How to Request Network Resources Just-in-Time using Smart Contracts

Tooba Faisal
King's College London, UK
tooba.faisal@kcl.ac.uk

Damiano Di Francesco Maesa
University of Cambridge, UK
dd534@cam.ac.uk

Nishanth Sastry
University of Surrey, UK
n.sastry@surrey.ac.uk

Simone Mangiante
Vodafone Group R&D, UK
simone.mangiante
@vodafone.com

Abstract—**5G promises unprecedented levels of network connectivity to handle diverse applications, including life-critical applications such as remote surgery. However, to enable the adoption of such applications, it is important that customers trust the service quality provided. This can only be achieved through transparent Service Level Agreements (SLAs). Current resource provisioning systems are too general to handle such variety in applications. Moreover, service agreements are often opaque to customers, which can be an obstacle for 5G adoption for mission-critical services.**

In this work, we advocate short-term and specialised rather than long-term general service contracts and propose an end-to-end Permissioned Distributed Ledger (PDL) focused architecture; which allows operators to advertise their service contracts on a public portal backed by a PDL. These service contracts with clear Service Level Agreement (SLA) offers are deployed as smart contracts to enable transparent, automatic and immutable SLAs. To justify our choice of using a permissioned ledger instead of permissionless, we evaluated and compared contract execution times on both permissioned (i.e. Quorum and Hyperledger Fabric) and permissionless (i.e. Ropsten testnet) ledgers.

Index Terms—**Distributed Ledger Technology, Smart Contract, Blockchain, 5G, Network Services**

I. Introduction

Network service requirements have changed considerably since the advent of 5G, and its promise to provide connections that have guarantees such as ultra-low low latency, high reliability or massive bandwidth. Such guarantees are required to enable life-critical applications such as remote surgery [1]. Moreover, the technological advancements and the need for automation in every industry have caused an exponential increase in the number of IoT devices, which are expected to reach 50 billion by 2030 [2]; hence, the surge in demand for network connectivity is inevitable.

In this evolving situation, network service operators face two major challenges: 1) It is hard to cope with the surge in demand for network resources, e.g., prior research has shown that service quality drops with an increased number of connections [3] and 2) Some connnections require *accountable* Quality of Service (QoS), e.g., a surgeon performing an emergency remote surgery over 5G during high congestion times needs a reliable connection with guaranteed bandwidth and latency bounds.

We propose a novel solution for the above problems with *Just-in-Time Resource Allocation* - that is, instead of fixed *long-term* contracts, operators offer dynamic *short-term* network service contracts with strict service level guarantees. The operators are expected to introduce more flexible and tailor-made service offerings considering their coverage and capacity and *if and only if* they can guarantee the Service Level Agreement (SLA) promised. This is advantageous for both operators and customers: operators can sell their peak-time contracts at higher prices and customers can shift their usage off-peak to get better value-for-money [4]. Moreover, it may be easier for operators to predict network behaviour in short-term rather than long-term.

Providing short-term and dynamic network connectivity along with strict and guaranteed SLAs for millions, probably billions, of devices is not an easy task. This approach will cause substantial management overheads, and customers will need to contact operators more frequently for new service contracts, causing additional staff and operating expenses. Another problem is that SLAs are often opaque to the customer, making it difficult for them to detect an infringement.

To solve the aforementioned issues, we exploited the inherent transparency, automation, and immutability properties of Distributed Ledger Technology (DLT) and proposed an end-to-end **A**ccountable **J**ust-**I**n-**T**ime **(AJIT)** system architecture previously [5]. In our proposal, service contracts are deployed as smart contracts on a Permissioned Distributed Ledger (PDL) that records all the service provisioning and can provides an impartial record for SLA accountability. Our initial work focused on the network layer of our architecture and examined overheads incurred by the network to allocate resources. This paper focuses on the application layer and presents a Just-in-Time architecture for service provisioning, which outlines the building blocks for user access to the ledger to get service contracts (§III). We further analyse and compare permissioned and permissionless ledgers and analytically justify our choice of PDLs in §IV with further discussions on the impact of the consensus mechanism on their execution. We discuss the future directions and conclude this paper in §V.

