Blockchain-based SLA Management for Inter-Provider Agreements

Farhana Javed and Josep Mangues-Bafalluy
Services as Networks (SAS)
Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA) Castelldefels, Spain
email: farhana.javed@cttc.es, josep.mangues@cttc.cat

2022 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN) | 978-1-6654-7334-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/NFV-SDN56302.2022.9974899

***Abstract*—With features like immutability and transparency, blockchain and Distributed Ledger Technologies (DLT) can enable the telco industry to exchange services using smart contracts. Consequently, various 6G network stakeholders can participate in a marketplace for inter-provider agreements as either service providers or consumers. As, a blockchain-based 6G network can aid administrative domains in sharing resources (virtual network functions, services, or slices). However, such a dynamic environment requires strict Service Level Agreement (SLA) monitoring and management. Therefore, this paper considers a use case of a smart contract-based inter-provider agreement. We use novel solutions like IOTA Tangle to perform transactions, IPFS to store the hash of use case data files, and chainlink to access off-chain data feeds for SLA monitoring, reducing costs and increasing transparency. We also provide experimental evaluations of and divide the emulation into two phases. Phase 1 consists of choosing the approach and creating a smart contract (SC) (i.e., SC-Marketplace or SC-Auction). Furthermore, phase 2 consists of off-chain data feed to monitor SLA through chainlink. Finally, we measure transaction latency, response time, overall time consumption, and transaction & storage cost. The maximum latency observed in phase 1 is $\approx$ 25ms and $\approx$ 15s for phase 2. Similarly, the average response time for both phases is $\approx$ 14s $\sim$ 20s. Lastly, the results also explain that using IOTA-EVM, we can have fee-less transactions, and IPFS helps reduce the storage cost by up to $\approx$ 80%. However, it is concluded that adding chainlink adds additional cost for SLA data feeds.**

***Index Terms*—Blockchain, DLT, NFV, smart contract, SLA, Chainlink, IOTA Tangle, IPFS, inter-provider, 6G.**

I. Introduction

According to *European Vision for the 6G Ecosystem*, 6G will have to achieve many more objectives than simply delivering rapid mobile Internet access. For example, in 5G networks, *Network slicing* refers to separating a physical network into distinct logical networks known as *slices*. Each slice can be set up to provide unique network capabilities, and features.Therefore, cost and time-efficient solutions with satisfactory results can be made possible by network slicing and virtualization. For this purpose, a typical Service Level Agreement (SLA) establishes a set of Service Level Objectives (SLOs) in terms of multiple Quality of Service (QoS) metrics and specifies financial penalties for SLO violations by the service provider. Moreover, it is considered that a 6G network will provide dynamic multi-domain network slicing with numerous inter-provider agreements [1]. Therefore, 6G networks are expected to substantially increase the number of stakeholders involved in a given service offering due to softwarization. In this direction, deploying multi-domain network services requires using advanced solutions, such as *service federation*. *Federation* is the process of orchestrating services or resources across several domains in a multi-domain scenario [2].

An inter-provider SLA agreement is needed to support the orchestration of network slices across multiple administrative domains for secure and trustful processes. In this context, service providers are looking for new models (a platform or an Network Function Virtualization (NFV) marketplace to sell, lease or purchase resources and services with defined SLA agreements) where the on-boarding of assets and resource sharing can be performed in an automated and cost-effective manner at the same time providing ways for accountability with transparency for these SLA agreements. However, the third party must ensure both sides' openness and reliability as there are gaps in the literature regarding SLAs of 6G-enabled use cases. In addition, there is a lack of SLA monitoring and penalty assignments in case of SLA violations. Therefore, automatic SLA monitoring, verification, and enforcement are challenging for both sides (providers and consumers) to implement to prevent trust deficiencies.

In this regard, Distributed Ledger Technologies (DLT)/blockchain has the potential to address the difficulties associated with automating SLA monitoring and compliance [3]. Therefore, the contributions of this paper are to propose DLT as a solution to support and provide trust in these open markets using DLT as SLA monitoring and management. Furthermore, on top of the automation and SLA management, blockchain and smart contracts remove the need for costly intermediaries resulting in transparency and trustworthiness for inter-provider agreements. Another contribution of this paper is to consider a 6G network use case of inter-provider agreement for multi-administrative domains, including registration and service offering using two approaches (i) marketplace advertisements and service discovery and (ii) reverse auction. For this use case, we aim to leverage solutions that support the current limitation of Ethereum while keeping the concept of decentralization alive. Our solution combines IOTA Tangle and InterPlanetary File System (IPFS) to minimize the overall cost for 6G use cases. For SLA monitoring, we use DLT-based Oracle, i.e., chainlink oracle. Chainlink oracles help bring the off-chain data to Decentralized Applications (DApps) and smart contracts. Therefore, the main contribution of this paper is to

propose a DLT-based automatic SLA monitoring system for ensuring transparency and trustworthiness for inter-provider agreements. We leverage the decentralized characteristics of DLT for SLA monitoring. Also, we propose an approach for SLA monitoring that can facilitate trustworthy SLA enforcement. We leverage DLT's inherent capabilities to achieve an immutable record of services which can be helpful for complete a posterior audit data trials and log violations against SLA. Additionally, we evaluate the proposed system for satisfying SLA, specifically for response time and added latency.

