Blockchain-Based Slice Orchestration for Enabling Cross-Slice Communication at the Network Edge

Konstantinos Papadakis-Vlachopapadopoulos*, Ioannis Dimolitsas*, Dimitrios Dechouniotis*, Eirini Eleni Tsiropoulou†, Ioanna Roussaki*, Symeon Papavassiliou*

*School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece

†Dept. of Electrical and Computer Engineering, The University of New Mexico, Albuquerque, NM, USA

{cpapad, jdimol,ddechou}@netmode.ntua.gr, eirini@unm.edu, ioanna.roussaki@cn.ntua.gr. papavass@mail.ntua.gr

***Abstract*—The Internet of Things (IoT) paradigm, with a grand variety of devices in large numbers, commonly combined with resource and power limitations dictates new challenges and constraints. The emerging Edge Computing paradigm arises promises to address these issues by placing cloud-type resources, closer to IoT devices. In this context, network slicing is common practice at the network edge. A network slice groups logically isolated computing and network resources. Under the 5G and Edge Computing umbrella, multi-administrative domain services have emerged as operational challenges but also giving economic incentives to the providers and the users. This study introduces a Blockchain-based solution, aligned with the European Telecommunications Standards Institute (ETSI) Network Function Virtualization (NFV) standard, for automated orchestration, enabling cross slice communication between different domains. The main focus is placed on enabling trust between untrusted parties, minimizing resource consumption, management and development overheads, as well as providing security and crash-fault tolerance.**

***Index Terms*—Blockchain, Smart Contracts, Edge Computing, NFV Orchestration**

I. Introduction

The ever-increasing plethora of the Internet of Things (IoT) complex services, along with the emergence of modern applications and the growing interactions between smart devices in smart city context, makes the Edge Computing [1] a determining factor in meeting the requirements of the 5G network establishment. The Edge Computing paradigm offers cloud computing capabilities at the edge of the network, where mobile end-devices with limited computing capabilities have to meet the high bandwidth and low latency demands of IoT-based applications [2], [3]. Under this setting, many cloud providers invest on small-scale data centers at the edge of the network, referred to as edge clouds (ECs) throughout the rest of this paper. Due to the stringent Quality of Service (QoS) constraints and the finite available resources, the automated and dynamic orchestration and management of ECs is prerequisite for successful service delivery.

Similarly to cloud computing, EC resource orchestration relies on the Network Function Virtualization (NFV) [4] and Software Defined Networking (SDN) [5], [6] technologies, which enable network slicing. A Network Slice consists of a set of Virtual Network Functions (VNFs), which are actually isolated Virtual Machines (VMs) with dedicated resources running a specific application. A network slice is usually deployed in a single EC. As NFV model evolves, apart from typical network services (e.g., routing and firewall), VNF refers to any type of services, which can be consumed by multiple tenants. Towards this direction, Cross-Slice Communication (CSC) can be beneficial for both tenants and providers [7]. Leasing off-the-shelf services reduces the administration overhead of the tenant-consumer and it is profitable for the tenant-provider. Furthermore, from the provider perspective, CSC leads to minimization of allocated resources for the shared services that in turn leads to the increase of available resources for the deployment of other slices and services. However, CSC should not violate the isolation property, which is a basic asset and principle of network slicing. Thus, for the realization of the CSC, it is very important to develop the necessary shared policies and slice brokering solutions.

In recent years, there has been a significant raise of interest around Blockchain technologies both from industry and academia. One of the main aspects motivating this interest, is that Blockchain technology enables parties without trust between them, to interact without a central trusted authority, communicating in a decentralized fashion, maintaining the same functionality, while all parties can retain the certainty for the outcome of the transactions. Thus, Blockchain allows the deployment of distributed trustless networks. Also, Blockchain exploits cryptography, and cryptographic hashes of every transaction are stored in a block along with the cryptographic hash of the previous block, creating in this way the Blockchain. In addition, those hashes must be approved and agreed by all participants in the network. All those characteristics guarantee non-repudiation, immutability, data integrity, and security. Initially, Blockchain was introduced with Bitcoin [8] and was strictly binded with a cryptocurrency. Later, Blockchain frameworks, such as Ethereum [9] and Hyperledger Fabric [10] came with a currency or without and introduced smart contracts to enable custom business. Smart contracts are basically scripts, which reside on the Blockchain and have a unique address, and when invoked by address they execute transactions following predetermined rules, and the encrypted records of those transactions are stored in the Blockchain ledger. The automated execution of smart contracts make them applicable for incorporating the business logic in the deployment of smart applications in EC.

