informatics

Article

On Blockchain-Based Cross-Service Communication and Resource Orchestration on Edge Clouds

Konstantinos Papadakis-Vlachopapadopoulos [1,*], Ioannis Dimolitsas [1], Dimitrios Dechouniotis [1], Eirini Eleni Tsiropoulou [2], Ioanna Roussaki [1] and Symeon Papavassiliou [1]

[1] School of Electrical and Computer Engineering, National Technical University of Athens, 15780 Athens, Greece; jdimol@netmode.ntua.gr (I.D.); ddechou@netmode.ntua.gr (D.D.); Ioanna.Roussaki@cn.ntua.gr (I.R.); papavass@mail.ntua.gr (S.P.)
[2] Department of Electrical and Computer Engineering, The University of New Mexico, Albuquerque, NM 87131-0001, USA; eirini@unm.edu
* Correspondence: cpapad@netmode.ntua.gr; Tel.: +30-210-772-1449

Abstract: With the advent of 5G verticals and the Internet of Things paradigm, Edge Computing has emerged as the most dominant service delivery architecture, placing augmented computing resources in the proximity of end users. The resource orchestration of edge clouds relies on the concept of network slicing, which provides logically isolated computing and network resources. However, though there is significant progress on the automation of the resource orchestration within a single cloud or edge cloud datacenter, the orchestration of multi-domain infrastructure or multi-administrative domain is still an open challenge. Towards exploiting the network service marketplace at its full capacity, while being aligned with ETSI Network Function Virtualization architecture, this article proposes a novel Blockchain-based service orchestrator that leverages the automation capabilities of smart contracts to establish cross-service communication between network slices of different tenants. In particular, we introduce a multi-tier architecture of a Blockchain-based network marketplace, and design the lifecycle of the cross-service orchestration. For the evaluation of the proposed approach, we set up cross-service communication in an edge cloud and we demonstrate that the orchestration overhead is less than other cross-service solutions.

Keywords: network services; blockchain; service marketplace; edge cloud; resource orchestration

check for updates

Citation: Papadakis-Vlachopapadopoulos, K.; Dimolitsas, I.; Dechouniotis, D.; Tsiropoulou, E.E.; Roussaki, I.; Papavassiliou, S. On Blockchain-Based Cross-Service Communication and Resource Orchestration on Edge Clouds. *Informatics* **2021**, *8*, 13. https://doi.org/10.3390/informatics8010013

Academic Editor: Antony Bryant

Received: 31 January 2021
Accepted: 21 February 2021
Published: 26 February 2021

1. Introduction

With the advent of the Internet of Things (IoT) and 5G era, modern time- and mission-critical applications with stringent requirements have been developed for every section of human activity. These complex end-to-end applications consist of network services that form a structured service chain. However, the abundant cloud resources cannot guarantee the delay requirements, and this motivates the need for the infrastructure operators to provide computing capabilities closer to end users. Thus, presently, the Edge Computing service delivery model is the most promising one to meet the requirements of applications derived from 5G verticals, Industry 4.0 [1] and smart cities [2,3]. As with cloud computing, the resource orchestration at the edge of the network relies on the Network Function Virtualization (NFV) [4] and Software-Defined Networking (SDN) [5] concepts that facilitate network slicing. A network slice is actually a structured chain of network services that perform specific tasks of the application logic. This allows infrastructure providers to lease customized bundles of computing, storage and network resources, to their tenants allowing the addressing of the service-specific (non-)functional requirements in an isolated fashion. Many standardization initiatives, i.e., ETSI Multi-access Edge Computing (MEC) (https://www.etsi.org/technologies/multi-access-edge-computing (accessed on 31 January 2021)) and ETSI-NFV (https://www.etsi.org/technologies/nfv (accessed on

31 January 2021)) aim to describe and standardize the specifications on the automated resource orchestration and support of the entire lifecycle of network services.

The evolution of 5G ecosystems and verticals requires the deployment of end-to-end network services over multiple domains or multi-administrative domain. Under this multi-tenant and multi-provider setting, the isolation of network slicing, which is its greatest advantage till now, impedes the cross-service communication (CSC). Thus, next-generation network service marketplaces (NSMs) aspire to overcome this obstacle and enable tenants to create tailor-made service chains using off-the-shelf network services that can consume other services in a secure manner. With this capacity, a multi-tier NSM should provide various functionality elements for both service provider and consumer. At the top layer of the marketplace, functionalities such as easy registration, enriched description and on-boarding of services should be provided to service providers. Furthermore, a service discovery and lease mechanism are essential for the tenants, who search for the network service to be included in their custom service chain. The next downward layer is responsible for handling the high-level information of every service chain and automating the resource orchestration of the network services. At the bottom layer, the service chain is instantiated by the resource orchestrator.

At the middle layer of the network service marketplace, various transactions between the evolving entities occur. Although a centralized trusted authority and a trust management mechanism provide the desired level of trust, this is usually performed through non-automated functionality and requires a lot of human intervention. Furthermore, the complexity of trust management significantly increases in the case of multi-domain or multi-administrative resource orchestration. Contrary to centralized solution, Blockchain enables entities in trustless environment to make transactions in a decentralized manner and guarantees the integrity of the outcome of the transactions to all participants of the Blockchain network [6]. The transactions are stored in a ledger in the form of blocks and the order of the blocks is based on cryptographic hashes. The process that the blocks are produced ensures the immutability, data integrity, non-repudiation and security of the Blockchain ledger. Furthermore, Blockchain platforms such as Hyperledger Fabric [7], provide smart contracts that enable an event-based execution of transactions based on predefined rules and conditions. This is of high importance in incorporating any business logic as automated functionality in the Blockchain network.

This article proposes a Blockchain-based NSM and a resource orchestration mechanism that enables the CSC in edge cloud (EC) and the establishment of the NSM. This mechanism is based on Hyperledger Fabric, a permissioned Blockchain platform, and leverages the functionality of smart contracts to automate the interaction among network services that are part of different service chains within an EC. Capitalizing on and extending our previous work [8], the main contributions of the article are summarized as follows:

- At the user layer, we specify the essential functionalities, i.e., registration, service discovery, service lease and billing, for both network service providers and consumers to become stakeholders of the marketplace.
- At the Blockchain layer, we update the asset definitions of [8] to facilitate cross-service orchestration with minimum possible data volume on the distributed ledger and preserve privacy of the users regarding their Network Slice information and structure.
- At the NFV layer, we provide an analytical description of our novel Service Orchestrator that establishes CSC orchestration leveraging Blockchain capabilities.
- We provide proof of concept results that demonstrate the applicability and efficiency of the proposed approach in terms of deployment time and reserved resources.

