Trustless Intermediation in Blockchain-based Decentralized Service Marketplaces

Markus Klems, Jacob Eberhardt, Stefan Tai, Steffen Härtlein, Simon Buchholz, and Ahmed Tidjani

Information Systems Engineering (ISE)
TU Berlin, Germany
{mk,je,st}@ise.tu-berlin.de
{haertlein, simon.buchholz, a.tidjani}@campus.tu-berlin.de

Abstract. Service marketplaces promise an open platform for sellers and buyers of IT services. The marketplace design usually assumes that market functions, such as match-making, transaction settlement, and dispute resolution are performed by intermediaries in a centralized system. We propose the concept of trustless intermediation to enable new forms of decentralized service marketplaces. By leveraging blockchain-enabled smart contracts we eliminate the need for trust in marketplace intermediaries and reduce barriers of entry, lock-in, and transaction costs, by removing now obsolete trust-establishing mechanisms. `Desema`, our decentralized service marketplace prototype, is a first implementation of this concept that is based on the Ethereum blockchain in combination with IPFS, a peer-to-peer distributed file system.

1 Introduction

A service marketplace enables IT service providers to sell software services and service consumers to discover and use services [14, 18]. Despite high hopes, service marketplaces have, so far, not been successful on a larger scale. Systems like the Universal Description, Discovery, and Integration (UDDI) registry have never attracted a critical mass [12]. Deficiencies in current service markets manifest in market barriers, low competition, insufficient service substitutability, insufficient service information, and high transaction costs [15]. A substantial body of work has addressed problems of insufficient service information by offering techniques to improve service descriptions (including service level agreements) and service discovery from a service-oriented computing perspective. Problems related to marketplace pricing strategy and incentive structures for marketplace participation, however, remain.

In this paper, we focus on the role of trusted intermediaries in service marketplaces. Centralized marketplaces often lead to deficiencies in the form of lock-in effects and market barriers [7, 15]. Dependency on intermediaries can lead to disadvantages for buyers and sellers if their objectives do not align with those of the intermediaries [4, 6, 8, 10].

We propose a concept for a decentralized and trustless service marketplace which is not provided and governed by a single trusted party, but instead by a community of individuals that participate in the marketplace. The design and prototypical implementation of our marketplace system is based on distributed systems technologies that enable decentralization, in particular on a blockchain that can execute smart contracts. We introduce the concept of trustless intermediation in the context of service marketplaces and describe how service discovery, transaction settlement, and dispute resolution can be realized without a trusted third party. This design could potentially overcome fundamental problems, such as lock-in, of traditional centralized service marketplaces. With our approach, we offer an open system that evolves through the combined effort of a community instead of the strategy of a single organization. Barriers of entry due to sole reliance on buyer/seller reputation, as described in [7, 15], can be reduced by creating a trustless system in which no reputation system is needed to establish trust between sellers and buyers. Our system prototype contributes to the design and development of future blockchain-based decentralized (service) marketplaces and helps to identify relevant technical and non-technical challenges.

2 Background & Related Work

A service marketplace is an online marketplace where suppliers can sell software services [14]. Service consumers can buy and use these software services for composing higher-level services and for building applications [18].

Centralized marketplaces provide mechanisms to facilitate efficient spot trades between large numbers of sellers and buyers by providing match-making and payment transaction processes that are accompanied by trust-building mechanisms, most importantly, reputation and dispute resolution systems. However, relying on reputation as a trust building factor alone can lead to entry barriers for new sellers from whom buyers are less inclined to purchase [7, 15]. A main problem of current online dispute resolution systems is enforcement, which is particularly challenging in cross-border e-commerce where no standardized global legal instruments for enforcing contracts exist [11]. Besides a marketplace provider, **trusted intermediaries** might offer additional market functions, in particular match-making (service and price discovery), transaction settlement, and legal/regulatory functions [9]. Reliance on trusted intermediaries can be problematic for buyers and sellers. Trust can be exploited or betrayed, for example, if a match-making intermediary can obtain higher revenues by matching certain buyers with certain sellers [4, 8, 10], if the marketplace provider forces sellers to vertically integrate with its technology platform [6], or in extreme cases if law enforcement take-down or "exit scams" terminate the existence of a marketplace [16].

