DIaC: Re-Imagining Decentralized Infrastructure as Code Using Blockchain

Rabimba Karanjai, Keshav Kasichainula, Lei Xu, Nour Diallo, Lin Chen, and Weidong Shi, *Senior Member, IEEE*

(Invited Paper)

***Abstract*—With the recent advances in concepts like decentralized "cloud" and blockchain-enabled decentralized computing environments, the legacy modeling and orchestration tools developed to support centrally managed cloud-based ICT infrastructures are challenged by such a new paradigm built on top of decentralization. On the other hand, decentralized "cloud" and computing infrastructures need to support many Dapp use cases. As the complexity of these targeted application scenarios increases, there is an urgent need for developing automation and modeling tools for deploying and managing decentralized infrastructures. Instead of creating such tools from scratch, a natural approach is extending mature infrastructure modeling tools for Dapps and decentralized computing environments. To this end, in this work, we have developed extensions to the TOSCA domain-specific language to support smart contract specification of decentralized computing infrastructures for supporting Dapps, where smart contracts or chain codes manage a decentralized computing environment. The result is blockchain-based orchestration and automation for decentralized "cloud" and computing environments that use existing infrastructure as code tools to deploy and manage decentralized applications.**

***Index Terms*—TOSCA, smart contracts, blockchain, infrastructures.**

I. Introduction

MODERN ICT (information and communication technology) infrastructures are becoming increasingly sophisticated and infeasible to manage manually. Various technologies have been developed to support efficient and flexible physical ICT infrastructure management, such as computer virtualization, software-defined networks, and network function virtualization. To further automate the deployment, configuration, and management of centralized infrastructures like a cloud-based ICT center, system administrators have been using orchestration tools [1] to dynamically deploy services. These orchestration tools are usually centrally managed by so-called orchestrators. An administrator fully trusts the orchestrator to manage and coordinate the lifecycles of ICT components (e.g., computation, storage, and communication resources), which constitute a deployed service. To further improve the management of the ICT infrastructure, the concept of "infrastructure-as-code" was developed [2], where a system administrator describes the target infrastructure and orchestration objectives as documents using a domain-specific language. This concept brings several advantages, such as faster configuration and environment provisioning, greater consistency, and minimized risk/enhanced security.

One of the most popular domain-specific languages for infrastructure definition is TOSCA (Topology and Orchestration Specification for Cloud Applications [3]), which helps manage and automate the cloud-based deployment of services. With TOSCA, administrators can focus on the intended functionalities of the infrastructure and only need to describe the end goal of the envisioned service deployment environment. The interpretation and implementation of the description are delegated to the orchestrator. One of TOSCA's fundamental properties is cloud platform agnostic, which makes it ideal for managing multi-cloud ICT infrastructure. There are numerous implementations of TOSCA based on its standards (e.g., Alien4Cloud (https://alien4cloud.github.io/), Cloudify (https://github.com/cloudify-cosmo), and OpenTOSCA [4]).

With the introduction of blockchains [5], smart contracts [6], and the recent advance in decentralized "cloud" and blockchain-enabled decentralized computing environments, the legacy modeling, and orchestration tools are challenged by this new emerging decentralized computing paradigm, where computing infrastructures consist of fully decentralized components. The creation and maintenance of such infrastructures are often facilitated by the principles of crypto-finance or crypto-economy, which provides various incentive mechanisms for sustaining a decentralized ICT infrastructure. The objectives of decentralization are to eliminate centrally managed components and reduce reliance on trusted entities. These are not aligned with the original design goals of orchestration tools like TOSCA. On the other hand, decentralized "cloud" and computing infrastructures need to support a wide range of Dapps (decentralized applications) use cases. As the complexity of these targeted application scenarios increases, there is an urgent need for developing automation and modeling tools for specifying, deploying, and managing decentralized infrastructures.

Manuscript received 20 May 2023; revised 2 September 2023; accepted 26 September 2023. Date of publication 18 October 2023; date of current version 15 April 2024. The associate editor coordinating the review of this article and approving it for publication was A. Veneris. *(Corresponding author: Rabimba Karanjai.)*

Rabimba Karanjai, Keshav Kasichainula, Nour Diallo, and Weidong Shi are with the Department of Computer Science, University of Houston, Houston, TX 77204 USA (e-mail: rkaranjai@uh.edu; kkasichainula@uh.edu; ndiallo@uh.edu; wshi3@uh.edu).

Lei Xu is with the Department of Computer Science, Kent State University, Kent, OH 44240 USA (e-mail: xuleimath@gmail.com).

Lin Chen is with the Department of Computer Science, Texas Tech University, Lubbock, TX 79409 USA (e-mail: lin.chen@ttu.edu).

Digital Object Identifier 10.1109/TNSM.2023.3325768

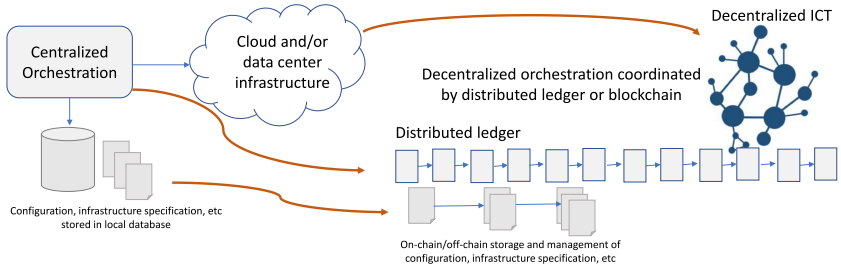

Fig. 1. From centralized to decentralized: both management and infrastructures.

To overcome these limitations and keep all the benefits of infrastructure as code, we propose a Decentralized Application Infrastructure (DIaC), which facilitates the automated management of decentralized computing environments and Dapps using blockchain technology. DIaC also relies on a specific language to describe ways to manage the decentralized resources, which are encoded and processed in the form of smart contracts. Instead of creating such tools from scratch, DIaC extends existing mature infrastructure modeling tools like TOSCA for Dapps and decentralized computing environments. Another benefit of extending TOSCA is that it potentially can support mixed ICT infrastructures that comprise both cloud and decentralized computing resources. The efforts bring a missing piece of the puzzle for realizing fully decentralized computing infrastructures.

In summary, the contributions of this work include: (i) We have developed a detailed design of DIaC, which enables the provisioning and management of decentralized ICT infrastructures with a unified, general-purpose, comprehensive, and standard-based infrastructure modeling tool; (ii) The new tool supports modeling, orchestration, and management of decentralized computing environments by leveraging blockchain technology and smart contracts; (iii) We have provided examples to demonstrate the usefulness of DIaC; and (iv) We have implemented a prototype of DIaC and conducted preliminary experiments on it to demonstrate its practicality.

II. Background

We briefly review background technologies in this section.

Decentralized "cloud" The concept behind the decentralized "cloud" is to enable a new paradigm of computing infrastructures based on decentralization and blockchain-facilitated economic models. Different from the traditional cloud computing model that depends on very few trusted service providers, decentralized "cloud" leverages peer-to-peer based nodes for computing (e.g., [7], [8], [9], [10], [11], [12]) These systems often apply blockchain or distributed ledger-based designs to provide incentives to the users who contribute resources, for instance, using utility tokens as payment for computing resources. The proposed platforms can target either general-purpose computing, for instance, Dfinity [12], or specific applications like databases, artificial intelligence, or machine learning-oriented use cases (e.g., [13]). To address privacy concerns, some works proposed to use of trusted execution environments (TEEs) [14] for processing data.

Infrastructure modeling and TOSCA: For deploying, configuring, and managing services and software in cloud-based environments, system administrators apply automation tools for provisioning server nodes and instantiating software on them. Typically, this is achieved by using domain-specific language to describe the targeted infrastructures as templates and their orchestration intents in policy files. There are several such languages backed by the industry and cloud service providers [15], [16]. TOSCA is one of the standard-based modeling languages for handling cloud-based infrastructure deployments and orchestration [17]. It was developed by OASIS (Organization for the Advancement of Structured Information Standards) as a cross-cloud standard. Many cloud modeling languages are compatible with TOSCA.

Smart contracts: A smart contract, also called a "chain code" in the context of Hyperleger Fabric, is software that can function as a general-purpose application to interact with blockchain data. Smart contracts implement the business logic of a blockchain application. Smart contracts are executed by blockchain nodes and verified using consensus. Many smart contract languages have been developed in the past few years. In the case of Fabric, it provides an open-source SDK that supports chain code written using popular programming languages such as GoLang, Nodejs, and Java.