II. Related Work

This study is focused on Distributed Ledger Technology for *Accountable Service Level Agreements (SLAs)*. The two main classes of distributed ledgers are permissionless (e.g., Bitcoin and Ethereum [6]) and permissioned (e.g., Hyperledger Fabric [7] and Quorum [8]). In this work, we propose that customers choose between available service contracts on a permissioned ledger, interacting with the ledger through a DApp [9]. A similar concept of a mobile application which allows users to choose between network contracts is presented by [4], in which authors show that it is an advantage for both the operator and the customers: for customers to shift their usage to off-peak times and for operators to manage congestion by offering discounted services during less-congested times. However, this work neither addresses the problem of accountability nor allows the customers to choose between operators.

A blockchain-centred DApp for multi-domain architecture is presented by [10], which proposes smart contracts to be used as SLAs to enable transparency in contract agreements between administrative domains. Further proposals to apply smart contract as SLAs in the Cloud computing environment are presented by [11] and [12]. [11] is focused on SLA management with smart contracts in a decentralised computing environment, and Uriarte et al.'s [12] major focus is more inclined towards dynamic SLA management and they propose an architecture to transform SLA into smart contracts. Also, neither of them discuss resource provisioning through short-term and dynamic service contracts

In [13] the authors suggest adopting smart contracts to model simple service agreements for sharing network resources in a decentralised fashion among home and business as small-cell providers. Our proposal, instead, considers traditional service provisioning by operators controlled base stations.

As far as we know, no work addresses the problem of accountability and provides an end-to-end architecture of smart contract focused SLAs.

III. System Design

Our system proposal is to set up a guaranteed Service Level Agreement *just-in-time* between an operator and customer. This is done by recording the agreement as a smart contract deployed on an appropriate Distributed Ledger Technology (DLT).

The key design requirement is that the DLT should be able to support the expected number of transactions. Although transaction rates of permissionless block chains (e.g., Bitcoin allows 7 transactions per second and Ethereum 15 transactions per second [14]) might be insufficient, recent work [15] has shown that up to 20,000 transactions per second can be sustained with Permissioned Distributed Ledgers (PDLs). Therefore, we advocate the use of PDLs.

Three different type of entities interact with this system: 1) **Customers** require network services and request for contracts - they have limited access to the ledger through a DApp and don't take part in the consensus, 2) **Service Operators** advertise their services on a portal and allocate services to the customers upon request - they run the consensus and deploy the service contracts on the ledger 3) **Regulatory Authorities** only observe the whole system for any misbehaviour - they take part in the consensus.

Our system is built atop a specific type of DLT, Permissioned Distributed Ledger (PDL), maintained by a consortium of operators and regulatory authorities. We envision a DLT centered market place where operators can advertise their available services, backed by smart contracts deployed on a PDL. Most countries or legal jurisdictions only have a limited number of operators. Therefore, considering the nature of the architecture, we believe that a permissioned scenario is the best-fit.

The distributed consensus nature of DLTs guarantees that the system remains honest as long the required threshold of participants (dependent on the particular distributed consensus algorithm used) behaves honestly. This means that even if the network is maintained by the operators, they can not cheat customers without the existence of a big enough cartel of operators. Moreover, the transparent nature of the DLT allows the regulatory authorities to act as watchdogs. This, coupled with the fact that the operator nodes identities are known in advance, due to the permissioned nature of the DLT, allows for an accurate pinpointing in case of misbehaviour.

In addition to participants, the system model also contains a Distributed Application (DApp), see Figure 1. This application can be installed as a mobile or desktop application and will run a discovery service every specified time interval to maintain an updated log of available operators in the vicinity along with their offered contracts and corresponding service quality. Service providers advertise their service contracts, and the customers purchase them through this DApp. When a user wants to buy a service contract, the DApp will list all the available contracts from the operators as per user's preference. When a customer chooses a contract and pays for it, the DApp sends the payment to the contract escrow, where it is kept until the contract is completed, and notifies the operator to start the services.

