) Do you need a blockchain? In: 2018 Crypto Valley Conference on Blockchain Technology. IEEE, pp 45–54. <https://doi.org/10.1109/CVCBT.2018.00011>
- Xing X, Chen Y, Li T, Xin Y, Sun H (2021) A blockchain index structure based on subchain query. *J Cloud Comput* 10(1):1–11. <https://doi.org/10.1186/s13677-021-00268-0>
- 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. IEEE, pp 53–61
- Mühlberger R, Bachhofner S, Ferrer EC, Di Ciccio C, Weber I, Wöhrrer M, Zdun U (2020) Foundational oracle patterns: Connecting blockchain to the off-chain world. In: 2020 International Conference on Business Process Management. Springer, pp 35–51. https://doi.org/10.1007/978-3-030-58779-6_3
- Shi Z, Farshidi S, Zhou H, Zhao Z (2021) An auction and witness enhanced trustworthy sla model for decentralized cloud marketplaces. In: 2021 ACM International Conference on Information Technology for Social Good (GoodIT). ACM, pp 109–114. <https://doi.org/10.1145/3462203.3475876>
- Uriarte RB, Tiezzi F, De Nicola R (2014) SLAC: A formal service-level-agreement language for cloud computing. In: 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing. IEEE/ACM, pp 419–426. <https://doi.org/10.1109/UCC.2014.53>
- de Brito Gonçalves JP, Gomes RL, da Silva Villaca R, Municio E, Marquez-Barja J (2020) A service level agreement verification system using blockchains. In: 2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS). IEEE, pp 541–544. <https://doi.org/10.1109/ICSESS49938.2020.9237735>
- de Brito Gonçalves JP, Lima Gomes R, da Silva Villaca R, Municio E, Marquez-Barja J (2020) A quality of service compliance system empowered by smart contracts and oracles. In: 2020 IEEE International Conference on Blockchain (Blockchain). IEEE, pp 532–538. <https://doi.org/10.1109/Blockchain50366.2020.00077>
- Sahai A, Machiraju V, Sayal M, Van Moorsel A, Casati F (2002) Automated sla monitoring for web services. In: 2002 International Workshop on Distributed Systems: Operations and Management. Springer, pp 28–41. https://doi.org/10.1007/3-540-36110-3_6
- Sun L, Singh J, Hussain OK (2012) Service level agreement (sla) assurance for cloud services: A survey from a transactional risk perspective. In: Proceedings of the 2012 International Conference on Advances in Mobile Computing & Multimedia. ACM, pp 263–266. <https://doi.org/10.1145/2428955.2429005>
- Alkandari F, Paige RF (2012) Modelling and comparing cloud computing service level agreements. In: Proceedings of the 2012 International Workshop on Model-Driven Engineering for High Performance and Cloud Computing. ACM, pp 1–6. <https://doi.org/10.1145/2446224.2446227>
- Labidi T, Mtibaa A, Gaaloul W, Tata S, Gargouri F (2017) Cloud sla modeling and monitoring. In: 2017 IEEE International Conference on Services Computing. IEEE, pp 338–345. <https://doi.org/10.1109/SCC.2017.50>
- Anithakumari S, Chandrasekaran K (2015) Monitoring and management of service level agreements in cloud computing. In: 2015 International Conference on Cloud and Autonomic Computing. IEEE, pp 204–207. <https://doi.org/10.1109/CCAC.2015.28>
- Lu R, Liang Y, Ling Q, Li C, Wu W (2021) Double auction and profit maximization mechanism for jobs with heterogeneous durations in cloud federations. *J Cloud Comput* 10(1):1–22. <https://doi.org/10.1186/s13677-021-00249-3>

18. Dibaj SR, Miri A, Mostafavi S (2020) A cloud priority-based dynamic online double auction mechanism (PB-DODAM). *J. Cloud Comput.* 9(1):1–26. <https://doi.org/10.1186/s13677-020-00213-7>
19. Shi Z, de Laat C, Grosso P, Zhao Z (2021) When Blockchain Meets Auction Models: A Survey, Some Applications, and Challenges. Available at arXiv 2110.12534. <https://doi.org/10.48550/arXiv.2110.12534>
20. Marchesi L, Marchesi M, Tonelli R (2019) ABCDE – Agile Block Chain Dapp Engineering. Available at arXiv 1912.09074. <https://doi.org/10.48550/arXiv.1912.09074>
21. Xu X, Weber I, Staples M (2019) Architecture for blockchain applications. Springer, Cham
22. Klemperer P (1999) Auction theory: A guide to the literature. *J Econ Surv* 13(3):227–286. <https://doi.org/10.1111/1467-6419.00083>
23. Zhou H, Shi Z, Ouyang X, Zhao Z (2021) Building a blockchain-based decentralized ecosystem for cloud and edge computing: an ALLSTAR approach and empirical study. *Peer-to-Peer Netw Appl* 14(6):3578–3594. <https://doi.org/10.1007/s12083-021-01198-z>
24. Liu Y, Lu Q, Xu X, Zhu L, Yao H (2018) Applying design patterns in smart contracts. In: 2018 International Conference on Blockchain. Springer. pp 92–106. https://doi.org/10.1007/978-3-319-94478-4_7
25. 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: 2019 IEEE International Conference on Computer Communications. IEEE. pp 1567–1575. <https://doi.org/10.1109/INFOCOM.2019.8737580>
26. Tapas N, Longo F, Merlino G, Puliafito A (2020) Experimenting with smart contracts for access control and delegation in IoT. *Futur Gener Comp Syst* 111:324–338. <https://doi.org/10.1016/j.future.2020.04.020>
27. Bakos Y, Halaburda H (2021) Tradeoffs in Permissioned vs Permissionless Blockchains: Trust and Performance. Available at SSRN 3789425. <https://doi.org/10.2139/ssrn.3789425>
28. Peng L, Feng W, Yan Z, Li Y, Zhou X, Shimizu S (2021) Privacy preservation in permissionless blockchain: A survey. *Digit Commun Netw* 7(3):295–307. <https://doi.org/10.1016/j.dcan.2020.05.008>

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► [springeropen.com](https://www.springeropen.com)