With the rise of Dapps supported by decentralized ICT infrastructures using distributed ledgers or blockchains, governance and managing such environments have become increasingly complex. In a smart contract-based Dapp platform, smart contracts can be applied for managing Dapps and decentralized computing infrastructures. In this environment, multiple Dapps can share the same blockchain for infrastructure orchestration. Different from data center and cloud-based computing environments, there is a significant lack of understanding, knowledge, standards, and tools for automating governance, proliferation, and management of decentralized computing environments. Figure 1 highlights the differences between centralized ICT and decentralized computing environments.

III. Overview of DIaC

One approach to automating the management and governance of decentralized infrastructures and Dapps is to let the technology play itself out, perhaps mirroring the growth of blockchains in the last fourteen years. Here we argue that a better alternative is to leverage what we have learned, built, and developed for centralized ICT and extend the existing tools

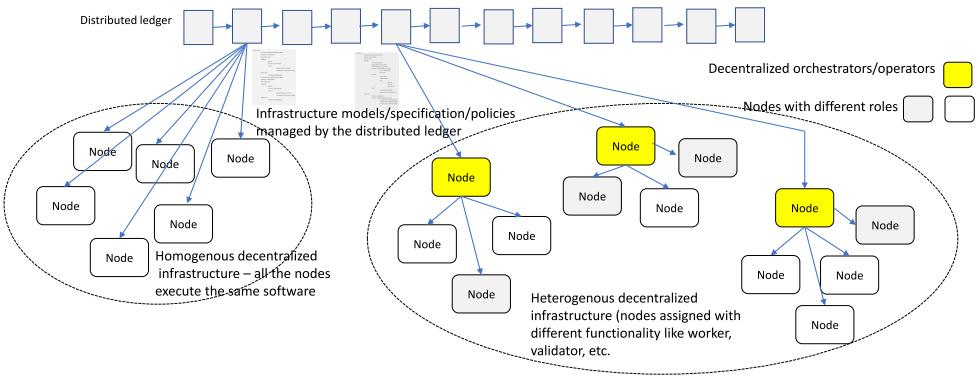

Fig. 2. Overview of DIaC.

and standards for decentralized computing platforms, including a hybrid of these two paradigms. Under such objectives, we have developed DIaC. This section will discuss the individual components and provide examples to illustrate the model.

Requirements: Our efforts are geared towards achieving the following requirements:

- *Decentralized orchestration and management of blockchain-based ICT infrastructures to support Dapps.*
- *Ensuring the security of infrastructure descriptions, configurations, settings, policies, and meta-data using blockchain-based security guarantees.*
- *Compatibility with existing orchestration standards like TOSCA.*
- *Universal and generic modeling support for various applications and use cases, allowing the tools to be used with any Dapp or blockchain system.*

However, developing a framework that meets these requirements is challenging as existing models are not designed for decentralized environments or blockchain-based applications. Instead, they rely on centralized orchestration, and cloud-based economic models rather than decentralization concepts and principles.

Fig. 2 provides an overview of the environment. The framework uses a blockchain/distributed ledger to manage the components and services required by a Dapp. It extends declarative-oriented languages like TOSCA and implements the extended language as smart contracts or chain codes. The governance entities of Dapps specify the intended infrastructure or operating environment as on-chain data objects using smart contracts/chain codes. Participating nodes retrieve these descriptions from the blockchain and assemble them into deployment documents (e.g., TOSCA instance templates). The nodes then deploy and configure the local node/nodes by interpreting and realizing the extended TOSCA documents. Rather than having centrally managed orchestrators, participants are independent and autonomous decision-makers incentivized to act in the best interest of the Dapp.

The distributed ledger tracks all nodes and ensures everyone follows the same rules and procedures. In Figure 2, the nodes are color-coded based on their roles - worker nodes (white) and validator nodes (light pink). The yellow nodes are decentralized orchestrators who manage the infrastructure based on on-chain data.

All the smart contract data related to a decentralized infrastructure can be updated following governance principles. Virtual machine and container images can be stored off-chain, for instance, using IPFS [18]. Links to the off-chain images are stored on-chain.There are several existing solutions that use IPFS as a container image, one example described in [19] which is IPFS-based container image distribution. Our framework can leverage the IPFS-base container distribution to store data as a function of the yellow nodes in Figure 2.

On-chain data objects related to a decentralized infrastructure can be stored as key-value pairs that can be realized using maps or dictionaries. To import and export infrastructure documents, the system supports either YAML-based or JSON-based documents. For a smart contract, an import function parses the received YAML document and stores the retrieved key-value pairs into on-chain data objects. An export function converts on-chain data objects into a YAML or JSON document.

Reasons to use TOSCA: Instead of developing a new infrastructure modeling framework from scratch, it is preferred to leverage the existing standards. TOSCA is a good option because it is open and vendor-agnostic. TOSCA can model infrastructures in a standardized manner, enable portability, facilitate solutions to increase interoperability, support reusable test cases, and improve automation. TOSCA support multiple ways to describe the same infrastructure, which provides great flexibility. It is important to point out that our concept, extending infrastructure modeling standards to Web3 and decentralized infrastructure environments, is not restricted to TOSCA. The concept can be applied to other standard-based modeling frameworks. In addition, all the existing infrastructure modeling standards need non-trivial extensions to be applicable for Web3 and decentralized applications because they were originally developed for centralized environments where orchestration and management are all based on a trusted central party.

Fig. 3 illustrates the flow of the decentralized infrastructure deployment process. Concepts of the TOSCA modeling standard are extended to adopt both abstract views of a decentralized infrastructure topology (type-level TOSCA) and the implementation level of nodes (instance-level TOSCA). For a Dapp environment, type-level documents encapsulate requirements, policies, and governance rules and offer a high-level view of the decentralized infrastructure topology.

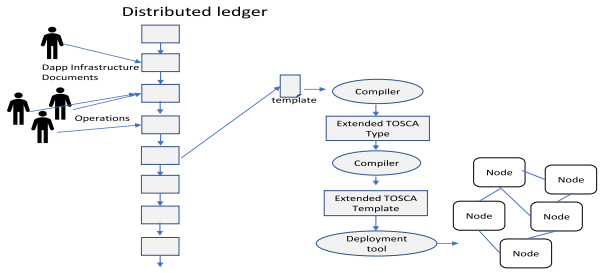

Fig. 3. Workflow of orchestrating decentralized infrastructure.

For a Dapp project, a governance entity can create initial documents reflecting an abstract view of the Dapp infrastructure. Then, the records are stored as on-chain data using multiple smart contracts (detailed explanation with examples in Section IV). The governance entities can update the documents using blockchain transactions. Participants can retrieve relevant data from a Dapp environment from the blockchain and assemble the data into comprehensive TOSCA documents.

Using TOSCA compiler, the documents will be translated to an instance-level model that can be used for implementing nodes in a decentralized system. Infrastructure documents stored on-chain can be modified over time. Local nodes synchronize with the blockchain. Infrastructure updates can trigger a reconfiguration of local nodes.

The proposed framework ensures that Dapp administrators deploy change in the form of infrastructure change/reconfiguration. It will follow the process below:

- *The changes are written into a TOSCA-based transaction. The extensions proposed in this paper work as a placeholder for the rules set for the deployment.*
- *The proposed changes will be vetted through on-chain governance and/or permissions (e.g., voting for adoption of the changes, verification of signatures, checking policies).*
- *A parser takes these changes and writes them into smart contracts executed into the change.*
- *Once validated using consensus, on-chain states will be updated with the new infrastructure specification, protected by the blockchain.*
- *A network of operators connecting to the blockchain will export the states, assemble them into TOSCA documents and convert them into actionable building blocks by a TOSCA interpreter.*
- *These are then deployed as changes into the targeted system.*

The above steps ensure that the infrastructure changes are not bound to any centralized component prone to failure and provide auditability with integrity protection.

IV. Detailed Design of DIaC

We present the detailed design of major components of DIaC in this section.

