DSES: A Blockchain-powered Decentralized Service Eco-System

Zhenfeng Gao, Yushun Fan*, Cheng Wu
Tsinghua National Laboratory for Information Science and Technology
Department of Automation
Tsinghua University
Beijing 100084, China
gzf13@mails.tsinghua.edu.cn
{*fanyus, wuc*}*@tsinghua.edu.cn*

Jia Zhang
Department of Electrical and Computer Engineering
Carnegie Mellon University
Silicon Valley
Moffett Field
CA 94035, USA
jia.zhang@sv.cmu.edu

Chang Chen
Ziggurat Technology
Zhongguancun SOHO 1010
Beijing 100190, China
calvin@ziggurat.cn

***Abstract*—Existing service ecosystems typically rely on some centralized service registries (e.g., ProgrammableWeb.com) as "middle people" to record service behaviors thus to provide service ranking and recommendation. Excessive centralization increasingly becomes the bottleneck and hinders the further growth of the service ecosystems. As the first attempt to apply the fundamental technique underneath the emerging Bitcoin network into the field of service oriented computing, this paper proposes to build a service ecosystem as a decentralized blockchain-oriented service network, called Decentralized Service Eco-System (DSES). Whenever any activity occurs in the system (e.g., APIs are used together in a published mashup), all involved parties will individually store and maintain a copy of the detailed record (provenance) at their local databases. Such a distributed database-oriented solution will enable services who do not fully trust each other to maintain a set of global states. In this way, service discovery and recommendation can be realized in a distributed manner that promises higher scalability and maintainability. As a proof of concept, a prototyping system of DSES is constructed using the real-world data from ProgrammableWeb.com, based on the INKchain, a newly open-source consortium blockchain mechanism extending the Hyperledger Fabric.**

***Keywords*-service ecosystem; blockchain; smart contract; Hyperledger Fabric; INKchain**

I. Introduction

As Service-Oriented Architecture (SOA) and Cloud Computing are widely adopted, the amount of published web services on the Internet has been rapidly growing [1]. Service providers are interconnecting their offerings in unforeseen ways, giving rise to web service ecosystems [2]. By reusing existing services (i.e., APIs), software developers are able to quickly create service compositions (i.e., mashups) to meet complex functional needs and offer additional business values [3]. A Web service ecosystem thus becomes a logical collection of Web services as well as their compositions. For example, ProgrammableWeb.com, consisting of more than 18,000 services and 6,000 mashups as of December 2017, represents by far the largest service ecosystem [4].

Most existing service ecosystems typically rely on some centralized service registries (e.g., ProgrammableWeb.com) as "middle people" to record service behaviors (i.e., service interactions in past activities within the service ecosystems), thus to provide service ranking and recommendation. Such a centralized architecture comes with trust and scalability issues. On the one hand, all records (i.e., information about services, mashups and usage records) are stored at and managed by a centralized server. Only if users trust the centralized registry, they will participate in contributing to it (e.g., provide sound information about services and utilize services to create mashups). On the other hand, people's active participation contributes to the reliability and trustworthiness of the service registry, based on which service discovery, ranking and recommendation could be realized. In reality, however, services provided by different developers in the service ecosystem may not trust each other. Furthermore, existing centralized registry lacks effective incentive mechanisms to encourage service providers and users to participate actively. As a result, excessive centralization increasingly becomes the bottleneck and hinders the further growth of the service ecosystems nowadays.

In recent years, blockchain technologies are taking the world by storm, largely thanks to the success of Bitcoin [5]. A blockchain, also called distributed ledger, is essentially an append-only data structure maintained by a set of nodes that do not fully trust each other. In its original design, Bitcoin's blockchain stores coins as the system states [6]. Since then, the technology has grown beyond crypto-currencies to support user-defined states. Taking Ethereum [7] for example, it enables any decentralized and replicated applications, known as DApps, with the help of smart contracts. Recently, increasingly more industry organizations have made efforts to develop new blockchain platforms where participants are authenticated. Such systems are called permissioned or

*Corresponding Author

2159-6190/18/$31.00 ©2018 IEEE
DOI 10.1109/CLOUD.2018.00011

consortium blockchains. Applications of different domains are being designed and implemented [8]–[10] as well. Particularly, Hyperledger Fabric [11] realizes the identity management mechanism required under enterprise scenarios, and has become a widely-used open-source permissioned blockchain platform aimed at business uses [12].