In the federated cloud environment, a central trust manage-

978-1-7281-8915-4/20/$31.00 ©2020 IEEE
DOI 10.1109/QRS-C51114.2020.00033

ment service is responsible for the communication between the federated ECs [11]. In this study, we leverage the distributed nature of Blockchain, its immutable ledger of transactions, and smart contracts to replace the centralized trust management with a distributed trustless framework that facilitates the automated CSC. This advantages counterbalance any concern about the performance overhead and scalability of the Blockchain technology. The CSC establishment requires some preparation steps, such as the instantiation of shared Network Services (NS) and VNFs, and the creation of connection points for virtual links. Additionally, other functionalities, such as monitoring and billing, are involved in the CSC process. Toward the automated resource orchestration and management, we use Hyperledger Fabric and leverage the high level of automation of smart contracts to initiate the communication between slices, which are deployed in geographically dispersed federated ECs. With this capacity, the main objectives of this research work are (a) to minimize the orchestration overhead, and (b) reduce the computing resources for the CSC establishment and the related cost. To this end, we present a proof of concept design and implementation for the Blockchain-based CSC lease, orchestration and lifecycle support. We define all the entities, their interactions and the required parameters and unique identifiers needed to perform all the operations for the CSC operations and management.

The rest of the paper is organized as follows. In Section II, the related literature is presented. Section III introduces and describes the architecture and its main components, while Section IV presents the necessary smart contracts for management, initiation, and orchestration of the CSC alongside with billing. Finally, Section V concludes the paper.

II. Related Work

This section presents the most relative studies on cloud resource orchestration, trusted and trustless management of federated clouds. The automated deployment of cloud applications has attracted the interest of both the academia and industry. Open Source Management and Orchestration (OSM) [12] provides an open source Management and Orchestration (MANO) stack aligned with ETSI NFV Information Models. In combination with a Virtual Infrastructure Manager (VIM), such as OpenStack [13], they provide automated management of the application's lifecycle, including instantiation, configuration, and maintenance. OSM and OpenStack are primarily used on large scale data cloud data centers and they mainly support VM-based deployment. Kubernetes [14] is an open-source system for automated orchestration of applications. Kubernetes is based on containers and focuses on application deployment, scaling, and management, and is appropriate for IoT-based or smart applications. At the same time, serverless computing is an emerging service delivery model that alleviates the developers from the resource orchestration overhead and offers a pay-as-you-go billing model [15]. This model is still in its infancy and more suitable for stateless applications [16]. In the federated cloud environment, trust management of serverless platforms remains an open challenge.

Regarding trust management of federated heterogeneous clouds, the authors in [17] proposed a trust management framework based on Service Level Agreement (SLA) and reputation. A multi-criteria methodology, named Fuzzy Analytic Hierarchical Process, was used to evaluate the performance of cloud services and update the reputation score of each cloud. Pustchi et al. proposed a OpenStack-based trust mechanism for federated clouds [18]. They enabled domain-trust through temporary mapping rules, which define a set of accepted remote users or groups to local domain users and groups. In [19], a reputation management mechanism for federated clouds was introduced. This mechanism focused on multi-tenancy and feedback from the cloud providers regarding the users was used to enable service differentiation among the tenants and calculate the reputation of both tenant and cloud provider. A geometric representation method was used to illustrate how the changes of user's reputation affect the provider's ones.

The use of Blockchain technology in the resource orchestration and management of edge/cloud infrastructure is an open and challenging research area and many on-going studies and platforms have already been proposed. In [20], the authors introduced a Trusted Orchestration Management (TOM) architecture as an early solution for container-based edge architectures, using Blockchain to provide trustworthy mechanisms in untrusted environments focusing on identity, provenance and orchestration management. TOM is based on smart contracts and is aligned with the W3C-PROV standard [21]. The authors of [22] presented a work in progress for Virtual Machine Orchestration, originally composed by the Orchestrator, the centralized Virtual Machine Orchestration Authenticator, and the Virtual Machine Manager. This framework aspired to replace the centralized authentication intermediary mechanism with a distributed one with the use of private Blockchain technology. In [23], a private Blockchain system was deployed for the management of cloud tenants and services, providing identity management, authentication, delegation authorization of services, and charging functionalities with the use of smart contracts. Finally, Afraz et al. [24] proposed the use of Blockchain for 5G network slice brokering by implementing a variation of the double auction mechanism with smart contracts in the market scenarios, where there is a lack of trust between market players. Finally, in [25] and [26], a proof of concept blockchained-based implementation using distributed applications (DApps) is presented, for multi domain service orchestration. Also the authors analyze three use case scenarios (MEF, 3GPP and ETSI NFV standards) discussing discussing standarization opportunities araound blockchain-based DApp.

We then analyze three use case scenarios pursued by on-going work at standards development organizations, namely MEF, 3GPP, and ETSI NFV, discussing standardization opportunities around blockchain-based DApp

This work exploits the Blockchain technology to provide a trustworthy mechanism for the communication between network slices in a multi-administrative environment, where VNF-deployed services can be leased by different tenants.

Furthermore, the exploitation of the smart contracts minimizes the slice orchestration overhead and the essential computing resources for the CSC establishment.