The sequence of actions required to set up connectivity is depicted in Figure 2:

1) Contracts are advertised on a public portal accessible through the DApp, so that a customer can select one as per their requirements. Customers are required to approve an appropriate payment with the contract request. These funds will be held in escrow, to be paid to the operator upon successful delivery of service.

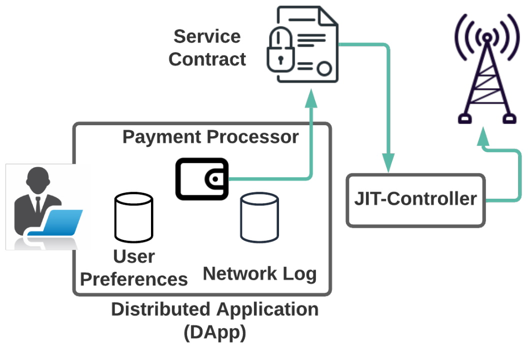


Fig. 1. Architecture of the proposed **D**istributed **App**lication (DApp) - It is run on users' device and acts as gateway to pull/push data from/to the PDL. JIT-Controller [5] performs network level resource allocation.

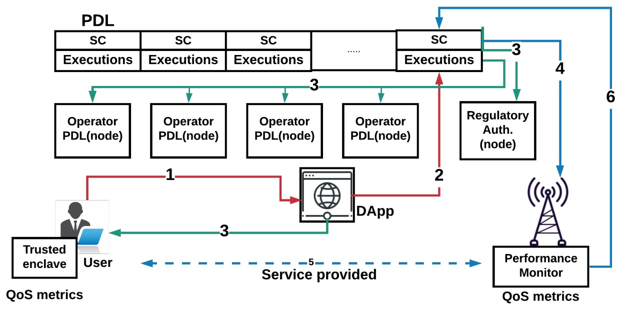


Fig. 2. Just-in-Time Resource Allocation - Architecture

2) Once the customer requests a service, the DApp sends an activation request, signed by the customer, to the corresponding smart contract residing in the PDL. Note that on top of any payment to the operator if the smart contract executes successfully, the customer incurs a cost simply to execute it (to prevent denial of capability attacks, where a malicious customer requests services and makes an operator reserve network resources without any intention of using and paying for the services).
3) The smart contract is executed, first checking with the operator whether there are available resources that can be reserved in order to provide the requested quality of service. Depending on the jurisdiction and the nature of the contract, regulators may also need to be informed about the potential service contract being agreed upon.
4) Once the operator confirms that it is able to deliver the service (and if needed, regulator approval is obtained), appropriate resource reservations are made on operator hardware such as base stations. Then, the initial state is setup to monitor the network connection during run time.
5) At this point, everything is setup, and the operator provides service. Details of the service provided are recorded in a dedicated state channel established between customer and operator.
6) At the end of service, the operator provides proof from the state channel that the agreed upon service has been delivered, and claims the service payment locked in escrow by the contract.

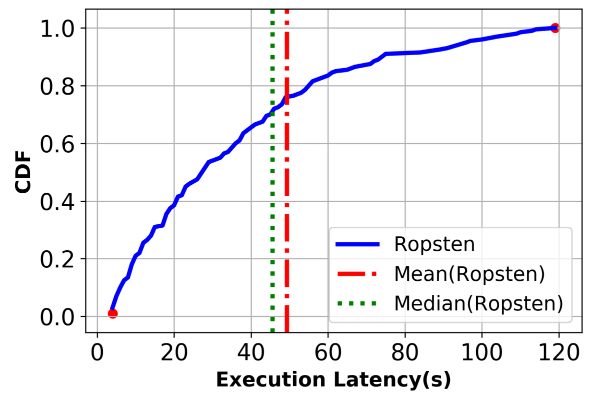
IV. Evaluation

As our study is focused on *Accountable Allocation of Network Resources*, to measure the viability of our proposal it is important to measure the overheads incurred by the employed PDL and smart contracts. The rationale behind adopting a PDL over a permissionless approach is explained in §III. However, to measure the practical impact, we deployed 200 service contracts on two different permissioned ledgers (i.e. Quorum [8] and Hyperledger Fabric [7]) and a permissionless ledger (i.e. the public Ethereum Ropsten testnet).

