5GaaS: DLT and Smart Contract-Based Network Slice Management in a Decentralized Marketplace

Kurdman Rasol*, Alfonso Egio*, Miguel Catalan-Cid*, Leonardo Lossi†, Hélio Simeão‡, Shuaib Siddiqui*
*i2CAT Foundation, Spain. {kurdman.rasol, alfonso.egio, miguel.catalan, shuaib.siddiqui}@i2cat.net
†Nextworks, Italy. l.lossi@nextworks.it
‡Ubiwhere, Portugal. hsimeao@ubiwhere.com

***Abstract*—The proper orchestration of end-to-end network slices demands dynamic and meticulous resource management while addressing the complexities of multi-tenancy and multi-service scenarios. In this context, integrating network orchestrators with distributed ledger technologies has gained significant attention for its potential to implement decentralized finance marketplaces and service-level agreements using smart contracts. This approach can support the evolution of business models beyond traditional network-sharing agreements, such as crowdfunding. To this end, we propose the 5G-as-a-Service (5GaaS) system architecture, which leverages distributed ledger technologies and smart contracts to orchestrate and optimize network slicing, enabling ubiquitous computing and connectivity in 5G networks. We evaluated the feasibility of this system across various Ethereum testnets, demonstrating the cost-effectiveness, scalability, and minimal latency of the 5GaaS system, making it suitable for seamless integration into existing telecommunications frameworks.**

***Keywords*—DLT; Network Slicing; Smart Contracts**

I. Introduction

The landscape for Mobile Network Operators (MNOs) is undergoing a significant transformation. As MNOs transition from non-standalone (NSA) to standalone (SA) 5G deployments, a crucial shift takes place, fueled by softwarisation and virtualization paradigms. The adoption of 5G SA will unlock advanced network slicing functionalities, which can revolutionize MNOs' capabilities by allowing the creation of multiple virtual networks on the same physical infrastructure and adapted to specific service requirements.

Network slicing will be key for adopting network sharing and multi-tenancy [1]. The worldwide expansion of 5G networks is expected to prompt operators to deploy small cell infrastructure on a massive scale, which in turn requires finding suitable urban spaces with both backhaul and energy availability. Therefore, network sharing is becoming increasingly necessary to facilitate large-scale commercial deployments. These developments are largely based on a service-oriented architecture for managing the life cycle of slices.

In these scenarios, the motivation for exploring the application of blockchain technology in network slicing management and orchestration has grown [2]. One notable avenue involves the integration of blockchain technology into platforms with the aim of streamlining accounting processes. To this end, the 5G-as-a-Service (5GaaS) project proposes a novel concept that empowers MNOs and service providers to offer highly differentiated services to vertical industries through network slicing. The main challenge involves integrating dynamic finance and accounting processes for digital assets into an already existing and complex network service orchestration layer. The goal is to achieve this in a minimally disruptive way through asynchronous workflows that support end-to-end real-time dynamic operations in an open and decentralized marketplace environment.

The remainder of this paper is structured as follows. Section II presents the related work on the network slicing concept and the applicability of blockchain technology in the 5G marketplace using smart contracts. Section III introduces the architecture necessary to support the 5GaaS system. Section IV covers the integration of the Slice Manager and DLT using a Smart Contract System, while Section V describes its application to a crowdfunding scenario. Section VI evaluates the cost of adopting the solution. Finally, Section VII expounds conclusions and future work.

II. Related Work

Collaboration among multiple stakeholders in a multi-domain and multi-tenancy business model has been identified as advantageous in the context of 5G service deployment and operation [3]. In the context of 6G, multi-tenancy will be also a key feature to address sustainability through cost-effective infrastructure and services [4]. In pursuit of a neutral and decentralized billing model, the natural choice leads to the utilization of DLT technologies, which offer trust and security to create a safe environment for resource negotiation. DLT features such as consensus and immutability can be exploited through a decentralized marketplace to create an economic platform for service providers to exchange services, virtual network functions (VNFs), or resources.