III. System Architecture

A. Proposed ETSI NFV-Based Architecture

The proposed architecture is based on the ETSI NFV Standard [27]. A specific region that includes multiple ECs is referred as *Availability Zone*. The proposed architecture is shown in Figure 1 and is detailed below, following a top-down description. In particular, at the top level, there is the Management and Orchestration (MANO) layer, where the OSM operates as the NFV orchestrator. Subsequently, the lower level includes the ECs. Each EC provides the physical infrastructure and offers the resources for network slice deployment, under lease basis. Each EC infrastructure is managed by the corresponding VIM, which interacts with the OSM layer, in order to manage and control the NFV Infrastructure (NFVI). The OSM is able to interact with multiple VIMs to perform the necessary NFV orchestration in a multi-domain administration environment. The NFVI is actually one of the most important functional blocks in the NFV architecture, as it describes the hardware and software components on which virtual networks are built. NFVI actually creates the virtualization layer, in which hardware resources are logically partitioned to deploy the functionalities of each VNF, increasing the interoperability between them and therefore enabling the network slice deployment. In such a way, network slices are composed of a set of VNFs and Network Services (Figure 2), so they deliver a plethora of different services as an isolated network service chain to the end-user. The network slice is allocated the essential computing and network resources to respect the QoS requirements of the slice tenant.

The above described architecture is aligned with 5G directives, where multiple ECs offer their resources for the deployment of time- and mission-critical applications at the edge of the network. It is worthwhile mentioning that, in our case, the EC provider is not necessarily equivalent to the slice provider. A network slice instance can be deployed in a single edge cloud, which means that the corresponding computing and network resources can be either located in a specific location, or in multiple provider domains. In both cases, the optimal resource allocation is of major importance to meet the performance requirements alongside with the cost reducing from the network slice provider perspective. Towards enabling a service market among slice providers, a network service of a deployed slice can be leased and included in the slice of a different tenant, who is interested in deploying its slice. To this end, OSM can automatically configure and establish the communication between different network slices by sharing already deployed network services.

Based on this ETSI NFV aligned architecture, we use the permissioned Blockchain solution of Hyperledger Fabric to automate slice orchestration for establishing cross-slice communication at the network edge. The primary goal of our Blockchain based solution, is to minimize the allocated resources for the purposes of CSC realization. Additionally, it establishes and provides trust in a trustless environment between tenants, immutable transaction logs, security, and more importantly eliminates single points of failure due to its distributed nature.

As depicted in Figure 1, the minimum requirements for our solution is the deployment of a Fabric peer and a Fabric Certificate Authority (CA) per EC. As detailed in section IV, every EC is considered as an individual organization in Fabric architecture and joins the common Fabric Channel in order to communicate and perform transactions. Extra channels, private data, access control lists can be used in order to ensure that confidential information would be shared only to permitted participants. However, though considering these challenges is of high practical importance, it remains out of scope of this study, and is part of our future research.

A Fabric peer is a Blockchain node that stores all transactions on a joining channel. Every peer has a ledger database and the instantiated Chaincodes, known also as Fabric smart contracts. All peers and the Orderer participate in a Fabric channel where the Chaincodes are instantiated as depicted in Figure 1. The CA is responsible for managing user certificates such as user enrollment, user revocation etc. Hyperledger Fabric is a permissioned Blockchain network, thus, only permitted users or entities can join the network and additional restrictions can be applied to different functionalities after joining the network. The Orderer is one of the most important components used in the Fabric consensus mechanism and belongs to the Fabric networks administrative components, therefore it is placed at the providers administrative layer in Figure 1. The Orderer is a service responsible for ordering transactions, creating a new block of ordered transactions and distributing a newly created block to all peers on a channel. Fabric offers different ordering service implementations. We use Raft, a crash fault tolerant ordering service based on the implementation of Raft protocol [28]. The basic components of the Fabric architecture for our solution and their functionalities are presented in this section. Subsequently, the specific interaction of the components of our solution (i.e., Fabric components, MANO, and VIMs) will be presented in detail in section IV.

B. ETSI NFV Network Slice Instance

In order to enable cross-slice communication, it is necessary to describe the main components of a Network Slice Instance (NSI), as defined in the OSM model. The Network Slice Instance in OSM is described by the Network Slice Template (NST), which is a descriptor file that includes the necessary information about a slice deployment.The main arguments in the NST descriptor are about the slice identifier (ID), the name of the slice, and the service type of the slice (e.g., eMBB). A network slice is composed of a set of NSs which includes the corresponding VNFs and Virtual Network Connections between them. In the NST descriptor, the *netslice-subnet* fields define the included NS using their unique ID. Also,

