An end-to-end trusted architecture for network slicing in 5G and beyond networks

Sabra Ben Saad* Adlen Ksentini* Bouziane Brik ‡

Abstract

5G opens the door not only for new network services but also for a new business model involving new business actors. Particularly, the introduction of Network Slicing encourages the appearance of new stakeholders, namely Verticals or network slice tenants, Network Slice Providers, and Resource Providers. The Network Slice Provider sells end-to-end network slices (virtual end-to-end mobile network) to the Vertical, while leasing virtual and physical resources from Resource Providers to deploy and instantiate these end-to-end network slices. Consequently, a trusted relationship among these actors should be established and maintained. To fill this gap, in this paper, we propose a Blockchain-based trust architecture that aims to: (1) create and negotiate the deployment of an end-to-end slice network; (2) manage automatically the Service Level Agreements (SLA) established between the involved actors; while guaranteeing security and anonymous transactions.

Keywords: Service Level Agreement, 5G End-to-End slicing, Security, Smart Contract, Blockchain.

1 Introduction

The evolution of 5G comes to us faster than can be imagined offering a great opportunity for new business model innovations [20]. Network Slicing is one of the key enablers of 5G, which consists of creating virtual end-to-end mobile networks tailored to the application needs[1],[21]. The network slicing concept allows multiplexing virtualized and distinct logical networks on the same physical network infrastructure. Through network slicing, 5G supports three types of network slices supporting services with specific performance requirements: enhanced Mobile Broadband (eMBB), massive Machine-Type Communications (mMTC), and ultra-Reliable Low-Latency Communications (uRLLC)[2],[23].

Usually, an end-to-end network slice comprises three main sub-slices: Radio Access Network (RAN), Core, and Transport networks [1] corresponding to three distinct Technological Domains (TD). Therefore, to deploy an end-to-end network slice, the Slice Provider (SP) has to lease resources for each sub-slice on each TD and from different Resource Providers (RPs).

First of all, the Vertical (V) chooses the network slice type that perfectly matches its needs, requesting the creation of a new network slice using a blueprint or a network slice template provided by the SP. Next, this template will be translated by SP into specific slice resource requirements for each sub-slice component or TD, such as needed computing resources, network resources, radio resources, etc. SP uses a resource broker mechanism to select which RP will deploy a sub-slice for each TD.

*Communication Systems, Eurecom Sophia Antipolis, France. Email: name.surname@eurecom.fr

‡DRIVE EA1859, university of Bourgogne Franche-Comté, France. Email: bouziane.brik@u-bourgogne.fr

The selection process relies on SP's specific algorithm, which considers several criterion, such as cost, Quality of Service (QoS)[22], reputation, etc. Once the needed resources by each sub-slice are allocated, meaning that the end-to-end network slice is ready to be deployed, SLA contracts are signed between V and SP as well as between RPs and SP. The SLA specifies what customers can expect from SP. SLA is used to check if a defined service is delivered as contracted and helps managing QoS degradation. Indeed, SLA defines QoS requirements such as bandwidth, throughput, and latency, without specifying the technology to be used to deliver a particular service. Moreover, SLAs include different sections on the concerned parties, the service fee, the validity period, and the compensation value used in case of SLA is not respected (i.e., when the performance levels are not met). In 5G, SLA should indicate the QoS related to the pre-defined slice types, i.e. mMTC, eMBB, and uRLLC.
In this paper, we propose a trust architecture to create and manage end-to-end network slices in 5G ensuring security and anonymous transactions. The proposed architecture relies on Blockchain and Smart Contracts to (1) negotiate and deploy an end-to-end network slice; (2) manage automatically SLA signed between all the involved actors, the slice tenant, SP, and RPs. First, we propose a trust architecture model that allows the resource broker, used by SP, to select the appropriate RP that will deploy a sub-slice for each TD. The resource selection is formulated as an Integer Linear Program (ILP), where the aim is to maximize the RP's reputation while minimizing the cost. Once the RPs are selected and SLA established, the proposed trust architecture framework allows monitoring the Key Performance Indicator (KPI) of the deployed end-to-end network slices, in order to verify if SLAs are violated by RPs, and hence automatically compensate V and SP. Finally, the RP reputation is build using the information on the SLA violation; RP's reputation increases if it has respected the established SLA, otherwise it decreases. Computer simulation has been used to validate the proposed framework in terms of (1) resource negotiation and RP selection to deploy an end-to-end network slice; (2) ability to detect SLA violation and compensate both V and SP.

To the best of our knowledge, this is the first work that proposes a trust architecture to negotiate and deploy end-to-end network slices, and automatically manage the SLA signed between the involved actors. A previous work [3] has introduced a Blockchain-based brokering mechanism to deploy end-to-end network slices. Although we share the same concept of using a Blockchain-based broker, in this paper we devise a selection algorithm based on RP reputation, which is an important criterion to establish trust relations among the involved actors. Regarding SLA management, the work in [4] introduces SLA trust management in the context of a simple scenario of a Cloud Resource Provider and a tenant. Despite the fact that we share the concept of using Blockchain, we address in this work a more complex scenario where not only one SLA has to be managed, but different SLAs involving different stakeholders.
The rest of the paper is organized as follows. Section II introduces the concept of Blockchain and Smart Contracts. Section III details our proposed trust architecture for an end-to-end network slice negotiation, creation and SLA management. Section IV presents the performance evaluation of the proposed algorithms and trust model framework. Finally, the conclusions are drawn in section V.

2 Blockchain and Smart Contract

Blockchain [5] is a chain of blocks, in other words a database that stores information about transactions like date, persons participating in transactions, time, and amount transferred. A copy of the Blockchain is spread over many computers. These computers are called nodes constituting a network. It also stores unique codes called "hash" which are cryptographic codes created by special algorithms such as "Proof of work" used in Bitcoin transactions[6]. Actually, when a transaction

occurs, it will be checked by all nodes. For each transaction, a new block is created. This block is sent to every node in the network. If a transaction is approved by the majority of the nodes using a special algorithm, then the block is added to the existing Blockchain.

More than that, Blockchain platforms are of different types: (i) Permissioned, which means that the Blockchain network is private and its access can be granted by particular participants, and the management of the network is typically done by a specific administrator; (ii) Permission-less, which means that the Blockchain network is public; or (iii) Consortium, which is a hybrid of permissioned and permission-less [7]. Additionally, Consortium or hybrid Blockchains are similar to the permissioned networks; but the management of the chain is carried out by multiple administrators.

Blockchain can be used in several fields, such as healthcare, Internet of Things (IoT)[8], online shopping, etc. Likewise, Blockchain implementation has extended far beyond the technology's applications in cryptocurrencies, moving toward more mainstream adoption and launching new opportunities. Take the example of smart cities, where the Blockchain can be used for user authentication [9]. We can especially mention the case of Estonia, where citizens or "e-residents" can access several public services by using a Blockchain-based digital ID card, while all-access history to the public's health record is registered in the Blockchain. Also, it is used for medical information management.

On the other hand, a Smart Contract is a feature associated with Blockchain. The definition of a Smart Contract based on Nick Szabo, taken from the original publication [10], is an agreement between several parties in the form of computer code. It allows automated transactions when one or more contract conditions are met without resorting to a third party. They are distributed and, therefore, stored in a public database that cannot be modified, such as a Blockchain. Plus, Smart Contract can control valuable things like the balance of user account (number of ETH) or other parameters, as well as the transfer of value among users. Ether (ETH) is a unit of currency, serving to compensate the nodes for storage and processing of Smart Contracts (1 Ether equals 213.20 EUR).

Smart contracts are used in Decentralized Application (DApp), which is a server-less peer-to-peer application that can run on a particular Blockchain written for a specific application. Moreover, Smart Contract is a self-executing contract where the terms of an agreement between the buyer and the seller are previously agreed before and are directly written into lines of code. Similar to a programming language, Smart Contract is a collection of functions and data residing at a specific address on the Blockchain. In the same way, data can be queried or altered by calling functions of the Smart Contract or by making a transaction to it. These functions are also executed automatically on every node in the network according to the data included in the triggering transaction [11]. Then, the Smart Contract's execution outcome is validated and agreed on by all distributed and trusted nodes of the network. Moreover, DApps are frontend user apps that interact with the Smart Contracts by initiating specific transactions that call functions within the Smart Contract.