With the help of blockchain technology, the full potential of network slicing can be extended from a single domain to a global scale in a seamless and automated way [5]. By using smart contracts to enforce Service Level Agreements (SLAs), the reliability of the offered global services can be significantly improved. Furthermore, integrated pay-as-you-go charging can be implemented using blockchain technology [6]. Authors in [7] introduced the Slice Leasing Ledger, where each involved stakeholder (consumer or provider) possesses a unique digital key for signing and verifying transactions. The goal of this approach is to facilitate verifiable transactions for charging, billing, and SLA agreements between consumers

978-3-903176-66-9

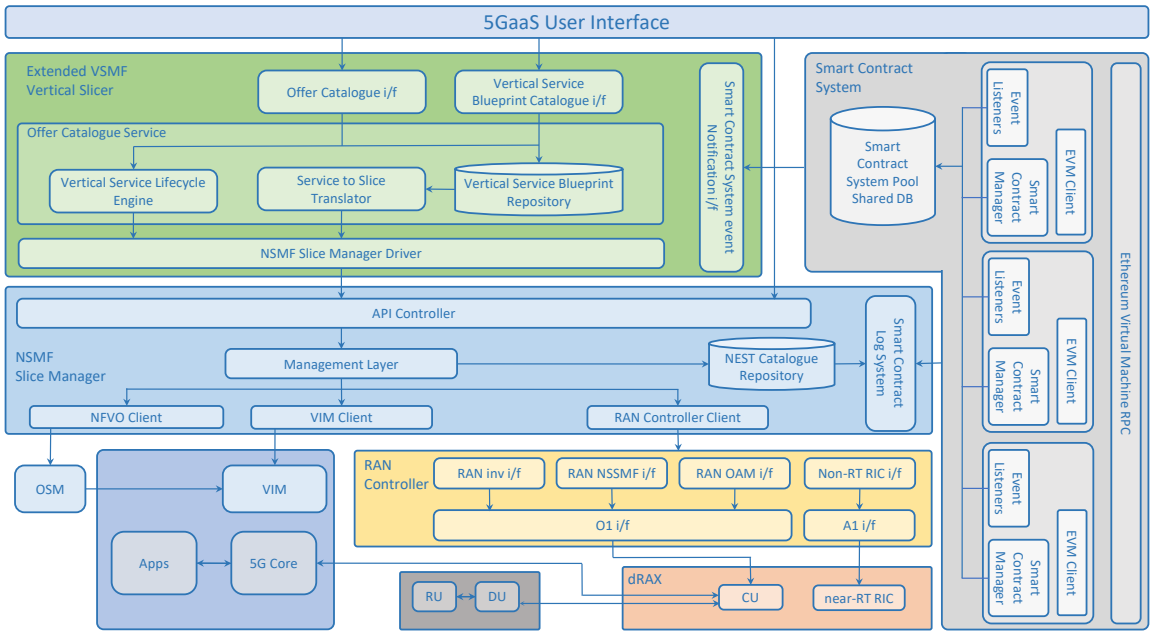

Fig. 1: Modular architecture of the 5GaaS framework

and providers. Furthermore, leveraging Blockchain technology streamlines the process of service creation, saving time and enabling more dynamic operations. However, it's worth noting that no implementation or performance analysis has been provided for this concept.

The system described in [8] is based on the Hyperledger blockchain to support roaming bill automation between different operators. Since Hyperledger is a permissioned network, it requires some centralized governance from the start, allowing it to strictly control nodes and smart contracts in a closed environment. Therefore, it cannot be considered a fully decentralized solution. Authors in [9] delve into the application of Ethereum smart contracts related to 5G NFV infrastructure, facilitating interactions between consumers and providers. Notably, the transactions are directed towards the primary blockchain, eschewing a peer-to-peer workflow in favor of optimizing performance and scalability. However, the evaluation doesn't include a real integration with a MANO framework neither with a slice management system.

To this end, this paper presents 5GaaS, a blockchain-based decentralized marketplace for 5G network slicing, which centers around the adoption of a Slice Manager as the central orchestration layer. The 5GaaS marketplace connects supply and demand in a single digital marketplace, simplifying the trade of assets and services, using the programmability of smart contracts.

III. 5GaaS Design

The 5GaaS platform is composed of several components, each with its own distinctive function and objective. In this section, we describe each component and how they interact within the 5GaaS platform, which is depicted in Figure 1.

The 5GaaS User Interface (UI) is the gateway for service providers and consumers in the 5G-as-a-Service marketplace. It features Ethereum wallet integration for secure user authentication. Users register and log in using Ethereum wallets via MetaMask web browser extension, ensuring a safe and streamlined experience. The UI collects and transmits user information and facilitates offer creation and purchase. Service providers can create offers, and users can easily access and purchase them through the UI.

The 5GaaS UI interacts with the Vertical Slicer (VS)[10], which constitutes a fundamental element within the management and orchestration (MANO) components of the 5GaaS platform. The primary objective is to bridge the gap between the specialized needs of verticals, focusing on business and service-level requirements, and the challenges faced by service providers when interacting with MNOs to automate the setup, coordination, and configuration of 5G networks to fulfill these demands. The VS facilitates browsing, customization, and purchase of vertical service offers (VSO) while shielding users from the underlying complexities of the 5GaaS MANO stack.