The rest of the paper is organized as follows: Section II presents the related literature, and III discusses the proposed approach. Section IV showcases the evaluations performed, and finally we conclude at the end in Section V.

II. Related work

Recently, there has been an interest in SLA monitoring for emerging 6G network requirements. In this regard, a DLT-focused resource reservation idea is the blockchain network slice broker [4] based on [5]. The authors present an idea where the consumer (tenants) can request network services from a provider (Mobile Network Operators (MNOs). The SLAs for the allocation are recorded to a distributed ledger through smart contracts. However, the actual resource usage at a device is not recorded, and the problem of SLA monitoring in network sharing is not addressed. An idea presented in [6] is the extension of Network Slice Broker (NSB) [5]. It provides a blockchain-focused architecture for network slice auctions. In this work, infrastructure providers allocate network slices through an intermediate entity, the intermediate broker, which further gives resources to tenants. However, NSB does not discuss the problem of SLA monitoring. In [7], the authors present an End-to-End (E2E) architecture for network sharing based on blockchain and smart contracts. The authors focus on permissioned distributed ledger; also, the process of SLA monitoring is not addressed.

However, it may be unclear what party should be trusted as the SLA management authority after establishing an SLA [7]. This question becomes even more pertinent where failures are less tolerable. Therefore, we suggest that no single entity should have sole authority over SLA lifecycle management because of numerous critical concerns, including breach, conflict of interest, single point of failure, and lack of awareness of E2E ecosystem requirements [8].

Therefore, the main contribution and focus of this work are to enable SLA monitoring, where the data for SLA monitoring is accessed off-chain and stored on the blockchain and used for SLA enforcement. Furthermore, chainlink off-chain brings automation and transparency, which is critical in SLA monitoring. Moreover, effective SLA monitoring and enforcement mechanisms are required. In the context of inter-provider agreements, traditional SLA practice can be inefficient due to the limitations, including less or no transparency, manual SLA enforcement, and trust in a single authority. Such restrictions can impair SLA compliance. To address this matter, we believe any improvement to current SLA practice can integrate blockchain and DLT as a solution. We use novel solutions like chainlink to enable off-chain data access and reduce the overall transaction cost using IOTA-EVM. We also propose to use IPFS to minimize the storage cost of the inter-provider agreement.

III. Blockchain-Based SLA Management for Inter-Provider Agreements

In inter-provider agreements or multi-administrative use cases, a domain usually outsources resources from different providers to meet end-user demand. Therefore, the SLA plays a vital role in inter-system orchestration and mediation for inter-provider agreements [9] and a monitoring and enforcement mechanism for SLAs is required. However, in the context of inter-provider agreements, typical SLA practice might be inefficient due to limitations such as little or no transparency, manual SLA enforcement, and reliance on a sole authority. These constraints may hinder SLA compliance.

Consequently, the integration of blockchain and DLT can increase transparency and automation. Therefore, we use the blockchain's audibility and self-enforcement characteristics to impose accountability and responsibility [9]. Blockchain and DLT can play an active role, allowing activities to be recorded on a shared ledger in the form of transactions. These transactions must adhere to the defined SLA terms. In addition, a blockchain oracle (i.e., chainlink) can obtain the SLA metrics from the off-chain source, facilitating SLA monitoring.

The convergence of two technologies, such as blockchain/DLT and service federation, frames the complexity of a phenomenon that requires a deep understanding of involved technologies and a clear comprehension of how the underlying building blocks could work together for Blockchain and DLT-based 6G network use cases. Therefore in Figure 1, we begin by illustrating the definition of two high-level technological stacks as a stepping stones to understanding the characteristic of the underlying building blocks. The building blocks are as follows (i) A Domain (provider or consumer), (ii) Decentralized Application (DApp), (iii) Oracle, (iv) Smart Contract (SC), (v) IOTA Tangle, and (vi) Ethereum blockchain. Here the initial block, *Domain*, is AI/ML platform and integrated into the management and orchestration (MANO) workflow described in the 5Growth project[1]. We aim to extend this framework by adding the blockchain for SLA management.