Several works advocate for the usage of Blockchain and Smart Contracts in 5G. In [12], the authors summarize the key opportunities offered by Blockchain and Smart Contract technologies in 5G networks. The authors propose using Blockchain-based security techniques for 5 scenarios in 5G. The first scenario, "5G Infrastructure for Crowdsourcing", allows smaller infrastructure investors to roll out cellular towers that will be a part of the overall operator's infrastructure. These investors are registered, managed, and also automatically paid upon the use of the resources implicating secure technologies such as Blockchain and Smart Contracts. The second scenario, "5G Infrastructure Sharing", in which a seller Mobile Network Operator (MNO) offers telecommunication services either (i) cellular towers or (ii) a subset of these towers. Here, Blockchain can be used in managing and tracking resource usage and sharing. The third scenario, "International Roaming", is one of the challenging issues in the telecom sector as it involves brokers and third parties to settle down

payment, and charges rules among them. In 5G, many parties are going to be involved in the usage of networks. These parties may include multiple operators, intermediary networks, and international intermediary exchanges. Smart Contracts can be implemented to accomplish a Blockchain-based payment and roaming in which charges and consumption per usage are recorded and tracked. The fourth scenario,"Network Slicing", where a Network Slice Broker (NSB) is used to perform network slicing by exposing service capabilities of the mobile operator network during the brokering mechanism. A Blockchain and Smart Contract with decentralized storage, like Storj[1] or IPFS[2], can be used to replace NSB functionalities. The fifth scenario is "Management and Authentication of mMTC/ uRLLC" where millions of IoT devices are expected to be connected, and some scenarios may require few milliseconds of latency. Incorporating such a large number of different IoT devices opens up the possibility for new business models and services to be offered to mobile customers. Blockchain and Smart Contracts, with decentralized storage, are more powerful and attractive alternatives to these centralized operators, in which such management functionalities can be performed in a decentralized manner.
In [13], Blockchain was also used to build a verification platform between IoT devices and Baseband Unit (BBU) where user access information is stored on the chain and the Smart Contract is used to perform automatic user authentication. Similarly, the authors in [14] apply Blockchain to build a trusted authentication architecture for Cloud Radio Access Network (Cloud-RAN) in the 5G era. In [15], the authors propose an architecture for the core network of Long Term Evolution (LTE), where a distributed database is used instead of having a centralized database (Home Subscriber Server (HSS)) to maintain subscriber information. Indeed, when an operator configures a new Subscriber Identification Module (SIM) card, the SIM information is distributed within the Consortium Blockchain through a new block. Then, this block will be available to all the nodes (core networks) among the chain. Furthermore, when a user performs any activity such as subscribing to a new offer or charging the SIM card, this information is stored in another block. After its validation by the other nodes, the block will be published to all the other nodes within the chain as a copy in the distributed ledger. The ledger can have the same role as HSS to store user subscription information. When a user sends an attach request to Evolved Packet Core (EPC), the latter retrieves the information of International Mobile Subscriber Identity (IMSI) from the message and performs a null transaction to the Smart Contract. Then, the Smart Contract will check if the user belongs to one of the operators who are already in agreement with Consortium Blockchain. Hence, the Smart Contract acts as distributed Self Organizing Network (SON) features to handle self-transactions among mobile operators in return for sharing small cells' infrastructure.

3 Proposed Trust Architecture

As stated earlier, there are two separate steps to create and manage an end-to-end network slice. Firstly, SP selects the RPs in which the different sub-slices composing the end-to-end network slice requested by V will be deployed. This step ends when the SLAs are signed between, on one hand the SP and V actors and on the other hand, between the SP and RPs actors. The second step concerns the SLA management, which consists of monitoring the SLA violation and applying for penalties and compensation if deemed appropriate. For both steps, we are proposing a trust architecture to guarantee the well-functioning of such a complex system that involves different stakeholders and actors.

[1]https://storj.io/
[2]https://ipfs.io/

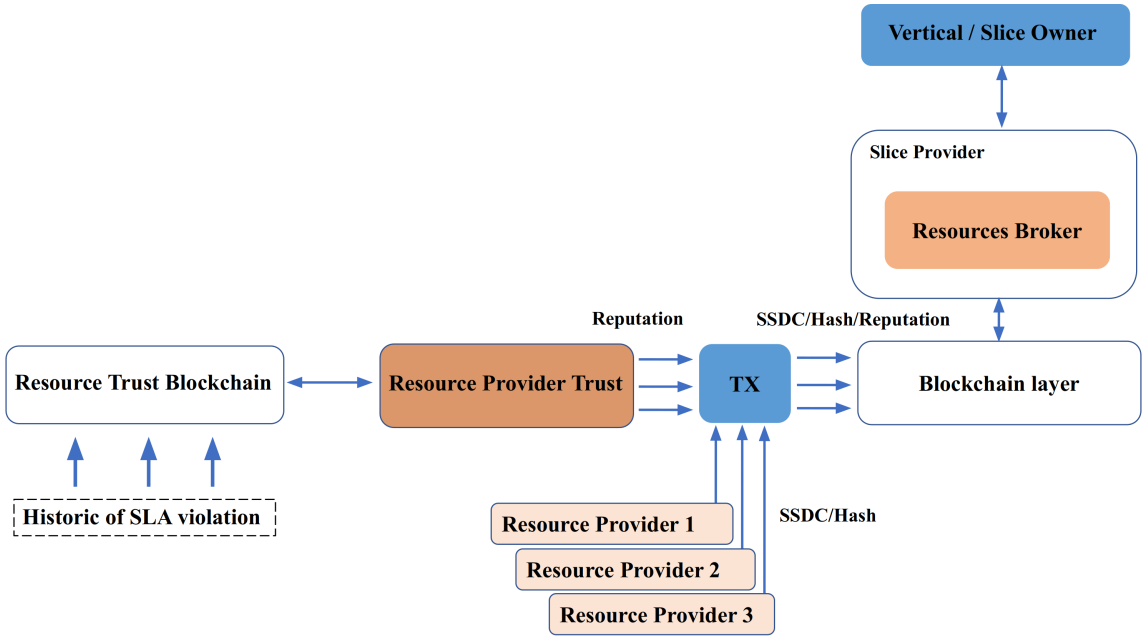


FIGURE 1: Trust architecture for end-to-end network slice creation.

3.1 Trust architecture for end-to-end network slice negotiation and creation

3.1.1 System overview

It is well established in 5G that a network slice is composed of sub-slices (virtual or physical resources) that belong to different TD; i.e., RAN, Core Network, Data Center domain. An end-to-end network slice is composed of Virtual and Physical Network Functions (VNF and PNF) deployed on different technological domains and stitched together to build an end-to-end slice through specific mechanisms or protocols. When a V requests the creation of an end-to-end network slice to SP, the latter decomposes it to sub-slices and negotiates for each sub-slice the necessary resources from the TD Resource Providers. This procedure is defined in [3] and is as follows. V or the slice owner requests the creation of a network slice using a template or a blueprint. This template may contain high-level information. The SP will translate the template to specific slice resource requirements, such as the number and types of sub-slices, PNF, VNF, central processing unit (CPU), I/O, memory, storage, etc. The sub-slice components are translated to resources of a TD. Actually, these resources of a TD can be computing (such as CPU, I/O), storage, radio (evolved Node B (eNB), Central Unit - CU, Distributed Unit - DU, Remote Radio Head - RRH / Remote Radio Unit - RRU), and transport (e.g., Virtual Local Area Network (VLAN), Virtual Private Network (VPN)). We assume that a slice is composed of several TDs, noted as $NS = \{TD_1, TD_2, TD_3, \ldots, TD_n\}$. For each TD, SP describes the needed resources according to the slice type. For instance, in the computing resource domain, they could include the number of CPUs, number of Virtual Machine (VM) or container instances, etc. For the radio domain, resources could be related to the functional split type, the MAC scheduler algorithm, the number of Physical Resource Blocks (PRB), etc. On the other hand, transport domain resources may include the type of link (bandwidth, latency), number of VLANs, front haul link capacity, VPN links, QoS, etc. We define $R(TDi) = \{p_1, p_2, p_3, ..., p_m\}$, as the set of parameters requested by TD_i. In this context, a trusted model is needed to guarantee the creation and the instantiation of the sub-slices to build the end-to-end network slice, particularly knowing the presence of different business actors, namely V or slice owner, SP, RP.

We illustrate in Fig. 1 a novel trust architecture that allows negotiating and deploying an end-to-end network slice. The proposed architecture includes all the above-mentioned actors in addition to a third-tier entity that creates and maintains the reputation of RP, namely Resource Provider Trust. SP owns the Blockchain layer, which is used to generate and negotiate contracts between the slice and RPs. The negotiation between the SP and RPs is done per sub-slice. The sub-entity at the SP in charge of this negotiation is the resources broker. The latter selects the different RPs, per TD, that maximizes a specific objective function. An example of the latter could be the reduction of the deployment cost [3]. In this work, we envision a sub-slice deployment brokering mechanism as series of small contracts. Each contract has a unique identifier and some data fields and can also perform actions such as creating a new contract or updating the Blockchain state. Contract actions are triggered by on-chain data updates (i.e., the creation of a new contract). Each sub-slice generates a contract. The end-to-end slice is ready for deployment once all the contracts regarding its sub-slices are negotiated and finalized.

3.1.2 Sub-slice Brokering

Once each sub-slice is described in terms of resources (i.e., R(TDi)), the request for each sub-slice is sent to the Blockchain layer. Each sub-slice will generate a contract to be negotiated. Once the query for a sub-slice arrives at the Blockchain layer, a Sub-Slice Contract (SSC) is created and published. This contract specifies the necessary resources needed by the sub-slice (i.e., R(TDi)) and the duration of the sub-slice. The different RPs are notified of the new SSC. Here, all the SSCs are visible on the Blockchain. RPs respond by publishing Sub-Slice Deployment Costs (SSDC), which specify the cost that the RPs are willing to charge for providing the necessary resources for each component of the sub-slice R(TDi). The Resource Provider Trust module records the trust value of the corresponding resource in the SSDC as appended information in the Blockchain layer. The original SSC collects all the related SSDC and arbitrates according to specific objectives (e.g., cheapest, best in terms of quality, or other criteria). All other contracts are terminated, and the winning contract is used to deploy the sub-slice components. All the related information about the sub-slice deployment are recorded in the Blockchain managed by SP. Relevant information on the different interfaces allowing to access to the sub-slices, such as the stitching interface (the Resource Orchestrator (RO) interfaces and their description), are compiled in a Sub-Slice Deployment (SSD) document.