To address the aforementioned issues of conventional service ecosystems, in this paper, we migrate the idea of blockchain into service ecosystem and introduce the framework of a Decentralized Service Eco-System (DSES). DSES builds a service ecosystem based on a decentralized, service-oriented blockchain network. In order not to reinvent the wheel, DSES leverages the INKchain* as the low-level blockchain solution for two major reasons. First, identification and authority controls are the unique features of permissioned blockchains, which are helpful to allow qualified developers or users to participate in DSES thus promote the quality and reliability of information stored in the system. Second, the systematic characteristic of asset component in INKchain makes it possible to issue customized token "*SToken*" as the media of value transfer and realize the incentive mechanisms in DSES. Be more specific, function blocks are implemented as smart contracts (i.e., chaincodes) on a supervisor level. Whenever any activity occurs in DSES, all involved parties (services and mashups) will individually create a detailed record through the designed interfaces in the chaincode. Illegal operations will be denied and the agreed-upon records will be permanently stored and maintained at the local database of each involved node in the network. Such a distributed-database-oriented solution will enable services who do not fully trust each other to maintain a set of global states. In this way, service discovery and recommendation with higher scalability and maintainability can be realized.

To the best of our knowledge, this paper is the first attempt that designs and implements a decentralized service platform. The contributions of this paper are summarized in three-fold:

1) A new concept of "Decentralized Service Eco-System" (DSES) is created, as the first attempt to utilize the blockchain mechanism to address the problems encountered by conventional centralized service ecosystems.

2) As a proof of concept, the overall framework of DSES is implemented based on INKchain. We present how to build a service ecosystem and utilize its smart contract instrument to realize the basic functions in a service ecosystem and provide incentive mechanisms to maintain the vitality of the platform.

3) As a case study, a prototyping system of DSES is constructed using the real-world data from ProgrammableWeb.com.

The remainder of this paper is organized as follows. Section II introduces background and illustrates our motivation to build a decentralized service platform. The overall framework and detailed design of smart contracts are provided in Section III. In Section IV, the results of implementing a DSES prototype is presented. Section V presents further discussions on the feasible improvements and applications of DSES. Section VI discusses related work and finally Section VII concludes the paper.

*https://github.com/inklabsfoundation/inkchain

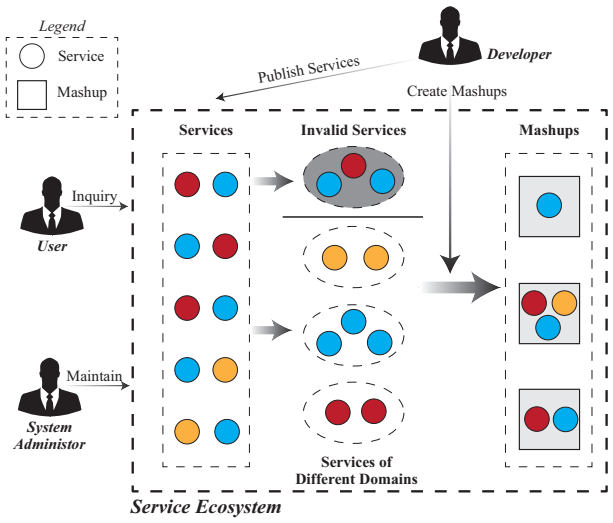

Figure 1. Framework of Conventional Centralized Service Eco-System

II. Background and Motivation

A. Service Eco-System

Service ecosystems are logical collections of Web services (APIs) as well as their compositions [2]. Existing service eco-systems are usually organized in a centralized manner, as shown in Figure 1, with the legend at the top left corner.

In general, there are three kinds of roles that play a significant part in service ecosystems: developer, user, and system administrator. **Developers** (i.e., service providers) register and publish their original services (i.e., APIs), which are the basic components of the service ecosystem. Different services may carry different functions and different application scenarios, reflected by different colors in Figure 1. With the evolvement of the service ecosystem, some services may become invalid, and others gradually form different application domains according to their functions. To meet complex function needs and offer additional business values in a faster development cycle, developers reuse existing services to create mashups (i.e., service compositions). A mashup may invoke one or more services to implement specific functionalities. The second role of a service ecosystem is **User**. Users usually propose inquiries to search for services or mashups in the ecosystem and then directly utilize them under different circumstances. Furthermore, in a centralized service ecosystem, **System Administrators** play

an important role in maintaining the stability, security and prosperity of the service ecosystem. Without their efforts, the service ecosystem could be more vulnerable, and have less people's attention and participation.