The development of **decentralized marketplaces** is driven by a desire to establish systems without a central marketplace provider and without trusted intermediaries. Motivations include reducing barriers of entry [7, 15], reducing fees [2], increasing resistance to shut-down [13], and improving privacy [17].

There are a few initiatives for building decentralized anonymous marketplaces based on blockchain technologies, such as OpenBazaar, Beaver, and Ties Network. OpenBazaar [2] is a decentralized marketplace for trading goods, using multiple crypto-currencies for payment and settlements. Transactions are performed directly between peers without the involvement of a trusted intermediary. The objective of OpenBazaar is to create a free marketplace without fees and no possibility of censorship. Beaver [17] has a reputation system which is Sybil attack resistant and keeps the anonymity of reviewers without relying on a trusted third party. Another approach for designing decentralized marketplaces, described in the Ties Network Whitepaper [3], is to designate special roles to users, e.g., for resolving conflicts and to make anonymity optional in favor of more traditional reputation based trust-building approaches.

3 Trustless Intermediation

As a new idea, we introduce the concept of trustless intermediation which replaces traditional trusted intermediaries, such as centralized service registries and payment providers. We define **trustlessness** as a system property which guarantees rules of interaction that are known to and agreed upon by all participants of the system, and which cannot be unilaterally changed. These guarantees are enforced through, what we call **trustless intermediation**, a set of mechanisms for *decentralizing the enforcement of rules in a system*, thereby removing the need for and existence of trusted intermediaries.

In the following, we propose two main mechanisms for implementing trustless intermediation in a system: through (1) a set of smart contracts, and (2) supporting actors. By using smart contracts, the rules of interaction between sellers and buyers are transparent and self-enforced. If any participant deviates from the rules, the consequential actions, e.g., compensation and punishment, are also known and automatically enforced through smart contracts. Supporting actors provide functions that exceed the capabilities of smart contracts, yet, need to be carefully selected and incentivized, to avoid that they neglect or abuse the marketplace function that they have been assigned.

Based on this concept, we describe functions of a decentralized service marketplace from the perspectives of both service providers (sellers) and service consumers (buyers). We show how service registry, transaction settlement, and service delivery features can be realized with smart contracts and supporting actors in lieu of trusted intermediaries. Figure 1 illustrates the roles and functions in a blockchain-based decentralized service marketplace for a service lifecycle.

Service Registry. A service registry enables basic match-making between service providers and consumers. Service providers can publish their service descriptions to a service registry and service consumers can discover services they need. We realize the service registry with a smart contract that contains references to service providers and service description documents. For this purpose, the service registry contract needs to maintain state of the provider-to-service mapping, allow providers to publish and update service descriptions, and enable

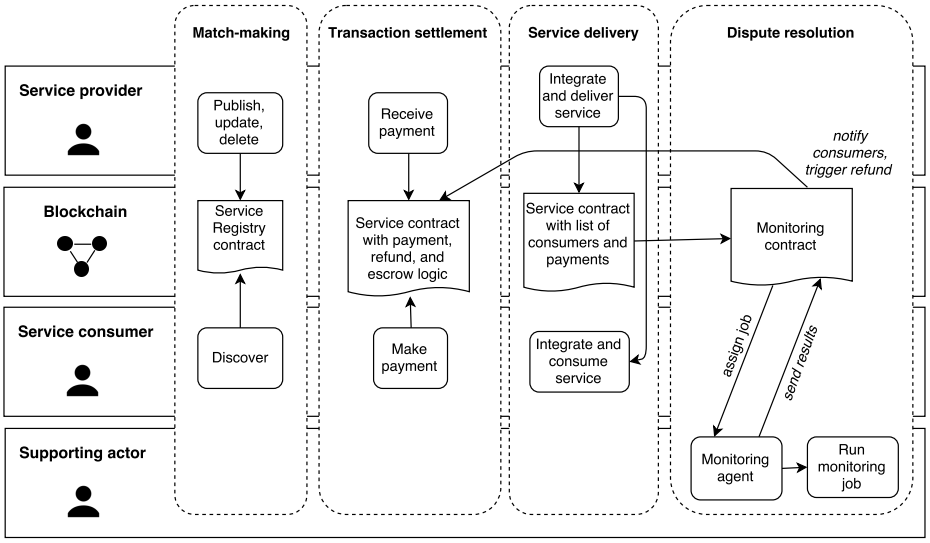

Fig. 1. Roles and functions in a blockchain-based decentralized service marketplace.