3.1.3 Sub-Slice Deployment

Once the resources are negotiated and transcribed in SLAs, which are established, on one hand, between the SP and V and on the other hand, between the SP and each RP. Details on how the SLA is established using a trust architecture will be described in section 3.2. Once everything is settled, the Slice Orchestrator (SO) handling the Life Cycle Management (LCM) of the deployed network slices, will use the RO interfaces indicated in each SSD to: (i) instantiate and create the sub-slice; (ii) stitch the sub-slice with the other sub-slices to build the end-to-end network slice. Note that each RP uses its domain RO to manage and orchestrate its resources. RP exposes interfaces to allow other ROs or the SO to interact with the local RO.

3.1.4 Resource Provider reputation

The Resource Provider Trust module relies on the precedent experiences with RP, particularly on the respect of the SLA signed with a SP for a served TD resource. Indeed, a RP manages and provides

resources for more than one TD. To this aim, the Resource Provider Trust module monitors all SLA signed between a SP and a RP for sub-slices deployment. According to the monitored information on SLA, a Rep(i,j) of RP_i for serving TD_j is maintained and updated. The Rep(i,j), and hence the trust of RP when providing a resource of TD_j, is a function of the respected SLA by RPi. The reputation of a RP increases if a signed SLA by RP is respected and decreases otherwise.

$$Rep(i,j) = min(\Re p_{max}, Rep(i,j) + \triangle R) \textit{ if the SLA is respected.} \tag{1}$$

$$Rep(i,j) = max(\Re p_{min}, Rep(,ji) + \alpha.\triangle R) \textit{ if the SLA is not respected.} \tag{2}$$

Where $\triangle R$ is the reputation increase step, $\alpha.\triangle R$ is the reputation decrease step, $\Re p_{max}$ is the maximum value of the reputation, and $\Re p_{min}$ is the minimum value of the reputation. Details on the SLA monitoring will be introduced in the coming sections.

3.1.5 Algorithm of the resource selection

As stated earlier, the SP relies on a resource broker to select RPs that will serve the different sub-slices for each TD. In this work, the resource broker will implement a local algorithm that selects the best offer from the received SSDC. The proposed algorithm uses two criteria for the selection, namely, cost and Resource Provider reputation. The cost is included in the SSDC, while the reputation is obtained from the Resource Provider trust entity (i.e., Rep(i)).
The problem of selecting the RP for the resource of TD is to find an optimal trade-off between reducing the cost of deployment while increasing the Resource Provider's reputation. We formulate this problem using a Linear Program (LP). First, we denote by $C_{i,j}$ the cost of TD_j resource of RP(i), and Rep(i,j) is the reputation of RP(i) when serving resource of TD_j. Second, we assume that the network slice is composed of K sub-slices that need to be deployed using K TD resources. In addition, we assume that L RPs have responded to the SSCs published by the resource broker. We define a binary decision variable $x_{(i,j)}$ that indicates whether TD_i is deployed on resources provider j.
Knowing that the objective of the resource broker is to minimize the deployment cost and maximize the reputation of the user RP, we can formulate the problem as follows:

$$Minimize \sum_{i=1}^{K} \sum_{j=1}^{L} [\alpha(C_{(i,j)} x_{(i,j)} / C_{max}) - (1-\alpha)(Rep_{(i,j)} x_{(i,j)} / Rep_{max})] \tag{3}$$

Where the constraints are:

$$\sum_{j=1}^{L} x_{(i,j)} = 1 \tag{4}$$

$$x_{(i,j)} \in \{0,1\} \tag{5}$$

$$\alpha \in \{0,1\} \tag{6}$$

Equation 3 reflects the objective function that aims to minimize the cost and maximize the reputation. A weight (α) is used in order to balance between the two objectives. A higher value of α means that the targeted solution prioritizes the reduction of the cost, while a low value of α indicates that the solution is giving more priority for maximizing the reputation. Rep_{max} and C_{max} are the maximum values of Rep(i,j) and Ci_j. They are used to normalize the reputation and cost values to be used in the same objective function.
Constraint 4 ensures that a sub-slice resource of TD_j is deployed only once. Constraints 5 and 6 guarantee that $X(i,j)$ variables are binary and α is between 0 and 1, respectively.

By solving the ILP, the resource broker can select the optimal RP in which a sub-slice should be deployed. In section 4.1, we will study the solutions obtained by solving the ILP with Gurobi solver.

3.2 A Trust architecture for SLA management

3.2.1 SLA establishment

Once the resources are negotiated and the SP has selected RPs in which each sub-slice will be deployed, SLAs have to be established. On one hand, between the SP and V and on the other hand, between SP and each RP.
Generally speaking, SLAs are contracts between consumers and a service provider. An SLA specifies what customers can expect from a service provider. In 5G, SLA should reflect the QoS as related to the pre-defined type of slices, i.e. mMTC, eMBB, and uRLLC. The established SLA needs to be managed and monitored in order to ensure that the service is well functioning, and hence build the reputation of RPs.

The SLA management can be divided into eight phases, as shown in Fig.2. The first SLA establishment is between V and SP, while the second establishment is between SP and RPs. In the first step, V identifies which SP can deliver the necessary resources that meet a network slice's needs.

The second step is the negotiation between SP and V. The two actors should agree on the terms, target performance level for the end-to-end network slice, and the trust monitoring system. When one party does not agree, then the process has to start again until a consensus is reached between the different actors. The third step is the established agreement between V and SP. It involves the structure and definition of the SLA template that is going to be placed. In this phase, the two parties (V, SP) sign the contract and agree about the obligations and penalties. The fourth step is the negotiation between SP and RPs. SP should select RPs to serve sub-slices, hence to deploy the end-to-end slice. SP should select K TDs from L RPs. They should agree on the terms, target performance level for the sub-slice, and the trust monitoring system.

The fifth step is the established agreement between SP and RP. It involves the structure and definition of the SLA template that is going to be placed. In this phase, the parties (SP, RP) sign the contract and agree about the obligations and penalties. It is worth noting that steps four and five are repeated for each sub-slice to deploy.

The sixth step is the SLAs violation monitoring, which consists of monitoring the slice and sub-slice performance metrics. This step is important to detect if SP or RPs are meeting the defined performance level signed in SLAs. When the service is not as expected, V must be compensated by the SP, and the latter must be compensated by RP that did not respect the SLAs. All parties, i.e., V, RPs, and SP, should actively monitor the metric of the slice and the sub-slices using a trust monitoring system, which should provide reliable measurements. The last step is the SLA termination. Based on the terms defined in the definition of SLA, the termination of an SLA happens when the SLA validity expires or when an SLA violation is detected. Also, they end when V wants to finish this service.
On the other hand, a calculation of the compensation in the case of SLA violation should be calculated automatically. Finally, in the phase of the penalties for SLA violation, the payment of the compensation, calculated based on the penalties defined in the SLA will be transferred to the account of V and the SP automatically.