The Slice Manager (SM) is responsible for overseeing the provisioning and lifecycle management of Network Slice Instances within the virtualization infrastructure. Additionally, it is responsible for establishing communication with the Smart Contract System (SCS) for deploying the corresponding Smart Contracts. The SCS, in turn, notifies the VS when the Smart Contracts are deployed, enabling the VSO Information Model (IM) to be enriched with the requisite information for accessing the VSO Smart Contracts.

The SM plays a central role, particularly in providing the necessary logic for dynamically creating and managing slices, each defined as a collection of logical network partitions

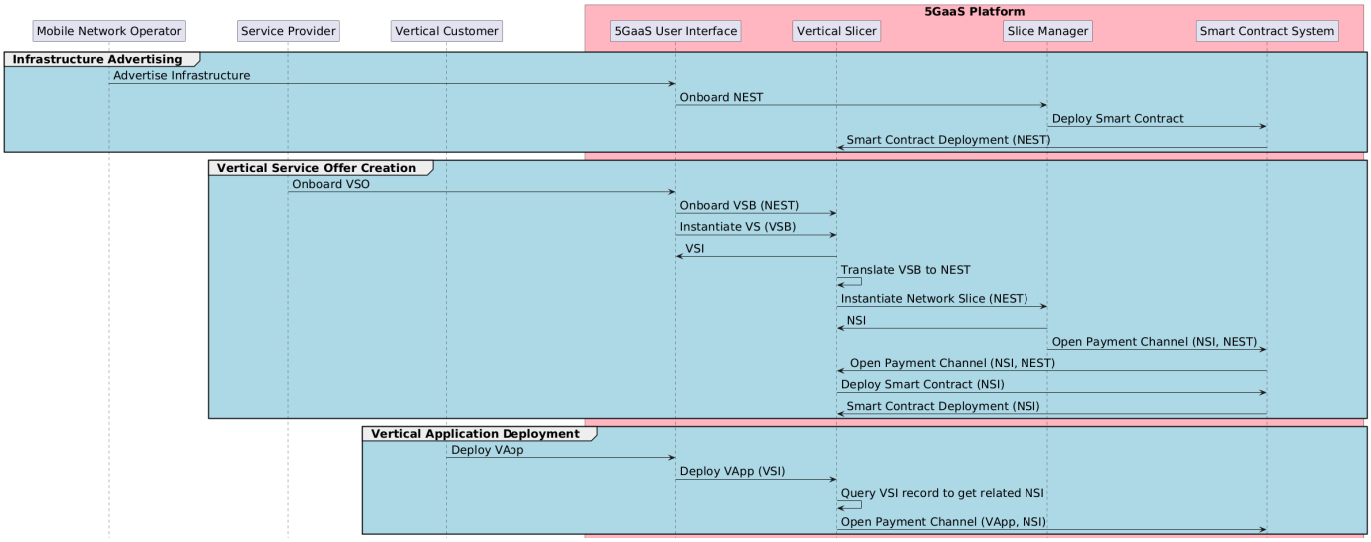

Fig. 2: Workflow between VS, SM and SCS comprising the different stakeholders' blockchain interfacing agents

or segments combined with the network services deployed on top of them [11]. The SM receives Vertical Slicer calls concerning the slice parameters required to deploy a service instance in accordance with the requirements of the issuer, provisioning and managing various computational, software, radio, and network resources.

First, the SM interfaces with Virtualized Infrastructure Managers (VIMs) to allocate and oversee cloud and edge computational resources, ensuring their optimal utilization in the context of service deployment. Secondly, it integrates with OSM which functions as a multi-VIM NFV management and orchestration stack compliant with the ETSI NFV architecture. Lastly, the SM integrates with the RAN Controller to align the RAN configuration with the requirements of the services. The RAN controller uses the NETCONF/O1 protocol to manage the configuration of the RAN nodes, including radio parameters and slice parameters.

Follows a description of the application of the 5GaaS platform to network slicing by detailing the workflows involving the Vertical Slicer, the Slicer Manager, and the RAN Controller

A. Vertical Slicer interactions

The 5GaaS MANO stack models a VSO as a Vertical Service Instance (VSI) composed by a Network Slice Instance built onto a NEST [12] and deployed over a virtual infrastructure managed by one or more VIMs. The Vertical Service functions, characteristics, capabilities, and mobile connectivity requirements are described on the VS by means of intents called Vertical Service Blueprints (VSB), whose information model is not defined by any standard. VSBs provides service-related parameters to enable the mapping onto one or more NESTs inside the SM catalogue defining the internal topology of the vertical application components and the service profile of the required network slice. This design allows the VS to translate the service-level and business-level requirements defined in the high-level service intent from the verticals to the concrete characteristics of the mobile connectivity to be offered by the operators according to a rule-based logic.