More specifically, let us take ProgrammableWeb.com[†] as an example. Since established in 2005, ProgrammableWeb.com has aggregated more than 18,000 services and 6,000 mashups up to December 2017. It is a centralized service ecosystem maintained by the official team of the platform. ProgrammableWeb.com, as the official journal and directory of the API Economy, aims at building a news and information source as well as a community of all kinds of API economy stakeholders. Its maintenance team establishes the website to provide people a portal to publish and search for various services and mashups.

B. Motivation to Decentralization

ProgrammableWeb.com has successfully established a community of all kinds of API economy stakeholders. Developers publish their new services or mashups on it, and users could find out information related to these services or mashups. Moreover, ProgrammableWeb.com updates news about APIs frequently, and sets up "API University" to provide valuable information and guidance for users and developers. However, there exist disadvantages due to the centralization of the system.

1) Trust Issues

Excessive centralization could result in trust problem. In a centralized service registry, services provided by different developers may not trust each other. A developer would play an active part in the service ecosystem (i.e., provides accurate and comprehensive information about services, and utilizes services to create mashups frequently) only if he/she trusts the authority of the centralized service registry. Conventional centralized platforms could not achieve this goal even with a lot of efforts and high cost. Without a service ecosystem containing credible information about all kinds of behaviors, it will hinder the development of service management, discovery and recommendation that promises higher scalability and maintainability.

2) Security Problems & Privacy Control

Centralized service platforms are usually susceptible to attack, making the massive data of services easy to tamper with. That is to say, in a centralized system, if the main servers of the centralized service ecosystems are invaded, the information of all the services and mashups may get falsified, bringing considerable economic losses. Furthermore, it is difficult to realize fine-grained authorization control on a systematic level.

3) Lack of Incentive Mechanisms

In ProgrammableWeb.com, developers publish their services and mashups voluntarily. There are no explicit incentive mechanisms in the existing systems. As a result, some service providers (i.e., developers) might have no motivation to develop new services or mashups actively.

[†]https://www.programmableweb.com/

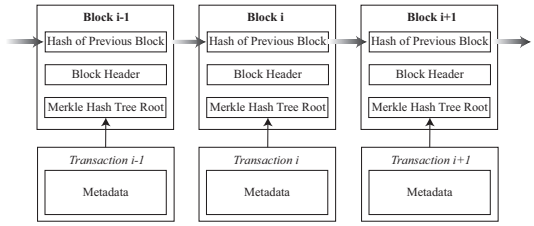

Figure 2. Typical Blockchain Structure

4) High Cost of Maintenance

Centralized platforms usually need administrator roles, as well as high infrastructure and human cost. Offering a community-oriented service platform that many community members will rely on implies spending billions of dollars in innovation research every year [13].

C. Blockchain Technology

A typical blockchain system usually consists of multiple nodes which do not fully trust each other [6]. Together, the nodes maintain a set of shared, global states and perform transactions modifying these states. Figure 2 shows the typical structure of a blockchain, where each block is linked to its predecessor via a cryptographic pointer, all the way to the first (i.e., genesis) block. Due to this structure, blockchain is also called distributed ledger.

A specific transaction in blockchain is a sequence of operations applied to global states. The core difference is that blockchain is decentralized, and each node has its own ledger while needing to reach a consensus in the whole blockchain system. Blockchain systems have advantages in decentralization, security, anonymity and untamperability. Lacking central points of trust or failure, blockchain has enabled a new class of decentralized applications [14].

Blockchain usually makes a trade-off between performance and trust. Generally, blockchain systems can be categorized as *public*, *private* or *permissioned*, described in detail as follows.

1) Public Blockchain

Public blockchains are maintained across peer-to-peer networks in a totally decentralized and anonymous manner [5], [7]. Anyone can join the network freely. Bitcoin [5] is the most well-known example of public blockchains. In order to determine which block to append to the ledger text, peers have to execute Proof-of-Work (PoW) consensus [15]. Ethereum [7] has grown beyond crypto-currencies to support user-defined states and Turing complete state machine models. It enables any decentralized applications (DApps) with the help of smart contracts. Public blockchains have advantages of enabling the ledger to be curated completely anonymously, and peers' willingness to hold a copy of the ledger and try to create new blocks.

2) Private Blockchain

In private blockchains, only specific organizations are allowed to join the networks. They are widely used in building blockchain-based underlying systems between several specific organizations. For example, two banks might negotiate to establish a private blockchain between them to assist with the reconciliation.