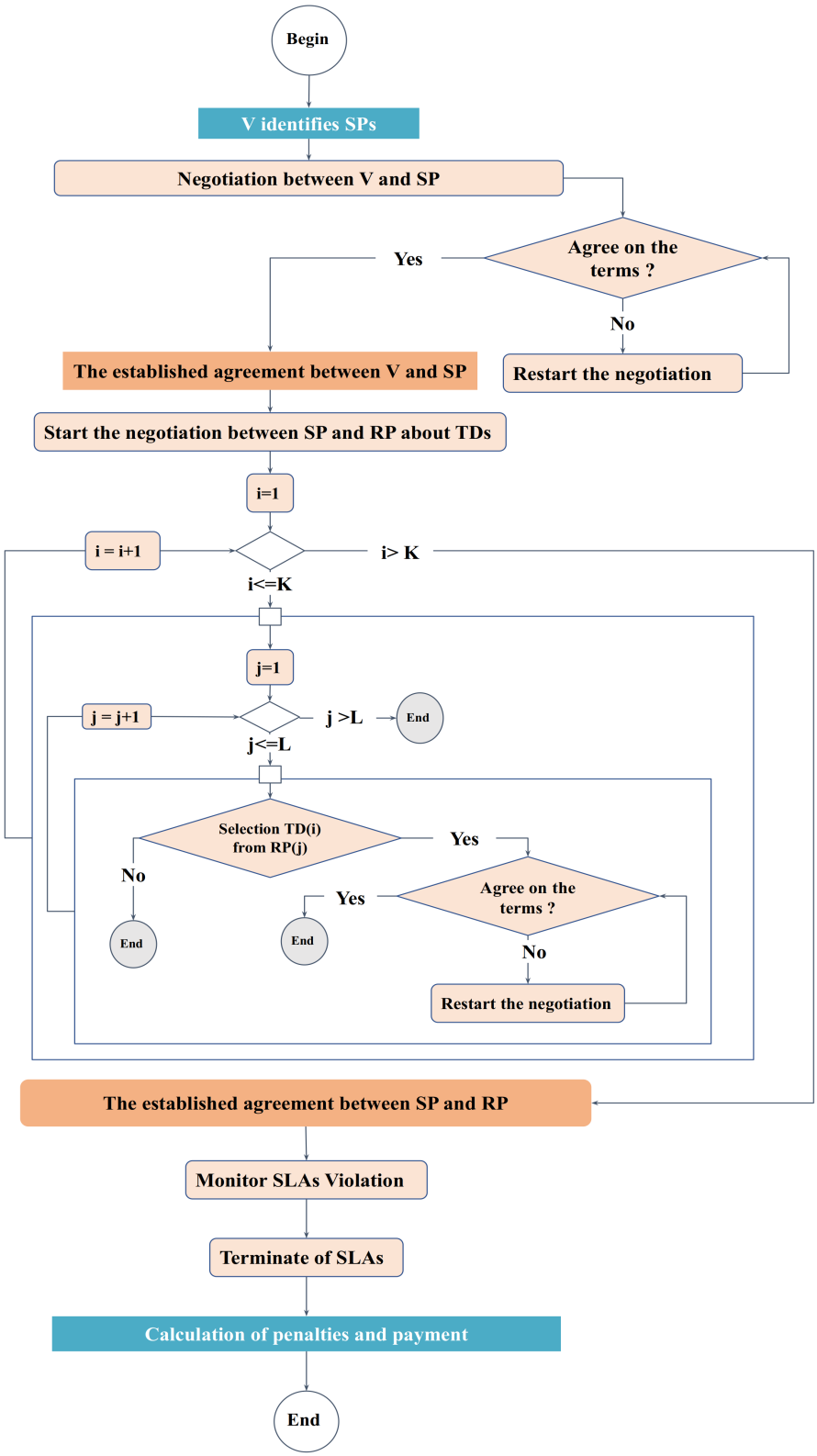


FIGURE 2: The life cycle of SLAs management

3.2.2 SLA Structure

Similarly to any contract, SLA is a structured form of elements set (mandatory and optional), namely the period of validity, the parties who signed the SLA, the services type, guarantees in terms of objectives, penalties, suspension, or termination of the contract and compensation.

The period of validity: As shown in Fig. 3, the period of validity of a SLA is defined by the date of beginning and the end of the network slice lifecycle. The latter can be modified in accordance with the conditions of termination.

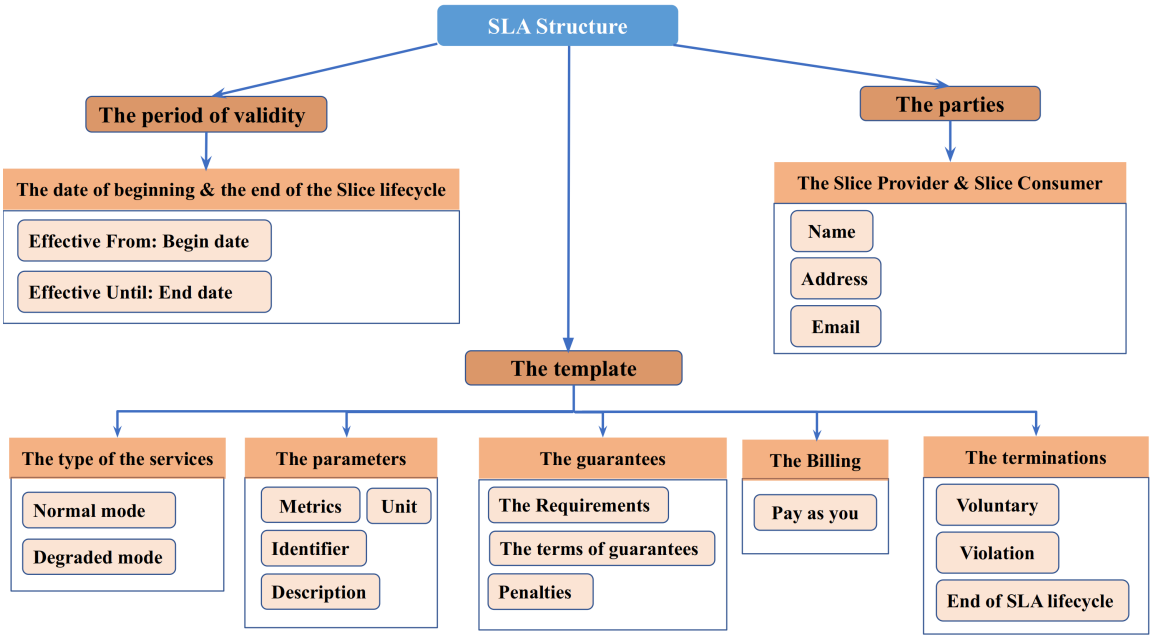


FIGURE 3: Service Level Agreement Structure

The parties: This section describes the parties involved in the contract, i.e., the three signatory parties: V, SP, and RPs. It includes the signatory parties as well as the trusted fourth party, i.e., the trust monitoring system, which is used to prove that the wanted service worked as expected, or was unavailable or less than the performance level signed in SLA. This part presents the properties of the parties, such as, the name, and the contact elements. In the studied case of Network Slicing in 5G, the first SLA is established between V and SPs. Later on, other SLAs are established between the SP and the selected RPs.

The template: It presents an essential part of the SLA. The template defines the parameters to ensure that the QoS offered are realistic and achievable. This part includes five elements: the services type, the parameters, the guarantees, the billing, and the termination condition.

The type of services: In 5G, there are different types of SLA signed. In this new generation, we have different types of slices with distinctive prioritization. Actually, 5G presents three network slice types, based on the typical characteristics required for use cases and Vs. The first type is eMBB (enhanced Mobile BroadBand) slice which is basically an extension of the 4G mobile broadband service. The second is uRLLC (ultra-Reliable Low Latency Communications) which

provides low-latency and reliable communication. The third type is mMTC (massive Machine Type Communications) which supports massive IoT devices with narrow bandwidth requirements. Also, Slices are independent from each other and provide unique services without interference with other slices. The core network slicing technology will facilitate the diversity of the new services offered. The variation of each slice priority needs dissimilar SLAs with different services. Moreover, the service provided can have one or more operating modes. For example, SP can offer a specific service with two modes: "normal mode" and "degraded mode". The advantage of degrading functionality is that SP can better plan its resources in short-term load peaks, without bothering V. Also, the service price will be adjusted according to the modes and is used to be attractive to V and profitable for SP.

The parameters: The parameters define the variables used in other contract sections by presenting particular elements such as the metric and the monitoring types. The metric is defined according to an identifier, a description (e.g., latency, throughput, reliability), and a corresponding unit (e.g., ms, bps, %). The monitoring specifies the nature of the evaluated metric such as average, maximum, and minimum. It also presents the evaluation window and the frequency of the values collection (period of the monitoring). As an example, for uRLCC slice, it is important to control the maximum value of latency.

The guarantees: This part presents three elements: requirements, terms, and penalties.

- The Requirements:

It presents the essential specifications for the slice's good functionality; a specification may be mandatory or optional. Moreover, it describes one or more preliminary technical elements to be completed, such as the version of a sensor used to connect to the mMTC slice.

- The terms of guarantees:

The terms of guarantees detail the service level objectives (SLO) of the contract. A guarantee is a composition of terms (e.g., reliability, latency, bandwidth, throughput) and objectives via operators such as "Or" and "And". As an example, for the good performance of an uRLLC slice, V needs 4 objectives: the bandwidth of the slice should be between 20 Mbps (as a minimum threshold) and 50 Mbps (as a maximum threshold), "and" also the slice reliability should be equal to 99.9999 %, "and", End-to-End latency should be between 20 ms and 100 ms, from RPs until the end-user, "and" finally, the throughput of eMBB slice be required to 100Mbps and 1Gbps.

- Penalties:

In case of a degradation of the end-to-end network slice performance (i.e., SLA violation), SP must pay compensation to V. In turn, the RPs involved in the SLA violation due to sub-slice resource degradation must compensate SP. The amount of reimbursement of each term is agreed upon among the parties and described in the penalties section of the SLA. As an example, when the latency of the end to end slice exceeds the threshold, it means that one or several RPs are not respecting the performance signed in the SLA, which leads that the end-to-end SLA is not respected. Accordingly, SP firstly pays compensation to V. Then, RPs responsible for this bad performance of latency will pay compensation to SP.

The Billing: In general, billing follows two types: "a pay package" or "pay as you go". In our case, we use the billing "pay as you go" which means that the service's price depends on the operating modes and penalties.

The terminations: A contract can be terminated automatically at the end date of SLA. The type of termination in this work could be as follows:

- Voluntary: when the V wants to finish this service.
- Violation: when the monitoring system detects a high level of violation.
- End of SLA lifecycle: when the validity of SLA ends.

3.2.3 SLA management

In this section, we will introduce the trust management architecture to manage the SLAs among V, SP, and RPs. The proposed architecture is illustrated in Fig.4. It complements Fig.1 with two new entities in respect to SLA management, namely the monitoring system and the Smart Contract. The monitoring system is a third-tier trusted entity that monitors the KPI as specified in the SLA established between V and SP and also between SP and RPs. The monitoring information will be used to check if an SLA is violated. The Smart Contract will contain all the signed SLAs and related information such as validity period, target performance level, price, compensation value, and relevant information. It stores the addresses of SP, V, RPs, monitoring system, and Resource Provider trust. These addresses are used to check the account balance, transfer funds, report SLA violations to the Resource Provider Trust entity, and allow only authorized addresses to interact with the Smart Contract.

Smart Contract: In the proposed architecture, the Smart Contract is managed by the SP. According to the monitoring system's information, it (Smart Contract) checks if one of the involved entities is violating the SLA. Moreover, it automatically gives compensation to V if the service is not satisfactory, and finds which RP has failed to support the performance defined in SLA. The concerned RPs will be charged automatically to pay penalties to the service provider in this case. It is worth noting that the compensation and penalties to pay are committed by all parties when the SLA is signed. To this aim, the Smart Contract includes functions that calculate the number of detected violations in an interval. Also, it is used to verify if the compensation interval has ended. At the end of the SLA life cycle, the compensation is paid based on the number of violations. Finally, a compensation function is called to transfer the compensation value for both V address and the SP address.

