A Trust architecture for the SLA management in 5G networks

Sabra Ben Saad
Communication Systems, Eurecom
Sophia Antipolis, France
sabra.ben-saad@eurecom.fr

Adlen Ksentini
Communication Systems, Eurecom
Sophia Antipolis, France
adlen.ksentini@eurecom.fr

Bouziane Brik
DRIVE EA1859, Bourgogne university
Franche-Comté, France
bouziane.brik@u-bourgogne.fr

***Abstract*—It is well established that 5G will impact not only the end-users by allowing several new services, but also the vertical industry and network operators business. 5G will open the business market to new stakeholders with the introduction of Network Slicing, namely the vertical or tenant, the network slice provider, and the infrastructure provider. 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 Infrastructure Providers to enforce these end-to-end network slices. Accordingly, there is a need to establish Service Level Agreement (SLA) among these actors to ensure: (1) that the service is well-delivered to the vertical and (2) the infrastructure providers are respecting their involvement with the network slice provider. To fill this gap, in this paper, we propose a trust architecture to automatically manage the SLAs and apply penalties and compensations if the SLAs are not respected by one of the involved actors.**

***Index Terms*—Service level agreement, 5G slicing, Security, Smart contract, Blockchain.**

I. Introduction

5G network slicing enables the multiplexing of virtualized and distinct logical networks on the same physical network infrastructure. Network slices are isolated end-to-end logical networks, which are classified into three major categories: Extreme/Enhanced Mobile BroadBand (xMBB), Ultra-Reliable Low Latency Communication (uRLLC), and Massive Machine Type Communications (mMTC). Each of these categories is characterized by its specific performance requirements. An end-to-end network slice comprises three sub-slices: Radio Access Network (RAN), Core, and Transport networks [1]. The resources of a sub-slice are supplied by resource providers in order to build an end to end network slice. In 5G, several stakeholders collaborate to provide an end-to-end network slice across different Technological Domain (TD) to a vertical or slice owner. The slice provider creates an end-to-end slice for the vertical by leasing resources for each TD, and from different infrastructure providers. When a vertical or a slice owner requests a network slice creation, it uses a blueprint or a network slice template. The slice provider will translate this template to specific slice resource requirements for each sub-slice component or TD, such as needed computing resources, network resources, radio resources, etc. Once the end-to-end slice is created, SLA are signed between the vertical and the slice provider, and between the resource providers and the slice provider. An SLA specifies what customers can expect from a service provider. The SLA is used to check if a defined service is delivered as contracted and helps manage Quality of Service (QoS) degradation. An SLA defines QoS requirements, e.g., bandwidth, throughput, and latency, without specifying the technology to be used in order to deliver a particular service. In 5G, SLA should reflect the QoS related to the pre-defined slice types, i.e. mMTC, eMBB, and uRLLC. In this paper, we propose an SLA trusted management framework based on the Blockchain concept for Network slicing 5G-ready networks. The proposed framework has as objectives: (1) monitor the Key Performance Indicator (KPI) as specified in the SLA established between the vertical and the slice provider, and between the slice provider and the resource providers; (2) verify if an SLA is violated; (3) automatically compensate the vertical and the slice provider. The proposed framework has been validated via computer simulation, which shows its ability to detect SLA violation, and accordingly compensates both the vertical and the slice provider. To the best of our knowledge, only one work [2] has addressed the trust management of SLA in the context of a simple scenario of a cloud resource provider and a tenant. Although we share the concept of using Blockchain, our paper addresses 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 concepts of Blockchain and the Smart Contract, while Section III details our proposed SLA trust management framework. Section IV describes the performance evaluation of the proposed framework. Finally, conclusions are presented in Section V.

II. Blockchain and smart contract

The Blockchain [3] is a chain of block which is a database that stores information about transactions like the date, the persons participating in transactions, time, and amount transferred. A copy of the Blockchain (BC) is spread over many computers. These computers are called nodes that constitute 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.