The rest of the paper is organized as follows. In Section 2, the related literature is presented. Section 3 describes the architecture and its layers, while Section 4 presents the functionality of the NSM. Section 5 describes the interactions of the automated CSC and Section 6 highlights the benefits of the proposed approach in a real experimental use case. Finally, Section 7 draws the conclusions and proposed possible future extensions of the presented work.

2. Related Work

In this section, the most recent and interesting studies and projects relevant to the cross-service, multi-domain and Blockchain-based orchestration are presented.

On top of the open-source MANO (OSM) [9], MESON orchestration platform enables the cross-slice communication within an Edge Point of Presence (EPoP) [10]. Aligned with ETSI-NFV [11] and ETSI MEC [12] architecture, MESON platform provides centralized service discovery through a service registry, EPoP selection and establishment of cross-slice communication. Based on functional and non-functional requirements, the EPoP selection is performed by a multi-criteria EC selection methodology that determines the most suitable EC for the slice deployment [13]. Then, through OSM, an automated mechanism establishes the actual cross-slice communication that aims to minimize the generated intra-EC network traffic and allocated resources [14]. Also aligned with ETSI-NFV architecture, FENDE marketplace [15] enables developers to offer their VNF-based services, while users can select Virtual Network Functions (VNFs) from the service and VNF catalogue, compose custom service chains and instantiate them in public or private cloud infrastructures. NECOS marketplace is a broker-based platform that enables the resource discovery, negotiation and selection for deploying network slice over multiple administrative domains [16], while the 5GEx platform focuses on cross-domain orchestration of network services [17]. With respect to the latter, a hierarchical architecture of multi-domain orchestrators enables the exchange of business-level information with customers, the service discovery and placement and supports the VNF configuration and monitoring.

Rathi et al. proposed a Blockchain-enabled multi-domain orchestrator at the edge of the network [18]. The objectives of this approach are the automated orchestration of network slices over heterogeneous networks and the Service Level Agreement (SLA) assessment for 5G services using smart contracts. Nubo is a virtual service marketplace created to connect providers and consumers of such services in an edge/cloud computing environment [19]. Nubo is basically a web portal built over and extending Saranyu [20], which is a decentralized application (DApp) built on top of Quorum private Blockchain [21]. Saranyu is used for the management of cloud tenants and services, providing identity management, authentication and charging functionality with the use of smart contracts. The authors in [22] present an experimental prototype for a multi-administrative domain federation of 5G services using distributed ledger technologies. Its objective is to provide a distributed federation solution where different administrative domains and providers participate in a permissioned Blockchain network with a single node. A single generic Federation Smart Contract acts as a distributed authority to enable secure and trusted interactions for functionalities such as registration to the federation, service announcement/discovery and service auctioning. The 5GZORRO project proposes an architecture focused on cross-domain security and trust resource orchestration mechanisms, by coupling Blockchain with AI-driven operations and service lifecycle automation in multi-tenant and multi-stakeholder environment [23]. Specifically, smart contracts are used to support multilateral agreements among all parties that are involved in the end-to-end service delivery. In [24], a Blockchain-based resource brokering mechanism for end-to-end slice deployment is proposed to secure transactions in the resource auctioning procedure. Upon a slice request, the slice is divided in sub-slices and an auction mechanism facilitates the resource providers to make bids for every sub-slice. The slice owner selects the best offers for each sub-slice. The authors of [25] proposes BSec-NFVO, a Blockchain-based system, to enable secure orchestration operations in virtualized networks ensuring auditability, non-repudiation and integrity. This mechanism is built on top of the Open Platform for Network Function Virtualization (OPNFV) (https://www.opnfv.org/ (accessed on 31 January 2021)). The solution ensures security in delivering and orchestrating end-to-end NFV service chains with small performance overheads. In [26], a proof of concept Blockchain-based implementation using distributed applications is presented in the context of operational phases to support multi-administrative domain networking for multi-domain service orchestration. Also, the authors analyze three use case scenarios (MEC, 3GPP and ETSI-NFV standards) discussing standardization opportunities around Blockchain-based DApps as enablers for multi-domain network services.

Our work complements and extends the above approaches and creates a Blockchain-based cross-service mechanism for ECs that alleviate the orchestration overhead and facilitates the establishment of NSMs. Leveraging the capabilities of smart contracts, the service provider and consumers can easily trigger and monitor the lifecycle operations of their services. Furthermore, based on the power of the smart contracts and OSM, the proposed Service Orchestrator enables the automated establishment of the cross-service communication with the minimum resource requirements and instantiation overhead.

3. Network Service Marketplace Architecture

3.1. NFV-Related Definitions

Towards the establishment of CSC and NSM, it is important to describe the basic NFV assets that are taken into account in the proposed solution. Aligned with the ETSI-NFV architecture [11], a service chain corresponds to the network slice, which describes the main interactions as well as the network communications between the services. Based on OSM information model [9], a service chain is described in the Network Slice Template (NST). An example of such NST is shown in Listing 1. The main fields of the NST are the *netslice-subnet* and the *netslice-vld*. The *netslice-subnet* objects are the declared services of the slice. For each service, the slice owner determines the corresponding network service descriptor—which in OSM model is defined as *nsd*—in the *nsd-ref* key, the name of the service and a Boolean key about enabling sharing for a specific service with other slices, in the *is-shared-nss* field. Therefore, a slice owner can share any of the services that is included in an NST. The *netslice-vld* objects correspond to the networking management of the service chain. A *netslice-vld* object consists of various parameters. These parameters refer to both Management and Orchestration (MANO) level information (e.g., the identifier, name of the virtual network—vld) and to the Virtual Infrastructure Manager (VIM) level, such as configuration about VIM networking. Also, the connection points for each service, which is attached to that vld, are included. A network service consists of one or more VNFs, in which the necessary functionality of the service is implemented. Each VNF constitutes of a specific instance at the VIM level. The interested reader can refer to [8] for more details about network service and VNF descriptors.