consumers to find services they need. For decentralized service discovery, service description documents need to be replicated across multiple nodes in the network. To avoid trust, each consumer should receive and maintain an up-to-date replica of the entire service catalog. In case of updates or shutdowns of services, the impact should be as small as possible for the service consumers. Since providers can remove services from the service catalog and with that prevent purchases, it is desirable to support a reasonable phase-out process. One approach to incentivize timely announcements of service interface and usage changes is to collect a deposit from service providers on service registration. A smart contract serves as an escrow and only refunds the deposit if a version change was announced by sending a notification to the smart contract with a certain leadtime before service removal. If the announcement has not been made (within the agreed-upon leadtime), the deposit is paid out to the service consumers. A similar approach could be applied to breaking changes during updates.

Transaction Settlement. For each service on the marketplace, a service contract is deployed. This contract contains the business logic for payments and refunds. In order to consume a service, a user invokes the service contract to make a payment in a virtual currency and in the same transaction adds her authentication information to the contract, which is later needed for authenticating the consumer's service requests.

Different payment models can be encoded in a service contract, such as:

- *Time slices:* Once a consumer has selected a service, she subscribes to the service by paying a fee to gain access to the service for a certain time.
- *Utility computing:* A service consumer pays for a certain type of workload, such as the number of requests, the operation types, the payload size, etc.
- *Subscription:* A service consumer pays a subscription fee that gives access to a service up to a certain workload limit.

Service Delivery. The service providers needs to distinguish paid-for service requests from unpaid service requests. We consider two approaches:

- *Proxy Service:* Requests are not sent to the target service directly, but to a proxy verifying the sender's authority before forwarding the request to the designated target.
- *Signature Library:* Sender and receiver both use a library to sign outgoing messages and validate incoming messages.

The Proxy Service approach simplifies service integration for both consumer and provider who do not have to take care of message integrity and caller authorization. On the downside, a provider-side Proxy Service would be a single point of failure for all services using it, and a Proxy Service that is powered by a supporting actor would require trusting a third party.

Using a Signature Library, consumer and provider can directly integrate functions for signing, and signature verification, respectively. Both parties are thereby enabled to freely choose which endpoints require authorization and how to handle errors, such as insufficient funds. On the downside, this solution shifts integration effort to consumer and provider.

Following our main objective of trustless intermediation, we prefer the Signatur Library approach, which consists of the following three steps:

1. The consumer signs the payload of service requests. Along with the payload, the signature and public key are sent in the request header.
2. Signature and public key are used by the receiving service to verify that the message body has not been altered.
3. The service verifies that the address belongs to a paying service consumer, processes the request, and sends a success or error response back.

This process requires consumer-side and provider-side service integration. Step 1 requires a signature library that must be used by the consumer to sign all requests that invoke services which have been purchased on the marketplace. Furthermore, a consumer might want to automate the process of making service payments to avoid request errors due to lack of funds. The provider needs to integrate a signature verification library to perform steps 2 and 3.

Dispute Prevention and Resolution. Disputes between provider and consumer can occur, e.g., if a consumer has paid for a service that is frequently unavailable. We identified the following approaches to prevent and resolve disputes: micro-payments, escrows, and escrows with supporting actors.

Micro-payments. One approach to dispute prevention is to allow service consumers to frequently buy short time slices or small units of service access. This limits the consumer's monetary loss in case of a service unavailability. However, there is no direct punishment for the provider's unavailability. As a disadvantage, a service consumer must continuously add deposits to the service contract.