Actually, when a transaction occurs, it will be checked by the node. For each transaction, a new block is created. This block is sent to every node in the network. If a transaction approved by the majority of the nodes using a special algorithms, then the block is added to the existing blockchain. Finally, an update is distributed across the network. So, it

maintains transaction records without a central authority with autonomously and automatic processes using Smart Contracts (SC). Blockchain can be used in several fields, such as healthcare, Internet of Things (IoT) and online shopping. Also, blockchain is envisioned for building a brokering mechanism [4] by a network slice provider in a 5G network.

On the other hand, smart contract is a feature associated with blockchain. The definition of a smart contract based on Nick Szabo taken from the original publication [5], 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. SC 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). More than that, the choice for such an SC language and blockchain is due to the fact that the calculation of a dynamic compensation requires complex calculations. Moreover, The smart contract contains many functions for the calculation and also the verification of the sender. In addition, the BC will facilitate the efficient and the secure operation of 5G Network Slice Broker.

III. PROPOSED APPLICATION ARCHITECTURE

A. SLA establishment

Before establishing the different SLA among the different stakeholders, there is a phase where the resources are requested by the vertical, and negotiated between the slice provider and the different resource providers. Mainly, this procedure is defined in [4] and is as follows. The vertical 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 slice provider will translate the template to specific slice resource requirements, such as the number and types of sub-slices, Physical Network Functions (PNF), Virtual Network Functions (VNF), CPU, I/O, memory, storage, etc. The sub-slice components are translated to resources of a Technological Domain (TD). A TD can be a computing resource domain (such as CPU, I/O), a storage domain, a radio domain (eNB, Central Unit - CU, Distributed Unit - DU, Remote Radio Head - RRH / Remote Radio Unit - RRU), and transport domain (e.g., VLAN, VPN). For each TD, the slice provider describes the needed resources according to the slice type. For instance, in the case of the computing resource domain, they could include the number of CPUs, VM instances, etc. For the radio domain, resources could be related to the functional split type [6], the MAC scheduler algorithm, the number of Physical Resource Blocks (PRB), and others. Transport domain resources, on the other hand, may include the type of a link (bandwidth, latency), number of VLANs, front haul link capacity, VPN links, QoS, etc. A brokering mechanism at the slice provider selects the different resource providers, per TD, that maximize a specific objective function. An example of the latter could be the reduction of the deployment cost. Besides, broker examples are given in [4]. Once the resources are negotiated, SLAs have to be established, on one hand, between the slice provider and the vertical; on the other hand, between the slice provider and each resource provider.

B. The trust architecture

1) SLA management: As explained in the previous section, SLAs have to be established between the vertical and the slice provider, and between the slice provider and the resource providers. 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 the resource providers. In this section, we will introduce the Trust management architecture to manage the SLAs between the above-mentioned entities.

The proposed architecture is illustrated in Fig. 1. The monitoring system is a third-tier trusted entity, which monitors the KPI as specified in the SLA established between the vertical and the slice provider, and between the slice provider and the resource providers. The monitoring information will be used to verify 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 slice provider, vertical, resource providers, monitoring system and resource provider trust. It is used to check the account balance, to transfer funds, to report SLA violation to the resource provider trust, and to allow only authorized addresses to interact with the smart contract.

2) Smart Contract: In the proposed architecture, the smart contract is managed by the slice provider. 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 the vertical if the service is not satisfactory and finds which resource