a) Framework and Workflow: In this paper, we examine a use case in which blockchain and smart contracts support inter-provider agreements for 6G networks using a market-place and auction mechanism. In addition, this inter-provider agreement utilizes a blockchain-based oracle for SLA management. Figure 2 illustrates the interactions or transactions (a transaction is one of the primary activities that plays a crucial role since it can change or update the state of the smart contract) involved in this use case. These components include:

[1]https://5growth.eu/

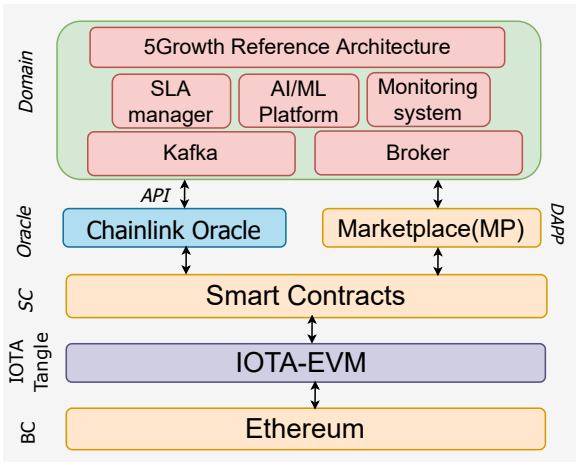

Fig. 1: Proposed building blocks for Blockchain/DLT-based SLA monitoring

Domain$_j$ as Provider (*P*), Domain$_i$ as Consumer (*C*), Broker (*B*), Marketplace (front-end) (*MP*), Smart Contract (*SC*) (a) SC-Auction, (b), SC-Marketplace, Chainlink Oracle (*CLO*), IOTA-EVM (IEVM), Ethereum, and IPFS.

We employ the blockchain-based marketplace referred to above as *Marketplace* (MP) to enable consumers (C) and service providers (P) to engage in buying and selling services and forming agreements based on requests. In addition, we use Ethereum and its smart contracts functionality as the foundation for a decentralized and transparent platform for multi-party or inter-provider negotiation between a growing number of stakeholders (e.g., Virtual Network Function (VNF) providers, multi-administrative domains) that provide an E2E network slice to consumers (e.g., verticals).

One of the components of our use case to automate the process of inter-provider agreements is *Broker* (B). The network slice broker concept depends on the service provider's capacity to automatically negotiate network slice requests from external tenants based on the current service/resource availability of the service/infrastructure provider [5].
Furthermore, We create numerous agreements for specific demands and scenarios using smart contracts, as shown in Figure 2 and listed above as *SC-Auction* and *SC-Marketplace*. *SC-Auction*, is when the consumer wants to start a reverse auction, and consequently, the provider will submit bids to the smart contract. The smart contract will choose the provider, and the winner will be notified and further provide the services. Additionally, an SLA smart contract will be created. On the contrary, for *SC-Marketplace*, the providers will publish their available services on the open marketplace for consumers to see. A consumer can choose the services listed on the marketplace from a specific provider. The provider can provide the services, and a new SLA smart contract will be created for *SC-Marketplace*.

Lastly, in our use case, we employ *IOTA-EVM* and *IPFS*. As previously mentioned, Ethereum has performance difficulties, such as transaction and gas fees. A *transaction fee* is a cost to Ethereum to accomplish a particular action. Due to Ethereum's Proof-of-Work (PoW) consensus mechanism, the Ethereum gas charge is a concern. Therefore, we employ the IOTA-EVM Tangle [10] shown in Figure 2 to address this issue. Direct Acyclic Graph (DAG)-based IOTA Tangle is a distributed ledger. IOTA can assist in the execution of Ethereum smart contracts on a network without transaction fees as of October 2021. Therefore, we merge IOTA with Ethereum for our use case resulting in a cost-effective solution. Similarly, IPFS further reduces storage costs. Thus we employ it to lower storage costs for the use case [11].

In the case of SLA monitoring, the smart contract can not access the off-chain data. For that, we aim to use Chainlink Oracle (CLO). Chainlink enables smart contracts on blockchain to access off-chain data feeds from any external Application Programming Interface (API) source in a highly tamper-resistant and reliable manner.

The workflow of the proposed use case in Figure 2 is explained further. We consider both sides, the provider side (P) on the left and the consumer side (C) on the right. In addition, we illustrate the interaction with the blockchain in the center.

