Antevski, K.; Bernardos, C. J. Federation in dynamic environments: can blockchain be the solution? In: *IEEE Communications Magazine, 60(2), Feb. 2022, Pp. 32-38*

DOI: https://doi.org/10.1109/MCOM.001.2100585

Federation in dynamic environments: Can Blockchain be the solution?

Kiril Antevski, Carlos J. Bernardos
Universidad Carlos III de Madrid, Spain

***Abstract*— Deploying multi-domain network services is becoming a need for operators. However, achieving that in a real operational environment is not easy and requires the use of federation. Federation is a multi-domain concept that enables the use and orchestration of network services/resources to/from external administrative domains. In this article, we first characterize the federation concept, and involved procedures, to then dive into the challenges that emerge when federation is performed in dynamic environments. To tackle these challenges, we propose the application of Blockchain technology, identifying some associated high-level benefits. Last, we validate our proposed approach by conducting a small experimental scenario using Tendermint, an application-based Blockchain.**

I. Introduction

Nowadays, to satisfy both vertical customers and end-users, service providers need to dimension and manage their infrastructure –composed of diverse computing, storage, and networking devices– to fit the expected demand. In case of an infrastructure failure (e.g., burned data center) or a sudden increase of the number of users for a given service (e.g., concerts), service providers are often not capable to react on time to maintain a service operational.

In these situations, thanks to virtualization, it is less costly and more time-efficient to enable an operator to use services and/or resources owned by other operators/providers than to expand the local infrastructure. Besides environmental benefits, a lessor benefits by lowering the operational cost, and a lessee satisfies users just-in-time without suffering any penalties. This process defined as *federation*, enables operators to orchestrate services/resources from external administrative domains. Federation has been already explored for Network Function Virtualization (NFV) based platforms [1]–[5], paying special attention to how to extend the Management and Orchestration (MANO) stack to enable federation of services and resources. While minor differences can be found in the definition of federation, a common assumption is a presence of a pre-established agreement among service providers. However, settling an agreement is time-consuming and suitable only for static environment.

The goal of this work is to explore how Blockchain technology can complement NFV MANO service providers to accomplish federation in dynamic scenarios – such as the on-demand deployment of virtual access points to expand remote control of Edge robots [6]. Although challenging [7], in this article we propose a solution validated through a small-scale experiment that *federates* and *heals* a simple virtualized service.

As our initial contribution, we provide a first deep dive into the federation concept in Section II, enumerating different relevant characteristics and the associated possible options that can be considered. We then describe the steps involved in a generic federation process, before identifying the challenges that arose when federation targets a dynamic environment. This sets the background required to analyze, in Section III, how Blockchain technologies can be used to improve federation of services and/or resources among different domains. We validate our proposal through an experimental evaluation in Section IV. Finally, we conclude our work in work in Section V, where we also point out some future research directions.

II. Federation: characterization, involved procedures and challenges in dynamic environments

We can define network federation as the feature of orchestrating services or resources across several domains in a multi-domain scenario. Federation is an enabler for operators to satisfy their customers' demands by being able to orchestrate and use services/resources from external administrative domains, effectively extending their footprint.

We use the term "consumer domain" to refer to the administrative domain that consumes federated services or resources, and "provider domain" to refer to the administrative domain that provides federated services or resources. Both domains have incentives to federate: the consumer domain to satisfy the vertical requirements, and the provider one to generate an extra income from unused resources.

A. Federation characterization

Federation is indeed a broad concept, and as such, has been already tackled in existing works [1]–[6], [8], [9], which consider different aspects that we represent in Table I. By doing so, we try to spell out what might be involved in a federation process and what variables could be in place. The first column of Table I presents the main federation characteristics as found in relevant previous works, along with possible options for each characteristic in the second column (briefly described in the third column).