Escrows. A service contract can contain an escrow mechanism. Service providers, for example, could be required to make a deposit to their service contract before offering a service. If a certain share of service consumers report dissatisfaction, the provider would lose that deposit and the escrow contract

would use it to compensate consumers. This approach further eliminates trust that consumers would otherwise need when purchasing a provider's service, but needs careful incentive design as sybil and collusion attacks need to be prevented.

Escrows with supporting actors. A supporting actor could serve as a monitoring agent who periodically checks service availability and stores the monitoring results in a smart contract (monitoring contract). These results can then be used to resolve disputes and compensate consumers, e.g., refund a consumer's payment or force the provider to pay a fine in case of service unavailability. Here, the smart contract acts as an escrow and requires a provider deposit so that the fine payment is guaranteed. To incentivize participation in the marketplace, supporting actors need to be rewarded, e.g., by paying them a reasonable fee.

4 Decentralized Marketplace System

In the following, we describe `Desema`, our **de**centralized **se**rvice **ma**rketplace prototype, which is available as open source software on Github [1]. `Desema` is a peer-to-peer system which connects service providers and consumers through a shared public blockchain network, Ethereum, and a distributed data storage system, IPFS. Figure 2 shows the high level system architecture with two `Desema` clients. The rich client offers marketplace users a web-based graphical user interface. On the left side is a service provider, Bob, who wants to sell API access to his service. For this purpose, Bob uses a local `Desema` client on his computer to register and publish his service. On the right side of figure 2 is a service consumer, Alice, who accesses `Desema` through her own local client. Alice finds Bob's service in the service catalog, decides to consume the service, and agrees to deposit a payment. Her purchase is facilitated through the service contract. After the purchase, Alice integrates Bob's service into her application. Her application invokes Bob's service with a signed request. For signing a request, the private key of Alice's user account is used. Afterwards, the request body is hashed and the hash is signed. The public key belonging to the private key used is added to the returned signature object, both of which are added to the request header. By calculating Alice's address from her public key and comparing it to his list of paying consumers, Bob's service can identify Alice as a paying consumer and verify request message integrity using a signature library.

Trustless Distributed Data Storage. Business processes in `Desema` are managed on the blockchain. Storing service metadata and other larger data object on the blockchain would, however, be inefficient and expensive. As a solution to this problem, we introduce an approach for trustless distributed data storage by which only data references are stored on-chain. Instead of using an arbitrary name as an identifier, the identifier is computed from the off-chain stored data itself. Off-chain data changes would immediately change the on-chain identifier and invalidate the reference. Furthermore, data integrity can be checked at any time by computing the identifier from the original data in a smart contract and comparing it to the reference. For off-chaining `Desema` service metadata, we use the InterPlanetary File System (IPFS) [5], a public peer-to-

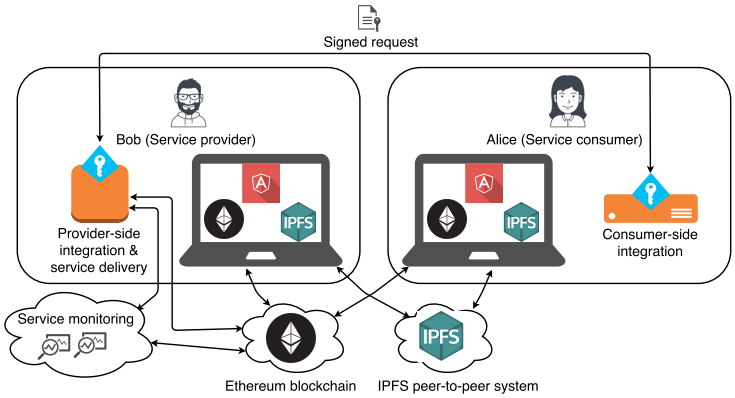

Fig. 2. `Desema` system architecture.

peer file system which addresses files by their hashes. IPFS peers host their own files as well as copies of other's files to ensure availability. Service metadata is stored off-chain in an IPFS directory structure which contains all service versions in separate sub-folders, and only a file reference address is stored in Ethereum. Using IPNS (InterPlanetary Name Space), we can also support mutable content at a fixed address. As IPNS requires cryptographic authorization by the service owner, trustlessness is not impacted.