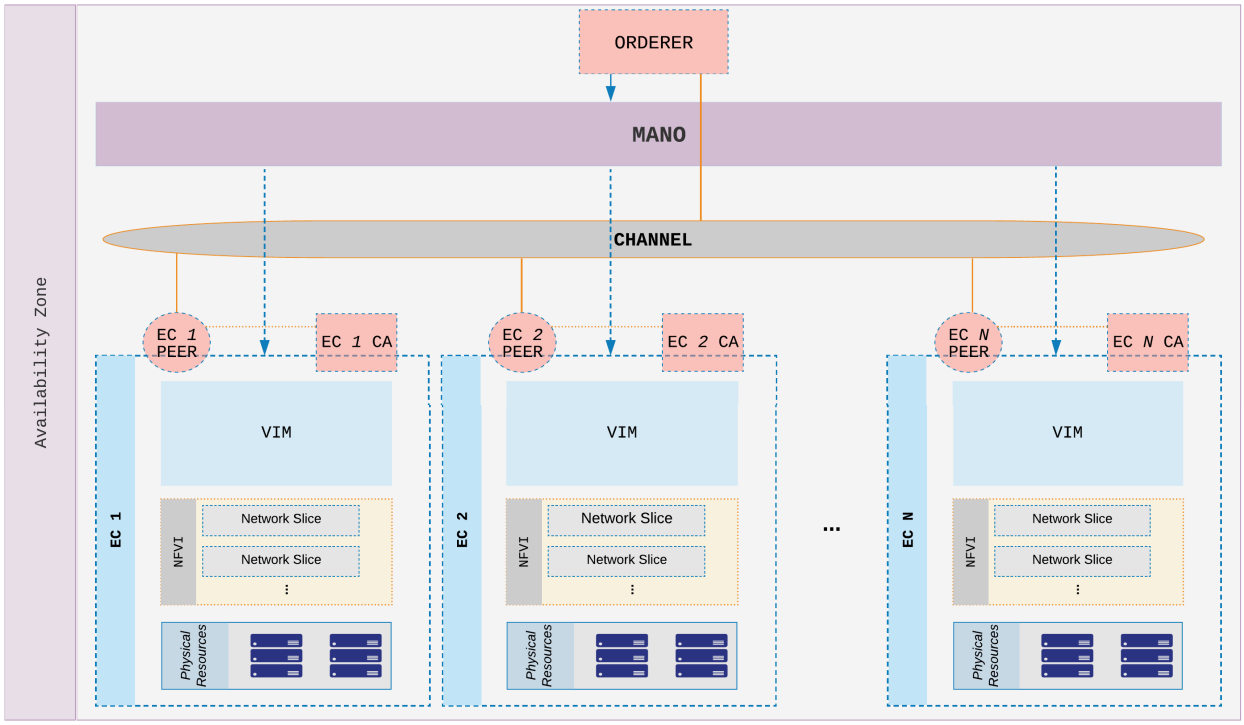

Fig. 1. System Architecture

the option for NS sharing is defined in the *is-shared-nss* argument. Furthermore, the NST includes the network slice virtual networks in the *netslice-vld* field. In this argument, the NS connection points are included, in order to create virtual links between them. The connection points of each NS are the key argument in service sharing.

Figure 2 illustrates a cross-slice communication scenario, where two network slices are deployed, with one shared NS between them. The *network-slice-1* consists of three NSs: *network-service-1-1, network-service-1-2*, and *network-service-shared* and two Virtual Networks: *vld1-1* and *vdl1-2*. In the descriptor of the shared NS, three connection points are defined, while the slice owner sets as *true* the *is-shared-nss* parameter in the NST descriptor. Then, another slice owner is interested for deploying a new slice (NST ID *network-slice-2*). This slice will include two new NSs, namely *network-service-2-1* and *network-service-2-2*, and the shared NS of the first network slice, *network-service-shared*. A connection point of the shared NS is necessary to create a virtual link between the shared NS and the *vld-2*, the virtual network of the second network slice. In such a way, the CSC is established without creating a new instance of *network-service-shared* and the instantiation time, the consumed resources, and the monetary cost are reduced for both slice owners. Beyond the basic set up of CSC, further functionalities, such as shared policies and billing, must be realized in order to provide a secure communication scheme in this multi-domain environment. As it is described in the next section, Blockchain appears a powerful option to achieve this objective.

IV. Blockchain Based Service Discovery, Lease, and CSC Orchestration

In [29], challenges, architectural concepts, and proposals for multi-domain operation have been described, including different departments within the same network operator, in an analogy to the setting in our research work, where different ECs in the same availability zone are provided by a single provider. For multi-domain operation, two alternatives are highlighted:

1) "Configuration driven": In this option, the different functional blocks to be interconnected are statically configured with the necessary information to form the relation with the other parties.
2) "Auto-discovery": In this option, the different functional blocks advertise their own information and prepare the information to be used to form the relation. This form of interconnection assumes the implementation of a discovery mechanism in the NFV-MANO functional blocks.