On the left side of the Provider (*P*), the process starts with OSS/BSS as (step 0) to create offers in the Figure 2. Next, the Broker registers the provider domain on the blockchain through a front-end (step 1). Once registered, the Broker requests to store and publish service offers on the blockchain (step 2). Afterward, the process is addressed by Marketplace (MP). First, the MP triggers an update SC (step 3) to store service offers. Next, the SC component stores the service offers files in the IPFS (step 4) and hash these files in the Ethereum (step 5). Once the process is completed, the IOTA-EVM component (*IEVM*) adds this transaction to the ledger (step 6), and *SC* can notify that the offers have been published (step 7).

As mentioned above, this use case aims to provide the consumer (i) select and order from the listed services or (ii) initiate a request for submissions of bids and start an auction to address the consumer's needs. Therefore, we consider two scenarios generating two different smart contracts: 1) SC-Marketplace and 2) SC-Auction. On the right side of the Consumer (*C*), the process starts with BSS/OSS as step 0. This new request can be for resources listed on the marketplace, VNFs, or a slice. *Broker (B)* registers the consumer domain on the blockchain through a front-end as (step 1)

First, we consider scenario 1 for SC-Marketplace. Once registered, Broker selects one of the available services on the listed service and places an order (step 2). The (*MP*), records the request to *SC* (step 3). The SC component generates a smart contract for the specific request arrival (step 4). Once the new SC-Marketplace is created, the (*IEVM*) adds this transaction to the ledger (step 5), and *SC* can notify that the services are running (step 6). The SLA monitoring (step 7) phase starts when the services run. First, SC sends a request to Chainlink Oracle (CLO) for data feeds. Next, the *CLO* gets

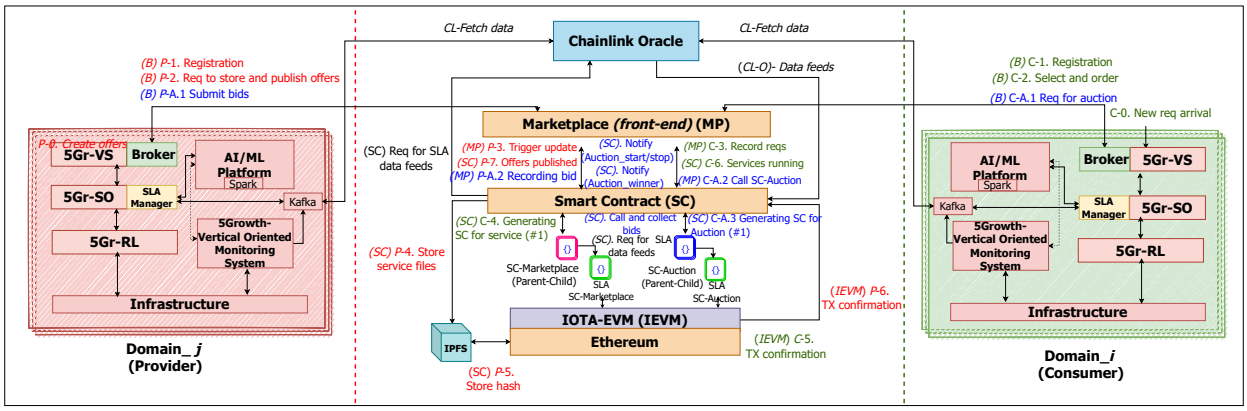

Fig. 2: Functional Blockchain/DLT-based SLA management for inter-provider agreement

SLA data feeds from Kafka, which is accessed through an API as illustrated in Figure 2. Afterward, these data feeds are sent to a smart contract, where they are compared to check for SLA compliance. Once the consumer *C* accepts the provided services and the SLA requirements are met, the agreed amount then can be transferred to the provider *P* (step 8).

Second, in scenario 2 for SC-Auction, the process is different from SC-Marketplace. After step 1 of registration, Broker (*B*) requests to start an auction (step 2). And the (*MP*) calls the SC for auction (step 3). The *SC* generates a smart contract for auction (step 4). Once the *SC* notifies the registered providers that the auction is started (step 5), the providers can submit their bids (step 6), and *MP* will record these bids in the smart contract (step 7). These bids will be collected and compared until the auction is stopped and a winner is declared and notified, as shown in Figure 2. Once the winner is notified, a new contract will be generated, and the provider *P* will start to run services for the consumer *C*. (step 8) *CLO* gets data feeds in any of the scenarios, such as for a marketplace or auction, which is later sent to SC as *data feeds* for SLA compliance. Once satisfied, the agreed amount will be transferred to the provider *P* from consumer *C* (step 9).

