AJIT: Accountable Just-in-Time Network Resource Allocation with Smart Contracts

Tooba Faisal
King's College London, UK

Damiano Di Francesco Maesa
University of Cambridge, UK

Nishanth Sastry
University of Surrey, UK
King's College London, UK

Simone Mangiante
Vodafone Group R&D, UK

ABSTRACT

New applications such as remote surgery and connected cars, which are being touted as use cases for 5G and beyond, are mission-critical. As such, communications infrastructure needs to support and enforce stringent and guaranteed levels of service before such applications can take off. However, from an operator's perspective, it can be difficult to provide uniformly high levels of service over long durations or large regions. As network conditions change over time, or when a mobile end point goes to regions with poor coverage, it may be difficult for the operator to support previously agreed upon service agreements that are too stringent. Second, from a consumer's perspective, purchasing a stringent service level agreement with an operator can also be expensive. Finally, failures in mission critical applications can lead to disasters, so infrastructure should support assignment of liabilities when a guaranteed service level is reneged upon – this is a difficult problem because both the operator and the customer have an incentive to lay the blame on each other to avoid liabilities of poor service.

To address the above problems, we propose AJIT, an architecture that allows creating fine-grained short-term contracts between operator and consumer. AJIT uses smart contracts to allow dynamically changing service levels so that more expensive and stringent levels of service need only be requested by a customer for short durations when the application needs it, and operator agrees to the SLA only when the infrastructure is able to support the demand. Second, AJIT uses trusted enclaves to do the accounting of packet deliveries such that neither the customer requesting guaranteed service levels for mission-critical applications, nor the operator providing the infrastructure support, can cheat.

CCS CONCEPTS

• **Networks** → Network management; **Peer-to-peer networks**; **Mobile networks**; **Network monitoring**.

KEYWORDS

SLAs, Smart Contracts, DLTs, PDLs

ACM Reference Format:
Tooba Faisal, Damiano Di Francesco Maesa, Nishanth Sastry, and Simone Mangiante. 2020. AJIT: **A**ccountable **J**ust-**i**n-**T**ime Network Resource Allocation with Smart Contracts. In *ACM MobiArch 2020 The 15th Workshop on Mobility in the Evolving Internet Architecture (MobiArch'20), September 21, 2020, London, United Kingdom.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3411043.3412506

1 INTRODUCTION

This paper is motivated by the key insight that new applications such as remote surgery and connected cars, which are being envisioned as the "flagship" or "killer" applications for 5G and beyond, are mission-critical applications. Failures in these applications can have disastrous consequences, possibly leading to loss of human lives. Therefore, to support such applications, the next generation of mobile architectures will have to greatly enhance their support for guaranteed service levels. Furthermore, to scale, such support will have to be provided efficiently and inexpensively.

We identify three limitations in current networks, in their ability to support mission-critical applications.

1) Service Level Guarantees are expensive for the consumer

Maintaining a high level of service guarantees can be expensive and mostly unnecessary for consumers. For example, consider a connected car, whose connectivity may be mostly used for non-critical applications such as entertainment, or traffic updates. This can be handled with a Service Level Agreement (SLA), not unlike today's "data" contracts, which offer a "best effort" service at an affordable price. However, the same car may, for a short duration, need to transmit information about emergency braking to other cars in its vicinity. This requires high reliability and low latency. A single stringent SLA that promises five-nines reliability and ultra-low latency all the time can be too expensive for the users of today's networks. At the same time, a standard "best effort" service level is not suitable during special occasions.

2) Stringent SLAs are difficult to honour for the operator

Stringent SLAs that need to be upheld at all times and in all locations can be difficult for operators to honour. For example, an operator may be able to support stringent SLAs for connected cars when they are in a region of the country where the operator has a dense and well-provisioned infrastructure, but the same operator may struggle to support even basic connectivity in the other areas. Even in well-provisioned geographic regions, an operator may temporarily be congested and therefore unable to offer high levels of reliability. Thus, even if customers are willing to pay, static SLAs that hold at all times and all locations can be impractical for operators to support when the consumer is a mobile endpoint [20] such as a car, and mobile architectures will need to support service requirements that may change dynamically efficiently.