A. Assumptions

In this work, we make certain assumptions about the decentralized computing environment. **Firstly**, we assume a blockchain or distributed ledger with smart contract support applied for managing Dapps and decentralized computing infrastructures. The blockchain records a sequence of transactions in append-only mode. As a result, it comprises ordered blocks created from a genesis block under a consensus mechanism (e.g., PoW, PoS [20], PoA, PBFT [21], DPoS [22]). Each block has a block height. Further, we assume that blocks are generated with a relatively constant speed. Details of the consensus mechanism and block structure are irrelevant. In this work, we use Hyperledger Fabric as a target blockchain environment. However, the concept and framework can be extended to other blockchains. We also assume that the blockchain supports smart contracts. It is plausible to adapt the model to blockchains without smart contracts where data objects are stored off-chain, and only hash values are kept on-chain. Such work is outside the scope of this paper. **Secondly**, we assume that multiple Dapps can share the identical blockchain for infrastructure orchestration in this environment. How this orchestration blockchain manages its own participating nodes or infrastructure is outside the focus of this paper. **Thirdly**, we assume that the underlying blockchain system for supporting the consensus and network communications is secure and reliable, which means that measures have been taken to prevent attacks at separate layers. Such attacks include but are not limited to, for instance, 51% attack, side-channel attacks, Sybil attacks, attacks to DNS, hard forks, quantum attacks, attacks due to smart contract language design flaws, etc. **Fourthly**, we assume that the blockchain is used for transactions of tokens and assets where users may own portfolios of such assets. Thus, the participants of the Dapps and the decentralized infrastructures are only motivated by financial incentives.

B. Extensions to TOSCA Concepts

In our system, governance entities describe their infrastructure requirements and envision topology, security settings, and orchestration intents as template files, using extended TOSCA language. These files are imported to the blockchain via transactions with data integrity and authenticity protected by the public keys of the transaction submitter. After verification, the submitted data is stored as on-chain states.

The current TOSCA specification is designed to support cloud-based environments with centrally managed orchestrators. Most of the key TOSCA concepts need to be extended to support crypto-economy-based and decentralized topology. A summary of these changes is listed in Table I.

Although the extensions of TOSCA involve the introduction of new data attributes, node types, and policies like stakes, operators, and decentralization-based optimization, the described changes to the domain-specific language itself are non-intrusive. They can be applied within the scope of the original language. This means that users do not need to modify the core template framework of TOSCA. The extended TOSCA with decentralization and Dapp support maintains the actual structures of TOSCA documents and are backward compatible with the TOSCA core concepts like types, instances, policies, requirements, and capabilities.

TABLE I
OVERVIEW OF EXTENSION TO TOSCA CORE CONCEPTS

TOSCA features	Changes	Smart contract modules
Metadata	Introduction of new data attributes related to decentralized environment and crypto-economic based computing models like a stake.	core, mapper
Node_type	New node types like operators, validators, proxy, and new characteristics like TEE support, cloud vs. non-cloud nodes.	identity, topology, node_templates
Node_template	New node templates for Dapps and blockchain-enabled infrastructures like worker nodes, validator nodes, proxy nodes, etc.	node_templates
Capabilities	New capabilities are supported like TEE capability.	topology, node_templates
Relationship	New relationships are defined as composed_of, operated_by.	topology, node_templates
Policies	New policies are added like approaches for managing decentralization, and usage of stake in infrastructure management.	topology, mapper

TABLE II
SMART CONTRACT MODULES

Modules	Features
Governance	Implementation of governance logic like voting for changes.
Core	Essential and common settings for Dapps and decentralized computing infrastructures including links to other contract modules.
Names & identifiers	Credentials (e.g., X509 certificates), user accounts (wallet addresses) including operator lists, identifiers.
Artifacts	TOSCA artifacts including links to virtual machine and container images.
Decentralized infrastructure specification & topology	Topology and type data of decentralized infrastructures.
Infrastructure templates	Template data of decentralized infrastructures after mapping.
Mapper	Smart contract that allocates operators to abstract infrastructure specifications.
Security	Security policies and configuration settings like blacklists, and permissions.

C. Smart Contract Modules

Instead of using a single, smart contract or chain code program, the management of decentralized infrastructures is split into multiple smart contracts. Each component can further be recursively decomposed into multiple contract codes. Each smart contract program only maintains a subset of an infrastructure model. This practice is common in smart contract design with certain advantages, such as reduced footprint of the individual contract, lower risk caused by compromised contracts, easier upgrades, etc. The set of smart contracts realizes a single namespace for a decentralized infrastructure.

There are several ways that on-chain states can be updated: (i) voting by the governance smart contract; (ii) update by an authorized user, for instance, the original creator of the smart contract; (iii) update by a set of authorized users with a scheme like multi-signature.

V. USE CASE EXAMPLES

The extended framework of infrastructure as contract codes leverages the concept of smart contracts for describing, configuring, governing, and orchestrating decentralized computing infrastructures. The adoption of existing and open standards allows rapid development of infrastructure templates and recipes that can reduce both time and cost. The framework can be applied to specify and manage a layer 1 blockchain infrastructure, describe infrastructure integration between Web2 computing resources and blockchains like a hybrid infrastructure comprising both decentralized components and traditional cloud-based components,[1] cross-chain or multi-chain infrastructure modeling, cloud-based provision of Web3 infrastructure. Because adopting service provider agnostic standards and the existing support for cloud-based infrastructures, DIaC makes it easy to integrate cloud infrastructure with Web3 where the physical infrastructure is cloud-based and the infrastructure model is on-chain based on standards that are already supported by the cloud service providers. Such interoperability support may provide tremendous advantages to both the cloud service providers and Web3 developers.

In the subsection below, we will provide two examples to demonstrate how the extensions work. It is worth pointing out that these examples are for illustration purposes, for instance, to provide examples for the constructs and concepts described in Table I and Table II in the previous section.

A. Sharding Network

We consider a simple sharding-based peer-to-peer computing network. The nodes are called workers. To prevent certain attacks, a stake is required for joining the network. Additional security enhancement can include measuring reputation scores or negative incentives like deposits. The network is partitioned into multiple shards where worker nodes are randomly allocated to the shards. Time is divided into epochs, and worker nodes are re-shuffled between epochs. The structure is abstract enough to capture many use cases like using the worker nodes for out-sourced computation or smart contract execution (e.g., [23]).

Governance: The system includes a group of smart contracts that exist on-chain and implement a decentralized public consensus mechanism for authorizing changes to the system. These changes can be configuration parameter updates, adding or removing new objects/templates, contingency mechanisms in case of system failures, etc. Users of the system participate in the decision-making process by staking the relevant crypto-asset and proposing changes to the system through smart contracts. The existing users can then vote on these changes within the implemented time-lock threshold, and, if approved, the changes take effect automatically through the execution of the proposed smart contracts. The mechanism allows anyone to stake their crypto assets and have a say in the governance of the system. Every proposed change and vote is recorded on-chain, making it tamper-proof and resistant to manipulation/forgery. The implementation takes inspiration from how

[1]One example is Chainlink functions that apply AWS Lambda and cloud as Oracle for blockchain.

governance mechanisms are implemented on popular decentralized finance (DeFi) Daaps on the Ethereum blockchain, e.g., MakerDAO [24] and Uniswap [25].

```
stake_threshold: 100   // must own minimal stake
max_operator_number: 5000   // number of operators
block_interval: 5s //
epoch_interval: 500 // 500 blocks
...
artifacts: myDappArtifactContract
topology:   myDappTopologyContract
template: myDappTemplateContract
security: myDappSecurityContract
identity: myDappIdentityContract
...
```

Core: The core chain code stores common attributes, metadata, and definitions for the sharding network. It can specify the minimal stake required for participation. Here we assume that the stake is divided into a unit stake, for instance, 100 stakes per worker node. This means that if a stake owner has 500 stakes, then he/she can contribute five worker nodes. Each worker node is administrated by an operator – similar to the concept of an orchestrator. Operators are stake owners. An operator can manage multiple worker nodes. The number can be determined according to the amount of stake owned. In addition, the core contract can keep a list of references that point to other smart contracts.

```
operator_type:
  description: "myDappOperator type"
  derived_from: myDappNode
  requirements:
    minimal_stake: {{core.stake_threshold}}
    minimal_reputation: {{core.min_reputation}}
operators:
  description: "node operators" // a list of candidate operators
  list: \\
    - name: "alice@myDapp"
      type: operator_type
      identity:
        cert:     // X.509 certificate
        type: "X.509"
        stake: 1000
    - name: "bob@myDapp"
      type: operator_type
      identity:
        cert:     // X.509 certificate
        type: "X.509"
        stake: 800
    ...
```

Names and identifiers: This contract keeps track of names, identifiers, certificates, and user accounts like qualified operator lists. Many blockchain systems have implemented built-in identity management, which can be leveraged. For Hyperledger Fabric, it applies X.509 based credentials. In the case of Ethereum, wallet addresses can be used. Stake owners who want to host worker nodes can add themselves as operators by sending a request transaction, which will invoke a function to handle it. The request will be verified, for instance, by checking whether the requester has a sufficient stake or whether the operator list is full before the stake owner is inserted into the operator list. There are functions defined for adding, removing, and validating operators.