As mentioned earlier, in a federation scenario, an administrative domain can be either a consumer or a provider. Depending on what is being federated, we refer to service or resource federation. In service federation [1], [3], [5], the provider domain deploys the service and provides the required connectivity for the consumer to use (i.e., *consume*) the federated service. Upon success, the provider domain is used as a proxy for the consumer domain to perform life-cycle management operations over the federated service. In

TABLE I

FEDERATION CHARACTERISTICS

Federation	Options	Description
Domain Roles	Consumer	Domain consuming federation
	Provider	Domain providing federation
What is federated?	Services	Service extension or new service (resource agnostic);
	Resources	Specific amount of virtualized resources;
Environment dynamicity	Pre-established & Static	Business agreements with known members (e.g., SLAs); Advertisement/discovery of federation capabilities;
	Open federation & Dynamic	Rapidly changing and unknown members; Requires higher security degree; Announcement/negotiation for federation;
Interconnection framework	Decentralized peering	Peer-to-Peer connection with each agreed administrative domain. Individual connections contain independent rules of interaction.
	Centralized	Single central entity manages the interaction and applies rules for all involved administrative domains
	Decentralized distributed	Peer-to-peer framework using consensus protocol for applying rules for all connections, maintain trust, security, or node failure.
Layer communication	Single	Orchestrator-to-orchestrator (same entities)
	Cross-layer	Cross-layer interaction;
Federation deployment	One-to-one	Requesting federated service or resource from a single domain
	One-to-many	Requesting federated services or resources from multiple domains; Simultaneous deploy & chaining

resource federation [4], the provider domain leases the control and management of the federated (virtualized) resources.

A key aspect of federation, being the core of what is later analyzed in this article, is how dynamic the environment is. There are two main scenarios that can be considered: (*i*) pre-established & static, and (*ii*) open & dynamic. We refer to a pre-established & static federation environment when administrative domains meticulously define federation relationships in advance, through business meetings, signing contracts and service-level agreements (SLAs). This process is time-consuming, long-lasting and precise, and defines in detail the rules and policies of the federation relationship. In a dynamic environment, there is a yearning for the contrary. Involved actors should allow for quick and brief federation relationships, and this can only be accomplished through open frameworks where administrative domains may join or leave at any time. This quite naturally leads to auction-based models for federation of resources and services. An open framework has to offer secure and trusty processes to enable domains to reach short-term agreements shortly before a service or resource is deployed, in a form of *dynamic SLAs* [10].

Also related to the dynamicity of the environment, another key federation characteristic is the nature of the interconnection between the involved domains. As shown in Table I, we can differentiate among: (*i*) centralized, (*ii*) decentralized-peering, and (*iii*) decentralized-distributed options. For static federation, either centralized

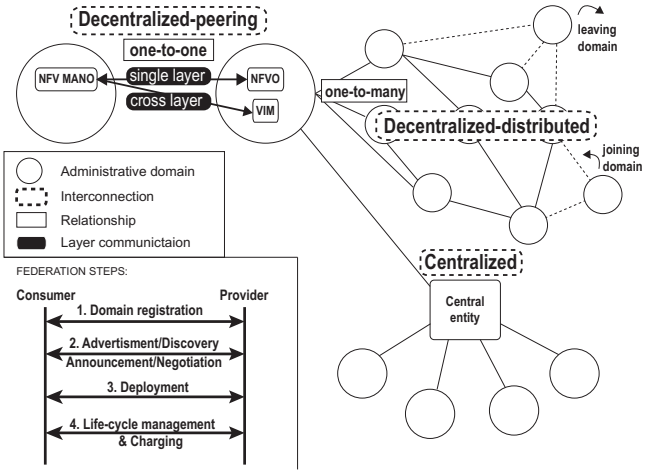

Fig. 1. Federation classification and steps

or decentralized-peering approaches are considered as most appropriate. In a decentralized-peering scenario, domains establish independent peer-to-peer connections to every administrative domain with an existing agreement. The maintained policies linearly increase with every new federation agreement, which would make this hard to handle and scale in dynamic environments. A centralized framework provides a single central entity to oversee and manage the federation among all involved administrative domains. A hybrid option is a decentralized-distributed framework, that relies on a consensus protocol to apply common rules and policies of interaction in a peer-to-peer network. Any domain joining or leaving does not alter the ratified behavior. Fig. 1 illustrates the different federation options and federation steps. In Section III, we describe how Blockchain technology can be applied as a decentralized-distributed network solution.