Considering the above features of the OSM Model and based on the ETSI-NFV architecture, we can design appropriate architectures and implement efficient mechanisms, in order to facilitate the cross-service interaction between services of different slice owners, leading to the network service marketplaces establishment.

```
# NST descriptor example of a Service Provider. The slice consist of 2 Services. The second of them is shared
# among other slice tenants.
nst:
-   id: provider_slice_nstd
    name: provider_slice_nstd
    SNSSAI-identifier:
        slice-service-type: eMBB
    quality-of-service:
        id: 1

    netslice-subnet:
    -   id: provider-service-1
        is-shared-nss: 'false'
        description: NetSlice Subnet (service) composed by 1 vnf with 2 cp
        nsd-ref: provider-service-1_nsd
    -   id: provider-service-2
        is-shared-nss: 'true'
        description: NetSlice Subnet (service) composed by 1 vnf with 3 cp
        nsd-ref: provider-service-2_nsd


    netslice-vld:
    -   id: slice_vld_mgmt
        name: slice_vld_mgmt
        type: ELAN
        mgmt-network: 'true'
        nss-connection-point-ref:
        -   nss-ref: provider-service-1
            nsd-connection-point-ref: nsd_cp_mgmt
        -   nss-ref: provider-service-2
            nsd-connection-point-ref: nsd_cp_mgmt
    -   id: slice_vld_data
        name: slice_vld_data
        type: ELAN
        mgmt-network: 'false'
        nss-connection-point-ref:
        -   nss-ref: provider-service-1
            nsd-connection-point-ref: nsd_cp_data
        -   nss-ref: provider-service-2
            nsd-connection-point-ref: nsd_cp_data
```

Listing 1. Provider's NSTD.

3.2. System Architecture

Our proposed Blockchain-based service marketplace architecture is designed in alignment with the ETSI-NFV standardization activities, and supports all phases and functionalities of a NSM, including registration, advertisement, leasing, orchestration, usage and billing. Figure 1 provides a high-level overview of the components of the NSM architecture. As it is shown, the architecture consists of three layers corresponding to different functionality elements and operational phases.

3.2.1. User Layer

The User layer provides the essential functionality required for the interaction of the users with the NSM, as it is shown in the left part of Figure 1. Through a client, portal and/or some dashboard, both service providers and consumers can register a service available for leasing, discover service features, request service lease and trigger service instantiation, and establish CSC. In addition, this layer provides to the users monitoring and billing functionalities for the leased network services. Any data storage or retrieval is performed by invoking the appropriate smart contracts through the Blockchain layer's interfaces.

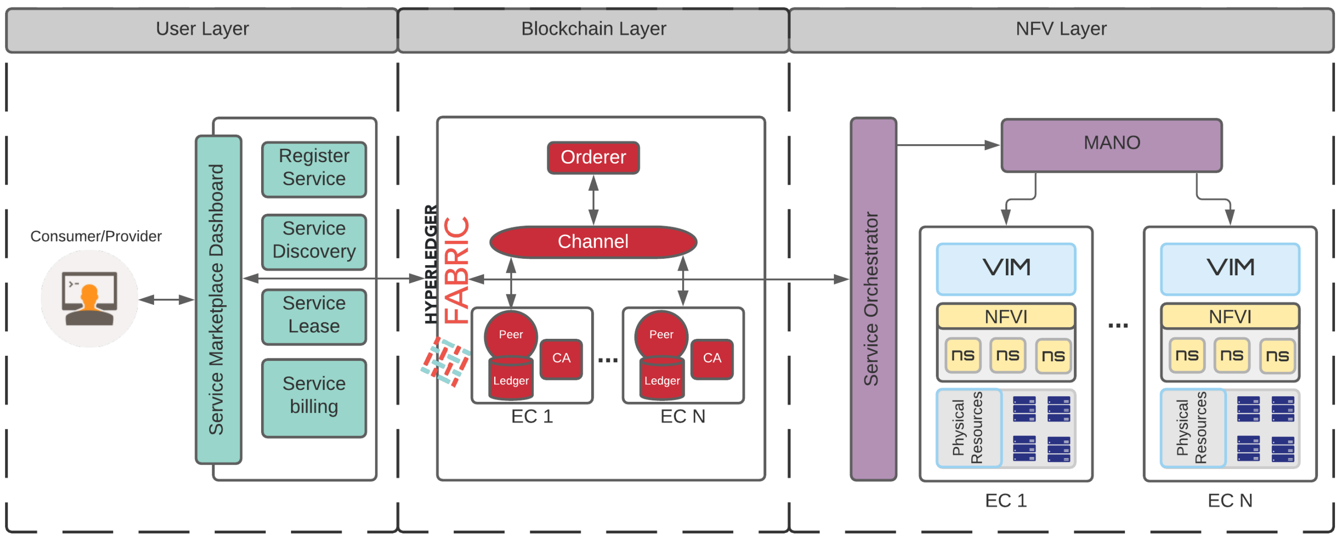

Figure 1. Network Service Management Architecture.

3.2.2. Blockchain Layer

The Blockchain layer handles through smart contracts all information required regarding users, services etc. This study leverages the Blockchain functionality, the smart contracts and the components described in [8]. Additionally, as it will be described in the following sections, we update and extend the above components. As shown in the middle part of Figure 1, the Blockchain layer architecture refers either to a single EC deployment or a multi-domain EC environment, which can be either single- or multi-administrative. In both scenarios, each EC is considered to be an individual organization in the Fabric Network. Each organization has instantiated a Fabric Peer with the smart contracts, a Certificate Authority to manage identities and a Ledger. The information on the Ledger is identical in every EC. All organizations join a common Fabric channel and finally the orderer handles the consensus mechanism and is responsible for ordering transactions, creating new blocks and distributing a newly created block to all peers in a channel. The Blockchain Layer also forwards all cross-service orchestration requests and information to the Service Orchestrator of the NFV layer, which is thoroughly analyzed in Section 4.2.

3.2.3. NFV Layer

Aligned with ETSI-NFV architecture, the NFV layer is responsible for the network service instantiation and the establishment of the cross-service communication. On top of this layer our extended Service Orchestrator offers additional functionality elements and abstractions to OSM, in order to enable fully automated CSC. The Service Orchestrator handles the above by interacting with OSMs APIs. Section 5.2 provides analytical description of the developed Service Orchestrator.