Trust-limited Monitoring. Monitoring is performed by supporting actors. We need to ensure that those actors do not compromise the integrity of the marketplace by returning inaccurate monitoring results. Trust in monitoring agents can be limited by randomizing the assignment of monitoring jobs to agents. The assignment is performed by the monitoring contract because otherwise a trusted third-party would be needed. Since the Ethereum Virtual Machine (EVM) cannot generate random numbers, we let the monitoring contract generate pseudo-random numbers that are difficult to predict. As an extension to random assignment, multiple nodes could be assigned monitoring jobs for the same service, whereby monitoring results can be determined through a quorum consensus, thereby further limiting trust in individual monitoring agents.

5 Conclusion

In this paper, we introduce the concept of trustless intermediation in service marketplaces based on blockchain technology and discuss approaches to overcome fundamental problems of traditional marketplace systems, such as barriers of entry, transaction costs and lock-in. We propose a design in which trusted intermediaries that operate a marketplace can be replaced with a set of rules encoded in smart contracts and enforced trustlessly in a blockchain network. As a proof-of-concept, we present a prototypical implementation of the aforementioned concepts. Based on the experience that we gained by building the prototype, we identify decentralized application engineering challenges. In particular, we address limitations of on-chain storage and propose a solution for

trustless and scalable distributed data storage. Open challenges include the design and development of more advanced and incentive-aligned approaches for trustless dispute resolution between service providers and consumers.

Acknowledgements. We thank our students Christian Kniep, Nikola Stavrevski, Ravish Aggarwal, and Xiaonan Qiao who contributed to the prototype development as part of a student project offered by the Information Systems Engineering (ISE) research group in the winter term of 2016/17.

References

1. Desema. https://github.com/markusklems/desema, accessed: 2017-06-02
2. OpenBazaar. https://www.openbazaar.org/, accessed: 2017-05-15
3. Ties Network. https://ties.network/, accessed: 2017-08-01
4. Armstrong, M., Zhou, J.: Paying for prominence. The Economic Journal 121 (2011)
5. Benet, J.: IPFS - content addressed, versioned, P2P file system. CoRR abs/1407.3561 (2014), http://arxiv.org/abs/1407.3561
6. Cornière, A., Taylor, G.: Integration and search engine bias. The RAND Journal of Economics 45(3), 576–597 (2014)
7. Einav, L., Farronato, C., Levin, J.: Peer-to-peer markets. Annual Review of Economics 8, 615–635 (2016)
8. Eliaz, K., Spiegler, R.: A simple model of search engine pricing. The Economic Journal 121(556), F329–F339 (2011)
9. Giaglis, G.M., Klein, S., O'Keefe, R.M.: The role of intermediaries in electronic marketplaces: developing a contingency model. Information Systems Journal 12 (2002)
10. Hagiu, A., Jullien, B.: Why do intermediaries divert search? The RAND Journal of Economics 42(2), 337–362 (2011)
11. Koulu, R.: Blockchains and online dispute resolution: Smart contracts as an alternative to enforcement. SCRIPTed 13, 40 (2016)
12. Legner, C.: Is there a market for web services? In: International Conference on Service-Oriented Computing. pp. 29–42. Springer (2007)
13. Olsthoorn, M., Winter, J.: Decentral market: self-regulating electronic market (2016)
14. Papazoglou, M.P.: Service-oriented computing: Concepts, characteristics and directions. In: Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE). pp. 3–12. IEEE (2003)
15. Schlauderer, S., Overhage, S.: How perfect are markets for software services? An economic perspective on market deficiencies and desirable market features. In: ECIS (2011)
16. Soska, K., Christin, N.: Measuring the longitudinal evolution of the online anonymous marketplace ecosystem. In: USENIX Security. vol. 15 (2015)
17. Soska, K., Kwon, A., Christin, N., Devadas, S.: Beaver: A decentralized anonymous marketplace with secure reputation. IACR Cryptology ePrint Archive 2016, 464 (2016)
18. Turner, M., Budgen, D., Brereton, P.: Turning software into a service. Computer 36(10), 38–44 (2003)