Regardless of the interconnection framework in use, administrative domains are composed of different layers (e.g., of different resources) and therefore communications may happen at different layers through different interfaces [5]. Single-layer federation connections are established between equivalent entities (e.g., orchestrator-to-orchestrator, shown on Fig. 1), laying on the same architectural layer using east/west interfaces. Hierarchical federation connections are cross-layer connections established between entities on different layers using different interfaces –northbound or southbound.

Last, but not least, we can also characterize how many domains are involved in a given federation instance. This is what we refer to as ”federation deployment”, and we consider two possible cases: one-to-one and one-to-many. A one-to-one federation deployment occurs when a consumer domain generates a federation request for a single provider domain to provide the required service or resource. In [8], this is referred to as the Resource Manager Role. In a one-to-many federation deployment, the consumer domain simultaneously requests several services to be provided by multiple provider domains. The consumer domain later orchestrates complex chaining of the federated services and/or resources

from the diverse provider domains [1]. Similarly, in [8], this is referred to as Aggregator or Hub role.

B. Procedures involved in federation

Although the ETSI NFV has defined the different orchestration and lifecycle management procedures (e.g., instantiation, termination, scaling, etc.), and has even explored different architecture options to support the collaboration among multiple federation domains, there is not yet a clear standardized set of procedures defined. It is critical to preserve interoperability between domains, thus minimizing the extent to which existing mechanisms (e.g., ETSI NFV MANO procedures) would need to be modified. We next summarize the high-level sequential steps that are involved in a generic service/resource federation process [4], [5]:

1) **Domain registration.** This is required regardless of the dynamicity of the environment. In a static federation, administrative domains establish peer-to-peer individual connections to every contracted administrative domain. In an open federation, the registration, although open to new administrative domains, might require a voting consensus to allow transparent decisions (of new members acceptance, governance, etc.). Security mechanisms and integrity checks are fundamental.
2) a) **Advertisement/Discovery.** In static federation environments, depending on the pre-established agreement, peering administrative domains periodically exchange or *advertise* information on available services/resources. For a large number of connections, an administrative domain uses polling or *discovery* instead of advertising. Based on the exchanged/shared information, a consumer domain is capable to generate a global view of available services or resources for federation, helping it to take a better federation decision.
   b) **Announcement/Negotiation.** In open and dynamic federation environments, domains renegotiate federation terms repeatedly. Different negotiation techniques are available: *bilateral*, *match-matching* or *autonomous* [11]. Match-matching and autonomous are more suitable for a centralized entity. A consumer requests federation by specifying a range of terms. The central entity matches potential provider domains that strictly match terms - *match-matching*, or by close-to-full fulfillment - *autonomous*. Upon a matched domain, both provider and consumer domains receive the connectivity details. In the *bilateral* case, more suitable for a decentralized-distributed interconnection scheme, the consumer domain broadcasts an *announcement* request for federation. Potential provider domains engage in a reverse-auction fashion by replying with bidding offers. The final decision is made by the consumer domain using internal policy criteria. The selected winning provider domain proceeds into fulfillment of the federation request.
3) **Deployment.** In this step, the ”winning” provider proceeds with the deployment of a federated service or resource. Upon successful deployment, both, the consumer domain and the provider domain, establish data plane connectivity and inclusion of the federated service/resource for the intended purpose. In an NFV MANO environment, these steps are repeated for every service extending/scaling/healing operation [3]. Finally, at the end of this stage, federated resources/services commence running as an integral part of the consumer domain.
4) **Life-cycle management & charging.** Once the federated resource/service is embedded and running, the consumer domain manages its life-cycle using the provider domain orchestrator as a proxy in a single layer communication scheme. In a hierarchical communication scheme, the control plane goes through the north/southbound interfaces (e.g., cross-layer on Fig. 1). Both domains monitor the federated usage and calculate the fee according to the established agreement. Note that the provider domain has a ”kill switch” or the ability to terminate the federation at any point in time [4]. As opposed to static environments, establishing a monitoring and charging process can be quite challenging in dynamic environments.