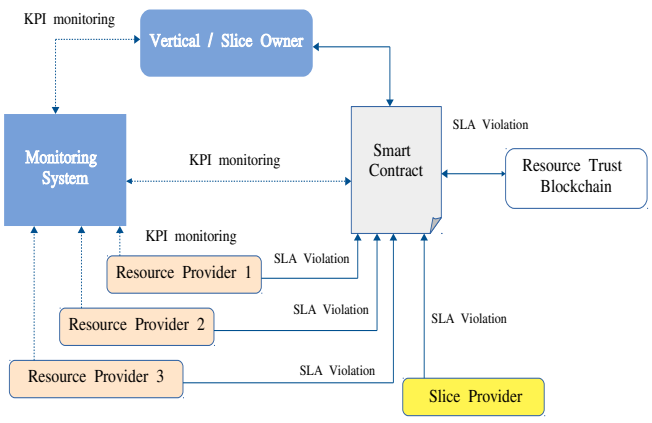


Fig. 1: SLA Management trust architecture

provider has failed to support the performance defined in SLA. The concerned resource providers 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 the vertical address and the slice provider address. In our case, smart contract uses algorithm 1 to allow the dynamic compensation for the Vertical, the slice provider and resource providers. Here, the SC is used for an automatic subscription payment without the need of a TTP (Third Party Trust). First, the SC contains SLA information about the validity period, Threshold of Terms such as latency and throughput, initial SLA price, and compensation value per term. The SC also contains addresses of the Slice Providers, Vertical, and Resource providers, as well as monitoring solution, in order to send the price and the compensation to the right parties. Second, the SC uses two special functions: one to check the account balance and to calculate the compensation, and one to transfer funds to the address. The first function works as follow: 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 violation will increase by one. These functions rely on time and use the block timestamp as a reference for the current time. As the SC is not able to automatically execute the function by itself, the monitoring solution must periodically call the SC to verify if the SLA is still valid or not. If the current block timestamp is above the SLA end time, the SC verifies if there is compensation to be paid. If not, it transfers the remaining SC balance to the SP (Slice Provider) and RPs (Resource Providers). Else, when the terms exceed the threshold during the life-cycle of SLA, a violation compensation value will be calculated and the final price will be sent to both the vertical and slice provider addresses stored by the SC, using the second function.

Inside the SC, the dynamic calculation of the compensation value assumes that the monitoring solution performs measurements every second. During the SLA life-cycle, when the measured performance in the end-to-end slice is different than the agreed performance level (threshold of latency or throughput in SLA signed between the vertical and slice provider), it means that one or more resource providers were not delivering the defined performance level. Thus, it is crucial to check which RP is responsible of this issue. Let's suppose that the slice is deployed using three TD, where each TD is provided by one RP. Also, our smart contract contains SLA information such as: "$Initial\_SLA\_price\_V$" that is the initial price supposed to be paid to the slice provider by the vertical as defined in the SLA, and "$T_{Latency/Throughput\_V}$" that is the threshold of metrics (Latency/Throughput) defined in SLA signed between V and SP. For the sake of clarity, the proposed

$$SumViolation(metric_k) = \sum SLAviolated(metric_k) \quad (1)$$

$$Comp(metric_k) = CompPerUnit(metric_k) * SumViol(metric_k) \quad (2)$$

$$Compensation = \sum_{n=1}^{NBmetrics} Comp(metric_k) \quad (3)$$

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 MS}
Data: $CompensationPerUintV, CompensationPerUintRP1, CompensationPerUintRP2, CompensationPerUintRP3, Initial\_SLA\_price\_V, Initial\_SLA\_price\_RP1, Initial\_SLA\_price\_RP2, Initial\_SLA\_price\_RP3, T_{Latency/Throughput\_V}, T_{Latency/Throughput\_RP1}, T_{Latency/Throughput\_RP2}, T_{Latency/Throughput\_RP3}, addressVertical, addressSP, addressRP1, addressRP2, addressRP3$
{Initialization}
Output: $SLA\_Price\_V, SLA\_Price\_RP1, SLA\_Price\_RP2, SLA\_Price\_RP3$
{Final SLA price will send from Slice Provider to Vertical and Final SLA Price send from Resource Provider to Slice Provider}
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 * CompensationPerUintV);*
    *SLA_Price_V_RP1 = Initial_SLA_price_RP1 - (Number_SLA_violation_RP1 * CompensationPerUintRP1);*
    *SLA_Price_RP2 = Initial_SLA_price_RP2 - (Number_SLA_violation_RP2 * CompensationPerUintRP2);*
    *SLA_Price_RP3 = Initial_SLA_price_RP3 - (Number_SLA_violation_RP3 * CompensationPerUintRP3);*
  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