4. Network Service Marketplace Operations

In this section, we describe the concepts and phases of the proposed Network Service Marketplace. The main concept and ambition of the NSM is to enable EC tenants, even from different providers or domains, to act as providers of their own services to candidate customers, or being consumers of off-the-shelf services instead of having to develop and deploy them on their own. Such services could be for example image recognition services, media caches etc. For that purpose, we implement a distributed Blockchain-based marketplace using Hyperledger Fabric. The Blockchain solution enables untrusted parties to interact and perform transactions in a trustworthy and secure environment, through smart contracts without any intermediaries involved, and use the Blockchain distributed ledger as a datastore for our marketplace. Additionally, as the proposed NSM is aligned with the ETSI-NFV architecture, any ETSI-NFV-based EC can join the marketplace by just installing a Fabric Peer with its ledger and the smart contracts instantiated, accompanied with a certificate authority. For the case of EC, this process requires a very small amount of resources and minimum administrative overhead from the providers. Below we present in detail all functionality of all the layers of the proposed NSM.

4.1. Marketplace Functionality

The User layer supports the interaction of both providers and users with the NSM and it provides the following functionalities,

- **Registration**: In the Registration phase, a tenant of an EC registers to the marketplace. Also, some minimum information described in Section 4.2 is provided about the tenant's Network Slice for orchestration purposes. In the registration phase, a tenant can also register a service for leasing providing information about the service to be leased, quotas pricing, etc.
- **Advertisement**: After a service is registered for lease and committed to the datastore, it is automatically retrieved and advertised in a service catalogue for the rest marketplace users, in order to browse and select it for lease in an automated fashion.
- **Discovery**: In the discovery phase, the users of the marketplace can browse and select a service for lease according to their needs through the service catalogue.
- **Lease**: After service selection, users request the leasing of a service for a specific time period, which can be renewed. Upon request, the leasing is granted by the smart contract and then the appropriate establishment of cross-service communication is performed automatically, and the leased service is ready to be consumed.
- **Usage**: In the usage phase of a leased service, depending on the pricing model of the service, the usage of the service is monitored and stored in the data store for logging, while the user can track the usage of the leased service.
- **Billing**: Depending on the usage monitoring data of a leased service and the pricing model defined by the service provider, billing is calculated by a smart contract and the appropriate fee is sent to the consumer for a specific predefined time period. Also, the users can monitor their usage and expected fees at any moment.

As an example, we present the workflow of the Register Lease function of the smart contract that corresponds to one the Lease functionalities of the user layer. This function grants a requested service lease, commits the Lease object in the data store as described in Section 4.2 and supports the CSC orchestration process. As depicted in Figure 2, when a user requests a lease, the client invokes the register function of the Smart Contract. All attributes of the Lease object are provided as input. Then, the Smart Contract checks if the requested Service for lease exists in the ledger. If the Service exists, then the Smart Contract checks the received service object from the ledger to validate that the service requested belongs to the grantor specified in the request. Lastly, the Smart Contract checks if the lease requested has already been granted and already exists in the ledger. If it does not exist, the lease is granted and committed to the ledger and the client is informed for the lease success. If any of the above validations fails, the smart contract returns to the client the appropriate error message. The above functionality corresponds to the Request Lease operation as depicted in Figure 3. More details about all developed smart contracts can be found here (https://gitlab.com/cpapad/bcpaper (accessed on 31 January 2021)).

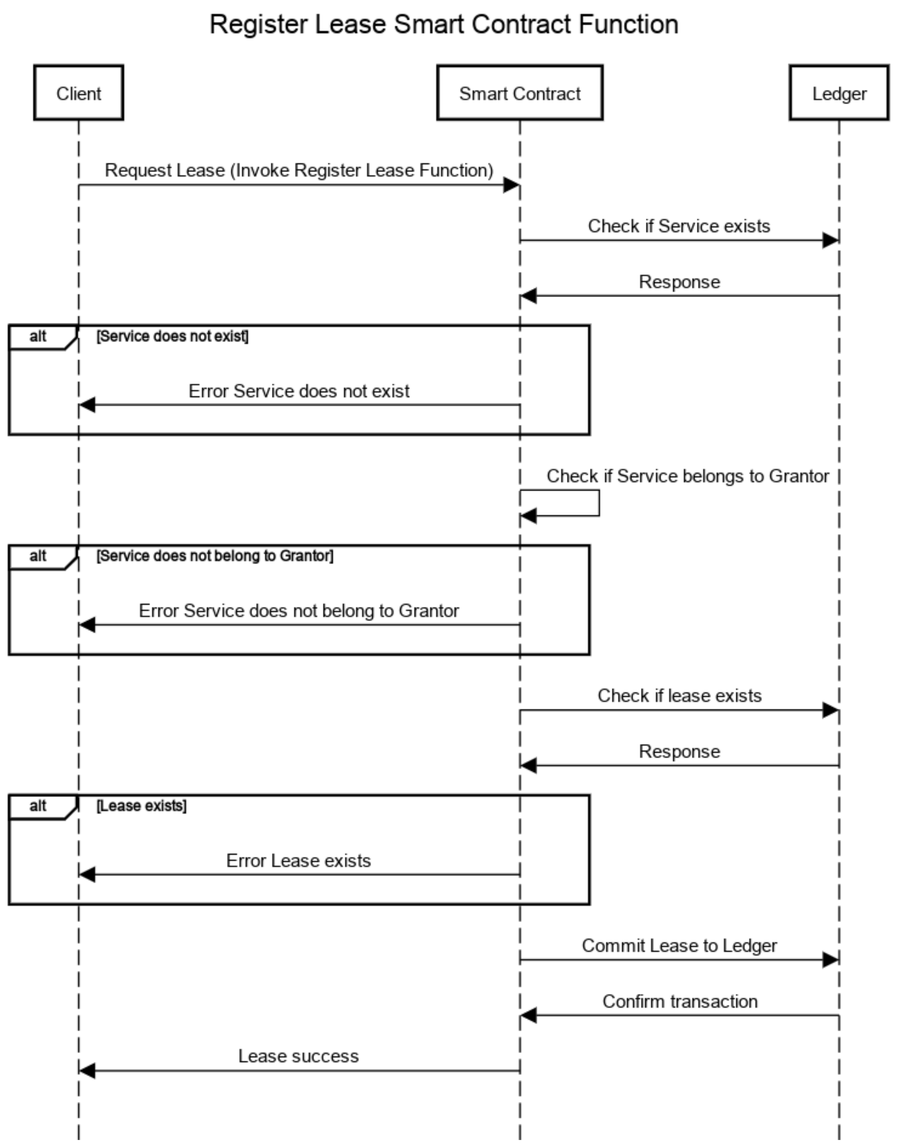
Register Lease Smart Contract Function
Client
Smart Contract
Ledger
Request Lease (Invoke Register Lease Function)
Check if Service exists
Response
alt
[Service does not exist]
Error Service does not exist
Check if Service belongs to Grantor
alt
[Service does not belong to Grantor]
Error Service does not belong to Grantor
Check if lease exists
Response
alt
[Lease exists]
Error Lease exists
Commit Lease to Ledger
Confirm transaction
Lease success

Figure 2. Register Lease Smart Contract Function.