As illustrated in Figure 2, first, the MNO advertises through the platform's user interface an infrastructure suitable for deploying 5G-related services and applications using the Slice Manager. Then, a service provider can then select and configure it according to the requirements of suitable vertical application use cases. This involves creating a network slice that can be utilized by a third party (the vertical customer) to deploy their own application instances.

B. Slice Manager interactions

The Slice Manager holds responsibility for resource partitioning at three distinct levels within the 5GaaS framework. These levels encompass infrastructure, network slice, and vertical service management. At the infrastructure level, the SM receives slice requirements. It then creates and activates the radio and computing virtualized infrastructure at the RAN, Edge, and Data Center levels, while reserving resource chunks and associating them with a specific end-to-end Slice ID. At the network slice level, the SM deploys and configures the 5G Core network. At the vertical service level, third-party actors can deploy a specific set of services published on any segment of the 5GaaS marketplace. The SM configures DNS and DHCP within a slice, making VNFs reachable within the network slice. However, the SM delegates VNF deployment, monitoring, scaling, and migration operations to the corresponding orchestrator.

Within the workflow outlined Figure 2, the Slice Manager contributes to a validated NEST catalog, offering customizable parameters for networking and radio configurations to the Vertical Slicer. Moreover, it actively manages the lifecycle of NESTs by orchestrating their instantiation into service slices. NEST providers gain the capability to automate billing for NEST service instances on behalf of other stakeholders. Importantly, the integration allows for a nuanced approach

to billing, enabling the separation of charges for distinct components within the NEST template. For instance, billing for radio infrastructure usage can occur separately from the computational resources responsible for running the 5G Core, catering to the needs of different stakeholders.

The versatility of the SM is showcased through its support for multi-tenancy and its ability to handle diverse controllers across different locations at a granular level. This versatility extends to its role as the central integration point with the Smart Contract System. Each user of the SM can deploy their smart contract interfacing agent, enabling them to receive streaming messages regarding actions performed on their resources and services orchestrated by this component. Additional details on the integration of the SM and the SCS will be provided in Section IV.

C. Resource Controllers interactions

Once received the request to deploy a slice, the SM interacts with different resource controllers in order to initiate resource reservation, configuration and activation. The main orchestration procedures involving the RAN and 5G Core deployment and configuration, are described as follows:

- In the case of radio resources, the RAN Controller receives a request specifying the 5G cells included in the slice and the required configuration, which comprehends parameters such as the PLMNID of the operator, the S-NSSAI of the slice or the N2 configuration to reach the AMF (e.g., AMF IP, port, and VLAN). The RAN Controller then interacts with the CU-CP (i.e., setting AMF configuration) and CU-UP (i.e., settting PLMN and S-NSSAI identifiers) components associated with the selected cells.
- In case it is specified in the slice description, the SM is also able to interact with the VIM to deploy a 5G Core instance configured according to the slice parameters. We have implemented this through OpenStack and Open5GS, an open-source implementation of the 5G Core. The SM deploys all the required NFs and, mainly, configures the AMF (N2 configuration, supported PLMNID, supported S-NSSAIs), the UPF (N3 configuration), the SMF (supported S-NSSAIs), and the NSSMF (supported S-NSSAIs). Additionally, the Slice Manager may configure the Open5GS UE database with an initial list of allowed IMSIs and Data Network Names (DNNs). This allows the automated deployment of a end-to-end 5G slice ready to be used by the verticals.

Additionally, in case an application or service is included in the slice in the form of an NFV, the Slice Manager is able to interact with the NFVO and the VIM, respectively, OSM and Openstack, to deploy the application in the specified compute resources and connect it to the data plane of the deployed RAN slice (i.e., to the network connected to the N6 interface).

IV. Integration with DLTs and SCS

This section delves into the integration of the 5GaaS platform with DLTs and Smart Contract Systems. It explores the use of Apache Kafka for asynchronous integration, alongside the architecture of independent agents. Additionally, it details the implementation of state channels for payment processing, along with the importance of event tracking and visualization for transaction monitoring. Note that 5G offering parties could easily integrate their own permissioned blockchain network with 5GaaS by just changing a line in the configuration file and with the only requirement of supporting Ethereum Virtual Machine-like smart contracts. We selected Ethereum due to its flexibility to choose between public and own-deployed networks.