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 Throughout and to "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 input. We use the function *"CalculateCompensation"* to verify the performance of the latency or throughput in the RP1, RP2 and RP3. When the level of the latency or throughput is different than the threshold defined in SLAs, then the number of violations will increase by one, and we update the total number as in equation 1. After that, we have to calculate the compensation value (Compensation), which is calculated using equations 2 and 3. The compensation is then sent to the right address using the function *"SendPrice"*. So, if the values of the metric (latency or throughput) did not reach a threshold of SLA (e.g., 30 ms for latency), then the vertical is not compensated and it will pay the entire price defined in the SLA to the slice provider. However, if the values of metrics are exceeding the threshold, then the vertical will pay only the difference between the initial price and the compensation. Similarly, the slice provider will send only the difference between the defined price and the compensation to the resource providers. At the end, the vertical receives a portion of the SLA price corresponding to the compensation from the slice provider and the latter receives compensation from the resource providers.

IV. PERFORMANCE EVALUATION

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. The vertical application is represented by a client and a server. We measured the latency and bandwidth as key performances, periodically sent by the monitoring system to the SC.

The deployment of the SC was performed using Ethereum [7], a blockchain-based distributed computing platform. This program is executed by nodes on the blockchain in the Ethereum Virtual Machine. Also, we use Ganache [8], a local blockchain designed for development 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 [9], which is a development environment for the compilation and deployment of smart contracts. Solidity [10] is the programming language used to write the contract code, and truffle looks for ".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 the vertical is depicted in Fig. 2. The test works as follows, to evaluate the proposed solution. Firstly, the SC is deployed, and a new SLA is created following the parameters presented in Table I. Secondly, the measurements of latency or throughput are considered

TABLE I: Parameters of the Smart Contract

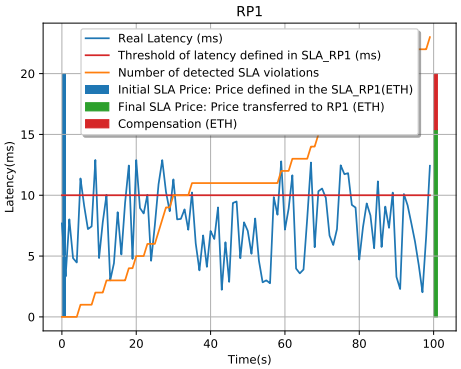
Parameters	Value
Price SLA (Vertical - Slice Provider)	100 ETH
Price1 SLA (Slice Provider - RP1)	20 ETH
Price2 SLA (Slice Provider - RP2)	20 ETH
Price3 SLA (Slice Provider - RP3)	20 ETH
Compensation	1 ETH
Compensation1	0.2 ETH
Compensation2	0.2 ETH
Compensation3	0.2 ETH
Threshold of latency defined in 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
Validity	100 s

the output of the trusted monitoring agent that performs periodic calls to the SC informing about SLA violations. If the monitored latency or throughput is above the threshold defined in the SLA; then the SC increases the number of SLA violations for the period until the end of the lifecycle of SLA. We considered two use cases: when the vertical requests uRLLC Slice or requests eMBB Slice

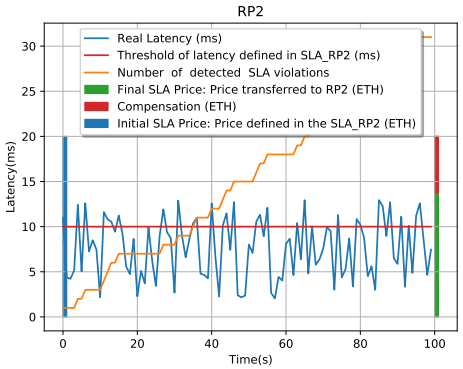
A. uRLLC Slice

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 the vertical to 30 ms, as depicted in Fig. 2d with a red line. Also, we suppose that the violation threshold of the latency in the resource providers 1, 2, and 3 should be 10 ms, as depicted in Fig. 2a, Fig. 2b, Fig. 2c 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 vertical and resource providers is depicted in Fig. 2; 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 the vertical (Fig. 2d) and 10 ms for resource providers 1, 2 and 3 (Fig. 2a,2b,2c). 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. 2. As a result, the initial price that is supposed to be sent to the slice provider (Fig. 2) will be decreased.