4.2. Service Data Store-Blockchain Functionality

As above mentioned, we leverage Blockchain technology and smart contracts to support the network service lifecycle of the NSM and use the distributed ledger as the NSM data store, while ensuring trust, security, authenticity, integrity and immutability. In this subsection, we focus on the data stored in the data store. The data store is one of the most important components of the marketplace as it facilitates the service advertisement, leasing and orchestration functions of the marketplace. The stored information on the distributed ledger contains data both for service advertisement and discovery, and cross-service orchestration. The data required for the orchestration operations are fully aligned with the ETSI-NFV standard and refer to information about Network Slices and Services linking to information also stored in OSM database. Delving into more details, Listing 2 illustrates the structure and format of the stored data.

Initially, the **Network Slice** object refers to the set of virtualized computing and network resources required to deploy an application. The *ID* and *Name* attributes refer to the Network Slice Template descriptor of the slice as described by ETSI-NFV, and the *Tenant* attribute points to the object stored in the ledger that contains all information about the slice owner.

The **Service** object refers to the advertised network services for lease in the marketplace. As it is shown in Listing 2, it contains the following attributes; *ID, Name, Shortname, Description, NsdName and Provider*. The attributes of *Name, Shortname* and *Description* are used solely for advertisement and discovery purposes and contain information and functionality description of the service for lease. The attributes of *ID, NsdName* and *Provider* are used for orchestration, management and billing operations. In particular, the *NsdName* argument indicates the name of the service's Network Slice descriptor that contains all information and descriptions for the slice deployment. The *ID* attribute indicates the unique identifier of the service's Network Slice instance. The *Provider* attribute points to a Network Slice object in the ledger that contains the information of the Network Slice of the provider and the tenant to which it belongs.

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

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


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

Listing 2. Data store formats.

Finally, the **Lease** object contains the stored information in the distributed ledger whenever a lease is granted. The *Grantor* and *Recipient* attributes refer to the Network Slice descriptors of the Provider and the Consumer respectively for orchestration purposes. The *Service* argument contains the ID of the service to be leased as described above, while the *Issue* and *Expiry* attributes indicate the duration of the granted lease. Lastly, the *Revokers* attribute is an array containing the identities of the users or entities that can revoke the lease.

Regarding the **Network Slice** and **Service** objects, we aim at maintaining only the essential information for orchestration in the ledger for the following two reasons. Initially, preserving the users' data privacy is one of the top priorities of the NSM. With this capacity, we only store the name of the Network Slice's descriptor in the data store and not the whole descriptor which would reveal internal details and structure of a deployed slice. When orchestration is requested, our Service Orchestrator can retrieve the whole descriptor securely from the OSM's API to perform the required operations to enable CSC without storing any data. This procedure and all related operations are thoroughly presented in Section 5.2. Secondly, storing only the name of the descriptor, instead of the whole or partial information of it, we preserve the minimum amount of data in the data store.

5. Network Service Lease and Orchestration

In this section, we describe the lifecycle and component interaction of an automated cross-service orchestration. At first, we discuss the interaction lifecycle of the three layers of the architecture depicted in Figure 1 and then we present, step by step, the operations and interactions performed in the NFV layer within this lifecycle.

5.1. Network Service Lease

In this subsection, we briefly describe the lifecycle of a network service lease and cross-service orchestration. As Figure 3 illustrates, a client or dashboard sends a lease request to the Blockchain layer invoking the smart contract. If the lease is successful, its information is stored in the distributed ledger and the client/dashboard receives the appropriate message. After the acknowledgement of the successful leasing, the client/dashboard requests the appropriate CSC orchestration from the NFV layer by contacting the Service Orchestrator. Upon completion of the orchestration, the NFV layer returns a confirmation message to the client and invokes the smart contract, so that the establishment of the CSC is committed and stored to the distributed ledger mainly for logging purposes.

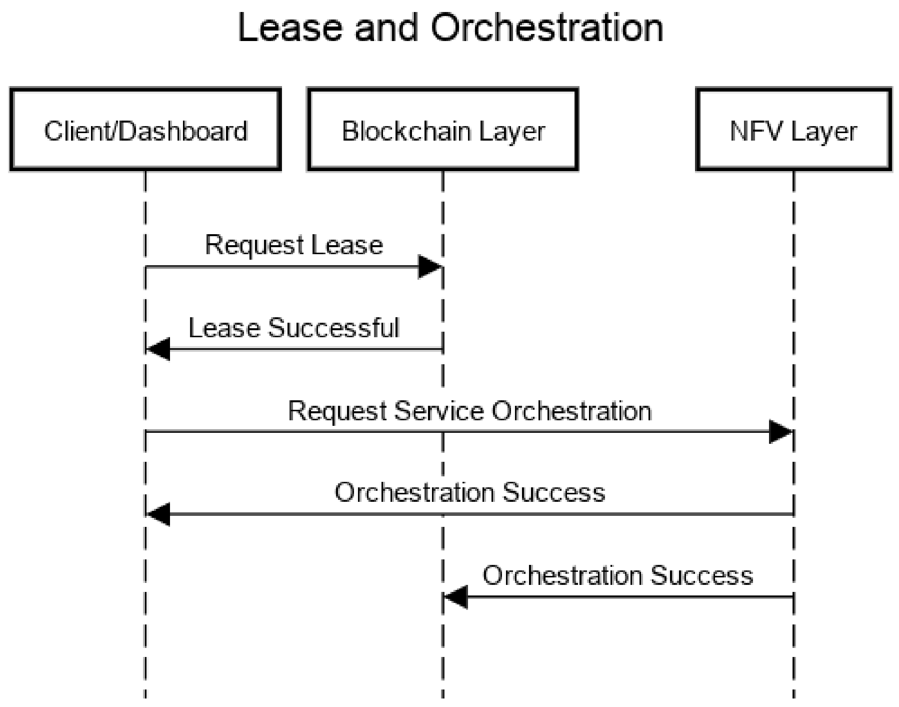

Figure 3. Lease and Orchestration Lifecycle.