A. API Choice and Operation Mapping with Apache Kafka

Slice Manager operations are offered through a synchronous REST API. On the other hand, many DLT technologies, especially the one targeted in this work, a Ethereum-like blockchain, rely on RPC-kind (remote procedure command) APIs. Since blockchain operations often exceed the duration of typical orchestration tasks handled by the Slice Manager, taking in some cases several minutes, these APIs feature an asynchronous nature. To further analyze the relationship between these two kinds of APIs, special attention has been given to the types of actions being mapped between the slice manager and the DLT, including but not limited to:

- Billing smart contract deployment according to infrastructure registration on the marketplace.
- Payment channel opening and closing, accounting for parts of a slice between infrastructure owners, service providers, and vertical use case customers.
- SLA incident reporting on the billing smart contract in case there's a defined penalty.

Accordingly, an asynchronous integration between the Slice Manager and the different DLT/Smart contract operations has been implemented to avoid additional friction between the slicing orchestration workflows and the DLT consolidation operations. To achieve this integration in an asynchronous operation mode, Apache Kafka has been used. The mechanics of the integration mainly rely on adding to the Slice Manager a logging system able to broadcast messages about operations performed by each of the users through Apache Kafka streams classified under relevant topics.

For instance, whenever a user registers a radio infrastructure, such as some number of 5G cells set up under a certain topology, the SM will broadcast a message regarding that event through a topic labeled as RAN infrastructure. This will be considered by the Smart Contract System developed to support this work, and accordingly, it will deploy a smart contract on behalf of the user who owns the infrastructure. Eventually, when a service provider requires using this infrastructure to deploy a service, the Smart Contract System middleware will transact with the specific smart contract to initiate a payment channel opening transaction in the DLT.

B. Internal Architecture of Independent Agents

Taking as a starting point the mentioned integration through the Apache Kafka message streaming system, we consid-

ered the ability to deploy multiple agents belonging to each stakeholder without the need for a central authority. These agents, which can be distributed among different and independent computational infrastructures (e.g., on private or public clouds), allow users to hold their private keys in their own custody and perform automated operations without the need to interact with a graphical user interface whenever a workflow requires a signature. Since the system relies on a pool of agent instances (one for each stakeholder) that can be independently hosted in a distributed manner, it horizontally scales by design on the number of users; each agent can operate independently as a Blockchain interfacing agent and also between each other in peer to peer protocols avoiding bottlenecks. In addition, having the ability to deploy these agents on stakeholders' own controlled infrastructure, guarantees that the private keys can be kept without the need for a central service that may imply risks as a single point of failure.

The mentioned agents consist of a software artifact that accepts configuration parameters, including the user's blockchain private key and their own ID recognized in the SM, among other assets like smart contract templates, parameters regarding payment deviation thresholds, SLA policies, etc. Internally, those agents comprise an event listener able to read the Apache Kafka streaming system and a smart contract manager able to perform smart contract deployment and automate smart contract transaction operations. Any of the stakeholder's agent has access to the SLA related events and designated stakeholders acting as referees could trigger penalties and closing conditions of the payment smart contracts in case their agent intercepts a SLA breach. To support that kind of operations, each agent contains an Ethereum Virtual Machine client that can be connected directly to an Ethereum node of any EVM compliant network.

C. State Channels for Payment Processing

Regarding blockchain trade-offs, which could impact the latency and/or friction of different operations like smart contract deployment times or transaction fees, the considered approach has consisted of using a state channel paradigm [13] regarding the support of billing or crowdfunding operations. The main idea behind state channels relies on a workflow that involves writing an entry in the smart contract deployed on the blockchain. This entry can be regarded as a state channel containing the digital signatures of the agents involved within the pool of instances connected to Apache Kafka. Such a state channel serves as an anchoring point for billing and SLA incident monitoring workflows. Once the entry is written, each participating agent can initiate peer-to-peer off-chain transactions. These transactions will ultimately conclude with a second operation, corresponding to the closure of the state channel, in accordance with the withdrawal conditions specified in the smart contract. This closure is executed through a final blockchain transaction.

Once these initial transactions take place in the main blockchain through specific smart contract procedure callings, the stakeholders' agents will hold a protocol exchanging messages through a common database that would be signed accordingly. Each of these messages contains an incremental payment commitment according to the billing conditions that can be anchored in the blockchain hosted smart contract at any time. For example, whenever a service operation reaches its end of life, the agent corresponding to the infrastructure owner supporting the service would trigger the payment channel closing method of the corresponding smart contract triggering the corresponding value transfer between both accounts.