In this study, we propose a blockchain-based solution aligned with the auto-discovery option. We use Hyperledger Fabric and Chaincode - the Hyperledger's term for smart

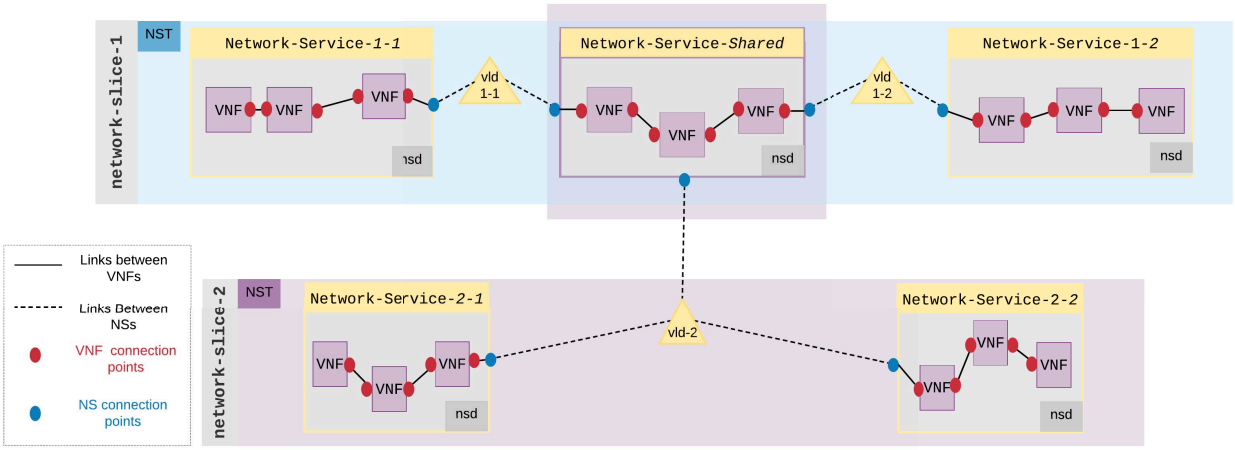

Fig. 2. Cross-Slice Communication Using a Shared Network Service

contracts - for the management and automation of the auto-discovery, as well as the lease and orchestration of network slices provided by tenants in a trustless environment. Hyperledger Fabric supports different programming languages for Chaincode development (Java, JavaScript, TypeScript and Go). In our case, Go language has been selected to develop smart contracts, due to its simplicity and performance features.

A. Assets Definition

In order to achieve our objectives, we have to identify and define all involving assets required for a complete workflow. Initially, we have to define the Tenant. For the initial registration of a user or the advertisement of a service, the tenant is enrolled to a Fabric CA. This component can be configured to require Lightweight Directory Access Protocol (LDAP) user authentication before enrollment and retrieve the identity's attribute values, which are used for authorization. This functionality allows the easy integration in a fully automated workflow and enable the tenant to interact with the Blockchain in order to advertise its service, or discover and lease a provided service.

1) Network Services and Virtual Network Functions: The most important assets to be defined are the Network Service and Virtual Network Function. As depicted in Listing 1, we provide a first basic definition of all mandatory attributes needed to be known in order to lease VNFs and orchestrate the CSC. For NSs, the ID and Name, which are unique identifiers, are needed for recognition and orchestration. In addition the Tenant's (NS owner) information is included. For VNFs, we use *ID, Name, Short Name, Description* and *Provider*.The *ID* and *Name* attributes are used as unique identifiers likewise NS. The *Short Name* and *Description* attributes are mostly used to describe the Service Advertised to the user. Finally, *Provider* describes the ID of the NS that the VNF belongs. This is vital for many reasons. On the one hand, a VNF could be shared between multiple NSs, so *ID* not combined with the

```
//NetworkService data struct def
type NetworkService struct {
        ObjectType string `json:"docType"`
        ID         string `json:"id"`
        Name       string `json:"name"`
        Tenant     Tenant `json:"tenant"`
}

//VNF data struct def
type VNF struct {
        ObjectType  string `json:"docType"`
        ID          string `json:"id"`
        Name        string `json:"name"`
        ShortName   string `json:"short_name"`
        Description string `json:"description"`
        Provider    string `json:"provider"`
}

//Lease data struct def
type Lease struct {
        ObjectType string   `json:"docType"`
        Grantor    string   `json:"grantor"`
        Recipient  string   `json:"recipient"`
        Vnf        string   `json:"vnf"`
        Issue      int32    `json:"issue"`
        Expiry     int32    `json:"expiry"`
        Revokers   []string `json:"revokers"`
}
```

Listing 1: Assets definition

Provider attribute does not uniquely identify a specific instance of a VNF. On the other hand, towards automated management, as it will be described later, it is necessary to have the full ownership chain information, as which VNF instance belongs to which NS, and the tenant who owns this NS.

2) Lease: As shown in Listing 1, *Lease* describes the VNF lease to an interested user. Whenever a network slice should include shared NSs of other slices and establish CSC, the struct *lease* describes the asset that would be stored in the ledger after a successful CSC instantiation. The *Grantor* and *Recipient* attributes describe the NSs IDs of the NS provider and

consumer respectively. Based on these unique identifiers, the Orchestrator is able to realize all preparatory steps to establish the CSC and enables the Recipient to consume the provided service by the Grantor. In addition, with the NS ID, the owners information (tenant) can be retrieved for functions like billing and others. The *Vnf* field, describes the ID of the VNF to be leased. The *Issue* and *Expiry* attributes describe the period of the lease described as Unix timestamps. The *Revokers* array contains the identities enrolled to the Blockchain that can revoke the lease. The permission to revoke a CSC will be defined by policies and decisions of the provider, and the parties involved in the Lease.