3) Permissioned Blockchain

The definition of permissioned blockchains falls in between public blockchains and private ones. Only authorized organizations can join in permissioned systems. Permissioned blockchains require a set of trusted nodes tasked with creating new blocks. Compared with public blockchains, permissioned ones require less resources and are able to reach smaller transaction latency and higher throughput. Moreover, permissioned blockchains make it possible to control the set of participants tasked with maintaining the ledger. This feature increases its popularity among industrial communities. Hyperledger Fabric is an open-source project under the Hyperledger umbrella project[‡], which targets business applications [16], [17].

D. Blockchain Selection in DSES

INK consortium blockchain (INKchain), which aims at providing a trusted consortium blockchain for regional use cases, extends Fabric by providing asset accounts, as well as supporting asset transfer and other enhancements. We chose INKchain as the underlying blockchain solution for two major reasons.

1) INKchain is inspired by Fabric, both of which are permissioned blockchains supporting identification and authority controls. With INKchain, it is possible to allow only qualified developers or users to participate in DSES and avoid hostile behaviors. As a result, DSES could focus more on information maintenance instead of worrying about malicious behaviors.

2) The systematic characteristic of asset component in INKchain makes it possible to issue customized token "*SToken*" in DSES as the media of value transfer. Furthermore, with systematic interfaces provided by INKchain, incentive mechanisms in DSES can be realized through chaincode.

III. Decentralized Service Eco-System (DSES)

A. Scheme Overview

Figure 3 depicts the overall framework of DSES. Simply speaking, after receiving certification from Certificate Authority (CA), a developer/user can utilize its Software Development Kit (SDK)/APP to make invocation to update or query states through chaincodes. In actual application scenarios, after reaching agreements on the business logic between entities in the same blockchain network, it is possible to code the business logic into chaincodes to make sure that everyone obeys it. Any illegal operations (i.e., not defined in chaincodes or beyond the scope of authority) will have no effect on the global states. Peers and orderers form the Blockchain Network. Such a distributed database-oriented solution will enable peers who do not fully trust each other to maintain a set of global states. All the peers hold the full history of all activities since the service ecosystem was established, assuring the reliability and authority of the information. DSES will thus provides reliable and sound information for researches on service management, discovery and recommendation that promises high scalability and maintainability. Detailed explanations of the components are discussed as follows:

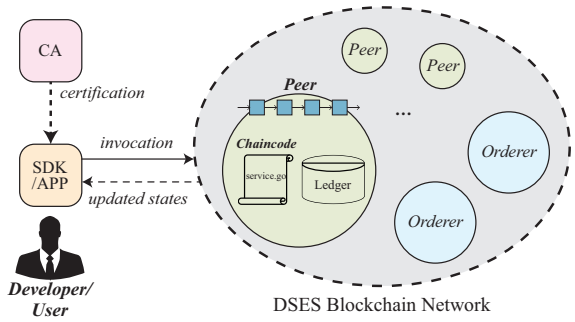

Figure 3. Overall Framework of Decentralized Service Eco-System

Developer/User: In DSES, there are only two types of participating roles (i.e., Developer and User). Developers or users could make several kinds of invocations (e.g., publish a new service, query service information, or create a new mashup), and broadcast them into the DSES blockchain network with the help of the SDK/APP component.

SDK/APP: The SDK/APP component implements business logic of the service ecosystem through provided interfaces in smart contracts. The app could run on any blockchain node, or be deployed on a centralized service. SDK/APP component is the application front-end part in DSES. Developers or users interact with it, and operations that will affect the states will generate a transaction through interfaces defined in the chaincode. The operations from the SDK/APP aspect are similar with those in centralized service repositories, bringing no burden for Developers or Users.

CA: The CA component could provide legal identity certifications to authorize developers or users to propose invocations through the SDK/APP component.

DSES Blockchain Network: The right-hand side of Figure 3 illustrates the structure of DSES blockchain network, which is the core component of DSES. DSES blockchain network is a distributed management system of transaction records in the scheme. All the developers and users contribute blockchain operations. Blockchain network based on INKchain distinguishes between two kinds of nodes: **Peers** and **Orderers** [12]. **Peer** is a kind of non-validating

[‡]https://www.hyperledger.org

node that functions as a proxy to connect clients (issuing transactions) to validating peers (Orders). It does not make real executions of transactions but may execute simulations and verify them. Chaincode links the blockchain network with the outside, providing interfaces for invocation, and intending to record data in a distributed Ledger. The data to be recorded is called "state," stored in a key-value form. An **Orderer** is a node on the network responsible for running consensus, validating transactions, and maintaining the ledger. In a practical application, the orderers and peers could be run by different organizations. No central server managed by a trusted third party is required. Thus there is no need for System Administrator role to provide maintenance.