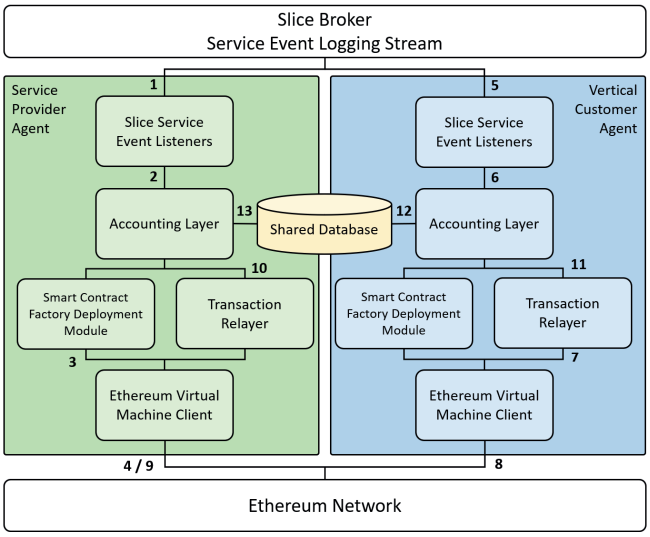

Fig. 3: SCS Agent: Detailed Interaction Diagram

D. Event Tracking and Visualization

In addition to the integration of the SM through the Apache Kafka system, various blockchain interfacing agents will generate notifications for each consolidated transaction on the blockchain. These transactions include smart contract deployment and disabling, as well as state channel opening and closing. These notifications will be managed through an event tracking Vertical Slicer endpoint, providing visibility into the smart contract status for proper UI visualization.

Figure 3 illustrates an exemplar workflow involving the SCS of two distinct instances: a Service Provider Agent and a Vertical Costumer Agent.

1) A message is streamed from the slice broker, conveying the registration of a new infrastructure stack.
2) This message undergoes processing by the provider's instance, triggering the deployment of a smart contract through the EVM client (steps 3 and 4).
3) Subsequently, a customer deploys a service on a slice created over the infrastructure, and the corresponding message is captured by the customer's instance (step 5).
4) Then, the customer's instance initiates the accounting process by opening a payment channel on the previously deployed smart contract (steps 6, 7, and 8).
5) The event of channel opening is monitored by the provider's instance, prompting the start of an execution thread that awaits payment validation (steps 9 and 10).