3) Fine-grained Assignment of Liabilities is challenging

Even if operators can support stringent SLAs, the critical nature of applications such as connected cars or remote surgery requires a fine-grained assignment of liabilities when the SLAs are violated: If a failure occurs on the operating table during remote surgery or an accident on the road, it is vital to be able to pinpoint precisely where the fault lies. In particular, the network has always been a "blackbox" and there is no easy solution for resolving whether a network operator is at fault due to not meeting SLA requirements. Both the network operator and the customer who obtains connectivity with a guaranteed level of service have every incentive to blame each other and avoid liability, and there is no Trusted Third Party (TTP), who can monitor and ensure accountability.

To address the above three limitations, this paper proposes the concept of using *smart contracts* to advertise and automatically execute SLAs in a *just-in-time* and *accountable* manner. Specifically, smart contracts are inherently transparent, immutable, and automated. Once executed, a smart contract binds the network operator to a service level and the customer to the payment for that service, thereby providing accountability on both sides. Smart contracts can be executed just-in-time when a particular type of service is required, and only for the given duration of time that that service level is needed, thereby providing service level guarantees in a *dynamic, fine-grained context specific* fashion. The ability to dynamically change agreed upon service levels in a fine-grained fashion will address the dual realities that customers may not need the same level of service at all times and that operators may not be able to provide such guarantees at all times and all locations.

Furthermore, the SLA can be tailored to specific contexts in a fine-grained fashion. As indicative examples, there could be SLAs that hold over specific *time windows* as mentioned above (e.g., an agreement about maximum allowed latency for the duration of a remote surgery), or particular *locations* (e.g., a zero packet loss guarantee when cars form a platoon [35] in a dangerous mountain pass road). SLAs can also be made specific in other ways. For example, a car in a platooning situation needs high-reliability connectivity only to reach other cars within the same platoon, rather than to all other Internet endpoints.

At the implementation level, such guaranteed levels of service require reservation of resources. In this effort, we propose to reserve resources through dedicated network slices, which can be created based on parameterised templates to fit various standardised use cases such as remote surgery or car platoons. To scalably monitor whether packets have been delivered in accordance with an SLA, we need detailed housekeeping from both the operator end and the consumer end about the delivery state of packets, the experienced loss and delay [10], etc. This housekeeping needs to ensure that neither the operator nor the consumer has a chance to lie, and both parties agree with the results. We achieve this by executing signed housekeeping code in a secure enclave that cannot be tampered by either party.

2 RELATED WORK

Accountability: Today's internet is a best-effort service, although there have been several calls for accountability. Andersen *et al.* [8] uses accountability in the sense of understanding who is responsible for a given packet. Argyryaki studied the feasibility of per-packet accountability, and understanding packet loss [9, 10]. Such techniques can be used for better monitoring in our architecture. A decentralised, hierarchical approach for Virtualised Authentication, Authorization and Accounting (V-AAA) in 5G is proposed by Wong [36]. However, V-AAA is limited to authorization and authentication of subscribers and tenants of a network rather than SLA monitoring. Operators measure QoS of their network using receipts generated by sample packets. Some operators may give sampled packets a preferential treatment, and a packet sampling algorithm to prevent such "prioritization attacks" is proposed by[28]. Further work in path quality monitoring is carried out by [18] and verifiable network performance is discussed in Network Confession [11].

Smart Contracts in Telecommunications: Our architecture advocates the use of smart contracts to provision network resources just-in-time. However, several factors need to be borne in mind. Smart contracts are software programs prone to errors [27] and are susceptible to attacks [12]. Formal verification [14] and analysis of safety [23] may be advisable as this is being applied to mission-critical applications. A simple method to program a privacy-preserving smart contracts is proposed by [24].

Network Slicing: Network Slicing [17] is a key component of contemporary 5G and beyond mobile architectures. Slice Orchestration in 5G through blockchains is proposed by [13], but their work is limited to conceptualising blockchain for slice creation, which is a modification of [33]. Our work expands beyond that to support mission-critical applications. Using network switches for chaining NFs and optimally forming Network Function Blocks, to avoid conflicts and minimise the cost of network policy, is discussed by [15], but they don't address the significant issue of ensuring accountability. *Performance Contracts*, a library to predict the performance of NFs, are introduced in [21]. [25] has proposed a heuristic algorithm to solve the problem of delay guarantees in real-time systems.