B. Blockchain Perspective

From a blockchain perspective, we scrutinize a transaction process. After being authorized, the user can send a transaction proposal to a peer through SDK/APP. The peer checks the proposal, simulates to conduct the transaction, and endorses the results. Then the endorsed proposal will be broadcast to orderers. The orderers validate the transactions and create new legal blocks. At last, the peers update their local ledgers according to the newly-received blocks. Any operation beyond the interfaces declared in chaincode or the authority setting is considered illegal, which will have no effect on the global states.

C. Security Requirements

DSES blockchain network is a decentralized system. Due to its industrial application scenario and derivation from Fabric, DSES supports crash tolerance through an ordering service based on Apache Kafka[§] to reach a consensus.

INKchain substantially integrates current technological advancement in the fields of distributed computing and security. INKchain takes advantage of cryptographic primitives such as Hash Function [18], Asymmetric Encryption [19] and Digital Signature [20]. As mentioned earlier, transaction proposals will be validated and endorsed during the transaction process. All such instruments ensure the reliability and security of the network.

D. Authority Control

Public blockchain platforms usually suffer from lack of permission control, and the information is completely exposed to the public. Permissioned blockchain is designed for enterprise scenarios, equipped with methods to realize authority control. Based on INKchain, DSES realizes it in three aspects. Firstly, the CA component implements the PKI service that can issue identify certification in advance, and distributes it to corresponding entities. Secondly, the INKchain could control different entities' access level of data and resources through fine-grained policy control. Last but not least, with the help of INK Account, we design

[§]https://kafka.apache.org/

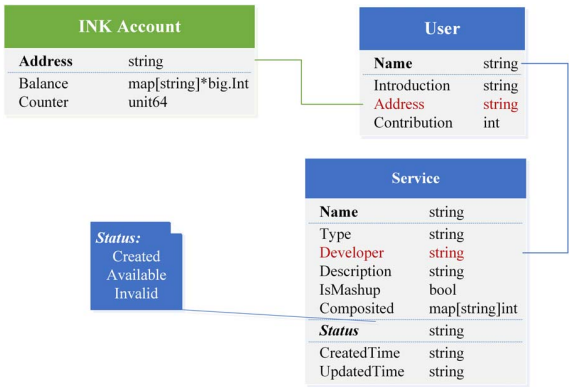

Figure 4. Design of Data Structure in Chaincode "service.go"

chaincode *"service.go"* to implement data access control functionally.

E. Design of Chaincode

In Fabric and INKchain, "chaincode" is the implementation of smart contract, spearheaded by Ethereum [7]. Chaincodes intend to make response to the transactions sent by SDK/APP, execute related codes and finally record data in a distributed ledger. We have made a deep investigation on the functionalities of service ecosystems, and designed a new chaincode *"service.go"* to implement related business logic. Moreover, we have added incentive mechanisms to stimulate people to participate more actively in making contributions to the service ecosystem. Details of the chaincode will be presented in the following sections.

1) Data Structure

The data structure in *service.go* is designed as shown in Figure 4 with the following core elements.

INK Account: INKchain has designed and implemented an account system called "INK Account," which can cater to a large number of anonymous users to manage digital assets and interact directly with the blockchain. As shown in Figure 4, the field **Address** uniquely identifies a record of INK Account. The **Balance** field records all kinds of tokens in an account. **Counter** is used for validation.

User: In the design of data structure, we do not distinguish Developers from Users mentioned in Figure 3. The field **Name** is the keyword of User structure. A user also needs to provide his/her brief introduction. Each user is the one-to-one correspondence of an INK Account through the **Address** field. The **Contribution** field is designed for recording the degree of a user's contribution to the service ecosystem.

Services: Structurally, we design the Service to record information of services as well as mashups. There exists a one-to-one correspondence between a service and a specific user's Name through the **Developer** field. We utilize a boolean field **IsMashup** to distinguish whether the service is a mashup or not. If it is a mashup, then the **Composited** field

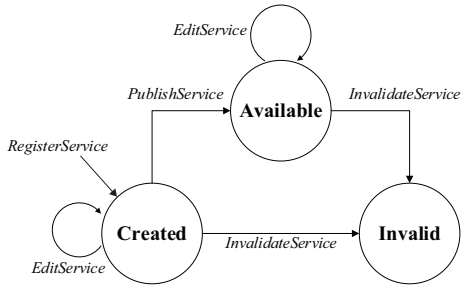

Figure 5. Status Change Logic of A Service

would record the invoked services (i.e., APIs). Specifically, we design a string field **Status** to record the current status of a service. Generally, we define three kinds of status about a service: *(a) Created:* a service is created but not published yet; *(b) Available:* a created service has been published; and *(c) Invalid:* an existing service has withered away.