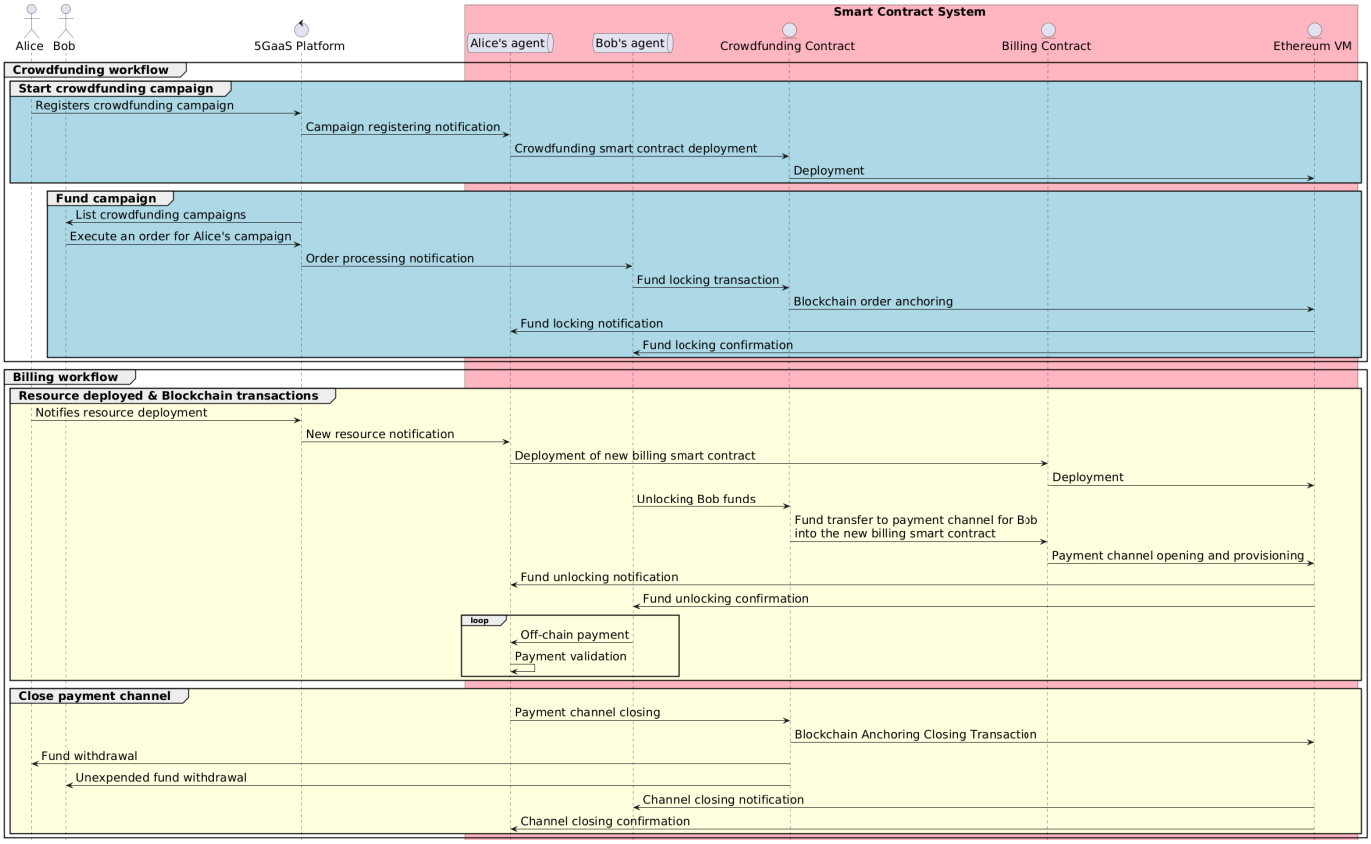

Fig. 4: 5GaaS Platform sample workflow regarding crowdfunding and billing processes (*blue background corresponds to the crowdfunding part of the workflow and yellowish one to the billing subprocess*)

.

6) Following the channel opening, the customer's instance initiates the signing process (step 11) and transmits appropriately signed off-chain payments through a shared database (step 12).
7) These payments are then subject to validation by the provider's instance accounting modules (step 13).

V. Application to Crowdfunding Scenarios

In addition to the billing scenario discussed in the previous section, we explored another use case involving the support of crowdfunding workflows. Recognizing that resource sharing is a fundamental aspect of slicing technology, it became imperative to facilitate a scenario where multiple interested stakeholders could contribute to the funding of infrastructure deployment. This includes accessing a decentralized finance marketplace tailored for 5G resources. The proposed approach in this work revolves around a crowdfunding smart contract. In this setup, a stakeholder, such as a mobile network operator, can register a commitment to invest in and deploy specific infrastructure based on the interest of potential customers looking to fulfill their business requirements by slicing and offering portions of the infrastructure as services or applications for their own customers.

The workflow for this scenario illustrated in Figure 4 that also includes previously described billing workflow; it involves the crowdfunding stakeholder deploying a dedicated smart contract linked to a hashed human-readable document. This document outlines conditions such as the features of the yet-to-be-deployed infrastructure, SLAs, deployment time deadlines, pricing conditions, etc. Once the crowdfunding initiative is consolidated in the blockchain, any interested stakeholder can lock funds as a commitment to subscribe to resource consumption. When the funds locked by interested stakeholders surpass a contract-defined threshold, the crowdfunding initiator is guaranteed to deploy the infrastructure and unlock the funds into a billing smart contract, similar to the one described in the previous section, initiating the payment processes seamlessly and automatically.

In the event that the infrastructure isn't deployed because the crowdfunding stakeholder fails to fulfill the initial commitment, the smart contract provides a mechanism to recover the locked funds for each interested party, along with accessing a pre-provisioned guarantee escrow secured by the crowdfunding campaign initiator as compensation for the loss of liquidity. Similar to the billing smart contract, this crowdfunding approach also supports cascading behavior, enabling seamless and automated decentralized finance operations in intricate scenarios, particularly in current 5G markets. This ensures trust and transparency among multiple stakeholders.

VI. Gas costs, deployment and typical transaction confirmation times

In Ethereum, gas is the unit used to measure the amount of computational effort required to execute operations, such as transactions and smart contract deployments. Each operation in Ethereum consumes a specific amount of gas, and users must pay in Ether (ETH), the native cryptocurrency of Ethereum. Computationally intensive operations require more gas, leading to higher costs. The practicality of any smart contract-based system depends on computational deployment costs, operational expenses, and confirmation times. Therefore, we considered these metrics to assess the feasibility of applying 5GaaS smart contract implementation in real-world scenarios.

The deployment times and gas units usage for typical transactions using the smart contracts in this study were measured across 200 runs on various testnets, with all values showing less than a 5% deviation. The results are presented in Table I. Time estimations were measured over the Goerli testnet, and gas units evaluation was performed using the Gas Usage Analytics for Hardhat tool[1].