C. Federation challenges in a dynamic environment

Next, we summarize the main challenges posed by multi-domain federation [7], [12] in dynamic environments (Table II), identifying how these challenges are tackled depending on the interconnection approach. We later elaborate (Section III) and propose how Blockchain can be the basis of a solution to all these challenges, but we first focus on how this is done for the centralized and decentralized-peering types of solutions:

- **Admission Control.** Administrative domains in an open federation are free to join or leave at any time. If a centralized interconnection is adopted, the central entity oversees the admission control, i.e., which domain is allowed to leave or join the network. If a decentralized-peering interconnection is used, the access can be completely open depending on each individual administrative domain. In both cases, the main challenge is to balance the trade-off between domain openness and preserving privacy, security, and trust. Highly secured frameworks and message exchanges may introduce higher delays or congest the federation interaction. To the contrary, an absolutely open admission may expose administrative domains to passive spoofing.
- **Availability.** The number of participating administrative domains changes over time in a dynamic environment. With centralized approaches, it is easier to monitor who is participating, though there might be inconsistencies if there are sporadic failures, due to the single point of failure nature of a centralized approach, which might lead to the federation becoming unavailable. Decentralized-peering solutions are inherently more re-

TABLE II
FEDERATION CHALLENGES AND HOW THEY ARE TACKLED THROUGH DIFFERENT INTERCONNECTION REALIZATIONS

	Interconnections		
Challenges	Centralized	Decentralized peering	Blockchain
Admission control	Central;	Open;	Distributed; Consensus voting;
Availability	High; Single point of failure;	Unknown; Fail-safe;	Balanced; Fail-safe; Incentive to participate;
Dynamic pricing & billing	Central auctioneer; Single-point control;	Autonomous; No control; No billing;	Autonomous by default; Token based billing
Multi-domain QoS	Central control & monitoring (dynamic SLAs);	None;	Smart contracts as dynamic SLAs; Off-chain oracles;
Security & Privacy	High security; Low privacy;	Low security; High privacy;	High security; High privacy;

silient to failures, but tracking administrative domains is more challenging and may introduce spoofing risks.

- **Dynamic pricing & billing.** Administrative domains have the incentive to increase the profit by adapting the federation price offerings, especially in a dynamic environment. A central entity, as an auctioneer, can change federation offerings and track the billing process. However, participating domains should voluntarily trust this federation control and pay for it. In the decentralized-peering scenario, the administrative domains autonomously set the price offerings. The difficult part is to quickly arrange secure agreements in the form of *dynamic SLAs*, that would establish a baseline for assuring billing. Additionally, employing mechanisms to identify other domains and securely implement a charging process is costly.
- **Multi-domain Quality of Service (QoS).** Guaranteeing the quality of service across federating domains is quite challenging in dynamic environments. In both decentralized-peering and decentralized-distributed, the major challenges are establishing dynamic SLAs and guaranteeing unbiased monitoring data [7]. Domains often disagree on the monitored data avoiding the responsibility in case of a low QoS or SLA breach. Often, for avoiding legal disputes, third-party entities are monitoring the SLAs. While in the centralized option, a centralized entity is responsible for establishing QoS across every domain, thus controlling and monitoring the whole process.
- **Security & privacy.** With a centralized interconnection schema, the administrative domains rely on the central entity. To increase security, the central entity demands more information from every administrative domain, which comes at cost of a lower privacy per domain. On the contrary, a decentralized-peering solution may achieve higher privacy (exchanging less information with peering domains), at the cost of lower security policies employed.

III. APPLYING BLOCKCHAIN TO FEDERATION

Blockchain can help to overcome most of the challenges posed by federation in dynamic environments (enumerated in Section II-C). Actually, ETSI has formed an Industry Specification Group (ISG) for Permissioned Distributed Ledgers (PDL) which lays the foundations for the application of Blockchain in globally open telecommunication networks [13], [14]. We next provide a short insight into Blockchain technology and then evaluate how Blockchain technology can be helpful for federation – considering the work done by the ETSI PDL.