3) Billing: Finally, templates should be created for billing purposes, depending on the nature of the VNF to be leased. For example, a CPU intensive service, such as video processing, face recognition, and others, could have quotas and billing rates in relation with CPU usage, whereas a video streaming cache could have billing rates and quotas related to network traffic.

B. Service Lease and Orchestration Workflow

In this subsection, we will present our Chaincode functionality and the full service lease and orchestration workflow in our proposed framework. Our Chaincode packs four smart contracts aligned with the different assets, namely *manage tenant, manage network service, manage VNF and manage lease*, as described previously. The first three contracts basically handle all the CRUD (Create, Read, Update, Delete) operations for the respective assets. When an asset is "deleted" is not actually deleted from the ledger and there is no alteration of the blocks of the Blockchain. On the contrary, the key-value of the asset is only removed from Fabric state database and if a new read operation requests the asset, it will not "exist" as it will not have any current state. However, all logs and previous values and states of the asset are accessible from the ledger. The above three contracts offer mainly management functionalities, thus, the corresponding analysis is omitted.

At this point, we will thoroughly explain the *manage lease* smart contract and the service lease and orchestration of the cross slice communication needed. In contrast to the previous assets, the lease asset only gets committed in the ledger upon a user's request. It is not a static asset, like tenant and network services, and it is essential for service advertisement, discovery, and all the management functions. This contract also provides the basic CRUD operations in a different manner. The Create operation is included in the orchestration workflow. Additionally to the CRUD operations, the manage lease contract offers functions to check if a lease is valid or expired, lease suspension, and lease revocation functions.

The lease registration and orchestration workflow is illustrated in the sequence diagram in Figure 3. Possible errors in consensus, or other aspects of the Blockchain functionality and life-cycle are omitted for clarity. For example, all Blockchain transactions are assumed successful and we do not present the case of an endorsement failure because the possibility of a transaction failure is irrelevant to the workflow and lifecycle of our proposed solution, and adding all these subcases would make Figure 3 to complex. So only lease registration and orchestration are presented with all possible outcomes and scenarios (lease exists, lease does not exist e.t.c).

The workflow begins when a user requests a lease. The information passed to the Chaincode include the Grantor's NS ID, the Recipient's NS ID, the VNF ID of the requested VNF for leasing, Issue (lease start time of the requested lease in Unix Time), Expiry (Expiry time of the lease in Unix Time) and an array of eligible Revokers of the lease as, described in Listing 1. Once the register Lease function is invoked, the Chaincode reads the ledger to check if the lease already exists or not. Depending on the response, the Chaincode will inform with the appropriate error message if the lease already exists. Alternatively, it will verify if the requested VNF exists in the ledger. If the lease does not exist, the user is informed with the corresponding error. On the contrary, as described in Listing 1, the lease is committed to the ledger. After the successful transaction, the Chaincode returns a confirmation of the lease grant and registration to the user and emits an event to the Orchestrator and exits. Fabric allows event based communication with subscribed and authorized clients. The Orchestrator receives the event with all the necessary parameters to Orchestrate the CSC, as described in section III-B.

Finally, the Orchestrator contacts accordingly the VIMs associated with the Network Services (Slices) involved in the lease in order to deploy all the necessary elements for the CSC to take place. Once the VIMs complete their tasks, they inform the Orchestrator for the success of operations and then, the Orchestrator invokes the Chaincode, which stores the orchestration success to the ledger for complete logging of the transactions and the orchestration operations, and emits an event to the user client to inform him that the orchestration has completed successfully.

C. Billing

In the CSC scenario, there are two main ways to manage the billing of the leased services. One available option for the provider is to provide the monitoring and billing calculations based on billing templates created by the tenant who provides the service for lease.The second option involves the implementation of intermediary slices in the CSC to manage the monitoring and billing depending also on the characteristics of the provided service (Network, Compute, etc.). However, both solutions present several challenges and issues that must be addressed. The first one creates administrative, development and resource overhead for the provider, and at the same time it requires absolute trust from the CSC counterparts. The latter requires extra resources for the CSC which are controlled by the EC provider. In that case, a charging model must be decide, which consequently introduce extra orchestration and management overhead.

Contrary to these options, and in order to overcome and address these challenges, we propose to utilize completely the