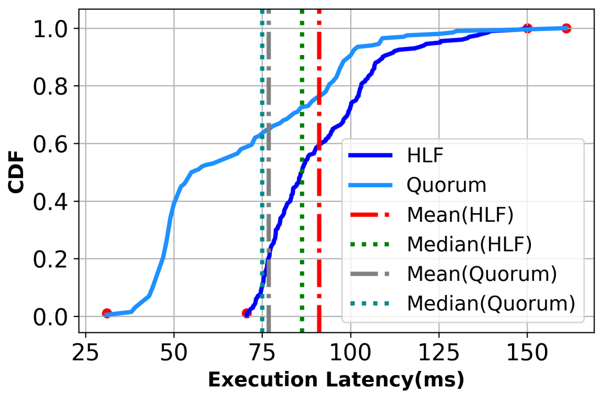
The example service contracts are executed on two local nodes of Hyperledger Fabric (Version 1.3.0) through a Hyperledger Burrow [16] running on a Ubuntu 64-bit, 16.04.7 virtual machine with 4.096 GB of RAM, and a 2.3 GHz Dual-Core Intel Core i5 processor. For Quorum we used an identical virtual machine as Hyperledger Fabric and installed two Quorum nodes with RAFT [17] consensus.

Overall, Quorum's Raft protocol is a leadership model, in which a single leader is elected to manage the replication of transactions to other nodes and is ideal for closed groups of nodes where fast block creation is required, ideally at the granularity of milliseconds [17]. In Hyperledger Fabric transactions are sent to endorsers first (in our case we have two endorsers) who endorse the transactions as per the endorsing policy of the ledger. Once the transactions are endorsed, they are sent to orderers (one orderer in our experimental setup), who use Apache Kafka to reach a consensus [18].

This difference in the speed of the two consensus algorithms is reflected in our analysis, in which Quorum outperforms Hyperledger Fabric on average. The mean execution time for Hyperledger Fabric is $\approx$ 91 ms which is a bit higher than Quorum's that is $\approx$ 68 ms. However, the standard deviation is $\approx$ 16 ms and $\approx$ 24 ms for Hyperledger Fabric and Quorum, respectively. That is, although Quorum has lower execution times than Hyperledger Fabric, the standard deviation is higher, meaning that more diversified execution times should be expected with Quorum. This is also visible from Figure 3(b), where Quorum is slower than Hyperledger Fabric in the worst case. This indicates that Hyperledger Fabric could be a more desirable alternative in application scenarios where the consistency of performances is more desirable than speed.



(a) Permissionless Ledger



(b) Permissioned Ledgers

Fig. 3. a) Execution latency in a permissionless environment - the min and max values are 4 sec and 119 sec. respectively b) Execution latency in a permissioned environment - The min and max values for **H**yper**L**edger **F**abric(HLF) are ≈ 70 ms and ≈ 150 ms respectively and for Quorum are ≈ 31 ms and ≈ 161 ms respectively. It is to be noted that in the permissionless case (a), the latency is in seconds, but for permissioned ledgers (b), latency is in milliseconds.

In addition to our prior work [5], we deployed another 200 contracts on Ropsten through a node with an Intel Xeon CPU E5-2660 (2.60 GHz with 20 cores) and 94G RAM. The analysis shows that it took an average of ≈ 49.3 sec to execute the same contracts on the Ropsten testnet with a standard deviation of ≈ 35 sec.

The difference between contracts execution times measured on permissioned and permissionless ledgers is an entire order of magnitude, confirming the performance advantages of permissioned ledgers. Another factor, however, is that the permissionless experiment was conducted on a possibly congested public testnet, while both permissioned experiments were set up on dedicated private testnets.