2) Functional Implementation

Based on the definitions of data structures, we implement the invoke functions as listed in Table. I. Related codes are provided in our github repository¶.

Table I
IMPLEMENTED FUNCTIONS IN CHAINCODE

Function	Description
RegisterUser	register a new user
RemoveUser	remove an existing user
QueryUser	query info about a specific user
RegisterService	register a new service (not mashup)
PublishService	publish a newly created service
InvalidateService	invalidate a specific service
CreateMashup	create a new mashup, compositing services
QueryService	query info about a specific service or mashup
EditService	edit info of a specific service or mashup
QueryServiceByUser	query services developed by one specific user
QueryServiceByRange	query services by a range
RewardService	reward a service's developer by another user

Generally, the invoke functions are divided into three categories: user-related functions, service-related functions, and reward-related functions. Leveraging INK Account, we are able to recognize an invoker's identity with help of chaincode stub interfaces such as **Transfer**, **GetAccount** and **GetSender**. Moreover, based on them, DSES has realized incentive mechanisms. To better explain, we will first explain how we design the functions in the chaincode, through the logic of services' status change as well as the design of *CreateMashup* function.

3) The Logic of Services' Status Change

As mentioned before, a service in DSES has three different status. Figure 5 illustrates how a service's status changes with different service-related functions.

4) Introduction of CreateMashup

¶https://github.com/gzf09/DSES

In this section, we present the detailed design of function *CreateMashup*, which is an important invoke function to make contribution to the development and prosperity of the service ecosystem. With the help of interfaces like **GetSender**, **Transfer** in INKchain, the overall process of creating a new mashup in *service.go* is illustrated as follows.

Process 1: CreateMashup

Input: Mashup information and list of invoked services
Output: Transaction record on the ledger
Procedure:
01. Get the mashup's creator's address through **GetSender**
02. Create a mashup record according to Service structure, where:
03. a) Field IsMashup is set true
04. b) Field Composited records all the invoked services
05. **For** every invoked service
06. Pay to its developer for providing related functionalities through **Transfer**(*to*, *"SToken"*, *amount*) [1]
07. **End**
08. Store the newly-created service into the ledger

[1] *to* is the address of the service's developer, *amount* determines how much token *"SToken"* is paid to the service's developer.

5) Incentive Mechanisms

Incentive mechanisms are designed from three aspects.

a) Contribution Record

In the User data structure, we use **Contribution** field to record the degree of a user's contribution to the service ecosystem. The value of contribution changes once the user publishes a new service or creates a mashup. The contribution of a user is calculated as follows:

$$contribution_i = \ln(n_s + n_m + 1) + \lambda \cdot \frac{n_i}{n_{si}} \qquad (1)$$

where $\ln(n_s+n_m+1)$ represents the evaluation of quantity. n_s is the total number of available services the user has developed, and n_m is the total number of mashups. To ensure $contribution_i > 0$, we add one to the total number of available services and mashups. Meanwhile, $\lambda \cdot \frac{n_i}{n_{si}}$ represents the evaluation of quality (i.e., whether a service is invoked more frequently by other developers), where n_{si} is the total number of invoked services developed by user i, and n_i is the total times that this user's services have been invoked by mashups. Here we introduce coefficient λ to adjust the preference of users' contribution in creating mashups. We use the logarithm function for quantity evaluation and liner for quality evaluation, because we pay more attention to developers' behavior of utilizing services to create mashups.

b) Token Transfer when Creating Mashups

Due to the specific feature of INKchain, we are able to introduce a customized token *"SToken"* (abbreviation for "Service Token") to realize related incentive mechanisms.

As mentioned in step 6 of Process 1, when creating a new mashup, the developer of the mashup has to transfer a fixed amount of *"SToken"* to each invoked service's original

developer. Developers thus are motivated to create more general and more valuable services to the service ecosystem.

c) Reward from Users

The invoke function *RewardService* is also part of the incentive system in DSES. A user can reward the developer of a service or a mashup with any amount of *SToken* as he/she likes. Through the interface **Transfer**, it is feasible to realize this capability based on INKchain.

IV. Prototyping System, Experiments, and Discussions

In this section, we describe the implementation of our prototype of DSES in a case study, along with performance evaluation results. Original test scripts are provided in the github repository.