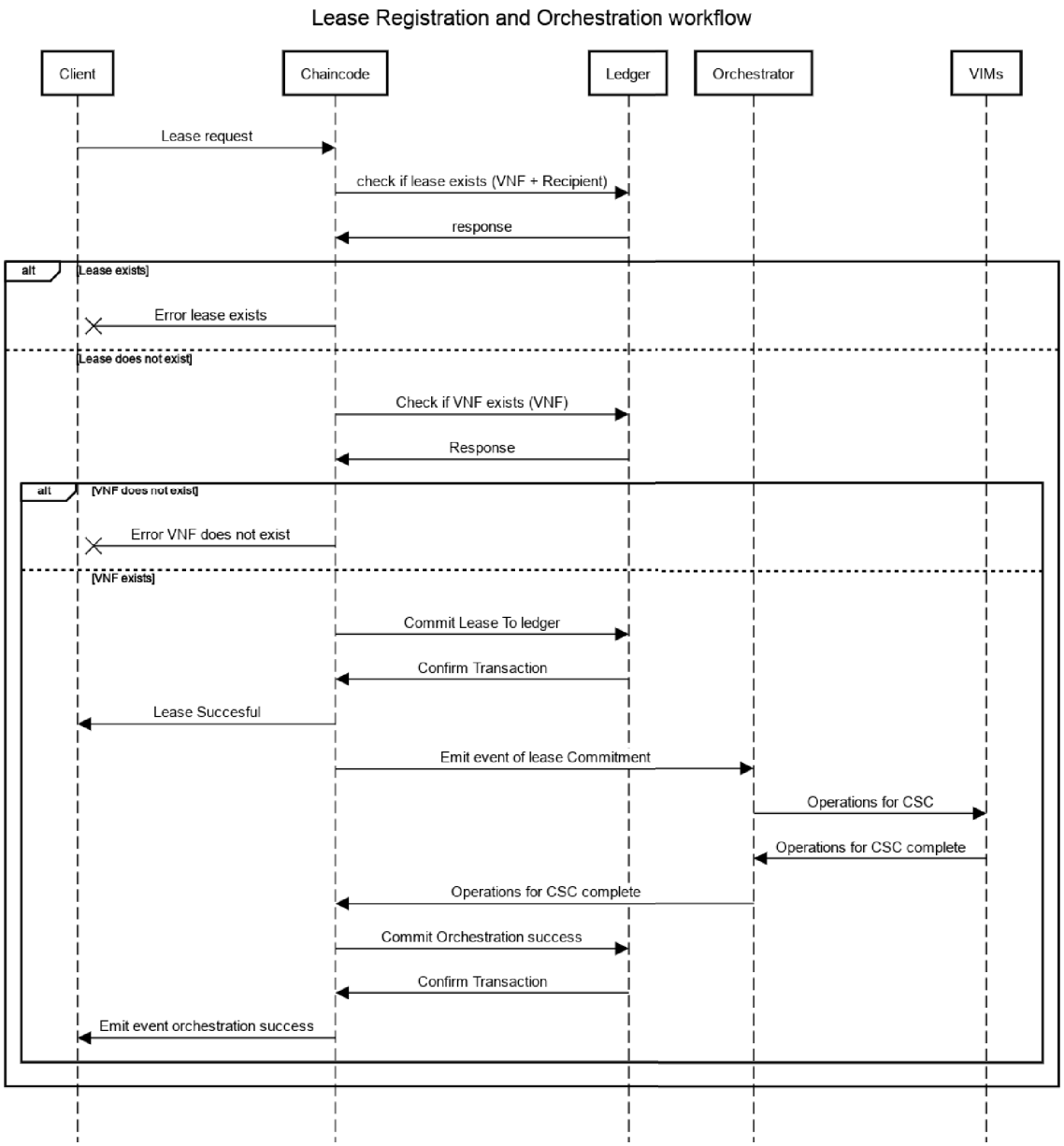

Fig. 3. Service Lease and Orchestration Sequence Diagram

billing management and process with smart contracts. With this capacity, no extra resources need to be reserved and no additional development and management should be performed by either the provider or the service consumer. Towards this direction, we need to define clearly the parameters needed to be submitted to the ledger or the smart contract and the respective assets to describe them to fit the needs of services of different nature and billing approaches. Also, we need to define how and where the monitoring will occur (Network Service, VNF or from the VIMs). Due to the overall benefits of this approach, providing trust between parties, low resource consumption, and minimum management and development overhead, we work towards this direction and will be included on the road to make this work a fully functional framework.

V. Conclusion

In this study, we proposed a proof of concept design and early implementation of a distributed framework for enabling cross-slice communication and orchestration over multi-domain EC environment, aligned with the ETSI NFV standard. The proposed architecture exploits Blockchain technology which enables parties to interact in a trustworthy manner, guarantee data integrity, high availability, fault tolerance and

immutable logs of transactions. Our solution aims at providing low resource demands and minimum extra administrative and development overheads for providers, tenants and service providers. We presented a first full service lease and CSC orchestration workflow and lifecycle, while also providing some early design principles and benefits for the billing functions in such an environment.

Our current and future research and development plan mainly focuses on the implementation of a fully functional framework. On this road we should define additional criteria, requirements and use cases aligned with the ETSI NFV standard, EC providers and industry practices, common use cases and design patterns. Furthermore, the fully implemented version of the proposed Blockchain-based framework, will be compared against centralized trust management solutions, in terms of performance, resource consumption and scalability.

ACKNOWLEDGMENT

The research work of Mr. Papadakis-Vlachopapadopoulos was co-financed by Greece and the European Union (European Social Fund- ESF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning" in the context of the project "Strengthening Human Resources Research Potential via Doctorate Research" (MIS-5000432), implemented by the Greek State Scholarships Foundation (IKY). The research work of Mr. Dechouniotis is co-financed by Greece and the European Union (European Social Fund- ESF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning" in the context of the project "Reinforcement of Postdoctoral Researchers - 2nd Cycle" (MIS-5033021), implemented by the State Scholarships Foundation (IKY).