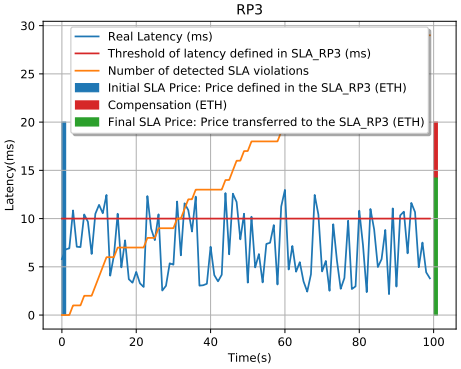
After the end of SLA, the billing model allows the dynamic compensation to the vertical and the slice provider and automatic payment without the need of a TTP using the SC to realize these transfers. First, it is expected that the vertical sends the price of the network slice before the start of the slice. The blue rectangle presents the initial price defined in SLA signed between the vertical and slice provider. The vertical deposits 100 ETH in the beginning. This fee is locked in the SC until the end of the SLA lifecycle. Once the SLA is finished, the funds are transferred to the slice provider and the resources providers. The Green rectangle presents the real price of SLA to be paid by the vertical after the calculation process, and the red rectangle value presents the compensation that will be paid to the vertical. At the end of the SLA lifecycle, the compensation will be calculated



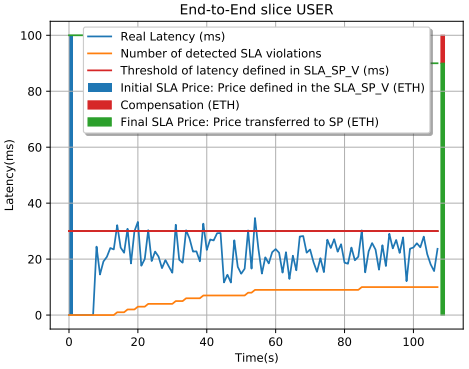
(a) Evolution of latency and balance of RP1



(b) Evolution of latency and balance of RP2



(c) Evolution of latency and balance of RP3



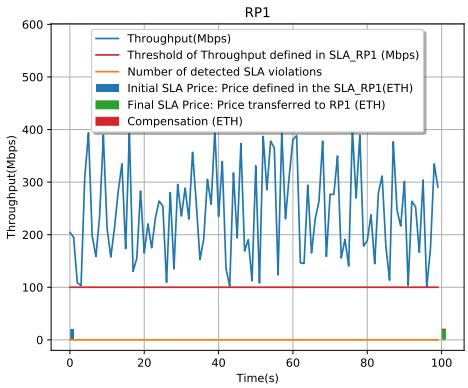
(d) Evolution of latency and balance of Vertical

Fig. 2: Evolution of latency and balance of resources providers and vertical

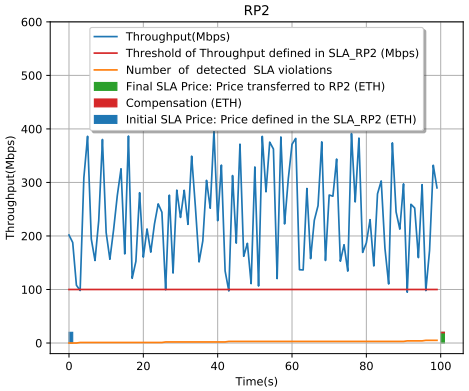
based on the counter during the slice usage. Actually, the real price that will be paid at the end is the difference between the initial price and the compensation. In our case, the initial price is 100 ETH, and the compensation value per unit is 1 ETH. So, the final price to pay is 100 - 1 x 10 (number of violations) = 90 ETH. Consequently, the final price will be 90 ETH. Meanwhile, the monitoring system also sends the measured values of latency at the resource provider 1, 2 and 3, which are participating in the running of uRLLC slice. As shown in Fig. 2a, Fig. 2b, Fig. 2c, when the SLA is starting, the resource provider 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. In 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. And the same process will happen to RP2 and RP3.