Dynamic, automated, and on-demand resource allocation in network slicing is discussed by [33] using signal-based methods and introduces a centralised entity called the *Network Slice Broker* which looks after operations such as resource assignment and admission control. A comprehensive structure of end-to-end SLA for 5G Networks, along with important metrics, is proposed by[20]. But this study is limited to the structure of the SLA. Network slicing policy enforcement between different Mobile Virtual Network Operators(MVNOs) and mechanisms to minimise interference between them is discussed in [16].

3 AJIT ARCHITECTURE

In this section, we present **AJIT**, our framework for **A**ccountable **J**ust-**i**n-**T**ime allocation of network resources to support guaranteed levels of service that can dynamically change in fine-grained, context-specific ways and hold over short time windows, or in specific locations. Our architecture is built on three pillars: (*i*) *Billing and accountability*, which is achieved through smart contracts, (*ii*) *resource reservation*, which is achieved through network slicing, and (*iii*) *tamper-proof monitoring and housekeeping for dispute resolution*, which is achieved through code running in a secure enclave. We describe each of these in turn:

3.1 Billing and accountability through smart contracts

A novel concept that enables our dynamic fine-grained switching among different service levels is the notion of very short duration service agreements between the customer and network operators. In order to quickly execute and reach such agreements, we propose to use ***smart contracts***, which are essentially short immutable (and potentially verified [14]) programs executing on a Distributed Ledger Technology (DLT).

Coding service agreements as smart contacts allows SLAs to inherit the classic properties of smart contracts: (*i*) **Transparency** – any party can access the service agreement logic and read its past activation history. This provides monitorability [34]. (*ii*) **Immutability** – once a JIT Contract is deployed, its logic can not be changed, this prevents any interested party from tampering with it, and (*iii*) **Automation** – the agreement execution (i.e., enforcement) is beyond the control of any interested party, i.e., they can only affect it as specified by its immutable logic [30].

3.1.1 *Permissioned Distributed Ledgers.*

A typical base station may need to support several thousand smart contract transactions per hour from customers [22]. Therefore, we advocate using a particular class of DLTs called Permissioned Distributed Ledgers (PDLs) [3]. Commercially available PDLs such as *Hyperledger Fabric* can support this volume of transactions [19], in contrast with the more common *permissionless* ledgers such as the one used by the bitcoin blockchain (which supports on the order of tens of transactions per second [31]).

The characteristic of PDLs is that they are managed by authorized members only, i.e., only recognised entities are allowed to update the shared distributed state. This allows to pinpoint any consensus misbehaviour to the correct perpetrator quickly. In our proposal, the PDL is managed by a consortium of participants comprising the different telecom operators of a country. Operators advertise various types of services as separate smart contracts. A customer buys a particular a limited fine-grained service (for a short duration, a specific location, and perhaps only to connect to specific other endpoints) by executing a smart contract from a given operator. This transaction is recorded in the PDL, binding the customer to make the requisite payment and the operator to the service level. Whether the operator can provide the service level is monitored throughout the duration of the contract, as detailed in Section 3.3. Failure to adhere to the SLA may result in penalties that can be automatically imposed through the smart contract, or with an audit that takes place at a later date.

3.1.2 *Participants' roles.*

We note that even in a PDL, the state updates are still agreed upon through a distributed consensus algorithm, so the members need not trust each other and may still compete with each other [37]. However, PDLs are still susceptible to majority attacks wherein the majority of parties (often 50%+1 or 33%+1 depending on the consensus algorithm employed) can collude to influence the ledger. Therefore, a set of chosen regulatory authorities (e.g., FCC in the US, or Ofcom in the UK) are employed to provide oversight through consensus level access. The participants of the PDL are discussed below:

Operators act as validator nodes, i.e., are responsible for updating the ledger by executing transactions (e.g., calls to JIT Contracts) received by both operators and customers. They have *read/write* access to the ledger, and are in charge of creating and deploying new JIT Contracts. We assume the network operators to be competing with each other, so their cooperation on the PDL management is achieved algorithmically, by them taking part in a distributed consensus protocol; there is no trust required between them.

Customers send requests to execute JIT Contracts. Customers only have *read* access to the PDL, i.e., they do not take part in the consensus protocol to maintain it, but they can still affect the state of the PDL by submitting smart contract execution transactions, which may then lead to the recording of new SLAs, and after the service is delivered, will trigger payments from customers to operators.