Artifacts: The contract for artifacts records virtual machine or container images and source code repositories like GitHub project links. The actual files are stored off-chain, for example using IPFS [18]. Only metadata is kept on-chain.

Decentralized infrastructure specification and topology: The contract is used to specify the envisioned topology in detail, like shards, and worker node details (e.g., link to the virtual machine or container image, link to any source codes, network

```
vm_worker_image:
  description: "worker node vm image"
  type: myDappArtifact
  properties:
    id: "92a78ef0-31d1-11ec-8d3d-0242ac130003"
    IPFS_CID:
    file_name: "myDappWorkerVM"
    attributes:
      -type: qcow2
      -size: 20G
      ...
    encryption:
      -algorithm: AES
      -key_distribution_method: "myDappSecretSharing"
      ...
```

```
topology:
node_groups:
  description: "node groups (shards)"
  derived_from: myDappTopology
  properties:
    name: "shard"+{{index}}   // shard name: shard1, ...
    occurrence: 20 // 20 shards
    workers_per_shard: 250
    epoch_interval: {{core.epoch_interval}}
  requirements:
    composed_of: worker_node
worker_node:
  description: "worker node"
  derived_from: myDappNode
  properties:
    occurrence: 5000
    image: {{artifacts.vm_worker_image}}
    ports:
      -target: 12345
       published: 12345
       protocol: TCP
    ...
  requirements:
    operator_selected_from: {{identity.operators}}
    ...
```

ports, service interfaces, and resources desired like GPUs). It defines the number of available worker nodes per shard and the number of shards. It can further specify requirements for the infrastructure like optimization policies. For instance, a policy requires that worker nodes administrated by the same operator be allocated to different shards. Moreover, topology details can be defined using relationships. In addition, validator node types can be specified for verifying the status of the worker nodes. A validator node can run pre-defined test requests against a worker node. In the validator node type, one can determine the commands for running these tests.

```
node_template:
  - id: ab76141a-31db-11ec-8d3d-0242ac130003
    description: "node template"
    type: worker_node
    properties:
      group: 1
      ipv4_addr:  169.25.138.132
      status:
        -alive: 6/15/2021 13:45:30.617
        -measurement:  // measurement results
    relationship:
      operated_by: "alice@myDapp"
  - id: bdc94556-31db-11ec-8d3d-0242ac130003
    description: "node template"
    type: worker_node
    properties:
      group: 2
      ipv4_addr: 239.245.96.116
      status:
        -alive: 6/15/2021 13:42:21.020
        -measurement:  // measurement results
    relationship:
      operated_by: "bob@myDapp"
  ...
```

Mapper: The mapper contract takes decentralized infrastructure specifications and data contained in other contract states as input. It computes infrastructure templates that can be converted later to deployable TOSCA templates. One essential function of the mapper contract is to allocate worker nodes to the operators. Other nodes of the blockchain network can verify the allocation plan. For security reasons, allocation is randomized. There are multiple approaches to implementing

a randomization function on-chain, for instance, using block hash as a random source in PoW-based blockchain or using VRF (verifiable random function) [26]. In this work, we utilize VRF to provide randomness to the mapper contract. Reconfiguration is necessary after updating of other parts of the infrastructure specification. This means that changes to other contract states can trigger the execution of the mapper contract. An alternative approach is that each contract tracks whether its state is dirty – and requires infrastructure reconfiguration. When a transaction is received from exporting infrastructure templates, it will trigger the re-execution of the mapper function that re-configures the infrastructure templates. Orchestrators periodically check the blockchain for infrastructure updates.

```
node_blacklist: [node_uuid, ...]  // list of nodes
operator_blacklist: [operator_name, ...]
permissions:
  - artifacts:
      update_by: admin_user
      admin_users: []
  - core:
      update_by: governance
    ...
...
```

Security policies and configuration setting: The contract stores security policies and related settings like maintaining blacklists of operators, worker nodes (using their IP addresses), and compromised credentials. It also keeps track of permission lists like who can update certain infrastructure specifications and on-chain states, such as permitted accounts that can update artifacts when new images are released.

After worker nodes are created, orchestrators will upload data like IP addresses using a smart contract. Validator nodes can check the worker nodes by sending requests. To verify whether a worker node has minimal computing resources, the validator nodes can send carefully designed tasks to the worker node such as a PoW puzzle. A threshold response time can be used to determine if the worker node has sufficient computing resources. Another approach is to use a reputation score that measures the average task completion time for the worker node in the past. A third approach can be applied in the case of the TEEs where TEE measurement reports can include information like the CPU model and hardware setting of a worker node.

B. Decentralized Computing Network Using TEEs

The second example is a TEE-based outsourced computing environment for data processing and machine learning applications. Similar decentralized infrastructures have been proposed in the literature for running AI applications. The IPFS network stores encrypted disk images where data to be processed is kept. Worker nodes apply the TEEs for protecting data confidentiality when the data is processed. To simplify the discussion, we use AMD SEV TEEs [27] that provide out-of-box support for encrypted virtual machines. AMD TEEs implement memory encryption and protect virtual machines from the compromised hypervisor or local host [27], [28], [29]. In this work, we do not consider TEE attacks such as side-channel attacks which are outside the scope of this paper. We assume that the TEEs are secure enough for use cases of confidential computing. TEEs have been proposed for confidential smart contract execution (e.g., [30]).

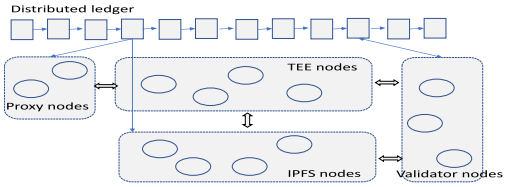

Fig. 4. A decentralized computing infrastructure with TEE worker nodes and other nodes with different roles (validators, decentralized storage nodes, SGX proxy nodes).

```
node_template:
  - id: ac81a42a-71db-55ea-7a3e-1b46ae36010a
    description: "proxy template"
    type: proxy_node
    properties:
      ipv4_addr: 169.25.118.12
      image: {{artifacts.vm_proxy_image}}
      enclave: {{artifacts.sgx_proxy_enclave}}
      credential:  // private key generated by enclave
        cert:    // X.509 certificate
        type: "X.509"
      cpu_arch:
        -vendor: "Intel"
        -model: "i7-8705G"
        -sgx:
          sgx1: true
          sgx2: false
          sgx_enclv: false
      status:
          -alive: 6/11/2021 09:15:24.401
          -measurement: //sgx remote attestation result
      relationship:
          operated_by: "carol@myDapp"
          connect_to: //AMD SEV virtual machine (id list)
            -2621acf8-338d-11ec-8d3d-0242ac130003
            -2621af14-338d-11ec-8d3d-0242ac130003
            -2621b00e-338d-11ec-8d3d-0242ac130003
             ...
```

Here we focus on the unique aspects of infrastructure specification for this use case. The contract code can define worker nodes with TEE properties in the infrastructure specification, like providing a list of optional TEE CPU models. After a worker node is launched, the system will verify that the virtual machine is launched under AMD SEV, a functionality supported by the AMD remote attestation protocol. This is achieved by submitting the measurement report. To work with the AMD TEE and remote attestation protocol out-of-box, we introduce a SGX-based proxy as a relay of disk decryption keys. This eliminates the dependence on a single party for holding disk decryption keys. The design applies secret sharing [31]. Shares of a disk decryption key are distributed to multiple entities in the system. After receiving the measurement report and verifying that a virtual machine instance is properly launched by an AMD SP using SEV memory encryption, encrypted key shares will be uploaded to the blockchain. The key shares can be retrieved and sent to the SGX proxy, where the shares will be assembled into the disk decryption key (k out of n shares is enough). The SGX enclave uses a secure communication channel with the AMD SP, created according to the AMD protocol. The SGX proxy uses this channel to send the assembled key that will be forwarded to the virtual machine instance. After receiving the key, the encrypted disk can be decrypted. Then the confidential virtual machine instance hosted under AMD SEV can process the decrypted data. It is worth mentioning that the idea of enclave proxy/relay has been implemented

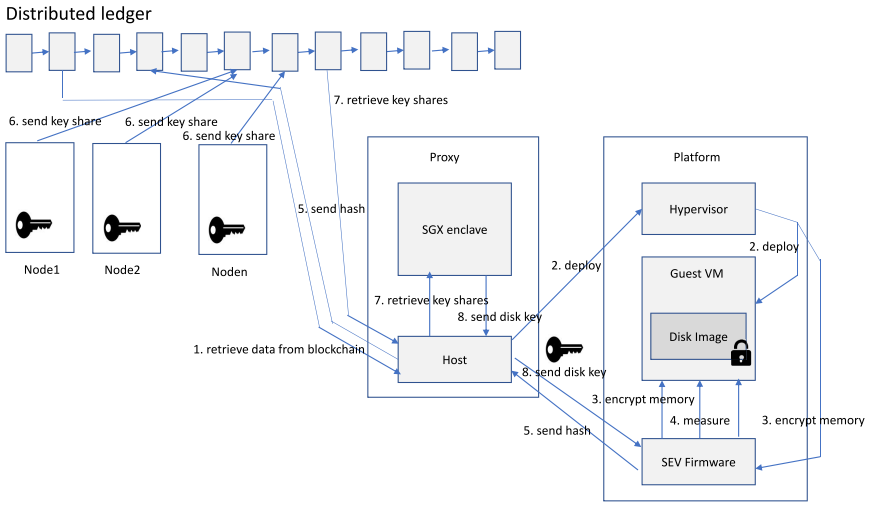

Fig. 5. TEE worker node and disk decryption key distribution using blockchain facilitated secret sharing.