A. Data Set

ProgrammableWeb.com has been accumulating a variety of services and mashups since established in 2005. We crawled the data from its inception to December 2017, including over 18,000 services and 6,000 mashups.

B. Implementation

We used INKchain Version 0.11.0 as the basic blockchain library and implemented our design of DSES. Our implementation environment, cryptographic algorithms and basic network settings are listed in Table II.

Table II
Implementation Environment & Network Settings

PC	2.6 GHz Intel Core i5, 8 GB Memory
Language	Go 1.9.2 linux/amd64
Containerization	Docker 1.35
Digital Signature Alg.	ECDSA (256bit key)
Hash Function	SHA-256
# of Peers	4
# of Organizations	2 (2 Peers in each Organization)
# of Orderers	1

As the first step to verify the feasibility and validity of our design, we implemented a DSES prototype in an experimental environment. All peers, orderers and client programs are containerized in Docker containers. The peers join a consensus protocol of the blockchain. Specifically, in the prototype, we focused on the implementation of basic business logics in a service ecosystem without a concrete front-end.

The designed functionalities and incentive mechanisms have passed the basic functional tests, with related test scripts in *test_service_init.sh*.

C. Performance Analysis

1) Requirements of Invoke Throughput

Take Programmable.com for example. Since established in 2005, the average daily service-publishing rate is about 4.10. The peak value of daily service-publishing is about 110, and the peak value of daily mashup-publishing is around 40. They are far less than the capacity of Fabric [6]. Above all, operations around a service ecosystem is usually not of high-frequency. Forked from Fabric, INKchain has made progress in enhancing the throughput by creating transfer set based on the read-write set in Fabric. Even if considering other invoke operations like creating new users or rewarding outstanding services, in general, the current performance of INKchain could meet the requirements of DSES.

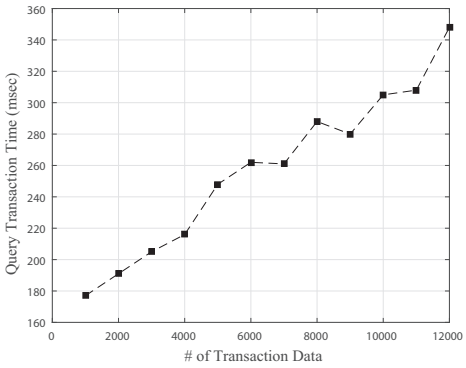

Figure 6. Processing Time for Query Invocation with Different Numbers of Transaction Data

2) Analysis of Query Performance

We conducted a number of experiments to test the influence of data size on query performance. We invoked *registerService* function to store different numbers of transaction data in the DSES blockchain network, and then made query invocation and recorded the processing time. We made the number of transaction data vary from 1,000 to 12,000, and recorded the average processing time of five queries using the *queryService* function. The results are summarized in Figure 6. In summary, the network query performance of DSES is feasible for practical use.

D. Further Improvements on DSES

More detailed design could be implemented to realize more complicated functionalities. A more complicated commercial production environment remains to be established to proceed to conduct more intensive tests about functionalities, performance and the effectiveness of incentive mechanisms. Peers and orderers could be deployed in VM machines on cloud platforms, managed with Kubernetes. Furthermore, the idea could be applied in other industrial scenarios such as tourism, building a decentralized tourism service ecosystem. Service providers publish all kinds of tourist services such as restaurants, hotels, tourist guides and so on. Users would pay specific token in order to use the services.

V. Related Work

A service ecosystem is a logical collection of web services as well as their compositions [1]–[3]. Existing works mainly

focus on systemic management or studying related data to make analyses and recommendation [21]. However, existing service ecosystems typically rely on some centralized service registries as "middle people" to record service behaviors, which could result in problems such as trust issues, security, authority control, and high cost of maintenance.

Bitcoin [5] is the first successful attempt to construct a commercialized decentralized system. Since then, blockchain technology has grown beyond crypto-currencies to support real applications. Ethereum [7], the first one realized smart contract, enables people to develop all kinds of DApps. In industry scenarios, Hyperledger Fabric [11], [16], [17] is the most popular solution among companies. As an extension of Fabric, INKchain is a newly open-source permissioned blockchain platform, enabling account systems and customized token issuance. Permissioned blockchains have advantages in authority control and higher performance [16]. Thus they have been widely used to develop decentralized applications [12], [22]. In this paper, as the first attempt to design a decentralized service ecosystem, we have built the overall framework DSES on top of INKchain. We realized functionalities of conventional service ecosystems, and designed incentive methods to maintain the sustainable development of service ecosystems.