Regulatory authorities are introduced as trustworthy entities on the consensus layer of the PDL. By taking part in the distributed consensus protocol, they can detect anomalous or fraudulent behaviour from the operators and cartel forming among operator coalitions. Note that regulatory authorities are not in charge of managing the system. Instead, they only act as “fall back” watcher nodes in case of disputes or misbehaviour. As such, they are not a Trusted-Third Party (TTP), and control still resides with operators.

3.2 Real-time Resource allocation with network slices

Network Slicing [32] is a mechanism that operators can use to provide isolation between different kinds of traffic, and reserve resources to provide different levels of service to different users. A user is allocated a particular slice based on their required type of service and isolation level. For example, a platoon of cars can get a network slice with very low-latency and high bandwidth. It is also possible that a user is allocated different slices, which raises questions of SLA Management[20]. These network slices are created with the chaining of NFs, which should be programmable to support today's service demands[15].

Numerous solutions have been proposed to create and assign users to network slices in near-real-time. For example, operators may migrate previously allocated resources with subsecond latency through primitives such as VM Fork [26]. Nour *et al.* [29] propose to create end-to-end network slices by stitching together “sub slices”, with each sub-slice representing a resource or set of resources from particular “technological domains” (e.g., radio, compute resource, transport protocol, etc.). Their simulations indicate that this stitching approach can lead to network slices creation time on the order of a few seconds.

3.2.1 *Just-in-Time Network Slicing.*

In our architecture, the slice creation starts with the execution of the smart contract when users request services. After the validation of PDL nodes, the respective smart contract is executed, and resources are assigned through a JIT- Controller (Figure 1). The controller maintains an internal database, which keeps track of the available resources and NFs, and their respective path metrics, which will be needed to honour a

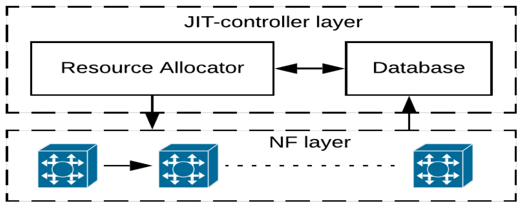

Figure 1: JIT-Controller creates network slices through consultation with the internal Database

particular SLA. With every incoming service request, the controller consults the database and stitches an end-to-end network slice from available resources that can satisfy the requested SLA. If such a slice cannot be created, this is communicated back to the PDL, nullifying the smart contract. Once the slice creation is confirmed with the smart contract, the resources and network functions used are marked as "reserved" or "in-use" within the database for the duration of the contract.

3.3 Tamper-proof monitoring with housekeeping code in secure enclaves

The difficulty in providing reliable service level measurements is that the two participating entities, the customer and operator, do not trust each other. Each has an incentive in cheating with their reported measurements: the customer may unduly claim SLA breach compensations (eg., by claiming that packets were delivered late or not at all), whereas the operator may pretend to satisfy an SLA, while actually providing lower levels of service, hence charging the customer an unfair premium.

To solve the above problem, when a dynamic fine-grained SLA is instantiated through a smart contract (as detailed in Section 3.1), the funds of the customer are kept in escrow by the contract. These funds are released when the operator produces an *SLA receipt* from the customer. However, to ensure that the customer does not cheat in the SLA receipts, the receipts are generated by code authored by the operator but run on customer premises within a trusted enclave. More precisely, an enclave resides on the customer device and is able to run code that cannot be tampered with by the customer and is therefore trusted by the operators. Similarly, we deploy an *edge monitor* on the operator base stations, i.e., a software component charged with interacting with the enclaves in monitoring the SLA.

3.3.1 ***Fine-grained SLA Monitoring.*** The SLA monitoring proceeds on both sides by dividing the service agreement's duration into small epochs over which SLA should be maintained. The edge monitor and its counterpart on the customer's enclave do the housekeeping to keep track of the level of service provided by the operator during that epoch (e.g., by counting the number of packets that successfully cross the boundary from operator to customer during the epoch, from which the bandwidth provided to the customer can be calculated). From this housekeeping tally, an *SLA receipt* is generated for the epoch on the edge monitor and its counterpart in the customer's enclave.

At the beginning of each epoch, the customer may be required by the operator to provide it with a signed version of the SLA receipt from the previous epoch, for service to be continued. Thus, the customer cannot cheat by discarding the receipts produced by the secure enclave. Within the ambit of the smart contract, the operator claims payment of funds held in escrow by producing these SLA receipts. Note that the operator cannot simply claim payment with the copy of the SLA receipt generated on the operator's base station, because this has not been signed by the customer. Note also that because the operator generates its copies of the SLA receipts, the operator can detect and stop service if the customer cheats by giving the operator a fake SLA receipt (e.g., by replaying an old receipt).