TABLE III
SUMMARY OF ATTACK SURFACE AND MITIGATION STRATEGIES

Layers	Threats	Analyses
Application use cases	Use case specific attacks like fairness, verification of computing results, load balancing	Addressed by use case specific defense mechanisms.
	Attacks to the TEEs including side-channel exploits	Addressed by approaches mitigating TEE security vulerabilities like security patches, side-channel resistant libraries, address space randomization, etc.
Our protocol & processes	Attacks to governance	Addressed by secure on-chain voting designs, delay of voting rights (defense against flash loan based attack).
	Attacks to orchestration or proxy nodes	Addressed by decentralization (e.g., partitioning Dapp infrastructure into independently managed components), on-chain blacklist of compromised nodes.
	Cheating on resources provided or availability	Addressed by having validator nodes and reputation scores.
	Attacks from compromised or malicious participating nodes	Addressed by enforcing minimal stake requirements, on-chain mapping policies (e.g., maximize decentralization), blacklist of offending nodes, and reputation scores.
	Hijack smart contract updates	Addressed by enforcing permission rules and different modes for updating smart contracts (e.g., multi-signature control, on-chain permission checking).
Smart contract & Toolchain	Reentrancy Attack,	Addressed in the system by smart contract security protection, model checking, and execution atomicity.
	DoS with unexpected revert	
	Unchecked call return value	
	Call stack depth limit exceeded	
Data	Various (e.g. data integrity, data availability)	Attacks not specific or dependent on our design and use cases, and mostly addressed by the underlying blockchain protocols and implementation.
Blockchain	Various (e.g., selfish-mining, double spending, bribery attack, Sybil attack)	
Network	Various (e.g., DDoS, eclipse)	

in systems like ARM Veracruz where an SGX proxy server is used for supporting cross-platform remote attestation in a heterogeneous TEE environment. A difference is that in our example use case, the SGX proxy is used for key distribution.

C. Remarks

As demonstrated by the examples, it is feasible to reuse and extend the existing standards for centrally managed infrastructures to decentralized and blockchain-based environments. Compared with centrally managed infrastructures, the extension exhibits the following unique characteristics: on-chain storage and management of infrastructure specification and definitions vs using a centralized database, decentralized orchestration of infrastructure deployment vs centrally managed orchestration, public verification of infrastructure attributes which is not a requirement for centralized orchestration, support of Blockchain based economic models like staking instead of cloud-based economic models, removal of a single point of failures, decentralized infrastructure governance and management, and enhanced security by using blockchain and smart contracts to secure infrastructure models vs using a centrally hosted database for storing infrastructure models and associated metadata. To summarize, the extended TOSCA using smart contracts is a general tool and standard-based approach for describing decentralized infrastructures to support decentralized application scenarios. Another point to highlight is that our work is different from applying blockchains to secure centralized orchestration, which is a separate research topic.

VI. Threat and Security Analysis

Regarding security vulnerabilities in blockchain-based systems, they can be categorized based on the software stack. Assuming that the blockchain system is appropriately implemented with defense mechanisms to reduce security risks, the security risk analysis will focus on vulnerabilities specific to the design. It is important to note that attacks on a Dapp built on the system and exploits of TEE vulnerabilities are not considered in this work [32].

Tamper proofing of the smart contracts: To prevent attacks on innovative contract programs for managing decentralized infrastructures, we use a DIaC tamper-proofing method. We assume that the smart contract language is secure. Before any transactions are made that reconfigure a decentralized infrastructure, they must first be verified on-chain. Only authorized entities can change infrastructure specifications, security policies, metadata, etc. We implement a multi-signature approach for updating smart contracts and on-chain states to mitigate the risk of losing or exposing private keys (e.g., [33]). The blockchain ensures security assurance, providing immutability and integrity for the infrastructure data stored on-chain.

Cheating nodes: A worker node can cheat by not providing resources. As we have already discussed, there are multiple approaches to mitigate such risk. Firstly, validator nodes can check the worker nodes by sending resource-intensive tasks to verify whether the worker node has sufficient resources. Secondly, reputation scores can be used to rank worker nodes based on their past task completion times. Finally, when worker nodes are TEE-enabled, measurement reports can be certified. The report can include information such as CPU model and hardware settings.

Compromised orchestrators: We assume that individual orchestrators can be compromised. To reduce the risk of a single point of failure, the framework relies on decentralized orchestrators. A decentralized infrastructure is maintained by multiple independent and autonomous orchestrators. Security breaches to a part of the infrastructure can be handled by isolating the compromised nodes from the rest of the system, for instance, using a blacklist (protected as on-chain data).

Malicious operators: We assume that operators are rational and financially motivated. The framework applies negative incentives to discourage undesired operator behaviors. A project can enforce that an operator must own a specific amount of stake to participate. It assumes that the majority of operators are trusted.

Attack against on-chain governance: An adversary may launch attacks against the governance mechanism like skewing voting outcomes using a flash loan-based attack. These attacks are not unique to our framework because on-chain governance is applied in numerous blockchain projects. Therefore, we can apply the same best practices and defense approaches to protect smart contracts and management processes.

Attack against the random source used for allocation: Depending on the use cases, specific security risks may arise if an adversary can manipulate the mapping process that allocates operators to the worker nodes. Hence, randomization plays a critical role as a defense. In this case, we assume that VRF itself is secure.

VII. Implementation and Evaluation

A. Extended TOSCA

TOSCA defines a very extensible modeling framework. In our case, many decentralization and blockchain-related concepts like a stake, and randomization between operators and worker nodes, are handled on-chain. The evolution of the decentralized infrastructure is also coordinated through smart contracts and blockchain transactions. Orchestrators/operators can retrieve infrastructure specifications and configuration as key-value export documents from the blockchain. The infrastructure modeling key-value pairs are assembled into a TOSCA template document that can be parsed and interpreted by a TOSCA interpreter. For prototyping, we leverage an open-source TOSCA interpreter. One such tool is the TOSCA parsing engine implemented in the Alien4cloud orchestrator (https://alien4cloud.github.io/). The tool supports TOSCA files in YAML format. Segments of infrastructure declaration in YAML can be exported from the blockchain using the smart contract function. The TOSCA interpreter can take the segments as inputs and synthesize a deployment scenario including a deployment plan from the declaration and specification. During deployment, artifacts will be downloaded from IPFS or source code repositories according to the deployment plan. The deployment plan in JSON format constitutes technical steps for provisioning nodes of a decentralized computing environment. The future extension of the TOSCA interpreter is to support accelerators for computing such as GPUs and FPGAs, and integrate the deployment of hybrid TEE computing environments.

B. Experiments

For experiments, we use Hyperledger Fabric V2.2 [34]. Hyperleger Caliper (https://www.hyperledger.org/use/caliper) is used as a benchmarking tool.