5.2. Network Service Lease Orchestration

The main interactions between the Service Orchestrator component of the proposed architecture, the MANO, the Dashboard and the Blockchain layer are presented in Figure 4. In this subsection, we describe these interactions, from the scope of the NFV layer, in order to clarify the key functionality of the proposed Service Orchestrator. As a first step, the interested client for CSC instantiation, posts a request on the Service Orchestrator API. The request consists of the NST descriptor (NSTD) name, which is onboarded in the OSM and describes the service chain of the client, alongside with the identifier of the specific service of the provider, for which the user is interested in cross-service communication, among the set of the registered services in the data store. At the second step, the Service Orchestrator requests the provider's NSTD for the CSC-available service and extracts the necessary data from the descriptor file. Looking back on Listing 1, where an example of a NSTD of a service provider is depicted, we assume that a client is interested in the network service with identifier *"provider-service-2"*. In order to update the client's NSTD, we need the information stored in the *netslice-subnet* object, which matches with the *id* attribute, and the *netslice-vld* objects for the corresponding connection points of the desired service. Listing 3 illustrates the extracted data in a JSON file for the *"provider-service-2"* service. After retrieving the mandatory data from the provider's NSTD, the Service Orchestrator requests the onboarded client's slice NSTD. Then, using the extracted data from the provider's NSTD, the Service Orchestrator updates the client's NSTD accordingly, in order to be able to access the shared service. The updates in the descriptor refer to the addition of the shared-service object of the *netslice-subnet* field in the JSON file, and the corresponding connection points attachment at a management-network and a data network of the client's slice. The additions take place in the YAML NSTD file of consumers, in the corresponding fields. Finally, the updated NSTD is used for the instantiation request. For the instantiation, the Service Orchestrator constructs a configuration object, with parameters that refer to the VIM layer, in order to enable the network connections between the services of the client and the provider. After a successful instantiation, every component of the proposed architecture receives an orchestration success message.

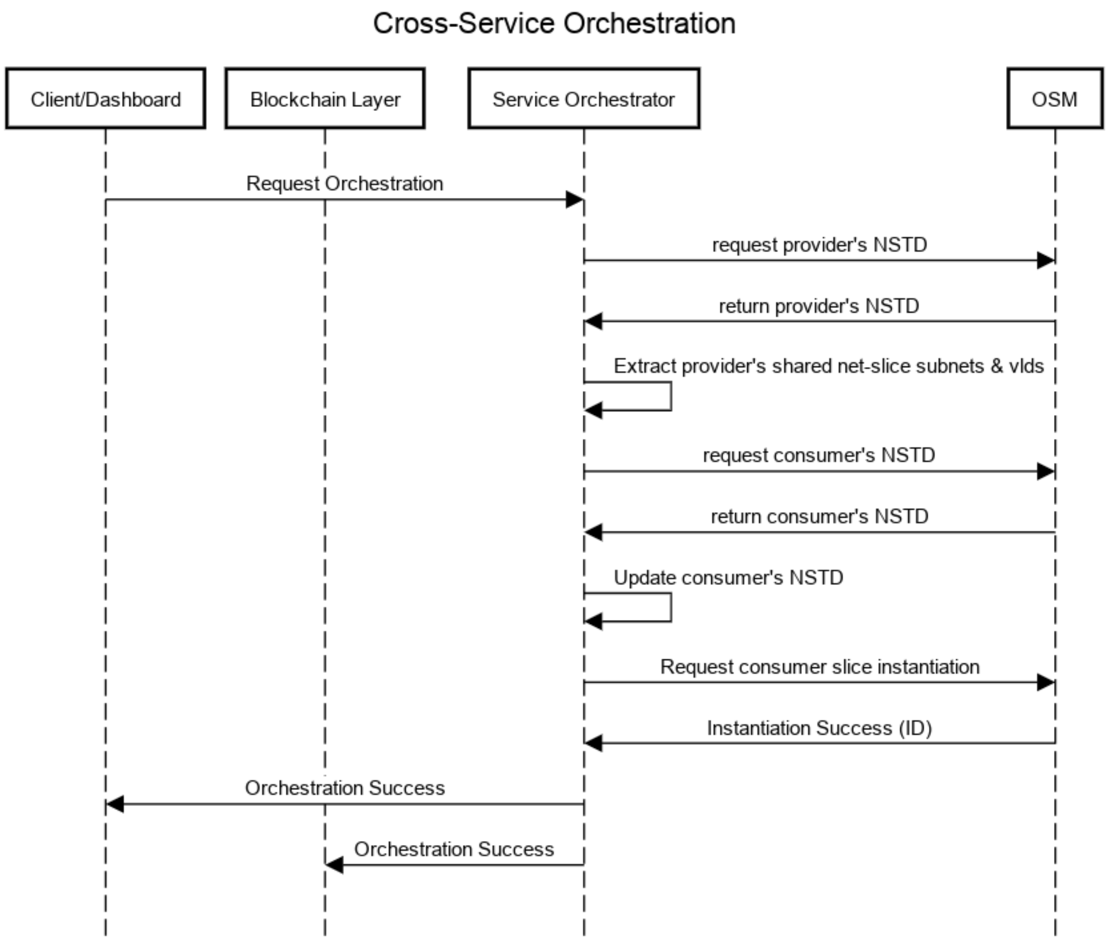

Figure 4. CSC interactions and operations.

```
{
        "netslice-subnet":
                {
                "id":"provider-service-2",
                "is-shared-nss": true,
                "description": "Provided service for cross-service interaction",
                "nsd-ref": "provider-service-2_nsd"
                },
        "mgmt-connector":
                         {
                                "nss-ref": "provider-service-2",
                                "nsd-connection-point-ref": "nsd_cp_mgmt"
                         },
        "data-connector":
                         {
                                "nss-ref": "provider-service-2",
                                "nsd-connection-point-ref": "nsd_cp_data"
                         }

}
```

Listing 3. Provider's extracted shared info.