In our case, Smart Contract uses algorithm 1 to allow the dynamic compensation for V and SP. Here, the Smart Contract is used for an automatic subscription payment without the need of a TTP (Third Party Trust). First, the Smart Contract contains SLA information about the validity period, threshold of terms such as latency and throughput, initial SLA price, and compensation value per term. The Smart Contract also contains addresses of SPs, V, and RPs, as well as the monitoring system, in order to send the price and the compensation to the right parties. Second, the Smart Contract uses two special functions; a first function to check the account balance and to calculate the compensation, and a second one to transfer funds to the address. The first function works as follows: when the measured value sent by the monitoring system is above the target performance level defined in the SLA, and the SLA life-cycle is not over yet, the number of violations will increase by one.

These functions rely on time and use the block timestamp as a reference for the current time. As the Smart Contract is not able to automatically execute the function by itself, the monitoring solution must periodically call the Smart Contract to verify if the SLA is still valid or not. If the current block timestamp is above the SLA end time, the Smart Contract verifies if there is compensation to be paid. If not, it transfers the remaining Smart Contract balance to SP and RPs. Else, when the terms exceed the threshold during the life-cycle of SLA, a violation compensation value will be calculated.

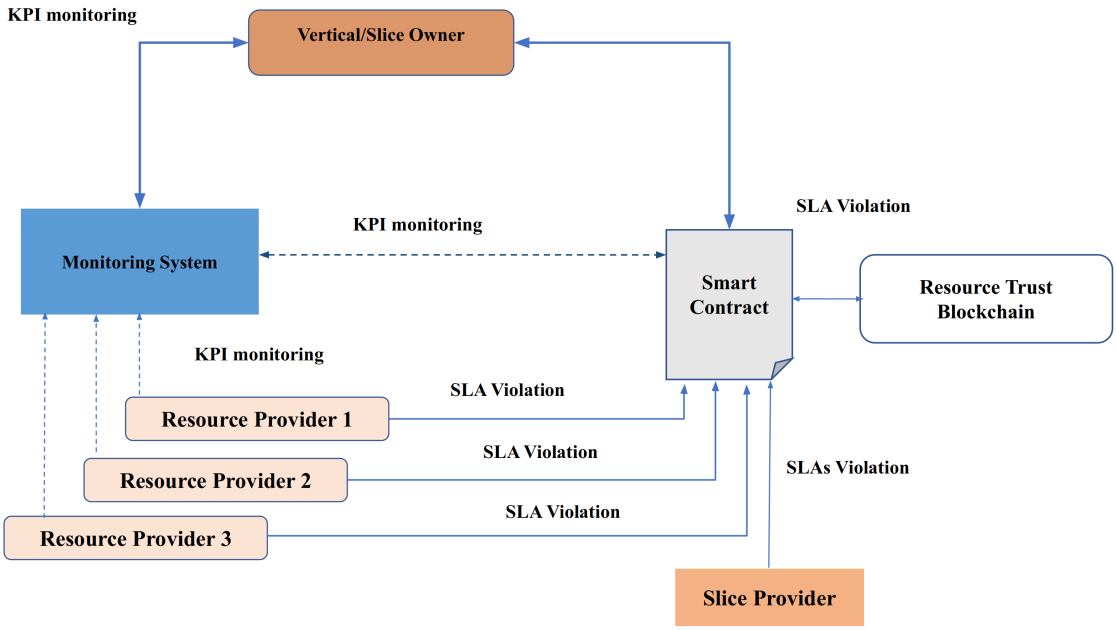


FIGURE 4: Trust architecture for end-to-end network slice creation

Then, the final price will be sent to both V and SP addresses stored by the Smart Contract, using the second function.

$$SumViolation(metric_k) = \sum SLAviolated(metric_k) \tag{7}$$

$$Comp(metric_k) = CompensationPerUnit(metric_k) * SumViolation(metric_k) \tag{8}$$

$$Compensation = \sum_{n=1}^{Number\_of\_metrics} Comp(metric_k) \tag{9}$$

Inside the Smart Contract, the dynamic calculation of the compensation value assumes that the monitoring solution performs measurements every second. During the SLA life-cycle, if the end-to-end slice's measured performance is different from the agreed performance level (threshold of latency or throughput in SLA signed between V and SP), then one or more RPs are not delivering the defined performance level. Thus, it is crucial to check which RP is responsible for this issue. Let's suppose that the slice is deployed using three TDs, where each TD is provided by one RP (RP1, RP2, RP3). In addition, our Smart Contract contains the SLA information, such as:

- *"Initial_SLA_price_V"* that is the initial price supposed to be paid to SP by V as defined in the SLA,
- *"Initial_SLA_price_RP1"* that is the initial price supposed to be paid RP1 by SP as defined in the SLA,
- *"T_Latency/Throughput_V"* that is the threshold of metrics (Latency/Throughput) defined in SLA signed between V and SP.

Algorithm 1 Algorithm of the Smart Contract

Input: V_L/T_RP1, V_L/T_RP2, V_L/T_RP3, V_L/T_V
▷ List of value send by the Monitoring System (MS)
Data: $CompensationPerUnitV$, $CompensationPerUnitRP1$, $CompensationPerUnitRP2$, $CompensationPerUnitRP3$, $Initial\_SLA\_price\_V$, $Initial\_SLA\_price\_RP1$, $Initial\_SLA\_price\_RP2$, $Initial\_SLA\_price\_RP3$, $T_{Latency/Throughput\_V}$, $T_{Latency/Throughput\_RP1}$, $T_{Latency/Throughput\_RP2}$, $T_{Latency/Throughput\_RP3}$, $addressVertical$, $addressSP$, $addressRP1$, $addressRP2$, $addressRP3$
▷ Initialization
Output: SLA_Price_V, SLA_Price_RP1, SLA_Price_RP2, SLA_Price_RP3
▷ Final SLA price will send from SP to V and Final SLA Price send from RP to SP
Function `CalculateCompensation`($V\_L/T\_RP1, V\_L/T\_RP2, V\_L/T\_RP3, V\_L/T\_V$):
  while *EndofSLA = false* **do**
    if $v(t, metric_{Latency/Throughput\_V}) \geq / \leq T_{Latency/Throughput\_V}$ **then**
      Number_SLA_violation_V =+ 1
      if $v(t, metric_{Latency/Throughput\_RP1}) \geq / \leq T_{Latency/Throughput\_RP2}$ **then**
        Number_SLA_violation_RP1 += 1
      else
      end
      if $v(t, metric_{Latency/Throughput\_RP2}) \geq / \leq T_{Latency/Throughput\_RP2}$ **then**
        Number_SLA_violation_RP2 =+ 1
      else
      end
      if $v(t, metric_{Latency/Throughput\_RP3}) \geq / \leq T_{Latency/Throughput\_RP3}$ **then**
        Number_SLA_violation_RP3 =+ 1
      else
      end
    else
    end
    *SLA_Price_V = Initial_SLA_price_V - (Number_SLA_violation_V * CompensationPerUnitV);*
    *SLA_Price_V_RP1 = Initial_SLA_price_RP1 - (Number_SLA_violation_RP1 * CompensationPerUnitRP1);*
    *SLA_Price_RP2 = Initial_SLA_price_RP2 - (Number_SLA_violation_RP2 * CompensationPerUnitRP2);*
    *SLA_Price_RP3 = Initial_SLA_price_RP3 - (Number_SLA_violation_RP3 * CompensationPerUnitRP3);*
  end
End Function
Function `SendPrice`($SLA\_Price\_V, SLA\_Price\_RP1, SLA\_Price\_RP2, SLA\_Price\_RP3$):
  if *EndofSLA = True* **then**
    send (SLA_Price_V) to ($addressSP$) ;
    send (SLA_Price_RP1) to ($addressRP1$) ;
    send (SLA_Price_RP2) to ($addressRP2$) ;
    send (SLA_Price_RP3) to ($addressRP3$) ;
  else
  end
End Function

For the sake of clarity, the proposed algorithm is not repeated for the two considered metrics. Hence, we will consider the following notation for the metrics: “Latency/Throughput”. For the eMBB slice, the metric to control is the “Throughput” while for the uRLLC slice, the latency is to control. Therefore, "Latency/Throughput" corresponds to throughput and “Latency”, respectively. If we aim to control more than one metric (latency and throughput) per algorithm, then we have to add another function similar to *"CalculateCompensation"* with different inputs.
It should be noted that:

- *"CompensationPerUnitV"* is the compensation value to be paid by SP to V in the case of one violation,
- *"CompensationPerUnitRP1"*, *"CompensationPerUnitRP2"*, and *"CompensationPerUnitRP3"* correspond to the value of compensation which will be paid to SP in case of one SLA violation by RP1, RP2 and RP3, respectively.

- *"addressVertical, addressSP, addressRP1, addressRP2, addressRP3"* are the corresponding addresses of the actors' accounts.

We use the function *"CalculateCompensation"* to verify the performance of the latency or the throughput in the RP1, RP2, and RP3 (see Algorithm 1). When the latency level or throughput level is different from the threshold defined in SLAs, then the number of violation will increase by one, and we update the total number as in the equation 7. After that, we have to calculate the compensation value ("Compensation"), which is derived using equations 8 and 9. The compensation is then sent to the right address using the "SendPrice" function. If metrics' values (latency or throughput) did not reach a threshold of SLA (e.g., 30 ms for latency), meaning no SLA violation, then the V will pay the entire price defined in the SLA to the SP. However, if the metrics' values exceed the threshold, then V will pay only the difference between the initial price and the paid compensation. Similarly, the SP will send only the difference between the defined price and the paid compensation to RPs. At the end, V receives a portion of the SLA price corresponding to the compensation from the SP, and the latter receives compensation from RPs. On the other hand, the Smart Contract will write the SLA violation in the Resource Trust Blockchain, indicating which RP has violated the SLA for which TD resource. The information will be used by Resource Provider Trust to update the reputation of the RPs, as described in the previous section.