Hyperledger Fabric is used to implement infrastructure contracts (chain code in Hyperledger terminology) and manage infrastructure key-value pairs, which save the inputs for creating a configuration YAML file. To test the system, we apply randomly generated infrastructures specified as key-value pairs. Table IV summarizes the data collected for *Submit Transaction* and *Export Infrastructure* with different infrastructure sizes (measured in the size of key-value pairs). The operation of *Submit Transaction* consists of two parts, reading the infrastructure input data (in JSON format in our experiments) and submitting a constructed transaction to the Fabric to store on the chain. We collected the total time it takes for this whole process to take place. Similarly, the generation of infrastructure export also consists of two steps. The first step is querying data from the chain and the second step is generating a configuration `.yaml` file from the returned data. For randomly generated infrastructures, the input data sizes used in the experiment ranged from 0.7KB to 5.4KB. We can observe that the time differences between the two operations are small even with the increased infrastructure size. Table V summarizes several important metrics on the operations of *Create Asset* and *Read Asset*. We used Caliper for collecting stats for the network. We observe that CPU usage for peers was

TABLE IV
PERFORMANCE OF CREATING ON-CHAIN INFRASTRUCTURE DATA (IMPORT) AND GENERATING INFRASTRUCTURE FILE (EXPORT)

File Size(KB)	Import infrastructure data (sec)	Generate infrastructure file - export (sec)
0.743	3.347	1.269
1.1	3.349	1.237
2.59	3.383	1.266
3.3	3.388	1.247
4.3	3.42	1.250
5.43	3.481	1.292

TABLE V
PERFORMANCE OF ASSET CREATING AND READING

Metric	Create Asset	Read Asset
Successes/ Fail	1000/0	10338/0
Send Rate(TPS)	60.5	11.52
Max/Min Latency(s)	11.52/0.46	0.19/0.001
Avg Latency(s)	6.95	45.4
Throughput(TPS)	45.4	175.0

around 14% and reaching to a Max value of 25%. For orderer nodes, we see a Mac of 10% and an average of 5%. For the blockchain network in our experiment, we saw an average of 100MB for peer nodes and 50MB for orderer nodes in terms of average memory utilization. Traffic In/Out for a peer node is around 11/8 (MB), and around 7/13 (MB) for an orderer node. In terms of disk reads, the peer nodes barely reach 1MB. But for disk write, we see that peers had 8MB. For orderer nodes, the disk write and disk read are around 13 MB and 4 MB respectively.

It is possible to further fine-tune Fabric to improve the performance of DIaC, which is orthogonal to the scope of this work. For instance, by removing certain bottlenecks like parallelizing endorsement policy verification and optimizing CouchDB bulk read/write, the overall throughput of Fabric can achieve 2,250 tps [35]. FastFabric achieved a throughput of 20,000 tps by reducing the computation and I/O overhead during transaction ordering and validation phases [36]. These optimizations do not require any interface changes to the Hyperledger Fabric and can be easily incorporated into the implementation of DIaC.

C. End-To-End Implementation

This section will discuss the experiments where we demonstrate DAI for a real-world use case. For this experiment, we selected the following templates from the sample AWS CloudFormation templates[2]:

- Amazon EC2 instance in a security group.
- Amazon EC2 instance with an ephemeral drive.

All the experiments were performed on an AWS EC2 instance with Ubuntu 22.04LTS. This was done to make sure we have the same network bandwidth and speeds for all the experiments. In Fig. 6 we discuss the flow of the experiments which consists of mainly three stages.

1) TOSCA. We use the TOSCA to write down the configuration details of the users and these details are then submitted as a transaction to the Hyperledger Fabric.

[2]https://aws.amazon.com/cloudformation/resources/templates/

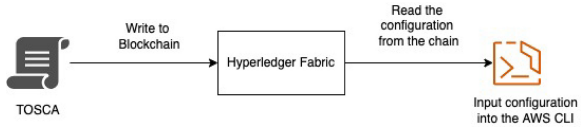

Fig. 6. Flow of the End-To-End Implementation.

TABLE VI
RESULTS FOR IMPLEMENTATION OF CLOUD FUNCTION TEMPLATES

Type of Instance	With Blockchain	Without Blockchain
EC2 instance in a security group	9.704s	3.867s
EC2 instance with an ephemeral drive	7.459s	2.306s

2) Hyperledger Fabric. We use the Hyperledger Fabric V2.2 for the experiments.
3) AWS CLI. We use the AWS CLI V2 for the creation of EC2 instances.

In the use case of Amazon EC2 instance in a security group basically means creating an Amazon EC2 instance running the Amazon Linux AMI. The AMI is chosen based on the region in which the stack is run. This example creates an EC2 security group for the instance to give you SSH access. In the other use case of Amazon EC2 instance with an ephemeral drive means to attach ephemeral drives using EC2 block device mappings. In this experiment, we also included creating the EC2 instance and attaching the created volume.

From the Table VI we can see that in use case 1 it takes around 5sec more time and in use case 2 a similar trend where it takes 5sec more. But in this experiment, we were able to showcase the usability of this kind of system in real-world scenarios.

D. Remarks

The proof-of-concept exercises with exemplary scenarios and experiments have demonstrated the feasibility of applying and extending a standard-based approach for describing, modeling, configuring, and deploying decentralized infrastructure. Such an approach may enable advantages such as improved interoperability, the opportunity of re-using already developed modeling tools, reduced cost, and barriers for adoption, transferability of knowledge and skills, enhanced manageability of decentralized infrastructures, and better automation. The standard-driven framework paves a road of infrastructure governance for the emerging area of Web3. With the ever-increasing number of decentralized systems (e.g., so-called layer 1, layer 2, layer 0) and in-depth integration of off-chain infrastructure with on-chain infrastructure, the need for a standard for modeling decentralized infrastructures will be imperative to prevent infrastructure fragmentation, Web3 silos, barriers of infrastructure governance and management. The described research work represents the first step towards achieving the overarching goals. In the following Sections IX and X, we analyze research challenges and discuss the directions of future work.

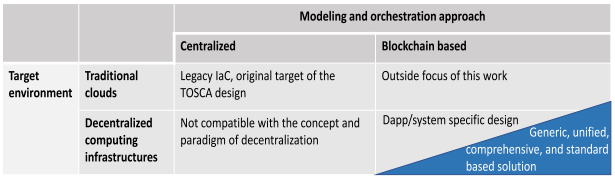

Fig. 7. Comparison of DIaC with other related works (scope of this work in blue background).

VIII. Related Works

While there has been a considerable amount of work in the field of centralized computing and the application of infrastructure-as-code for automation and DevOps, there has not been much consideration for Dapps and decentralized computing environments enabled by blockchain technology and the associated economic models. Infrastructure-as-code (IaC) is the practice of automatically configuring system dependencies and providing local and remote instances. To harness infrastructure-as-code, administrators write a blueprint that contains deployment specifications ready for orchestration in the clouds. A systematic study [37] on the existing IaC research shows the work that has been done in centralized infrastructure. Hummer et al. [38] propose a testing-based infrastructure. There has been research, applying blockchains for managing various facets of the cloud. A survey of blockchain and cloud is provided in [39]. These efforts significantly differ from our work because we target decentralized computing infrastructures and their needs for infrastructure management and automation (using smart contracts) instead of centrally managed cloud-based deployment.

There is plenty of research relying on the TOSCA standards as we see in the work by Artac et al. [40]. Similar to the approach in this paper and on which the extension has been proposed. In their work, Tsagkaropoulos et al. [41] present a set of TOSCA extensions to model edge computing applications.

Kim et al. created a TOSCA-based clustering service to virtualize network functions. A framework automates the configuration and management of virtual network functions (VNFs). Moreover, the framework is integrated with a cluster management service for high availability and auto-scaling of VNFs [42]. However, these TOSCA extension efforts focus on other centrally managed ICT infrastructure use cases like support for edge computing or network virtualization. None provide a standard-based modeling tool for decentralized infrastructures and Dapps, as described in this paper. These efforts are parallel to our work, so it may be plausible to incorporate some subset of the functionalities in our work.

Since [43], there has been work on multiple cloud deployments which tries to answer the same question on how to have a multi-cloud model of deployment. Donsuypae [44] proposed a two-fold approach in his Masters thesis, concerning Essential Deployment Meta Model (EDMM) and Business Process Model and Notation (BPMN). These are used to build partial models which then can be used to generate and deploy the infrastructure. Wild et al. [45] present a concept for decentralized cross-organizational application deployment automation. They introduce a global declarative deployment model that describes a composite cross-organizational application, which is split into local parts for each participant. Based on the split declarative deployment models, workflows have been generated that form the deployment choreography and coordinate the local deployment and cross-organizational data exchange.

Individual Dapp projects may adopt a customized approach for managing their computing environment like project-specific governance using a smart contract. This can lead to a lot of different scenarios including using confidential computing [46], [47], [48], machine generated code [49]. Our work distinguishes itself from such efforts because we aim at a generic, comprehensive, unified, and standard-based approach for modeling, provisioning, and managing decentralized computing infrastructures and Dapps rather than a makeshift design only applicable to a specific system or a single Dapp. We achieve our goal by extending TOSCA standard.

Our prior work [43] on the matter talks about our reference implementation for having a framework for decentralized infrastructure deployment using smart contracts. The current work builds on top of that and implements an end-to-end deployment scenario taken from AWS Cloudformation production-ready environment and something that Amazon itself recommends. We also were able to extend the architecture to allow for future extension of having a decentralized identity and bad code protection, as discussed in Future Works. We are also able to provide a benchmark on the end-to-end implementation compared to traditional non-blockchain deployment.