Contract	Dep. time	Dep. Gas	Tx time	Tx Gas
PaymentChannel	90 s	2.8e6	45 s	7.5e4
Crowdfunding	95 s	1.5e6	67 s	8.1e4

TABLE I: Average Smart Contract Deployment Times and Gas Usage Across 200 Transactions with <5% Deviation

The metrics indicate that the 5GaaS system is well-suited for widespread adoption by any telecommunications stakeholder consortium. It offers minimal friction, with only marginal costs (most expensive operation at current gas costs are of the order of units of USDs) and negligible latency impacting business operations. This suggests that the system can seamlessly integrate into existing frameworks, providing a scalable solution with efficient performance and cost-effectiveness for industry-wide deployment.

VII. Conclusions

The application of blockchain and smart contracts in network slice management offers a transformative approach for the secure, transparent, and efficient handling of transactions and interactions between different stakeholders in the 5G network ecosystem. 5GaaS decentralized marketplace is built on top of innovative solutions which facilitate and automate the composition, offering, billing and deployment of E2E network slices. The proposed framework relies on the integration a pool of user agents that scale horizontally and have automation capabilities to mirror hierarchically complex network slices and service deployment scenarios into smart contract business rules. Additionally, the smart contracts used implement state channels to ensure scalability and peer-to-peer operational capabilities. These workflows could bridge the gap, especially in 5G deployment and adoption, where many private and public companies and entities can benefit from cooperation in a frictionless marketplace. The results of this study, including the evaluation of gas costs, deployment times, and transaction confirmation times, indicate that 5GaaS is well-suited for adoption by telecommunications stakeholders, with minimal friction and negligible latency affecting business operations.

Future work includes the application of AI/ML-based strategies to further automate and optimize the selection and management of slice resources. Additionally, the smart contracts developed and tested so far only consider the option of executing value transfers in the native currency of the targeted Ethereum Virtual Machine (ETH for Ethereum and MATIC for Polygon). To mitigate the volatility associated with exchange values and ensure alignment with business requirements, an extension of these smart contracts to handle specific utility tokens and/or stablecoin tokens should be explored as well.

[1]https://github.com/cgewecke/hardhat-gas-reporter

Acknowledgment

Funding for this research has been provided by the European Union's (EU) Horizon Europe research and innovation programme XGain (Grant Agreement Number 101060294). This work was also supported by the Spanish Ministry of Economic Affairs and Digital Transformation, along with the European Union – NextGenerationEU, under the framework of the Recovery, Transformation, and Resilience Plan (PRTR) (Call UNICO I+D 5G 2021, ref. numbers TSI-063000-2021-12 – 6GENABLERS-DLT and TSI-063000-2021-15 – 6GSMART-EZ).

References

[1] M. Catalan-Cid and et al., "i2Slicer: Enabling Flexible and Automated Orchestration of 5G SA End-to-End Network Slices," in *2023 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2023, pp. 159–162.

[2] F. Javed and et al., "Distributed ledger technologies for network slicing: A survey," *IEEE Access*, vol. PP, pp. 1–1, 2022.

[3] A. Fernández-Fernández and et al., "Unlocking the path towards intelligent telecom marketplaces for beyond 5g and 6g networks," *IEEE Communications Magazine*, vol. PP, pp. 1–7, 03 2023.

[4] Hexa-X-II, " D3.1: Environmental and social view on 6G," Tech. Rep., 2023.

[5] S. Bao and et al, "Blockchain for network slicing in 5g and beyond: Survey and challenges," *Journal of Communications and Information Networks*, vol. 7, no. 4, pp. 349–359, 2022.

[6] N. Hamdi and et al., "A survey on sla management using blockchain based smart contracts," in *Intelligent Systems Design and Applications*. Cham: Springer International Publishing, 2022, pp. 1425–1433.

[7] J. Backman and et al., "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*, 2017, pp. 1–8.

[8] B. Mafakheri and et al., "Smart contracts in the 5g roaming architecture: The fusion of blockchain with 5g networks," *IEEE Communications Magazine*, vol. 59, no. 3, pp. 77–83, 2021.

[9] F. Javed and J. Mangues-Bafalluy, "Blockchain-based 6g inter-provider agreements: Auction vs. marketplace," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 1271–1277.

[10] X. Li and et al., "5growth: An end-to-end service platform for automated deployment and management of vertical services over 5g networks," *IEEE Communications Magazine*, vol. 59, no. 3, pp. 84–90, 2021.

[11] A. Papageorgiou and et al., "On 5g network slice modelling: Service-, resource-, or deployment-driven?" *Computer Communications*, vol. 149, pp. 232–240, 2020.

[12] GSMA, "Generic network slice template, version 7.0, ng.116,june 2022." *https://www.gsma.com/*, 2020.

[13] L. D. Negka and G. P. Spathoulas, "Blockchain state channels: A state of the art," *IEEE Access*, vol. 9, pp. 160 277–160 298, 2021.