V. Conclusion and Future Work

This paper envisions a new-era of modern resource provisioning mechanisms, where both customers and operators benefit from flexible and dynamic service contracts. We argued that short-term and dynamic service contracts with strict Service Level Agreement (SLA) guarantees are important for the viability of future life-critical applications of 5G. It can be challenging and resource-intensive to allocate resources to billions of devices for the short-term and provide them with an SLA guarantee. To solve this problem, we have proposed an end-to-end architecture based on Permissioned Distributed Ledgers (PDLs) which records SLAs as smart contracts. These service contracts are deployed on a PDL and are accessible by customers through a Distributed App (DApp). Adopting a Distributed Ledger enables automation, transparency, and immutability in the system: all the service metrics will automatically be recorded to the ledger through a smart contract, and any violation will trigger automated compensation, avoiding the huge overheads of traditional methods. Our evaluation suggested that a *Permissioned* Ledger is essential for scalability and manageable overheads.

We aim to extend this work by studying dynamic pricing through smart contracts in which machine learning techniques can forecast service quality and help service contracts set their pricing parameters accordingly.

References

[1] S. Baggioni, "Remote Surgery, Robotics and more - how 5G is helping transform healthcare," https://bit.ly/3oyCqKu, 2019, [Online; accessed 06-Oct-2020].

[2] Statista, "Number of internet of things (IoT) connected devices worldwide in 2018, 2025 and 2030," http://bit.ly/30tQpaA, 2021, [Online; accessed 08 March 2021].

[3] E. Obiodu, N. Sastry, and A. Raman, "Towards a taxonomy of differentiated service classes in the 5g era," in *2018 IEEE 5G World Forum (5GWF)*. IEEE, 2018, pp. 129–134.

[4] S. Ha, S. Sen, C. Joe-Wong, Y. Im, and M. Chiang, "Tube: Time-dependent pricing for mobile data," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, 2012, pp. 247–258.

[5] T. Faisal, D. Di Francesco Maesa, N. Sastry, and S. Mangiante, "Ajit: Accountable just-in-time network resource allocation with smart contracts," in *Proceedings of the ACM MobiArch 2020 The 15th Workshop on Mobility in the Evolving Internet Architecture*, 2020, pp. 48–53.

[6] E. Foundation, "Ethereum," https://ethereum.org/en/, 2020, [Online; accessed 23-Nov-2020].

[7] Hyperledger, "Hyperledger Fabric," https://bit.ly/3gyHD22, 2020, [Online; accessed 23-Nov-2020].

[8] Quorum, "Consensus Quorum Blockchain," http://bit.ly/2OgZGzZ, 2021, [Online; accessed 08 March 2021].

[9] M. Shiraz, A. Gani, R. H. Khokhar, and R. Buyya, "A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing," *IEEE Communications surveys & tutorials*, vol. 15, no. 3, pp. 1294–1313, 2012.

[10] R. V. Rosa and C. E. Rothenberg, "Blockchain-based decentralized applications for multiple administrative domain networking," *IEEE Communications Standards Magazine*, vol. 2, no. 3, pp. 29–37, 2018.

[11] P. Kochovski, V. Stankovski, S. Gec, F. Faticanti, M. Savi, D. Siracusa, and S. Kum, "Smart contracts for service-level agreements in edge-to-cloud computing," *Journal of Grid Computing*, pp. 1–18, 2020.

[12] R. B. Uriarte, R. de Nicola, and K. Kritikos, “Towards distributed sla management with smart contracts and blockchain,” in *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2018, pp. 266–271.

[13] E. Di Pascale, J. McMenamy, I. Macaluso, and L. Doyle, “Smart contract slas for dense small-cell-as-a-service,” *arXiv preprint arXiv:1703.04502*, 2017.

[14] M. Bez, G. Fornari, and T. Vardanega, “The scalability challenge of ethereum: An initial quantitative analysis,” in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, 2019, pp. 167–176.

[15] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, “Fastfabric: Scaling hyperledger fabric to 20,000 transactions per second,” in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2019, pp. 455–463.

[16] HyperLedger, “Burrow,” https://bit.ly/3nmcukT, 2020, [Online; accessed 12 Dec 2020].

[17] GoQuorum, “Raft Consensus Overview,” http://bit.ly/3rA1SAZ, 2021, [Online; accessed 08 March 2021].

[18] Apache, “Kafka documentation,” http://bit.ly/3vmkqXI, 2021, [Online; accessed 08 March 2021].