4 Performance Evaluation

In this section, we use computer simulation to evaluate the performances of the trust architecture framework in terms of (1) end-to-end network slice negotiation and deployment; (2) SLA management.

4.1 End-to-end network slice resource selection

For the simulations, we consider that four RPs respond to the SP's SSC. We assume that each RP provides an offer to run sub-slices for three TDs. We have changed the value of α (weight) for each run to observe the impact on the trade-off (between cost and reputation) when selecting the RPs to host sub-slices.

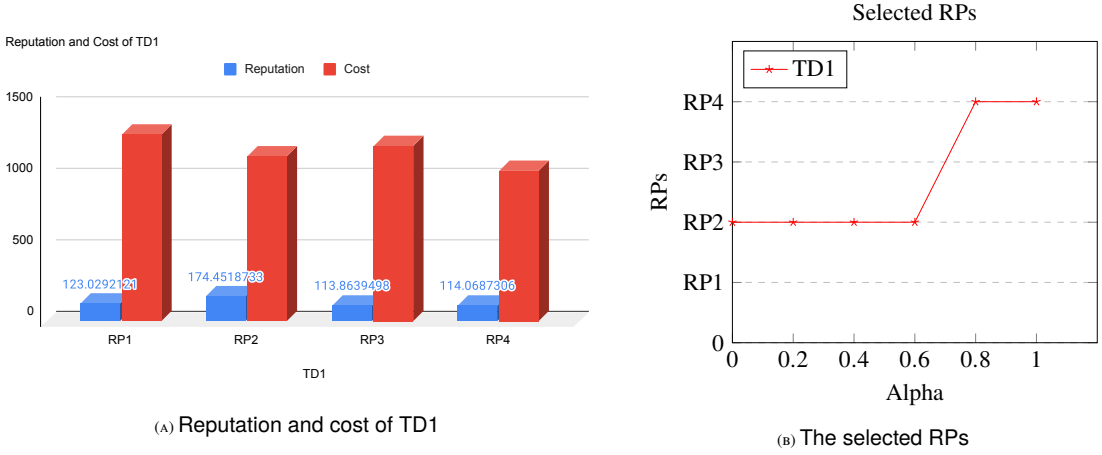
We use the Gurobi Jupyter Notebook to solve the ILP that allows selecting the RPs for each TD. Gurobi is useful for a wide variety of industries including manufacturing, financial services, energy and telecommunications as well as marketing campaign optimization and supply network design. As explained in the previous section, our objective function aims to find the resources based on their cost and reputation.

Table 1 summarizes the simulation results. First, we randomly generate cost and reputation for each couple (RP_i,TD_j). For each couple, the 2nd column indicates the reputation of (RP_i when serving resources of TD_j, while the 3rd column corresponds to the cost of RP_i to run a sub-slice in TD_j. For each value of α (from column 4th to 9th), 1 indicates that (RP_i has been selected to host a sub-slice of type TD_j; 0 otherwise.

To illustrate better the results of Table 1, we draw three Figures: Fig.5, Fig.6 and Fig.7. Each figure shows, per TD, the reputation and cost of the fourth RP and the selected RP for each value of α. For TD1 (Fig.5), we remark that RP2 is selected for low values of α (between 0 and 0.6), while for high values, RP4 is selected. We argue this by the fact that RP2 has the highest reputation and the second lowest cost. Hence, it is selected when the reputation is privileged over cost. Nevertheless, when α is higher than 0.6, i.e., the cost is prioritized, RP4 offering the lowest cost is selected. For TD2 (Fig.6), RP1 is always selected as it has the highest reputation and proposes the lowest cost.

TABLE 1: Results of simulation for the algorithm of resource selection while varying α.

(RP, TD)	Reputation	Cost	$\alpha = 0$	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$	$\alpha = 1.0$
(1, 1)	123.0292121	1303.276056	0	0	0	0	0	0
(1, 2)	415.0614412	848.8607873	1	1	1	1	1	1
(1, 3)	389.0051414	942.9657908	1	1	0	0	0	0
(2, 1)	174.4518733	1153.495538	1	1	1	1	0	0
(2, 2)	302.9867984	1837.715308	0	0	0	0	0	0
(2, 3)	266.1766331	542.1692793	0	0	0	0	0	1
(3, 1)	113.8639498	1224.255528	0	0	0	0	0	0
(3, 2)	361.1993909	1047.50991	0	0	0	0	0	0
(3, 3)	330.9289954	830.3358363	0	0	0	0	0	0
(4, 1)	114.0687306	1050.376848	0	0	0	0	1	1
(4, 2)	379.1780584	1241.622399	0	0	0	0	0	0
(4, 3)	350.4639782	548.1868774	0	0	1	1	1	0



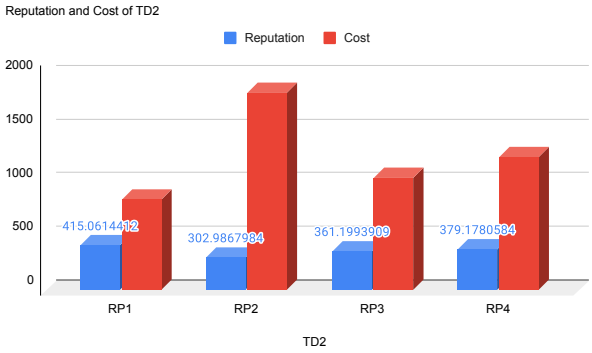
(A) Reputation and cost of TD1

(B) The selected RPs

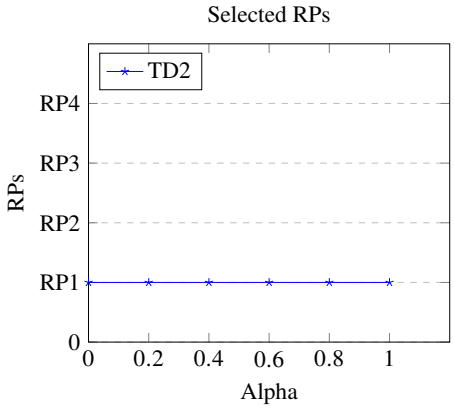
FIGURE 5: Resource provider selection for TD1

Regarding TD3, three different RP are selected. When α is low (between 0 and 0.2), i.e., reputation is privileged, RP1 is selected as it has the highest reputation to serve TD3 sub-slices. In contrast, when α is high (equal to 1), i.e., a lower-cost solution is privileged, RP2 is selected. The latter offers the lowest price. Finally, when a trade-off is sought (α between 0.4 and 0.8), RP4 is selected as it provides the better trade-off between reputation and cost.

Clearly, from these results, we deduce that the proposed selection algorithm to be run by the resource broker is able to find the best trade-off between maximizing the reputation and reducing the cost by well fixing α.

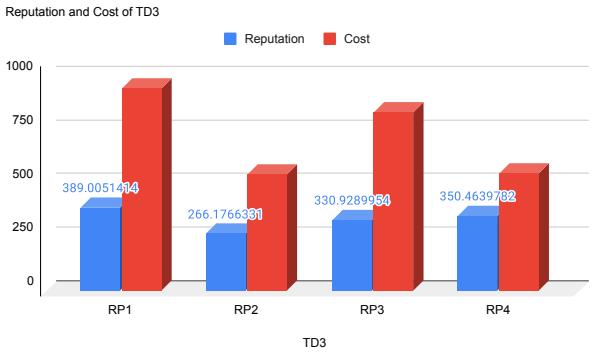


(A) Reputation and cost of TD2

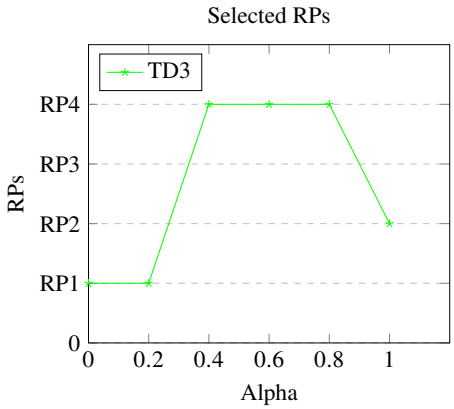


(B) The selected RPs

FIGURE 6: Resource provider selection for TD2



(A) Reputation and cost of TD3



(B) The selected RPs

FIGURE 7: Resource provider selection for TD3

4.2 Performance evolution of the Smart Contract

To evaluate the proposed SLA trust management framework, we choose two different metrics: the latency and throughput. In fact, these quantitative metrics are defined in SLAs, and it is possible to be controlled by the Smart Contract. We simulate a network slice deployed on three different TD; a different RP provides each one. V application is represented by a client and a server. We focused on measuring both the latency and bandwidth as key performances, which are periodically sent to SC by the monitoring system.