One issue with the above strategy is that PDL transactions are typically much slower than the line rates that need to be supported in mobile networks, and therefore payments cannot be generated at line rate. However, because the PDL guarantees the escrow, the payments can happen at any later time. Indeed, for higher efficiency, a single cumulative settlement for all epochs can happen after the flow has ended (the operator still checks the customer-signed SLA receipts with its own copies at the end of each epoch, to ensure that it is not being cheated mid-flow). Finally, we observe that some QoS measures may need information from multiple edge monitors. For example, consider two connected cars communicating with each other via the operator's network. In order to verify that the latency experienced by packets that enter the operator's network from one car (say *car1*) and exit to the other car (*car2*) is below a particular bound, we will need *SLA receipts* from the boundary point between the network operator and *car1*, as well as the network boundary between the operator and *car2*.

4 EVALUATION

In this section, we evaluate a key claim of our architecture – that network resources can be allocated "just in time". In particular, for a customer to request a new type of service with new guarantees of service levels, two steps need to be resolved: First, the customer has to invoke a smart contract, which then needs to be executed on the underlying distributed ledger technology. Second, a new network slice has to be stitched together from available resources to provide guaranteed service levels.

4.1 Profiling Smart Contract Delays

To study JIT Contract set up latency we have chosen to use smart contracts written in Solidity [5] and deployed on Ethereum's Ropsten testnet [4]. We have chosen Ethereum as it is the most popular smart contract-enabling protocol. Our system is suitable to be deployed on a PDL since all validators are known. As such, the actual system implementation would be deployed in a fine-tuned custom implementation, and the results obtained below on a public test blockchain should be considered approximate *upper bounds* on the delays incurred by smart contracts.

Each operator may implement a JIT Contract as arbitrarily complex as they wish. However, all JIT Contracts accepted by consumers should reflect, at least, the service activation logic expressed in Section 3. During our analysis, we consider a smart contract in Solidity implementing an example JIT Contract providing the main activation logic common to all JIT Contracts, i.e., from service activation

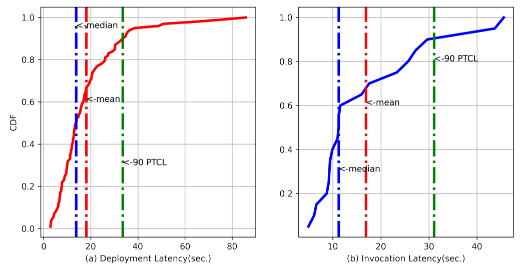

Figure 2: Cumulative Distribution Functions (CDFs) of Deployment (left) and activation (right) function invocation latency of the reference JIT Contract showing the mean, median and 90th percentile latency.

to the SLA receipt. We measure the time to deploy 100 clones of our reference JIT Contract and execute once the *activate* function for each of them. Note that *activate* is called by a customer, and it locks the agreed payment, notifies the owner to start providing the agreed service, and starts SLA monitoring.

The results are shown in Figure 2. The mean deployment time obtained is 18.101 seconds, with a standard deviation of 14.290 seconds. Note that Ropsten consensus wait imposes that transactions take 14 seconds on average to be accepted. So our results are not far from the achievable optimum. Moreover, each contract is deployed only once and can be deployed well in advance by the operator. Activate times are, instead, crucial to achieving Just-in-Time performances. The activate function invocation times are a little lower than deployment times, with a mean of 16.892 seconds and standard deviation of 11.943 seconds, but still under one minute on average.

.

4.2 Profiling resource reservation delays

Resources are reserved by creating or stitching together a network slice [29] by chaining appropriate network functions (NFs). As mentioned in Section 3.2, this requires checking with an internal database for available resources and NFs, commissioning them to the slice, and marking the resources as "reserved" or "in-use" within the database. Our evaluations are done in Mininet [2] and Open vSwitch [1]. To mimic a typical operator's network topology, NFs are switches arranged in a tree topology (with Core, Aggregation and Edge layers), with users attaching at the leaves. An explicitly designed Just-in-Time Controller controls the database and reserves the slices. As a preliminary back-of-the-envelop analysis, we submitted 70 user requests to the controller under varied packet loss rates. Figure 3 shows the slice creation latencies, suggesting that a satisfactory chain of NFs can be created quickly.