VI. Conclusions

In this paper, as the first attempt to bring decentralization to service ecosystems, we have described our design and development of the Decentralized Service Eco-System (DSES) based on a newly open-source consortium blockchain called INKchain. We have presented a prototype of DSES to demonstrate the feasibility and effectiveness of our design. Related experiments on performance have also been conducted based on a real-world data set.

In our future work, we will further design the details of the DSES toward a real-world production environment. In addition, more comprehensive experiments about the performance of DSES in a production environment will be conducted. Especially, we will study the cost incurred by such a distributed storage manner as well as each party's willingness to adopt such a kind of distributed storage. Moreover, we will tackle the data integration challenge when combining distributed data for unified calculation in DSES.

Acknowledgment

This research has been partially supported by the National Nature Science Foundation of China (No.61673230).

References

[1] V. Andrikopoulos, S. Benbernou, and M. P. Papazoglou, "On the Evolution of Services," *IEEE Transactions on Software Engineering*, vol. 38, no. 3, pp. 609–628, 2012.

[2] D. M. Barros A, "The Rise of Web Service Ecosystems," *It Professional*, vol. 8, no. 5, pp. 31–37, 2006.

[3] X. Liu, Y. Hui, W. Sun, and H. Liang, "Towards Service Composition based on Mashup," in *Proceedings of IEEE World Congress on Services (SERVICES)*. IEEE, 2007, pp. 332–339.

[4] T. W. Huang K, Fan Y, "An Empirical Study of Programmable Web: A Network Analysis on a Service-Mashup System," in *Proceedings of International Conference on Web Services (ICWS)*. IEEE, 2012, pp. 552–559.

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Consulted*, 2008.

[6] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," *arXiv preprint arXiv:1708.05665*, 2017.

[7] Ethereum, "Ethereum," in *https://www.ethereum.org/*.

[8] Ripple, "Ripple," in *https://ripple.com*.

[9] Melonport, "Blockchain Software for Asset Management," in *http://melonport.com*.

[10] J. Morgan and O. Wyman, "Unlocking Economic Advantage with Blockchain," 2016.

[11] Hyperledger, "Hyperledger Fabric," in *https://www.hyperledger.org/projects/fabric*.

[12] S. A., "Blockchain Based Distributed Control System for Edge Computing," in *Proceedings of International Conference on Control Systems and Computer Science*. IEEE, 2017, pp. 667–671.

[13] AnantJhingran, "How and Why to Transform Your Business into a Digital Ecosystem," in *https://www.programmableweb.com/news/how-and-why-to-transform-your-business-digital-ecosystem/analysis/2018/01/11*.

[14] e. a. Ali, Muneeb, "Blockstack: A Global Naming and Storage System Secured by Blockchains," in *Proceedings of USENIX Annual Technical Conference (USENIX ATC)*. USENIX, 2016, pp. 181–194.

[15] A. K. Garay, Juan A. and N. Leonardos, "The Bitcoin Backbone Protocol: Analysis and Applications," in *Proceedings of the 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2015, pp. 281–310.

[16] M. Vukoli, "Rethinking Permissioned Blockchains," in *Proceedings of ACM Workshop on Blockchain, Cryptocurrencies and Contracts (ACM)*, 2017, pp. 3–7.

[17] C. Cachin, "Architecture of the Hyperledger Blockchain Fabric," in $https://www.zurich.ibm.com/dccl/papers/cachin_accl.pdf$, 2016.

[18] X. Yi, "Hash function based on chaotic tent maps," *IEEE Transactions on Circuits & Systems II Express Briefs*, vol. 52, no. 6, pp. 354–357, 2005.

[19] M. Bellare and P. Rogaway, "Optimal asymmetric encryption," in *Proceedings of The Workshop on the Theory and Application of of Cryptographic Techniques Springer, Berlin, Heidelberg*, 1994, pp. 92–111.

[20] R. C. Merkle, "A Certified Digital Signature," in *Proceedings of Advances in Cryptology - CRYPTO '89, International Cryptology Conference, Santa Barbara, California, USA*, 1989, pp. 218–238.

[21] Z. Gao, Y. Fan, C. Wu, W. Tan, and J. Zhang, "Service Recommendation from the Evolution of Composition Patterns," in *Proceedings of IEEE International Conference on Services Computing*, 2017, pp. 108–115.

[22] S. Kiyomoto, M. S. Rahman, and A. Basu, "On blockchain-based anonymized dataset distribution platform," in *IEEE International Conference on Software Engineering Research, Management and Applications*, 2017, pp. 85–92.