6. Experimentation and Results

In this section, initial evaluation results are presented, aiming to demonstrate the benefits and feasibility of our proposed approach. For the assessment, the following experimental setup has been used. A virtual machine instance of 4 vCores and 8 GB of memory running Ubuntu 18.04 was used to deploy a test Hyperledger Fabric network v1.4 with two Fabric Peers with all instantiated smart contracts. OSM v8.0.4 as the MANO component, Openstack 5.4.0 as the VIM and the developed Service Orchestrator were deployed in a Virtual Machine (VM) with the same specifications. The Service Orchestrator is developed in Python (https://github.com/jdimol/cs_osm_client) and the API was built using the Flask framework. The client is implemented as a python script interacting with the Blockchain and NFV layer as depicted in Figures 3 and 4. Our evaluation scenario is performed using a single EC. We evaluate our solution in terms of orchestration time overheads and resource consumption compared to known practices in the literature for cross-service communication.

In terms of time overheads, OSM takes an average of 38 s to perform the cross-service orchestration (consumer slice instantiation) while all API and Blockchain interactions add an average of 2-s overhead. Thus, the interaction between the client, the Blockchain layer, the Service Orchestrator and the OSM corresponds to the 5% of the total cross-service orchestration time, which is considered insignificant for practical purposes. As far as scaling is concerned, depending on the version of Fabric and the programming language the Smart Contracts are developed, Hyperledger Fabric can perform from 194 transactions per second, up to 592 transaction per second (commit to ledger operation) for our assets size (https://hyperledger.github.io/caliper-benchmarks/fabric/performance/ (accessed on 31 January 2021)), while Openstack can process up to 5 transactions per second for network creation (https://docs.openstack.org/developer/performance-docs/test_results/openstack_load/index.html#create-neutron-networks (accessed on 31 January 2021)), which is the operation needed to enable CSC. Taking also into account that Openstack is one of the industries standards for Cloud, Edge and NFV infrastructures, it is evident that the service instantiation at the NFV layer is the bottleneck in terms of scaling for the establishment of CSC.

Additionally, the proposed architecture focuses on the minimization of the required computing and network resources for cross-service orchestration. For example, in typical cross-slice communication solutions, an intermediary network slice is deployed to establish the communication between the provider's slice and the consumer's one [10]. In terms of resources, this solution demands extra computing and network resources, such as VCPUs, Memory and Connection Points, in the VIM, where the slices are located. Considering the simplest scenario, which requires an intermediate network slice just for traffic forwarding, a VM with a minimal Linux distribution allocates at least one VCPU and 512 MB RAM. Furthermore, the more complex requirements the CSC establishment has the more resources the intermediate slice consumes. However, the resources of an EC are limited and the existence of many slices with CSC peers leads to the allocation of significant number of resources for the essential intermediate network slices. Furthermore, the CSC intermediary Slice instantiation from the MANO layer introduces an additional time overhead. On the contrary, the Blockchain-based CSC is minimal and only the existing connection points are routed and connected in the virtual network through the NST descriptor updates.

7. Conclusions

In this article, we introduced a Blockchain-based network service marketplace and a resource orchestration mechanism to enable cross-service communication in edge clouds. Based on the ETSI-NFV architecture, the proposed solution allows the stakeholders of the marketplace to interact trustfully in a multi-domain or multi-administrative environment. Furthermore, we introduce a novel Service Orchestrator that offers abstractions and functionality to automatically orchestrate cross-service communication for service consumption to enable off-the-shelf leasing of services. We presented this process following a step-by-step approach, from lease to service usage by the consumer. Our proposed solution brings promising results, as the essential operations to enable CSC add only an insignificant overhead to the total orchestration time. The rest of the time is consumed by the operations made by OSM and the according VIM to instantiate the CSC. In addition, compared to other proposed solutions for CSC that require intermediary slices, our work does not require extra resources to establish the cross-service communication. These metrics indicate that the proposed architecture and solution provides efficient, fully automated and seamless CSC orchestration.

Our future work will focus on extending the functionality of the network service marketplace in a completely multi-domain and multi-administrative environment. Towards this direction, the proposed Service Orchestrator should be modified and extended to support orchestration operations over WAN Infrastructure Manager (WIM) and multiple VIMs. Furthermore, we aspire to evaluate the extensions in terms of performance and scalability and deploy an experimental multi-domain EC setting with a web Dashboard.

Author Contributions: Conceptualization, K.P.-V., I.D. and D.D.; methodology, K.P.-V., I.D. and D.D.; software, K.P.-V. and I.D.; validation, K.P.-V. and I.D.; investigation, K.P.-V., I.D. and D.D.; data curation, K.P.-V. and I.D.; writing—original draft preparation, K.P.-V., I.D. and D.D.; writing—review and editing, E.E.T., I.R., S.P.; visualization, K.P.-V., I.D. and D.D.; supervision, S.P. All authors have read and agreed to the published version of the manuscript.

Funding: 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).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MDPI	Multidisciplinary Digital Publishing Institute
DOAJ	Directory of open access journals
NFV	Network Function Virtualization
ETSI	European Telecommunications Standards Institute
SDN	Software-Defined Networking
MEC	Multi-access Edge Computing
CSC	Cross-Service Communication
NSM	Network Service Marketplace
EC	Edge Cloud
OSM	Open-Source MANO
EPoP	Edge Point of Presence
VNF	Virtual Network Function
SLA	Service Level Agreement
NST	Network Slice Template
VIM	Virtual Infrastructure Manager
NSTD	Network Slice Template Descriptor
VM	Virtual Machine
WIM	WAN Infrastructure Manager