5 DISCUSSION

This paper considered an emerging class of *mission critical* applications such as remote surgery and connected cars, which require stringent service level agreements or SLAs, and a fine-grained assignment of liabilities in case of failures. Furthermore, to make it affordable, operators need to find ways to allow customers to obtain these guarantees only during short critical periods flexibly. Equally, to avoid liabilities and penalties, operators should also have the

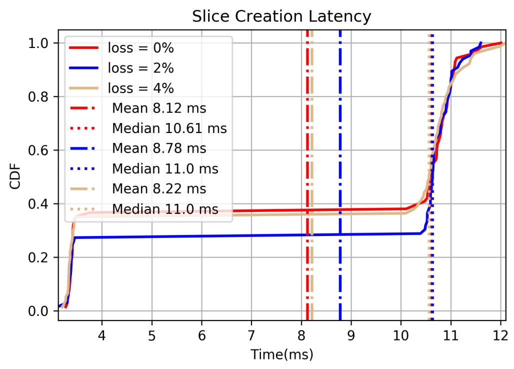

Figure 3: CDF of slice creation latency.

freedom to commit to strong SLAs only when they can guarantee the availability of resources.

To address these problems, we proposed an end-to-end architecture **AJIT** for **A**ccountable **J**ust-**in**-**T**ime allocation of network resources through smart contracts. We believe that radical changes are inevitable in the current contract design to make them ready for future service requirements, and most important among them is to be able to get network services at the finest possible granularity. In a conversation with the UK's major telecom operator, we understood that the current industry standard slice creation time could be up to a few minutes, depending on the size of the geographical area spanned by a slice. Thus, "real-time" requests from applications need to keep such constraints in mind, notwithstanding the previous research [29] and our simulations.

We proposed a solution for assigning liability based on SLA receipts. The presence or absence of signed SLA receipts can be used both for payment and for audting the network operator's performance at a later date for legal reasons. However, we note that the operator relies on the configuration of several hardware and software equipment in different network segments (e.g., radio access, transport, core). Most of the involved processes still depend on specific vendor's implementation or support, although standardisation efforts are helping their harmonization [6, 7]. Thus it may be difficult to understand whether and how blame can be split between the operator (due to configuration issues) and the vendor whose equipment is being used.

Our work marks the beginning of an era for automated and accountable resource allocation approaches, in which all partners can work with justice, honesty, and trust. We are evaluating our architecture with several strategies to optimize resource allocation latency. In the future, we also wish to extend this work to address the pricing strategies and benefits the operator and customer can achieve through this. Maintaining SLAs in smart contracts also achieves the orthogonal benefit of enabling operators to do business with peers/suppliers (such as vendors and other operators) and customers in an untrusted environment. This enables automation in billing, QoS monitoring, and compensation.