A. Blockchain: background and benefits

The *Blockchain* technology emerged as a key underlying mechanism for Bitcoin, providing a distributed, secure, and time-stamped ledger that records every transaction between anonymous users. A Blockchain is built of interconnected nodes that share a single ledger. The ledger contains distributed time-stamped blocks filled with transactions that can contain any data. Each block points to the hash of the prior block, generating a chain (or history) of blocks, back until the genesis block (*block 0*). New blocks are generated and validated by the nodes (e.g., any computing device) that are interconnected in a peer-to-peer Blockchain network. There are two types of Blockchain networks: permissionless and permissioned. Each new block needs to be validated by a consensus mechanism which is a key performance component. There are different consensus mechanisms that provide different levels of security, trust, and privacy. Summarizing, the main benefits of applying Blockchain are (more information regarding the use of Blockchain in networking can be found in [13], [14]):

- **Security.** The transaction data included in each block of the Blockchain is timestamped, tamper-proof and immutable. Data alteration is only feasible if at least 51% of the nodes are malicious/compromised.
- **Verifiability, integrity, and trust.** The state of the Blockchain is easily verifiable by all the members confirming an equivalent observed Blockchain state.
- **Smart Contracts.** Programmable applications that run as independent entities (or members) on top of a Blockchain (e.g., Ethereum). These applications have deterministic and atomic functions that can embed business logic and rules as in regular contract agreements.
- **Balanced privacy and transparency.** All transactions, state transitions, and blocks creations are transparent. Using cryptography enables private data to be encrypted and exclusive while maintaining the defined transition rules.
- **Third party absence.** The consensus mechanism enables collaboration among unknown members in a trusty manner without a third-party authority (e.g., central entity) to guarantee the integrity of the members.

The main drawbacks are (*i*) the immaturity of the technology; (*ii*) scalability and energy efficiency - the energy spent per transaction increases linearly with the network size.

B. Blockchain for dynamic and open federation

We envision the application of Blockchain to support federation in dynamic environments, in a way complementary to the existing architectural approaches (e.g., NFV MANO frameworks). Implementation on a public permissionless Blockchain can be costly due to costly transactions, whereas the cost for implementation and maintenance of a permissioned Blockchain is low [7].

We propose the design illustrated in Fig. 2, to apply Blockchain to open federation. Every administrative domain should deploy a Blockchain node connected to the East/Westbound interface of an NFV Orchestrator. We argue that maintaining this decoupling enables the independent evolution of both NFV and Blockchain technologies.

1) How can Blockchain solve the dynamic federation challenges: Just by deploying a non-customized (vanilla) version of a public permissioned Blockchain network (e.g., Ethereum, Hyperledger, Cosmos, Polkadot, and etc.), most of the challenges enumerated in Table II can be addressed.
Admission control is dependent on the Blockchain governance policy [13]. In a permissioned Blockchain, a common approach is to accept members via voting. Although domains may act maliciously and reject the entry of new members, domains typically have the incentive to increase the participants. If it is not the case, different Blockchain instances may operate in parallel.
Availability is guaranteed by the incentive of each domain to maintain an active Blockchain node. Therefore, this improves the Blockchain network security (avoiding 51% attacks), and (*ii*) increases the domain's *usage budget* (e.g., gas in Ethereum). In short, the 51% attack happens when a malicious user controls 51% of a Blockchain network, thus can modify all the transactions in every block. In case that a node fails, leaves or is compromised, the Blockchain network remains active and operational as well as the domain has access via other nodes by using its unique Blockchain address.
Security and privacy are established by limiting the *usage budget* and the use of cryptography. Newly joined domains have a lower limited *usage budget* or a limited number of federation announcements being unable to spoof or spam the participating domains. Communications between domains are recorded and validated as immutable transactions on the ledger. Cryptography is used to preserve the privacy of the data in the transactions exchanged [13].
Dynamic pricing and billing, and Multi-domain QoS require implementation of dynamic SLAs and QoS monitoring. The use of Smart contracts is a promising solution towards the integration of both dynamic SLAs and QoS monitoring. Smart contracts are deterministic and independent applications that reside on the Blockchain ledger. The ETSI PDL specification [14] provides a hint of how to employ QoS through an example scenario of using Smart contracts. It envisions a marketplace of SLAs where each Smart contract presents a specific service offering with QoS metrics. Customers, ready to deploy a service from the marketplace, need to send a payment Blockchain transaction to the specific Smart contract. A third-party entity is used (as an *oracle*) to monitor the QoS metrics and record the SLA fulfillment directly in the Smart contract. In the case that QoS is not satisfied, the Smart contract automatically sends back a Blockchain payment transaction to the customer Blockchain address with the penalty amount. Additionally, service providers as Smart contract owners can dynamically change the prices in every Smart contract, of course, prior to the customers making the deposit transaction. Similar ideas have been tackled in [10], [15].