Our approach differs from existing approaches by introducing the extension of TOSCA which handles the metadata needed for making the Infrastructure as Code pipeline work. Existing cloud deployment methods do not work with Dapps.

IX. Research Challenges & Future Works

In this work, we explored how Decentralized Application Infrastructure(DIaC) as a framework helps achieve decentralized infrastructure-based deployment. This enables us to also have the following properties

- Audit-ability. Since each of the deployment parameters is encompassed in the blockchain as commits, it gives precise granular insight into the deployment and audibility of the deploy parameters. This helps in separating deployment errors and failures. Existing work [50] has delved into how audit can be done at the VM instance level when deployed, but that is only applicable at the instance level. In a multi-cloud environment, it's crucial to have continuous monitoring and audit capabilities of deployment as shown by Torkura et al. [51].
  Even though DIaC gives the capability of auditing infrastructure deployment in multiple environments, it still doesn't give an easy way to surface that information or a way to monitor for bad changes. One important line of future work could to be have logic built into the transactions using smart contracts which prohibit certain kinds of deployments based on previous blacklisted or a list of pre-defined parameters.

- Defect prevention. Since DIaC isn't just a deployment method, rather works on certain conditions (the voting mechanism) it prevents bad deployment to some degree. However, Infrastructure as Code (IaC) by and large is prone to having insecure deployment templates as seen by the survey on configurations of Mozilla, Openstack and Wikimedia [52]. An open question [52] is whether we can mine for characteristics of these templates and have safeguards programmed as smart contracts directly into the blockchain, which will prevent deployment even if the consensus vote was to deploy. These can be divided into modular properties for generalization.
- Identity. At the moment there is no decentralized Identity Management component for the deployment. It just acts as a deployment infrastructure. In our prior published work [43] we had shown how we can have a decentralized infrastructure to deploy code, however, this can be more streamlined with a decentralized identity component, which can tie up multi-cloud deployments to a single identity based on project or even granular developer level, independent from cloud providers provided identity which is different for each provider.

A. Open Research Question

A more crucial future work would be trying to minimize the latency of the system so that it alleviates some of the performance penalties introduced by the blockchain layer. To optimize the performance of a blockchain-based system and minimize extra time, we can consider leveraging edge computing to distribute computational tasks, thereby speeding up block verification. Use lightweight cryptographic algorithms for decentralized authentication to enhance security without compromising speed. Regular performance evaluations can identify bottlenecks, enabling targeted optimizations like fine-grained access control to reduce computational load. Additionally, consider using faster consensus algorithms like Proof of Stake (PoS) and batching transactions to make better use of block space. These strategies collectively aim to reduce latency and improve the overall efficiency of the system. However each of these approaches will require careful study and experimentation to validate the feasibility and implementation.

Validation of computing resources: Different from the operational environment like a cloud where service providers are somewhat trusted. In a decentralized environment where resource providers are economic agents primarily driven by financial incentives, there is a major challenge in validating ICT resources in a decentralized environment.

Complex policies for the decentralized computing environment: The principle of decentralization prefers an open and permissionless environment for participants. However, such openness leads to new threat models and attack surfaces that can jeopardize the well-being of a decentralized computing infrastructure for running Dapps. To address a wide range of concerns in such an environment, policies, and governance mechanisms are necessary, for instance, who can qualify as validators in a Proof-of-Stake (PoS) based system. For instance, a set of policies can require that in a PoS blockchain system, no more than a certain number of validators with a combined stake of more than 50%, no existence of any subset of validators with size n situated in the same continent, no more than a certain percentage of validators using the cloud.

Decentralized infrastructure optimization and orchestration: Traditionally, management and optimization of ICT resources are centralized, which is not aligned with the principle of decentralization. New challenges will arise when centralized resource optimization toolsets are decentralized. In the new environment, orchestration, and optimization do not rely on trusted and centrally managed components. How to achieve efficiency and effective resource management in such a setting remains open research.

Improving implementation and performance: Beyond the current feasibility evaluation and proof of concept prototyping, future work includes a demonstration of fully functional implementation of concrete use cases using decentralized like IPFS, and comprehensive performance analysis on detailed functionalities such as policy updates and infrastructure governance.

As decentralized "cloud" and computing infrastructures continue to evolve to support a myriad of Dapp use cases, the complexity of managing these systems also escalates. Future research could explore the development of complex policy frameworks and governance mechanisms that are compatible with the decentralized architectures we have proposed. This could involve creating smart contracts that are not just functional but also policy-aware, thereby enabling more nuanced system management. Such advancements would be instrumental in achieving a balance between efficiency, security, and flexibility in decentralized computing environments, pushing us closer to the goal of full decentralization in general-purpose computing.

X. Conclusion

To develop a unified, general-purpose, comprehensive, and standard-based infrastructure modeling tool for decentralized computing environments and the Dapps empowered by them, we have extended the prominent TOSCA modeling and orchestration language, originally designed for cloud-based deployment, to meet the need of this paradigm shift of computing facilitated by decentralization. Instead of depending on centrally managed components for orchestration, our system leverages smart contracts and the security guarantees of the blockchains for coordinating decentralized infrastructure creation, provisioning, reconfiguration, and evolution. A prototyping environment is created using Fabric to demonstrate this new transformative framework. On top, we have implemented an end-to-end scenario with DIaC and compared it with traditional implementation to test the feasibility of the system. Showing the efficacy of the system in comparison with existing production-based solutions allows us to build future extensions for more robustness and completeness.

References

[1] D. Weerasiri, M. C. Barukh, B. Benatallah, Q. Z. Sheng, and R. Ranjan, "A taxonomy and survey of cloud resource orchestration techniques," *ACM Comput. Surveys*, vol. 50, no. 2, pp. 1–41, 2017.

[2] K. Morris, *Infrastructure as Code: Managing Servers in the Cloud*, O'Reilly Media, Inc., Sebastopol, CA, USA 2016.

[3] A. Brogi, J. Soldani, and P. Wang, "Tosca in a nutshell: Promises and perspectives," in *Proc.Eur. Conf. Service-Oriented Cloud Comput.*, 2014, pp. 171–186.

[4] A. Luzar, S. Stanovnik, and M. Cankar, "Examination and comparison of tosca orchestration tools," in *Proc. Eur. Conf. Softw. Archit.*, 2020, pp. 247–259.

[5] S. Nakamoto (Las Vegas, NV, USA). *Bitcoin: A Peer-to-Peer Electronic Cash System*. (Dec. 2008). Accessed: Jul. 1, 2015. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[6] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," Ethereum, Zug, Switzerland, White Paper, 2014, Accessed: Aug. 22, 2016. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[7] "Golem." 2020. [Online]. Available: https://golem.network/

[8] "Iexec." Accessed: Nov. 14, 2022. [Online]. Available: https://iex.ec/

[9] "Sonm." Accessed: Nov. 14, 2022. [Online]. Available: https://sonm.com/

[10] "Uchain." Accessed: Nov. 14, 2022. [Online]. Available: https://uchain.world/

[11] B. Yan, P. Chen, X. Li, and Y. Wang, "Nebula: A blockchain based decentralized sharing computing platform," in *Proc. Blockchain Trustworthy Syst.*, 2020, pp. 715–731.

[12] T. Hanke, M. Movahedi, and D. Williams, "DFINITY technology overview series, consensus system," Jan. 2018, *arXiv:1805.04548*.

[13] *Ocean Protocol: A Decentralized Substrate for AI Data and Services*, Ocean Protocol Found., Singapore, 2018.

[14] "Intel software guard extensions." Intel. Accessed: Nov. 14, 2022. [Online]. Available: https://software.intel.com/sites/default/files/332680-001.pdf

[15] A. Bergmayr et al., "A systematic review of cloud modeling languages," *ACM Comput. Surv.*, vol. 51, no. 1, pp. 1–38, Feb. 2018. [Online]. Available: https://doi.org/10.1145/3150227

[16] K. Kritikos, P. Skrzypek, A. Moga, and O. Matei, "Towards the modelling of hybrid cloud applications," in *Proc. IEEE 12th Int. Conf. Cloud Comput. (CLOUD)*, 2019, pp. 291–295.

[17] "TOSCA simple profile in YAML version 1.3." Accessed: Nov. 14, 2022. [Online]. Available: https://docs.oasis-open.org/tosca/TOSCA-Simple-ProfileYAML/v1.3/TOSCA-Simple-Profile-YAML-v1.3.html

[18] J. Benet, "IPFS—Content addressed, versioned, P2P file system," 2014, *arXiv:1407.3561*.