REFERENCES

[1] 2016. Open vSwitch. Retrieved from https://www.openvswitch.org/. Accessed on: 01 June 2020.
[2] 2018. Mininet. Retrieved from http://mininet.org/. Accessed on: 01 June 2020.
[3] 2020. Permissioned Distributed Ledgers. Retrieved from https://bit.ly/36XroXz. on: 01st June 2020.
[4] 2020. Ropsten Testnet. Retrieved from https://ropsten.etherscan.io/. Accessed on: 01 June 2020.
[5] 2020. Solidity Language. Retrieved from https://bit.ly/2BAbred. Accessed on: 01 June 2020.
[6] 3GPP. 2016. *Service requirements for the 5G system.* Technical Specification (TS) 22.261. 3rd Generation Partnership Project (3GPP). https://bit.ly/30eq4ya
[7] 3GPP. 2018. *Management and orchestration; 5G Network Resource Model (NRM); Stage 2 and stage 3.* Technical Specification (TS) 28.541. 3rd Generation Partnership Project (3GPP). https://bit.ly/2Y9xRdQ
[8] David G Andersen, et al. 2008. Accountable internet protocol (aip). In *ACM SIGCOMM 2008 conference on Data communication.* 339–350.
[9] Katerina Argyraki, et al. 2004. Providing packet obituaries. In *ACM HotNets-III.*
[10] Katerina Argyraki, et al. 2007. Loss and delay accountability for the Internet. In *IEEE ICNP.* 194–205.
[11] Katerina Argyraki, et al. 2010. Verifiable network-performance measurements. In *Proceedings of the 6th International COnference.* 1–12.
[12] Nicola Atzei, et al. 2016. A survey of attacks on Ethereum smart contracts. *IACR Cryptology ePrint archive* (2016).
[13] Jere Backman, et al. 2017. Blockchain network slice broker in 5G: Slice leasing in factory of the future use case. In *IEEE IoTs Business Models, Users, and Networks.*
[14] Karthikeyan Bhargavan, et al. 2016. Formal verification of smart contracts: Short paper. In *Proceedings of the 2016 ACM PLAS.* 91–96.
[15] Lin Cui, et al. 2018. Enabling heterogeneous network function chaining. *IEEE (TPDS)* 30, 4 (2018), 842–854.
[16] Salvatore D'Oro, et al. 2019. The slice is served: Enforcing radio access network slicing in virtualized 5G systems. In *IEEE INFOCOM.* 442–450.
[17] Xenofon Foukas, et al. 2017. Network slicing in 5G: Survey and challenges. *IEEE CommMag* 55, 5 (2017), 94–100.
[18] Sharon Goldberg, et al. 2008. Path-quality monitoring in the presence of adversaries. *ACM SIGMETRICS* 36, 1 (2008), 193–204.
[19] Christian Gorenflo, et al. 2019. Fastfabric: Scaling hyperledger fabric to 20,000 transactions per second. In *ICBC.* IEEE, 455–463.
[20] Mohammad Asif Habibi, et al. 2018. The structure of service level agreement of slice-based 5G network. *https://arxiv.org/pdf/1806.10426.pdf* (2018).
[21] Rishabh Iyer, et al. 2019. Performance contracts for software network functions. In *16th USENIX (NSDI).* 517–530.
[22] Olle Järv, et al. 2012. Mobile phones in a traffic flow: a geographical perspective to evening rush hour traffic analysis using call detail records. *PloS one* 7, 11 (2012).
[23] Sukrit Kalra, et al. 2018. ZEUS: Analyzing Safety of Smart Contracts.. In *NDSS.*
[24] Ahmed Kosba, et al. 2016. Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. *IEEE S&P* (2016).
[25] Rakesh Kumar, et al. 2017. End-to-end network delay guarantees for real-time systems using sdn. In *2017 IEEE RTSS.* 231–242.
[26] Horacio Andrés Lagar-Cavilla, et al. 2009. SnowFlock: rapid virtual machine cloning for cloud computing. In *ACM EuroSys.* 1–12.
[27] Loi Luu, et al. 2016. Making smart contracts smarter. In *ACM SIGSAC 2016.* ACM.
[28] Pavlos Nikolopoulos, et al. 2019. Retroactive Packet Sampling for Traffic Receipts. *ACM POMACS* 3, 1, 1–39.
[29] Boubakr Nour, et al. 2019. A blockchain-based network slice broker for 5G services. *IEEE Networking Letters* 1, 3 (2019), 99–102.
[30] Adrian Paschke et al. 2006. A categorization scheme for SLA metrics. In *Proc. Service-Oriented Electronic Commerce.* Gesellschaft für Informatik eV.
[31] Joseph Poon et al. 2016. The bitcoin lightning network: Scalable off-chain instant payments.
[32] Peter Rost, et al. 2017. Network slicing to enable scalability and flexibility in 5G mobile networks. *IEEE Communications magazine* 55, 5 (2017), 72–79.
[33] Konstantinos Samdanis, et al. 2016. From network sharing to multi-tenancy: The 5G network slice broker. *IEEE CommMag* 54, 7 (2016).
[34] James Skene, et al. 2007. The Monitorability of Service-Level Agreements for Application-Service Provision. In *WOSP.* 3–14.
[35] Vladimir Vukadinovic, et al. 2018. 3GPP C-V2X and IEEE 802.11 p for Vehicle-to-Vehicle communications in highway platooning scenarios. *Ad Hoc Networks* 74 (2018), 17–29.
[36] Stan Wong, et al. 2017. Virtualized authentication, authorization and accounting (V-AAA) in 5G networks. In *IEEE CSCN.* IEEE, 175–180.
[37] Karl Wüst et al. 2018. Do you need a blockchain?. In *CVCBT(2018).* IEEE, 45–54.