B. eMBB Slice

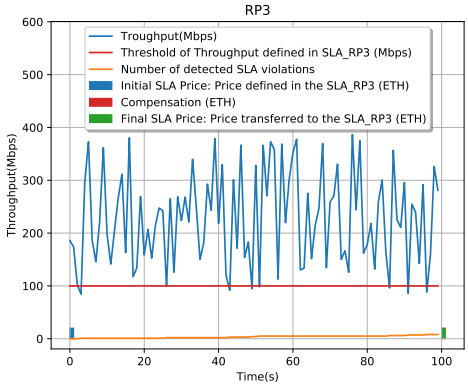
Another example we considered is the eMBB slice, which needs a high throughput between 100 Mbps and 1Gbps. Based on these values, the violation threshold was fixed to 100 Mbps, which is depicted using a red line in Fig. 3d. It worth noting that unlike latency, the end-to-end throughput is not composed of the throughput observed in each domain; each RP should support the same throughput. The SC will be used to ensure that the throughput observed at the vertical and provided by the resource providers is higher than this threshold. It can be seen in Fig. 3a, Fig. 3b, Fig. 3c 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 the slice provider 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 the vertical and slice provider. 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 the resource providers 1, 2 and 3 participating in deploying the eMBB slice. When the SLA is starting, the resource providers 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. In 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.



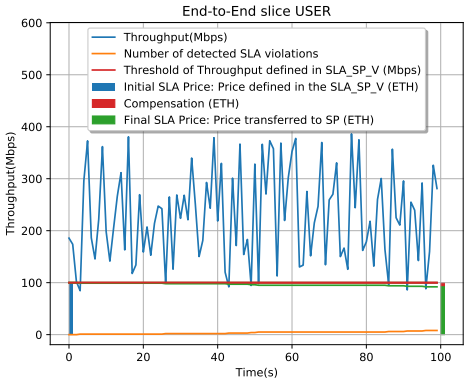
(a) Evolution of throughput and balance of RP1



(b) Evolution of throughput and balance of RP2



(c) Evolution of throughput and balance of RP3



(d) Evolution of throughput and balance of Vertical

Fig. 3: Evolution of throughput and balance of resources providers and vertical

V. CONCLUSION

This paper proposed an SLA Management trust architecture in 5G that aims at enabling the dynamic and automatic manages of the SLA during the network slice lifetime. The proposed approach based on smart contracts automatically manages the SLA, i.e., violation, billing process and payment by the vertical, and the compensation reimbursement by the resources providers and slice provider. Further, an implementation of the smart contract and blockchain was conducted to manage two QoS-related SLA (latency and throughput) examples. The obtained 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 introduced in this paper. Also, we envision to use the proposed architecture to build the reputation of the resource providers, when a resource provider replies to a sub-slice creation request, allowing a better selection of the resource providers by the network slice provider.

ACKNOWLEDGMENT

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

REFERENCES

[1] I. Afolabi, T. Taleb, P. A. Frangoudis, M. Bagaa and A. Ksentini, *"Network Slicing-Based Customization of 5G Mobile Services"* in IEEE Network, vol. 33, no. 5, pp. 134-141, Sept.-Oct. 2019.

[2] Eder J. Scheid, Bruno B. Rodrigues Lisandro Z. Granville, Burkhard Stiller, , *Enabling Dynamic SLA Compensation Using Blockchain-based Smart Contracts*, 2019 IFIP IEEE International Symposium on Integrated Network Management (IM2019), 2019.

[3] Euromoney, *How does a transaction get into the blockchain?*, 2019, https://www.euromoney.com/learning/blockchain-explained/how-transactions-get-into-the-blockchain, 2020.

[4] 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.

[5] Nick Szabo, *Smart Contracts*, 1994, https://www.fon.hum.uva.nl/rob/Courses/eInformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html, 2020.

[6] C. Chang, N. Nikaein, O. Arouk, K. Katsalis, A. Ksentini, T. Turletti, and K. Samdanis, "Slice orchestration for multi-service disaggregated ultra-dense rans," IEEE Communications Magazine, vol. 56, no. 8, pp. 70–77, 2018.

[7] 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

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

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

[10] Solidity, https://solidity.readthedocs.io/en/v0.7.1/, 2020.