References

1. Dechouniotis, D.; Athanasopoulos, N.; Leivadeas, A.; Mitton, N.; Jungers, R.M.; Papavassiliou, S. Edge Computing Resource Allocation for Dynamic Networks: The DRUID-NET Vision and Perspective. *Sensors* **2020**, *20*, 2191. [CrossRef]
2. Avgeris, M.; Spatharakis, D.; Dechouniotis, D.; Kalatzis, N.; Roussaki, I.; Papavassiliou, S. Where there is fire there is SMOKE: A scalable edge computing framework for early fire detection. *Sensors* **2019**, *19*, 639. [CrossRef]
3. Adeel, A.; Gogate, M.; Farooq, S.; Ieracitano, C.; Dashtipour, K.; Larijani, H.; Hussain, A. A survey on the role of wireless sensor networks and IoT in disaster management. In *Geological Disaster Monitoring Based on Sensor Networks*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 57–66.
4. Herrera, J.G.; Botero, J.F. Resource allocation in NFV: A comprehensive survey. *IEEE Trans. Netw. Serv. Manag.* **2016**, *13*, 518–532. [CrossRef]
5. Feamster, N.; Rexford, J.; Zegura, E. The Road to SDN: An Intellectual History of Programmable Networks. *SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 87–98. [CrossRef]
6. Huckle, S.; White, M. Socialism and the Blockchain. *Future Internet* **2016**, *8*, 49. [CrossRef]
7. Linux Foundation. Hyperledger Fabric. 2020. Available online: https://github.com/hyperledger/fabric (accessed on 24 February 2021).
8. Papadakis-Vlachopapadopoulos, K.; Dimolitsas, I.; Dechouniotis, D.; Tsiropoulou, E.E.; Roussaki, I.; Papavassiliou, S. Blockchain-Based Slice Orchestration for Enabling Cross-Slice Communication at the Network Edge. In Proceedings of the 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C), Macau, China, 11–14 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 140–147.
9. ETSI. Open Source MANO. Available online: http://osm.etsi.org (accessed on 24 February 2021).
10. Papathanail, G.; Pentelas, A.; Fotoglou, I.; Papadimitriou, P.; Katsaros, K.V.; Theodorou, V.; Soursos, S.; Spatharakis, D.; Dimolitsas, I.; Avgeris, M.; et al. MESON: Optimized Cross-Slice Communication for Edge Computing. *IEEE Commun. Mag.* **2020**, *58*, 23–28. [CrossRef]
11. ETSI GS NFV 002. Network Functions Virtualisation (NFV); Architectural Framework. 2013. Available online: https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf (accessed on 24 February 2021).
12. ETSI. Mobile-Edge Computing (MEC); Service Scenarios. 2015. Available online: https://www.etsi.org/deliver/etsi_gr/MEC/001_099/017/01.01.01_60/gr_MEC017v010101p.pdf (accessed on 24 February 2021).
13. Dimolitsas, I.; Dechouniotis, D.; Theodorou, V.; Papadimitriou, P.; Papavassiliou, S. A Multi-Criteria Decision Making Method for Network Slice Edge Infrastructure Selection. In Proceedings of the 2020 6th IEEE Conference on Network Softwarization (NetSoft), Ghent, Belgium, 29 June–3 July 2020; pp. 1–7. [CrossRef]
14. Fotoglou, I.; Papathanail, G.; Pentelas, A.; Papadimitriou, P.; Theodorou, V.; Dechouniotis, D.; Papavassiliou, S. Towards Cross-Slice Communication for Enhanced Service Delivery at the Network Edge. In Proceedings of the 2020 6th IEEE Conference on Network Softwarization (NetSoft), Ghent, Belgium, 29 June–3 July 2020; pp. 22–28. [CrossRef]
15. Bondan, L.; Franco, M.F.; Marcuzzo, L.; Venancio, G.; Santos, R.L.; Pfitscher, R.J.; Scheid, E.J.; Stiller, B.; De Turck, F.; Duarte, E.P.; et al. FENDE: Marketplace-Based Distribution, Execution, and Life Cycle Management of VNFs. *IEEE Commun. Mag.* **2019**, *57*, 13–19. [CrossRef]
16. Maciel, P.D.; Verdi, F.L.; Valsamas, P.; Sakellariou, I.; Mamatas, L.; Petridou, S.; Papadimitriou, P.; Moura, D.; Swapna, A.I.; Pinheiro, B.; et al. A Marketplace-based Approach to Cloud Network Slice Composition Across Multiple Domains. In Proceedings of the 2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France, 24–28 June 2019; pp. 480–488. [CrossRef]

17. Guerzoni, R.; Vaishnavi, I.; Perez Caparros, D.; Galis, A.; Tusa, F.; Monti, P.; Sganbelluri, A.; Biczók, G.; Sonkoly, B.; Toka, L.; et al. Analysis of end-to-end multi-domain management and orchestration frameworks for software defined infrastructures: An architectural survey. *Trans. Emerg. Telecommun. Technol.* **2017**, *28*, e3103. [CrossRef]
18. Rathi, V.K.; Chaudhary, V.; Rajput, N.K.; Ahuja, B.; Jaiswal, A.K.; Gupta, D.; Elhoseny, M.; Hammoudeh, M. A Blockchain-Enabled Multi Domain Edge Computing Orchestrator. *IEEE Internet Things Mag.* **2020**, *3*, 30–36. [CrossRef]
19. Kempf, J.; Nayak, S.; Robert, R.; Feng, J.; Deshmukh, K.R.; Shukla, A.; Duque, A.O.; Narendra, N.; Sjöberg, J. The Nubo Virtual Services Marketplace. *arXiv* **2019**, arXiv:1909.04934.
20. Nayak, S.; Narendra, N.C.; Shukla, A.; Kempf, J. Saranyu: Using smart contracts and blockchain for cloud tenant management. In Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, 2–7 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 857–861.
21. GitHub-ConsenSys/Quorum: A Permissioned Implementation of Ethereum Supporting Data Privacy. Available online: https://github.com/ConsenSys/quorum (accessed on 24 February 2021).
22. Antevski, K.; Bernardos, C.J. Federation of 5G services using Distributed Ledger Technologies. *Internet Technol. Lett.* **2020**, *3*, e193. [CrossRef]
23. Carrozzo, G.; Siddiqui, M.S.; Betzler, A.; Bonnet, J.; Perez, G.M.; Ramos, A.; Subramanya, T. AI-driven Zero-touch Operations, Security and Trust in Multi-operator 5G Networks: A Conceptual Architecture. In Proceedings of the 2020 European Conference on Networks and Communications (EuCNC), Dubrovnik, Croatia, 15–18 June 2020; pp. 254–258. [CrossRef]
24. Nour, B.; Ksentini, A.; Herbaut, N.; Frangoudis, P.A.; Moungla, H. A blockchain-based network slice broker for 5G services. *IEEE Netw. Lett.* **2019**, *1*, 99–102. [CrossRef]
25. Rebello, G.A.F.; Alvarenga, I.D.; Sanz, I.J.; Duarte, O.C.M. BSec-NFVO: A blockchain-based security for network function virtualization orchestration. In Proceedings of the ICC 2019-2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6.
26. Rosa, R.V.; Rothenberg, C.E. Blockchain-Based Decentralized Applications for Multiple Administrative Domain Networking. *IEEE Commun. Stand. Mag.* **2018**, *2*, 29–37. [CrossRef]