2) Our Blockchain solution for federation: The adaptation of the described PDL concept in a federation scenario implicates a new Smart contract creation for every new federation of services or resources. These Smart contracts represent dynamic federation SLAs that guarantee the QoS between the consumer and provider domains, but this approach presents some drawbacks. There is an added Smart contract deployment latency [14]. This added delay is due to the writing operation on the ledger. Reading operations on a Blockchain ledger are immediate, but the writing alters the ledger state. The speed of writing mainly depends on the consensus mechanism (e.g., Proof-of-Work, Proof-of-State, Byzantine Fault Tolerant, etc.), the network size, number of newly issued transactions, etc. Furthermore, the use of a third-party entity (oracle) for monitoring QoS metrics denatures the distributed concept, transforming it into a hybrid version of a centralized solution.

Our proposed design (Fig. 2) can be realized, (*i*) with a single Smart contract as an auctioneer; or (*ii*) without a Smart contract, on an application-based Blockchain.

The use of a single Smart contract, in our vision, defines a neutral set of rules that reflect the federation steps from Section II-B in the role of an auctioneer. The use of a reverse-auction model enables consumer domains to have customized federation announcements and provider domains diverse bids. In [6], we showcased the use of a single Smart contract for federation in a dynamic environment. Compared to the multiple Smart contracts case, the main difference is that the deployment delay is omitted. The Smart contract can record all the domains' interactions on a single Blockchain address. These records are used as proof that all procedures have been performed correctly and to enable billing.

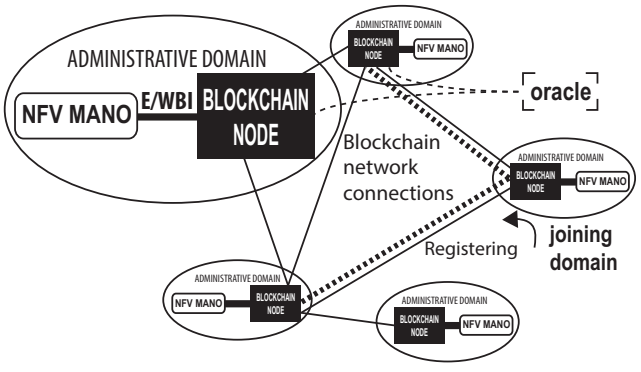

Fig. 2. Application of Blockchain to open federation

C. Application-based Blockchain for federation

Extensive use of Smart contracts may exponentially increase the ledger storage, due to the number of writing operations, even if it is a single Smart contract. As a consequence, new joining domains may encounter significant delays in syncing the ledger of the Blockchain, known as a *scaling issue*. New application-based Blockchains emerged to diminish the *scaling issue*. In comparison to general-purpose Blockchains, containing multiple Smart contracts (e.g., Ethereum), application-based Blockchains propose a single application per Blockchain, or vice versa. Examples of these are Tendermint/Cosmos SDK, Polkadot, etc.

In this work, we have explored the use of Tendermint[1], which is based on the Byzantine-Fault Tolerant (BFT) consensus mechanism. Block generation is achieved through voting, and there exist three types of voting: prevote, precommit, and commit. A block is added once it receives $2/3$ commit votes. Tendermint uses Application BlockChain Interface (ABCI) to reflect and broadcast changes of the specific application state as transactions on the Blockchain. In our case, changes in the federation application (e.g., federation announcements) are broadcast through the ABCI - E/WBI. The performance is not dependent on the processing power of the nodes (as in Proof-of-Work consensus) and the ledger storage is dedicated for a single application.

We next describe how federation steps (as described in Section II-B) can be implemented for a Tendermint federation application. First, a **registration** (step 1) is completed when the new administrative domains are admitted through voting of the already participating administrative domains, referred to as validators. Each newly joined administrative domain is voted by the validators to become a new validator node as well. More validators in the network translate in increasing the network security and improving the consensus mechanism. In a default BFT consensus, 2/3 of the validators need to approve the addition of a new block on the Blockchain or a new validator.