IV. Performance Evaluation

In this section, we evaluate our framework and provide the characteristics of the evaluated setup. We assume a scenario where the operations (transactions in the blockchain) will be performed between two parties; in our scenario, we consider two main stakeholders, i.e., the consumer and provider. Both consumer and provider are registered separately through a smart contract on an Ethereum network. This smart contract is deployed on an Ubuntu 14.04 with 2GB of RAM and 8GB of disk memory. The Ethereum blockchain uses solidity to write smart contracts. The Remix is used to test the proposed framework's performance. The following parameters characterize the emulated scenarios: the number of domains $D \in \{$consumer $Domain_i$, provider $Domain_j$ $\}$ increasing from 2 to 4 and the total number 300 of transactions. We are considering two phases: *Phase 1* starts after the domains have been registered in the blockchain, then the consumer either selects services (SC-Marketplace) or starts an auction (SC-Auction), and until the services start to run. *Phase 2* starts when the smart contract (SC-Marketplace or SC-Auction) sends requests to chainlink for SLA data feeds. In phase 1, we use IOTA-EVM, and for phase 2, we use Kovan testnet, which is supported by chainlink for data feeds.

Below we compare and analyze phase 1 and phase 2 in terms of latency, response time and transaction & storage cost.

A. Transactions latency

The transaction latency is the time interval since a transaction T_x is submitted until it is confirmed and available on the blockchain.

We are calculating the latency for phase 1 and 2 using the following expression: $Latency = T_{confirmation} - T_{submission}$. Once domains are registered in the blockchain, phase 1 will start, where the consumer will either buy a service listed in the marketplace or start an auction for a specific service request. The latency for phase 1 and phase 2 is shown in Figure 3. Here in phase 1, we send requests from the consumer for 300 transactions to purchase resources from the provider. Figure 3a shows the latency observed in the first phase of the use case as explained above. The transactions are performed using IOTA-EVM during phase 1. Because of its consensus mechanism, the IOTA-EVM experiences less latency, as illustrated in Figure 3 where the y-axis of 3a shows the latency observed in ms. Moreover, the average latency observed for marketplace contract (buy a service from the listed services) and auction contract (starting auction for a specific service request) is between 15ms to 25ms, while the number of domains D increased from 2 to 4. However, during phase 2, once the smart contract (for auction or a marketplace process) is confirmed and the SLA contract is created, the services start to run. The smart contract needs data feeds from off-chain for SLA monitoring. For that purpose, we use chainlink. We send a total of 300 transactions to request data feeds in both smart contracts (SLA-Marketplace and SLA-Auction). Therefore, we illustrate the added latency when smart contracts send a request to CLO for data feeds as explained in the workflow above III-0a. In Figure 3b, we illustrate the latency observed for both smart

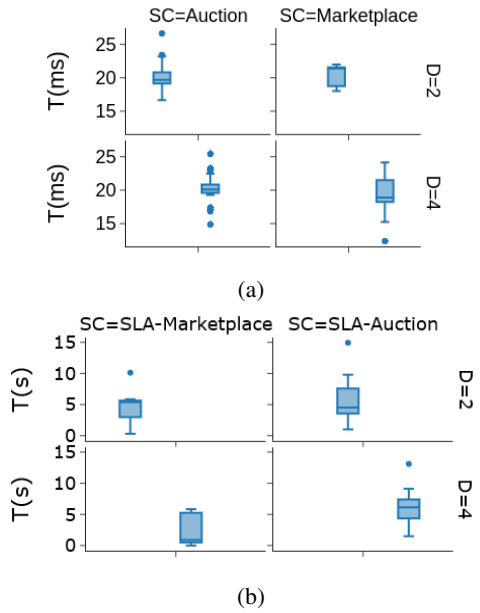

Fig. 3: Analysis and comparison of latency observed in Phase 1 (a) and Phase 2 (b)

contracts (SLA-Marketplace and SLA-Auction). It is observed that the chainlink adds additional latency as it uses Kovan - an Ethereum testnet. The average latency is observed and increased due to the consensus mechanism of Kovan. Kovan uses Proof-of-Authority (PoA), which affects how the transactions are confirmed and added to the blockchain. In summary, the total latency of phase 1 is lower than phase 2. It is orders of magnitude below the native Ethereum testnet (Kovan) with IOTA-EVM is (s vs. ms). Meanwhile, increasing the D from $D = 2$ and $D = 4$ adds slight latency. Additionally, the overall time taken in the process is discussed and illustrated later.