[19] K. Tokunaga, "P2P container image distribution on IPFS with containerd and nerdctl," in *Proc. Free Open Source Softw. Devel.' Eur. Meeting (FOSDEM)*, 2022, pp. 4–14.

[20] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Proc. Adv. Cryptology – CRYPTO*, 2017, pp. 357–388.

[21] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. 3rd Symp. Operating Syst. Design Implement.*, 1999, pp. 173–186.

[22] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1432–1465, 2nd Quart., 2020.

[23] "Harmony white paper. Version 2.0." Accessed: Jan. 10, 2021. [Online]. Available: https://harmony.one/whitepaper.pdf

[24] "The maker protocol: Makerdao's multi-collateral Dai (MCD) system," Maker.com. Accessed: Nov. 14, 2022. [Online]. Available: https://makerdao.com/whitepaper/White%20Paper%20-The%20Maker%20Protocol_%20MakerDAO%E2%80%99s%20Multi-Collateral%20Dai%20(MCD) %20System-FINAL-%20021720.pdf

[25] H. Adams, N. Zinsmeister, M. Salem, R. Keefer, and D. Robinson, "Uniswap v3 core," Uniswap, Tech. Uniswap, Brooklyn, NY, USA, Rep., 2021.

[26] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *Proc. 40th Ann. Symp. Found. Comput. Sci (Cat. No.99CB37039)*, 1999, pp. 120–130.

[27] "AMD SEV-SNP: Strengthening VM isolation with integrity protection and more," Adv. Micro Devices, Inc., Santa Clara, CA, USA, White Paper, Jan. 2020.

[28] D. Kaplan, "AMD x86 memory encryption technologies," in *Proc. 25th USENIX Security Symp.*, Aug. 2016, pp. 9–11.

[29] D. Kaplan, J. Powell, and T. Woller, "AMD memory encryption," Adv. Micro Devices, Inc., Santa Clara, CA, USA, White Paper, 2016.

[30] R. Cheng et al., "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts," in *Proc. IEEE Eur. Sympo. Security Privacy (Euro S&P)*, 2019, pp. 185–200.

[31] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *Proc. 28th Ann. Symp. Found. Comput. Sci.*, 1987, pp. 427–438. [Online]. Available: https://doi.org/10.1109/ SFCS.1987.4

[32] A. Nilsson, P. N. Bideh, and J. Brorsson, "A survey of published attacks on intel SGX," 2020, *arXiv:2006.13598*.

[33] M. Drijvers et al., "On the security of two-round multi-signatures," IACR, Bellevue, WA, USA, Rep. 2018/417, 2018, [Online]. Available: https://ia.cr/2018/417

[34] E. Androulaki et al., "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th Euro. Syst. Conf.*, 2018, pp. 1–15.

[35] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing hyperledger fabric blockchain platform," 2018, *arXiv:1805.11390v1*.

[36] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "Fastfabric: Scaling hyperledger fabric to 20 000 transactions per second," *Int. J. Netw. Manag.*, vol. 30, no. 5, p. e2099, 2099. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nem.2099

[37] A. Rahman, R. Mahdavi-Hezaveh, and L. Williams, "A systematic mapping study of infrastructure as code research," *Inf. Softw. Technol.*, vol. 108, pp. 65–77, Apr. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584918302507

[38] W. Hummer, F. Rosenberg, F. Oliveira, and T. Eilam, "Testing idempotence for infrastructure as code," in *Proc. Middleware*, 2013, pp. 368–388.

[39] K. Gai, J. Guo, L. Zhu, and S. Yu, "Blockchain meets cloud computing: A survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2009–2030, 3rd Quart., 2020.

[40] M. Artac, T. Borovssak, E. Di Nitto, M. Guerriero, and D. A. Tamburri, "Devops: Introducing infrastructure-as-code," in *Proc. IEEE/ACM 39th Int. Conf. Soft. Eng. (ICSE-C)*, 2017, pp. 497–498.

[41] A. Tsagkaropoulos, Y. Verginadis, M. Compastié, D. Apostolou, and G. Mentzas, "Extending tosca for edge and fog deployment support," *Electronics*, vol. 10, no. 6, p. 737, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/6/737

[42] M. Kim, T.-X. Do, and Y. Kim, "Tosca-based clustering service for network function virtualization," in *Proc. Int. Conf. Inf. Commun. Technol. Convergence (ICTC)*, 2016, pp. 1176–1178.

[43] R. Karanjai et al., "Decentralized application infrastructures as smart contract codes," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, 2022, pp. 1–9.

[44] N. Donsuypae, "Decentralized cross-organizational application deployment using multiple different deployment automation technologies," M.S. thesis, Inst. Archit. Appl. Syst., Univ. Stuttgart, Stuttgart, Germany, 2021.

[45] K. Wild, U. Breitenbücher, K. Képes, F. Leymann, and B. Weder, "Decentralized cross-organizational application deployment automation: an approach for generating deployment choreographies based on declarative deployment models," in *Proc. Int. Conf. Adv. Inf. Syst. Eng.*, 2020, pp. 20–35.

[46] K. Rabimba, L. Xu, L. Chen, F. Zhang, Z. Gao, and W. Shi, "Lessons learned from blockchain applications of trusted execution environments and implications for future research," in *Proc. 10th Int. Workshop Hardware Archit. Support Security Privacy*, 2022, pp. 1–8. [Online]. Available: https://doi.org/10.1145/3505253.3505259

[47] R. Karanjai et al., "Decentralized translator of trust: Supporting heterogeneous tee for critical infrastructure protection," in *Proc. 5th ACM Int. Symp. Blockchain Secure Crit. Infra.*, 2023, pp. 85–94. [Online]. Available: https://doi.org/10.1145/3594556.3594626

[48] R. Karanjai et al., "Dhtee: Decentralized infrastructure for heterogeneous tees," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, 2023, pp. 1–3.

[49] R. Karanjai, E. Li, L. Xu, and W. Shi, "Who is smarter? an empirical study of ai-based smart contract creation," 2023, *arXiv:2308.02955*.

[50] L. Litty and D. Lie, "Patch auditing in infrastructure as a service clouds," in *Proc. 7th ACM SIGPLAN/SIGOPS Int. Conf. Virtual Execution Environ.*, 2011, pp. 145–156. [Online]. Available: https://doi.org/10.1145/1952682.1952702

[51] K. A. Torkura, M. I. H. Sukmana, F. Cheng, and C. Meinel, "Continuous auditing and threat detection in multi-cloud infrastructure," *Comput. Security*, vol. 102, Mar. 2021, Art. no. 102124.

[52] A. Rahman, "Characteristics of defective infrastructure as code scripts in devops," in *Proc. 40th Int. Conf. Soft. Eng.: Companion Proc.*, 2018, pp. 476–479. [Online]. Available: https://doi.org/10.1145/3183440.3183452

Rabimba Karanjai is currently pursuing the Ph.D. degree in computer science with the University of Houston. His research interest spans from confidential computing, blockchain, virtual and augmented reality, high performance computing to safety in large language models, and machine intelligence.

Keshav Kasichainula has graduated from the University of Houston. He is currently working as a Senior Software Engineer with Teradata. His thesis was based on Adversarial attacks on Machine Learning models using Face warping. His area of interest are machine learning, artificial intelligence, blockchain, and databases.

Lei Xu is currently an Assistant Professor with Kent State University. His research interests include blockchain, cloud security, and applied cryptography.

Nour Diallo is currently pursuing the graduate degree in computer science with the University of Houston. He is working under the supervision of Dr. W. Shi with Secure and Intelligent Systems Lab. He has coauthored several papers on these topics in IEEE conferences and journals. His research interests include blockchain and mobile payment.

Lin Chen received the Ph.D. degree from Zhejiang University in 2013. He is an Assistant Professor with the Department of Computer Science, Texas Tech University. Before joining Texas Tech, he was a Postodoctoral Fellow with the Technical University of Munich and the Technical University of Berlin, and then a Research Assistant Professor with the University of Houston. He has published articles in refereed international journals, including *SIAM Journal on Computing*, *Mathematical Programming*, and *ACM Transactions on Algorithms*. His research interests include algorithms and complexity, and operational research.

Weidong Shi (Senior Member, IEEE) is an Associate Professor of Computer Science with the University of Houston. He leads the Secure and Intelligent Systems Lab with UH, where he works on enabling system technologies for services. His research interests include system security, blockchain, cloud computing, and computer architecture. He is also an active member of the IEEE Computer Society and ACM.