Once registered, an administrative domain may want to participate in an NFV service federation as a consumer domain. This domain creates a service federation **announcement** (step 2) including the service requirements, and it is broadcast as an event transaction. At this point, the reverse auction process starts. The consumer domain collects bidding offer transactions from potential providers. These transactions contain details that answer most of the service requirements. The consumer domain, based on an internal policy, elects a winning provider domain and sends a deployment transaction, while the rest are notified of the **negotiation** closure.

The provider domain proceeds with service **deployment** (step 3). Meanwhile, they exchange transactions on the interconnection details. Once the service has been deployed and the connections are established, the federated service is integrated and **life-cycle managed** (step 4) by the consumer domain. Both domains monitor different performance metrics. The QoS metrics are periodically recorded with transactions. These records are used as timestamps for **billing** information.

[1]https://tendermint.com/

IV. Experimental validation

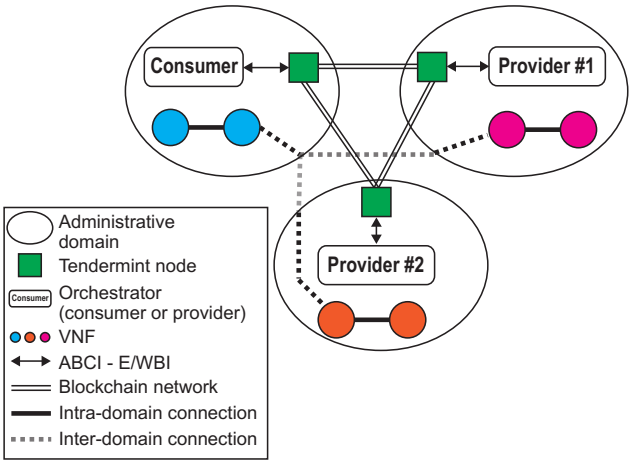

Fig. 3. Experimental setup

To validate the application of Blockchain in a federation scenario, we produced an experimental setup of three independent administrative domains as validators in a Tendermint network as shown in Fig. 3. Each domain is deployed as a mininet VM, an Ubuntu 14.04 virtual machine with 2 CPU cores, 2 GB of RAM, and 5 GB of disk memory.

The experimental testing is divided into two phases: federation and healing. First, we simulate an extension of service through federation, which requires two hosts from a consumer domain to have continuous communication (no packet loss) with two hosts from a provider domain. In the second phase, the federated service fails, and it is healed by performing a new federation.

Fig. 4 shows the event plot of 30 averaged experiments (also plotting the variance) for the three domains: the consumer, provider #1, and provider #2.

At the start, the consumer domain announces the service (1). Both provider domains (2) receive the announcement transaction and generate a bid-offer (3) containing all the service details and prices. The consumer domain receives the offers and elects a winning (4) provider domain. In the first phase that is provider #1. Both provider domains (5) receive the elected winner. It takes less than 7 seconds to finalize the announcement and negotiation. Then, the winning provider #1 is (6) ready to deploy the federated service. The consumer domain (7) sends the connection details transaction while the service is being deployed. Upon (8) service deployment, the interconnection between both domains is established using VxLAN and the federated service is up and running after less than 5 seconds. The consumer domain starts to continuously monitor the connection for the zero packet loss requirement.

In our experiments, the service is set to fail after 10 seconds, and the healing phase starts. The consumer domain issues a new federation procedure after a loss of two consecutive packets. Provider #1 is blacklisted as an unreliable

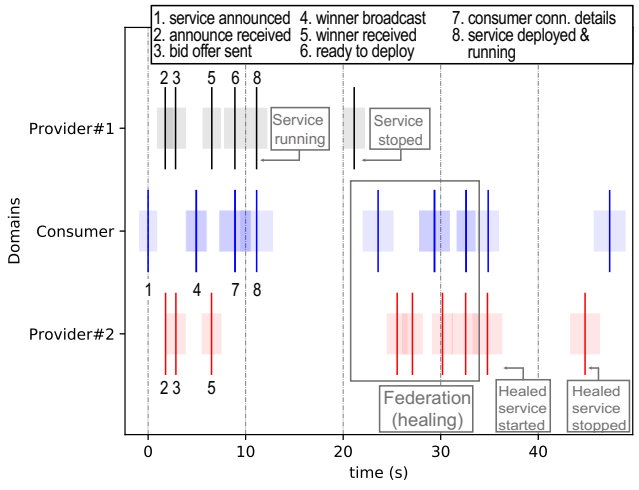

Fig. 4. Federation + healing procedures

domain, so provider #2 wins and deploys the newly healed federated service. It takes around the same time ($\sim$ 11 seconds) to federate or heal a service. As shown the Fig 4, the occurrence of the events vary with a maximum variance of 4 seconds.