B. Request arrivals and response time

In section III, we explain that the consumer can initiate a new request. The provider has the option of accepting or rejecting requests. We define the response time as the elapsed time between when the consumer sends a request and when the provider agrees with the request, and the service begins running. To monitor this, we start 10 to 30 requests from the consumer side to determine how long it takes for the service provider to respond. To calculate the response time we use: $R_T = Latency + t_{waiting}$. Here R_T is the sum of $Latency$ measured earlier, and $t_{waiting}$. Here $t_{waiting}$ is the time provider *P* will take to respond back to the consumer *C*. The $t_{waiting}$ consists of the moment the provider receives notification that a new request has arrived and the decision Provider P will make regarding whether to accept or refuse the request. In phase 2, the RT is calculated as well, but here the $t_{waiting}$ is the time it takes the CLO to respond to the SLA smart contract with data feeds. Figure 4 and 5 illustrates the RT observed in the scenario for a marketplace and auction. The 4 shows RT for phase 1 and 2 when $D = 2$ and $D = 4$.

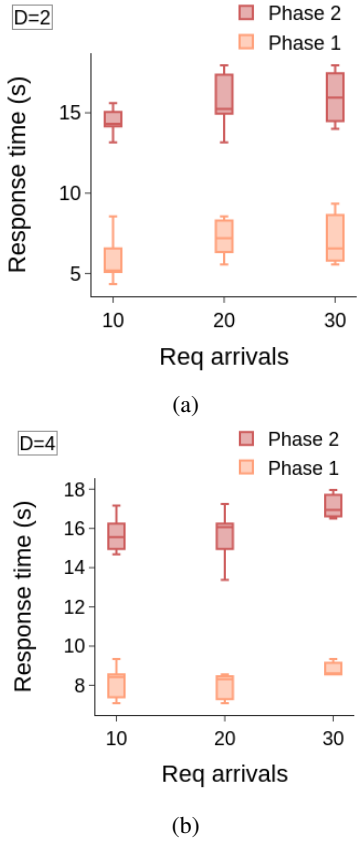

Fig. 4: Analysis and comparison of R_T for requests arrivals for SC-Marketplace (a) where $D = 2$ and where $D = 4$ (b)

The RT increases slightly when $D = 4$ for phase 1 and phase 2 since the number of D increases hence overall RT increases. Similarly, the 5 shows RT for phases 1 and 2 when D is 2 and 4. In this case, the RT increases slightly when $D = 4$. However, the latency and RT are not significantly increased in phase 2. The testnet used is Kovan, and it does not add significant latency compared to other testnet (i.e., Ropsten). The maximum RT observed for phase 2 is $\approx$ 20s when $D = 4$ in case of the auction and $\approx$ 18s for a marketplace. However, for phase 1, the average RT is less than $\approx$ 10s for marketplace and auction.

C. Transaction and storage cost

In our use case, the transaction cost is calculated in *ETH* using the formula: $Cost_{eth} = Cost_{gas} * gas_{price}/10^9$. The cost is calculated for both of the phases. We use USD as a homogeneous way of comparing costs from different sources. Therefore, we used the formula: $Cost_{USD} = Cost_{eth} * CurrentPrice_{eth}$ to calculate the cost in *USD*. In phase 1, we deployed our solidity smart contract using IOTA-EVM, and following its deployment, we examined the transaction details. As a result, we observed through the IOTA explorer that IOTA has no transaction fee due to how IOTA Tangle operates since IOTA reaches consensus on the authenticity of transactions without the participation of miners [10]. However,

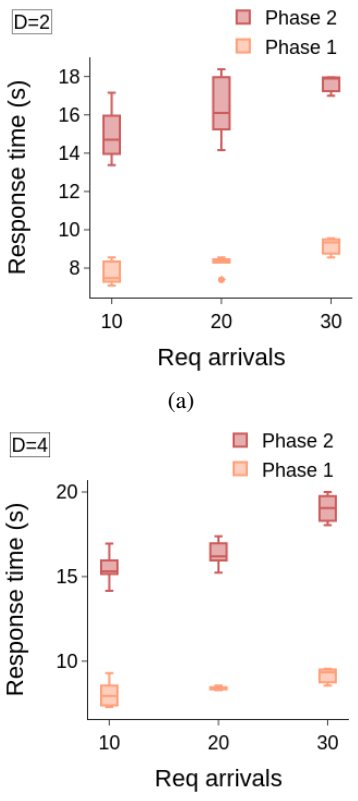

Fig. 5: Analysis and comparison of R_T for requests arrivals for SC-Auction (a) where $D = 2$ and where $D = 4$ (b)

in phase 2, we use chainlink for SLA monitoring and data feeds. For evaluation, we use Kovan testnet. Each request on the testnet costs 0.1 LINK (cryptocurrency of chainlink), which, according to the current market price, can be up to $\approx$ 0.63 USD. This means that sending up to 100 requests can cost up to $\approx$ 63.59 USD. Also, this cost is apart from the gas cost on the Kovan testnet. The average cost for 10, 20, and 30 requests for data feeds were $\approx$ 0.132262 ETH, 0.141100 ETH, and 0.145800 ETH, respectively. Therefore, the total price in USD can be $\approx$ 478.18. To further reduce costs, we intend to use IPFS. For emulation and fair comparisons, we save data in two ways: (1) on Ethereum and (2) on IPFS, with the hash stored on the blockchain. The data in the 6G networks' use case data includes service offers, negotiations, or other relevant data an operator may want to store. It is observed that cost can go up to $\approx$ 800 USD for storing 100KB of data.