REFERENCES

[1] D. Dechouniotis, N. Athanasopoulos, A. Leivadeas, N. Mitton, R. M. Jungers, and S. Papavassiliou, "Edge Computing Resource Allocation for Dynamic Networks: The DRUID-NET Vision and Perspective," *Sensors*, vol. 20, no. 8, p. 2191, 2020.

[2] G. Papathanail, I. Fotoglou, C. Demertzis, A. Pentelas, K. Sgouromitis, P. Papadimitriou, D. Spatharakis, I. Dimolitsas, D. Dechouniotis, and S. Papavassiliou, "COSMOS: An Orchestration Framework for Smart Computation Offloading in Edge Clouds," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–6.

[3] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Risk-Aware Data Offloading in Multi-Server Multi-Access Edge Computing Environment," *IEEE/ACM Transactions on Networking*, 2020.

[4] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[5] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[6] G. Mitsis, P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Intelligent Dynamic Data Offloading in a Competitive Mobile Edge Computing Market," *Future Internet*, vol. 11, no. 5, p. 118, 2019.

[7] I. Fotoglou, G. Papathanail, A. Pentelas, P. Papadimitriou, V. Theodorou, D. Dechouniotis, and S. Papavassiliou, "Towards Cross-Slice Communication for Enhanced Service Delivery at the Network Edge," in *IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 2020, pp. 1–10.

[8] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.

[9] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[10] Linux Foundation, "Hyperledger Fabric," Jun 2020. [Online]. Available: https://github.com/hyperledger/fabric

[11] D. Dechouniotis, I. Dimolitsas, K. Papadakis-Vlachopapadopoulos, and S. Papavassiliou, "Fuzzy Multi-Criteria Based Trust Management in Heterogeneous Federated Future Internet Testbeds," *Future Internet*, vol. 10, no. 7, p. 58, 2018.

[12] ETSI, "Open Source MANO," 2020. [Online]. Available: https://osm.etsi.org/

[13] OpenStack, 2020. [Online]. Available: https://www.openstack.org/

[14] Kubernetes, 2020. [Online]. Available: https://kubernetes.io/

[15] E. Jonas, J. Schleier-Smith, V. Sreekanti, C.-C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, J. Carreira, K. Krauth, N. Yadwadkar *et al.*, "Cloud Programming Simplified: A Berkeley View on Serverless Computing," Tech. Rep., 2019.

[16] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Risk-aware Social Cloud Computing based on Serverless Computing Model," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[17] K. Papadakis-Vlachopapadopoulos, R. S. González, I. Dimolitsas, D. Dechouniotis, A. J. Ferrer, and S. Papavassiliou, "Collaborative SLA and reputation-based trust management in cloud federations," *Future Generation Computer Systems*, vol. 100, pp. 498–512, 2019.

[18] N. Pustchi, F. Patwa, and R. Sandhu, "Multi Cloud IaaS with Domain Trust in Openstack," in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, 2016, pp. 121–123.

[19] S. Thakur and J. G. Breslin, "A Robust Reputation Management Mechanism in Federated Cloud," *IEEE Transactions on Cloud Computing*, 2017.

[20] N. El Ioini and C. Pahl, "Trustworthy orchestration of container based edge computing using permissioned Blockchain," in *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*. IEEE, 2018, pp. 147–154.

[21] "PROV-Overview," https://www.w3.org/TR/prov-overview/, (Accessed on 06/17/2020).

[22] N. Bozic, G. Pujolle, and S. Secci, "Securing virtual machine orchestration with Blockchains," in *2017 1st Cyber Security in Networking Conference (CSNet)*. IEEE, 2017, pp. 1–8.

[23] S. Nayak, N. C. Narendra, A. Shukla, and J. Kempf, "Saranyu: Using smart contracts and blockchain for cloud tenant management," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE, 2018, pp. 857–861.

[24] N. Afraz and M. Ruffini, "5G Network Slice Brokering: A Distributed Blockchain-based Market."

[25] R. V. Rosa and C. E. Rothenberg, "Blockchain-Based Decentralized Applications Meet Multi-Administrative Domain Networking," in *Proceedings of the ACM SIGCOMM Conference on Posters and Demos*. ACM, 2018, p. 114–116.

[26] 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.

[27] "ETSI - Standards for NFV - Network Functions Virtualisation — NFV Solutions," https://www.etsi.org/technologies/nfv, (Accessed on 06/17/2020).

[28] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm (extended version)," 2013.

[29] ETSI, "GR NFV-IFA 028 - V3.1.1 - Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Report on architecture options to support multiple administrative domains," https://www.etsi.org/deliver/etsi_gr/NFV-IFA/001_099/028/03.01.01_60/gr_nfv-ifa028v030101p.pdf, (Accessed on 06/11/2020).