V. Conclusion and Future Work

In this article, we have characterized federation, have described the basic steps it involves and evaluated the challenges it poses in dynamic scenarios, assuming NFV-based environments. We argue that the application of Blockchain can successfully tackle most of the challenges found, without impacting current widely-used NFV frameworks.

We have proposed a high-level design of a federation application-based Blockchain solution, which we have validated through an initial set of experiments using a Tendermint network of nodes from three administrative domains. Obtained results, though from a sample scenario, allow us to concluded that the time required to federate or heal an NFV service is in the order of tens of seconds, proving the feasibility of this promising technology. As future work, we plan to conduct a comparative performance study of Blockchains with different consensus protocols, and a big-scale NFV-MANO experiment, as in [3].

Acknowledgments

This work has been partially supported by EC H2020 5GPPP 5Growth project (Grant 856709).

References

[1] C. J. Bernardos et al., "5gex: realising a europe-wide multi-domain framework for software-defined infrastructures," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 9, pp. 1271–1280, 2016.

[2] A. Oliva et al., "5G-TRANSFORMER: Slicing and Orchestrating Transport Networks for Industry Verticals," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 78 – 84, 2018.

[3] J. Baranda et al., "Realizing the Network Service Federation Vision: Enabling Automated Multidomain Orchestration of Network Services," *IEEE Vehicular Technology Magazine*, vol. 15, no. 2, pp. 48–57, 2020.

[4] 5G-CORAL et al., "Deliverable 3.2 - refined design of 5g-coral," http://5g-coral.eu/wp-content/uploads/2019/06/D3.2.pdf, (Accessed on 02/05/2021).

[5] A. Francescon et al., "X–mano: Cross–domain management and orchestration of network services," in *2017 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2017, pp. 1–5.

[6] K. Antevski et al., "Dlt federation for edge robotics," in *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2020, pp. 71–76.

[7] R.B. Uriarte et al., "Blockchain-based decentralized cloud/fog solutions: Challenges, opportunities, and standards," *IEEE Communications Standards Magazine*, vol. 2, no. 3, pp. 22–28, 2018.

[8] GSMA, "Operator platform concept whitepaper," 02 2020, (Accessed on 02/22/2021). [Online]. Available: https://www.gsma.com/futurenetworks/wp-content/uploads/2020/02/GSMA_FutureNetworksProgramme_OperatorPlatformConcept_Whitepaper.pdf

[9] A. Boubendir et al., "Federation of cross-domain edge resources: a brokering architecture for network slicing," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 2018, pp. 415–423.

[10] R.B. Uriarte et al., "Distributed service-level agreement management with smart contracts and blockchain," *Concurrency and Computation: Practice and Experience*, p. e5800, 2020.

[11] V. Scoca et al., "Smart contract negotiation in cloud computing," in *2017 IEEE 10Th international conference on cloud computing (CLOUD)*. IEEE, 2017, pp. 592–599.

[12] K. Katsalis et al., "Multi-domain orchestration for nfv: Challenges and research directions," in *2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS)*. IEEE, 2016, pp. 189–195.

[13] ETSI, "ETSI ISG PDL 003 V1.1.1, Permissioned Distributed Ledger (PDL); Application Scenarios," December 2020.

[14] ——, "ETSI ISG PDL 004 V1.1.1, Permissioned Distributed Ledgers (PDL) Smart Contracts System Architecture and Functional Specification," February 2021.

[15] B. Nour et al., "A blockchain-based network slice broker for 5g services," *IEEE Networking Letters*, vol. 1, no. 3, pp. 99–102, 2019.

Biographies

Kiril Antevski is a PhD student at University Carlos III Madrid (UC3M), Spain.

Carlos J. Bernardos received his PhD from UC3M (Spain), where he works as associate professor.