Then, we save similar relevant use case files on IPFS for comparison. IPFS hash can be stored in a variety of formats. To further investigate the cost, we save it in three alternative methods: a string, an event log, and a struct. As a result, the cost of storing hash on the blockchain can be drastically decreased. The average price in ETH recorded as s a string, struct, or event is 0.0018, 0.002, or 0.0004, respectively. Furthermore, we observe that the maximum cost for storing the hash as a string is $\approx$ 3 USD. Therefore, we employ this

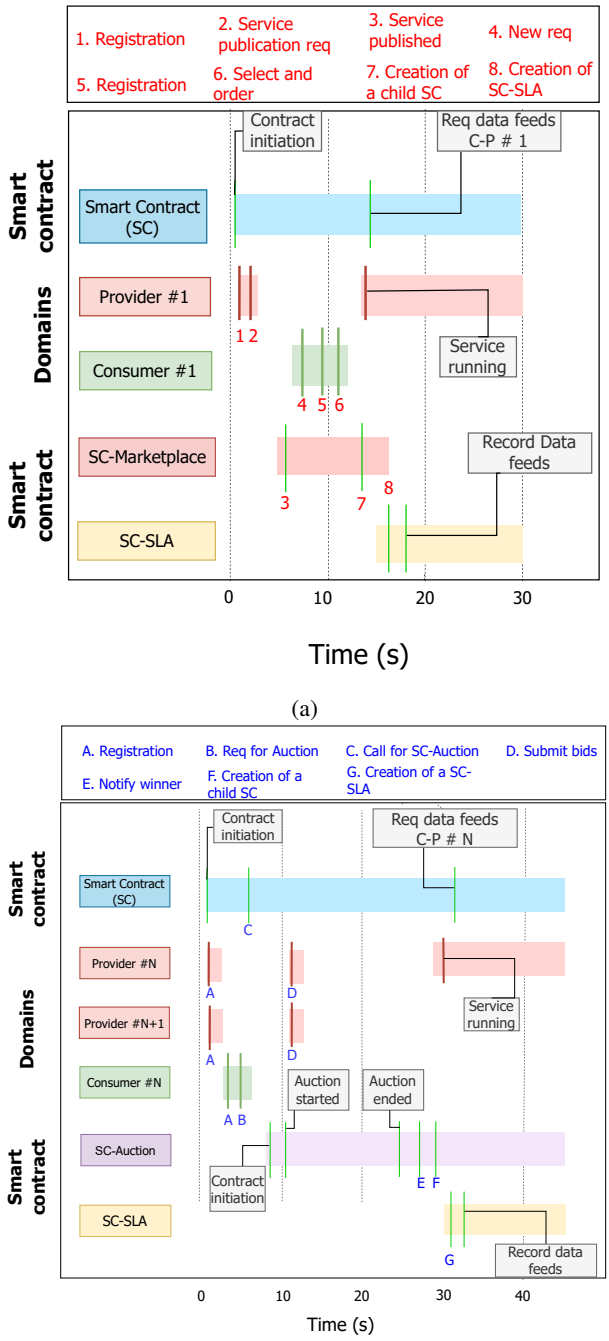

Fig. 6: Analysis and comparison of one completed service in time (s) for SC-Marketplace (a) and SC-Auction (b)

way to store the data a provider may like to store on IPFS, resulting in a cost savings of up to 80 percent in our use case.

D. Time consumption

As explained above, using $Latency$ and R_T, overall time consumption can be calculated for total completed services. A *completed service* can be defined as all the transactions performed, considering the steps mentioned in Section III for

both SC-Marketplace and SC-Auction. Also, SLA monitoring starts when the smart contract sends requests for SLA data feeds. Figure 6 shows the event plot of 10 averaged experiments for the three domains: the consumer, provider #1 for marketplace 6a, and additional provider #N+1 for auction in 6b. We also consider all the smart contracts created for these services. For example, it takes $\approx$ 18s until the CLO starts to send data feedback to record and monitor SLA in the case of SC-Marketplace. As for the auction, it takes $\approx$ 30s in SC-Auction to get data feeds back for SLA monitoring. All these steps are mentioned in Section III for both marketplace and auction separately.