The deployment of the Smart Contract was performed using Ethereum [16], a Blockchain-based distributed computing platform. This program is executed by nodes on the Blockchain in the Ethereum Virtual Machine. Also, we use Ganache [17], a local Blockchain designed for developing and testing. It stimulates a real Ethereum network, including the availability of accounts number funded with 100 Ether by default. It presents an interface that can be accessed on a port of the localhost in the same way one would connect to a real Ethereum node. Also, we use truffle [18], which is a development environment for the compilation and deployment of Smart Contracts. Solidity [19] is the programming language used to write the contract code, and truffle looks for

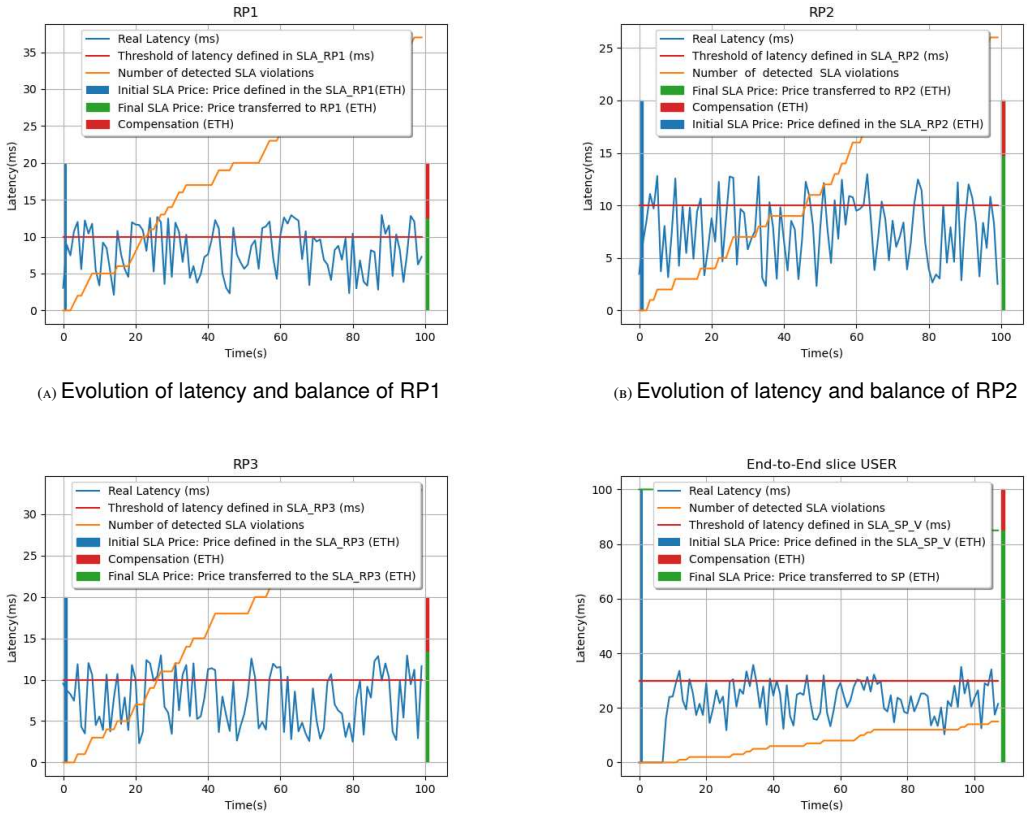
TABLE 2: Parameters of the Smart Contract.

Parameters	**Value**
Price SLA (V - SP)	100 ETH
Price1 SLA (SP - RP1)	20 ETH
Price2 SLA (SP - RP2)	20 ETH
Price3 SLA (SP - RP3)	20 ETH
Compensation	1 ETH
Compensation1	0.2 ETH
Compensation2	0.2 ETH
Compensation3	0.2 ETH
Threshold of latency defined in SLA_SP_V	30 ms
Threshold of latency defined in SLA_RP1	10 ms
Threshold of latency defined in SLA_RP2	10 ms
Threshold of latency defined in SLA_RP3	10 ms
Threshold of throughput defined in SLA_SP_V	100 Mbps
Threshold of throughput defined in SLA_RP1	100 Mbps
Threshold of throughput defined in SLA_RP2	100 Mbps
Threshold of throughput defined in SLA_RP3	100 Mbps
Validity	100 s

".sol" files to compile and migrate to the Blockchain. Moreover, we use Postman, which is an API testing tool. It can send the value of the metrics (throughput or latency) to the Smart Contract. The evaluation scenario stared from the monitoring system performing periodic requests. It sends the current values of the two metrics to the Smart Contract, every defined period. During a 100 seconds interval, the evolution of the latency performed in V is depicted in Fig. 8. The test works as follows, to evaluate the proposed solution. Firstly, the Smart Contract is deployed, and a new SLA is created following the parameters presented in Table 2. Secondly, the measurements of latency or throughput are considered the output of the trusted monitoring agent that performs periodic calls to the Smart Contract informing about SLA violations. If the monitored latency or throughput is above the threshold defined in the SLA; then the Smart Contract increases the number of SLA violations for the period until the end of the lifecycle of SLA. We have considered two use cases: when V requests uRLLC slice or requests eMBB slice.

4.2.1 uRLLC Slice

As specified by many uRLLC services in 5G, the needed latency is required to be between 5ms and 30ms. Based on these values, we fixed the violation threshold of latency for V to 30 ms, as depicted in Fig.8d with a red line. In addition, we suppose that the violation threshold of the latency in the



(A) Evolution of latency and balance of RP1

(B) Evolution of latency and balance of RP2

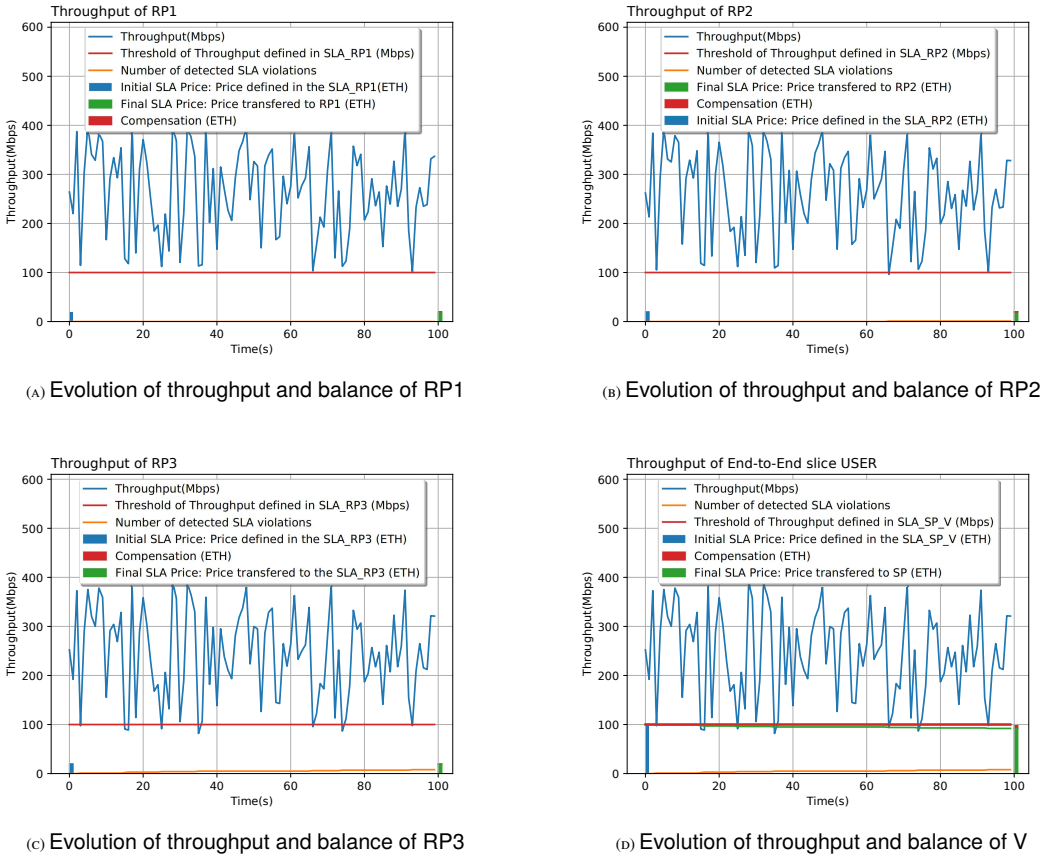
(C) Evolution of latency and balance of RP3

(D) Evolution of latency and balance of V

FIGURE 8: Evolution of latency and balance of RPs and V

RPs 1, 2, and 3 should be 10 ms, as depicted in Fig.8a, Fig.8b, Fig.8c with a red line. We argue this by the fact that we measure the end-to-end latency, which is composed of the latency experienced in each TD. During the 100 seconds interval, the evolution of the latency performed in the V and RPs is depicted in Fig.8; represented by a blue line. After that, the latency values in the end-to-end slice, RP1, RP2 and RP3, are iterated to verify each one against the target performance level defined in the SLA (i.e., 30 ms for V (Fig.8d) and 10 ms for RPs 1, 2 and 3 (Fig.8a,Fig.8b,Fig.8c). If the value is above the threshold, then the number of detected SLA violations will increase by one, which is depicted using an orange line in Fig.8. As a result, the initial price that is supposed to be sent to SP (Fig.8) will be decreased.