E. Discussion

To conclude the performance evaluation above, the DLT-based inter-provider agreements can be complicated because of the stringent requirement for B5G networks. For instance, the number of stakeholders will increase for B5G networks; therefore, novel blockchain architectures, techniques, and consensus algorithms are being studied and improved to increase the throughput of today's blockchain networks. Moreover, it is essential to see how they will impact the challenges related to the 6G network, such as scalability and transaction throughput will improve as they still require attention. Also, as illustrated above, off-chain data access can facilitate SLA management. However, a possibility of unintended events endangering the decentralized applications' underlying performance goals by bridging off-chain SLA data with on-chain smart contract data is present. Therefore, new trust-building techniques can make oracles like chainlink more accountable to avoid the SLA data breach sent to the smart contract triggering crucial decisions.

V. Conclusion

The current conventional solutions suffer from the SLA management and monitoring for demanding use cases of 6G networks such as multi-administrative domains. Blockchain and smart contracts offer prospects for SLA management in a decentralized and automated manner and independent of the control of a single central authority. However, traditional SLA management methods are insufficient because they lack transparency and automation, and the process requires manual management. Therefore, this study presents a conceptual framework for enhancing SLA management and bridging the gap towards improved QoS assurance in 6G use cases such as inter-provider agreements. We use chainlink to enable SLA monitoring as well as IOTA Tangle and IPFS to reduce the overall cost of the transaction performed on the smart contract for inter-provider agreements. To evaluate it, we use performance metrics such as latency, response time, overall time consumption, and transaction and storage cost. Compared to conventional negotiations, our approach shows through experiments that it takes less than $\approx$ 30s to form a contract (SC-Marketplace and SC-Auction) to purchase service. Furthermore, we have distributed emulation results in two phases. Phase 1 consists of sending a request to buy resources for SC-marketplace or SC-auction using the IOTA-EVM testnet until the services start to run. Whereas phase 2 consists of SLA monitoring. Phase 2 uses the Kovan testnet to get data feeds for SLA monitoring. The emulation results show that the latency observed in phase 1 is less than in phase 2 as phase 1 uses IOTA-EVM and phase 2 uses Kovan. Similarly, for response time, the average response time for phase 1 is less than $\approx$ 10s $\sim$ 12s and $\approx$ 15s $\sim$ 18s for phase 2. We use IOTA-EVM to have fee-less transactions and IPFS to minimize the storage up to 80% costs. However, phase 2, consisting of SLA monitoring, adds additional time and cost.

Acknowledgment

This work was partially funded by EU H2020 monB5G grant 871780 and Spanish MINECO grants TSI-063000-2021-54/-55 (6G-DAWN).

References

[1] V. Räisänen, "A framework for capability provisioning in b5g," in *2020 2nd 6G Wireless Summit (6G SUMMIT)*, pp. 1–4, 2020.

[2] J. Baranda, J. Mangues-Bafalluy, R. Martínez, L. Vettori, K. Antevski, C. J. Bernardos, and X. Li, "5g-transformer meets network service federation: design, implementation and evaluation," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pp. 175–179, IEEE, 2020.

[3] F. Javed, K. Antevski, J. Mangues-Bafalluy, L. Giupponi, and C. J. Bernardos, "Distributed ledger technologies for network slicing: A survey," *IEEE Access*, vol. 10, pp. 19412–19442, 2022.

[4] J. Backman, S. Yrjölä, K. Valtanen, and O. Mämmelä, "Blockchain network slice broker in 5g: Slice leasing in factory of the future use case," in *2017 Internet of Things Business Models, Users, and Networks*, pp. 1–8, IEEE, 2017.

[5] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, "From network sharing to multi-tenancy: The 5g network slice broker," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 32–39, 2016.

[6] L. Zanzi, A. Albanese, V. Sciancalepore, and X. Costa-Pérez, "Nsbchain: A secure blockchain framework for network slicing brokerage," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, 2020.

[7] T. Faisal, M. Dohler, S. Mangiante, and D. R. Lopez, "Beat: Blockchain-enabled accountable and transparent network sharing in 6g," *IEEE Communications Magazine*, vol. 60, no. 4, pp. 52–56, 2022.

[8] T. Faisal, J. A. O. Lucena, D. R. Lopez, C. Wang, and M. Dohler, "How to design autonomous service level agreements for 6g," *arXiv preprint arXiv:2204.03857*, 2022.

[9] K. Antevski and C. J. Bernardos, "Federation in dynamic environments: Can blockchain be the solution?," *IEEE Communications Magazine*, vol. 60, no. 2, pp. 32–38, 2022.

[10] S. Popov, "The tangle," *White paper*, vol. 1, no. 3, 2018.

[11] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.