After the end of SLA, the billing model allows the dynamic compensation to V and SP and automatic payment, without need to TTP using the Smart Contract to realize these transfers. First, it is expected that V sends the price of the network slice before the start of the slice. The blue rectangle, in Fig.8, presents the initial price defined in SLA signed between V and SP. V deposits 100 ETH in the beginning. This fee is locked in the Smart Contract until the end of the SLA lifecycle. Once the SLA is finished, the funds are transferred to SP and RPs. The green rectangle presents the real price of SLA to be paid by V after the calculation process, and the red rectangle value depicts the compensation that will be paid to V. At the end of the SLA lifecycle, the compensation will be calculated based on the counter during the slice usage. Actually, the real price that will be paid at the end is the difference between the initial price and the compensation. In our case, the initial price is 100 ETH, and the compensation value per unit is 1 ETH. So, the final price to pay is 100 -



(A) Evolution of throughput and balance of RP1

(B) Evolution of throughput and balance of RP2

(C) Evolution of throughput and balance of RP3

(D) Evolution of throughput and balance of V

FIGURE 9: Evolution of throughput and balance of RPs and V

1 x 12 (number of violations) = 88 ETH. Consequently, the final price will be 88 ETH. Meanwhile, the monitoring system also sends the measured values of latency at RPs 1, 2 and 3, which are participating in the running of uRLLC slice. As shown in Fig.8a, Fig.8b, Fig.8c, when the SLA is starting, RP 1 should receive 20 ETH at the end of the SLA. But when the performance of latency in RP1, RP2 or RP3 exceeds the threshold (10 ms), their counters of the SLA violation will increase by one. At the end, RP1 will not receive 20 ETH, but it will receive the difference between the initial price and the compensation. The latter is calculated in the Smart Contract. Similarly, the same process will happen to RP2 and RP3, as shown in Fig.8 b and c, respectively.

4.2.2 eMBB Slice

Another example we considered is the eMBB slice, which needs a high throughput between 100 Mbps and 1Gbps. Based on these values, the violation threshold was fixed to 100 Mbps, which is depicted using a red line in Fig.9d. It is worth noting that unlike latency, the end-to-end throughput is not composed of the throughput observed in each domain; each RP should support the same throughput. The Smart Contract will be used to ensure that the throughput observed at V and provided by RPs is higher than this threshold. It can be seen in Fig. 9a, Fig.9b, Fig.9c that when the throughput is lower than the threshold, the counter of SLA violations will be increased by one, which is depicted using an orange line. Consequently, the initial price supposed to be sent to SP will be decreased, which is depicted using a green dashed line in the figures. The blue rectangle presents

the initial price defined in the SLA signed between V and SP. At the end of the SLA lifecycle, the compensation will be calculated based on the counter during the slice use. Actually, the real price that will be paid at the end is the difference between the initial price and the compensation. In our case, the initial price is 100 ETH, and the compensation value per unit is 1 ETH. So, the final price to pay is 100 - 1 x 8 (number of violations) = 92 ETH. Therefore, the final price will be 92 ETH. On the other hand, the monitoring system also sends the value of throughput provided by RPs 1, 2 and 3, participating in deploying the eMBB slice. When the SLA is starting, RPs 1, 2 and 3 should receive 20 ETH at the end of the SLA, as signed in SLAs. However, when throughput in RP1, RP2 or RP3 are less than the threshold (100 mbps), their counters of the SLA violations will increase by one. At the end, RP1, RP2 and RP3 will not receive 20 ETH, but they will receive the difference between the initial price and the compensation, which is calculated in the Smart Contract.

5 Conclusion

This paper has introduced a trust management architecture that allows negotiating and deploying end-to-end network slices and managing automatically established SLA among involved actors. The proposed framework relies, on one hand, on Blockchain and Smart Contracts to ensure security and anonymous transactions, and on the other hand, on an ILP-based optimization model to select the most adequate resource provides. Through simulations, we proved the ability of the proposed framework to select the optimal RP for each sub-slice. In addition, results showed that the SLA management process was successfully automated using a decentralized solution and removing a third tiers player's dependency to handle the billing process. One of the future extensions of this work is to address the challenge of trust monitoring, as it is highly needed to improve the trust framework devised in this paper.

Acknowledgment

This work has been partially supported by the European Union's H2020 MonB5G (grant no. 871780) project.

References

[1] Ibrahim Afolabi, Tarik Taleb, Pantelis A. Frangoudis, Miloud Bagaa and Adlen Ksentini. *"Network Slicing-Based Customization of 5G Mobile Services"*, IEEE Network, 2019.

[2] Petar Popovski, Kasper F. Trillingsgaard, Osvaldo Simeone, and Giuseppe Durisi. *"5G Wireless Network Slicing for eMBB, URLLC, and mMTC: A Communication-Theoretic View"*, IEEE Access, 2018.

[3] Boubakr Nour Adlen Ksentini, Nicolas Herbaut ,Pantelis A. Frangoudis and Hassine Moungla. *"A Blockchain-Based Network Slice Broker for 5G Services"*, IEEE Networking Letters, 6 May, 2019.

[4] Eder J. Scheid, Bruno B. Rodrigues Lisandro Z. Granville and Burkhard Stiller. *"Enabling Dynamic SLA Compensation Using Blockchain-based Smart Contracts"*, IEEE International Symposium on Integrated Network Management, 2019.

[5] Euromoney. *"How does a transaction get into the Blockchain?"*, https://www.euromoney.com/learning/blockchain-explained/how-transactions-get-into-the-blockchain, 2020.

[6] Widya Nita Suliyanti, Riri Fitri Sari. *"Evaluation of Hash Rate-based Double-Spending based on Proof-of-Work Blockchain"*, International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea (South), 2019.

[7] Li Zhetao, Kang Jiawen, Ye Dongdong, Deng Qingyong and Zhang Yan. *"Consortium Blockchain for secure energy trading in industrial internet of things,"*, IEEE Transactions on Industrial Informatics, 2017.

[8] Nada Chendeb, Nour Khaled, and Nazim Agoulmine. *"Integrating Blockchain with IoT for a Secure Healthcare Digital System,"*, 8th International Workshop on ADVANCEs in ICT Infrastructures and Services (ADVANCE 2020), Candy E, 2020.

[9] Jong Ho Noh and Hun-Yeong Kwon. *"A Study on smart city security policy based on Blockchain in 5G Age,"*, International Conference on Platform Technology and Service (PlatCon), Jeju, Korea (South), 2019.

[10] Nick Szabo. *"Smart Contracts"*, https://www.fon.hum.uva.nl/rob/Courses/eInformationIn Speech/CDROM/Literature/LOT winterschool2006/szabo.best.vwh.net/smart.contracts.html", 2020.

[11] Konstantinos Christidis and Michael Devetsikiotis. *"Blockchains and smart contracts for the internet of things"*, IEEE Access, 2016.

[12] Chaer Abdulla, Salah Khaled, Lima Claudio, Ray Partha and Sheltami Tarek. *"Blockchain for 5G: Opportunities and Challenges"*, IEEE Globecom, USA, 2019.

[13] Hui Yang, Yizhen Wu, Jie Zhang, Haowei Zheng, Yuefeng Ji, and Young Lee. *"Blockonet: Blockchain-based trusted cloud radio over optical fiber network for 5G fronthaul"*, Optical Fiber Communication Conference, San Diego, California United States, 2018.

[14] Hui Yang, Haowei Zheng, Jie Zhang, Yizhen Wu, Young Lee and Yuefeng Ji. *"Blockchain-based trusted authentication in cloud radio over fiber network for 5G"*, 16th International Conference on Optical Communications and Networks (ICOCN), Wuzhen, 2017.

[15] Babak Mafakheri, Tejas Subramanya, Leonardo Goratti and Roberto Riggio. *"lockchain-based Infrastructure Sharing in 5G Small Cell Networks"*, 14th International Conference on Network and Service Management, Rome, 2018.

[16] Baran Köseoğlu. *"Data Science Studies: Ethereum, Blockchain-Based Distributed Computing Platform"*, https://medium.com/colendi/data-science-studies-ethereum-Blockchain-based-distributed-computing-platform-eae96cde70f7, 2019.

[17] Stefan Beyer. *"What is Ethereum Ganache?"*, https://www.mycryptopedia.com/what-is-ethereum-ganache, 2019.

[18] Mayank Sahu. *"What is Truffle Suite? Features"*, https://www.upgrad.com/blog/what-is-truffle-suite/, 2020.

[19] Ethereum. *"Solidity"*, https://solidity.readthedocs.io/en/v0.7.1/, 2020.

[20] Brik Bouziane and Ksentini Adlen. *On Predicting Service-oriented Network Slices Performances in 5G: A Federated Learning Approach*, 2020 IEEE 45th Conference on Local Computer Networks (LCN), International conference on Local Communications Networks, IEEE Computer Society, os Alamitos, CA, USA, nov, 2020.

[21] Bakri Sihem, Brik Bouziane and Ksentini Adlen. *On using reinforcement learning for network slice admission control in 5G: Offline vs. online*, International Journal of Communication Systems, 34, 7, e4757, https://doi.org/10.1002/dac.4757, https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.4757, e4757 dac.4757, 2021.

[22] Kandaraj Piamrat, Adlen Ksentini, César Viho and Jean-Marie Bonnin. *QoE-aware vertical handover in wireless heterogeneous networks*, Proceedings of the 7th International Wireless Communications and Mobile Computing Conference, IWCMC 2011, Istanbul, Turkey, 4-8 July, 2011, 95–100, IEEE, 2011.

[23] Ahmed Amokrane, Adlen Ksentini, Yassine Hadjadj Aoul and Tarik Taleb. *Congestion control for machine type communications*, Proceedings of IEEE International Conference on Communications, ICC 2012, Ottawa, ON, Canada, June 10-15, 778–782, IEEE, 2012.