Poster: QoS-aware Service Composition using Blockchain-based Smart Contracts*

Puwei Wang, Xiaohe Liu,
Jinchuan Chen
School of Information, Renmin
University of China
{wangpuwei,preminem,jcchen}@ruc.
edu.cn

Ying Zhan†
School of Humanities and Law,
Guizhou University of Finance and
Economics
zhanycathy@163.com

Zhi Jin
School of Electrical Engineering and
Computer Sciences, Peking University
zhijin@pku.edu.cn

ABSTRACT

Smart contracts that run on blockchains can ensure the transactions are automatically, reliably performed as agreed upon between the participants without a trusted third party. In this work, we propose a smart-contract based algorithm for constructing service-based systems through the composition of existing services.

ACM Reference Format:
Puwei Wang, Xiaohe Liu, Jinchuan Chen, Ying Zhan, and Zhi Jin. 2018. Poster: QoS-aware Service Composition using Blockchain-based Smart Contracts. In *ICSE '18 Companion: 40th International Conference on Software Engineering Companion, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3183440.3194978

1 INTRODUCTION

Blockchains and smart contracts have received significant attentions recently. The term "blockchain" comes from the Bitcoin network, in which transactions are recorded in a distributed ledger organized as a series of blocks. Each node maintains a copy of the whole data and they should follow the same consensus policy to obtain consistency. Smart contracts can be thought of as automated trustworthy workflow between parties without a central specific co-ordinator, which nobody controls and therefore everyone can trust.

In this work, we use the techniques of smart contracts for constructing service-based systems through the composition of existing services. The service-based systems are composed of services and exposed as composite services for use through standardized protocols, such as WSDL[1]. Service composition involves two kinds of players: *a service requester* and *a set of service providers*. The service requester chooses some service providers and composes their services to implement his desired system, while satisfying his constraints on the QoS (Quality of Service, e.g., response time, etc) and the budget. During the service composition process, they need to reach agreements on the QoS and prices. In general, the service requesters and the service providers do not know each other, and there are no trusted central organizations to guarantee the implementation of agreements among them. Recent work [3] proposed auction-based approach to reach agreements, but do not sufficiently discuss how to ensure the implementation of the agreements.

2 SMART CONTRACT-BASED ALGORITHM

In general, a QoS-aware composition process includes two steps: 1) for his desired system, the service requester identifies an abstract process that contains a set of tasks [2]; 2) the service requester chooses some service providers from the candidates to perform the tasks while meeting his QoS and budget constraints [7]. In this work, we assume the abstract process has been identified and our objective is to choose a service provider for each task.

In this work, we propose a smart-contract based algorithm for the QoS-aware service composition, running on the *Ethereum* platform [6]. *Ethereum* is an open-source, public, blockchain-based distributed platform for implementing smart contracts. Figure 1 shows the overall architecture. Suppose the service requester has three tasks t_1, t_2 and t_3. He installs a smart contract for each task on an arbitrary node of an *Ethereum* network. The smart contracts automatically select the service providers from candidates to perform the service requester's tasks and create the agreements with the selected service providers. For example, the smart contracts select the service provider s_2 to perform the task t_1, the service provider s_4 to perform the task t_2, the service provider s_3 to perform the task t_3,

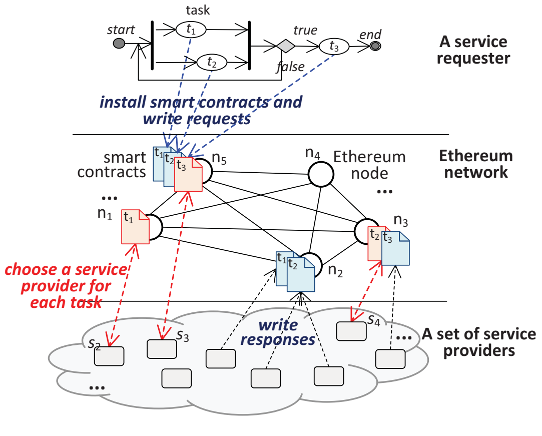

Figure 1: Overall Architecture

*The work was supported by the National Natural Science Foundation of China (NSFC) under grants 61003084, 61232007, 61472428 and 61003086.
†Corresponding Author

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3194978

and create the agreements on the QoS and prices. Finally, the smart contracts automatically execute the agreements, e.g., check the QoS the service providers offer and transfer the money promised to the service providers.

Suppose the service requester has a set of tasks $T = \{t_1, ..., t_m\}$. We divide the tasks T into two sets: one is *the set of assigned tasks* T_{acc} (we have selected a service provider for each task in T_{acc}); the other is *the set of unassigned tasks* T_{rej}. There is $T_{acc} \cap T_{rej} = \phi$ and $T_{acc} \cup T_{rej} = T$.

Algorithm 1: Smart Contract-based Composition Algorithm

```
1  $T_{acc} = \phi$; $T_{rej} = T$;
2  the service requester initializes a smart contract for each
     unassigned task $\forall t_i \in T_{rej}$ and deploys the smart contract on
     a node of Ethereum network;
3  //select a service provider for each task
4  while $T_{rej} \neq \phi$ do
5      for $\forall t_i \in T_{rej}$ do
6          the service requester stores his requests into the
             blockchain via the smart contract;
7          for $\forall s_j \in S_i$ do
8              the service provider stores his responses into the
                 blockchain via the smart contract;
9          end
10         if the requests can be satisfied then
11             the smart contract selects a service provider and
                 creates an agreement on the QoS and price;
12             move $t_i$ from the unassigned set $T_{rej}$ to the
                 assigned set $T_{acc}$;
13         end
14     end
15 end
16 //run the composite service
17 for $\forall t_i \in T_{acc}$ do
18     the smart contract executes the agreement between the
         service requester and the service provider;
19 end
```

In the beginning, all tasks have not yet been assigned (Line 1). For each unassigned task $\forall t_i \in T_{rej}$, the service requester installs a *smart contract* and then writes the selection & pricing rules (how to select the service providers and how to determine the prices of the tasks) into the smart contract. The smart contract are propagated to the whole *Ethereum* network (Line 2). The service requester writes a set of *requests* into the blockchain (Line 6). A request includes a desired QoS and a reserve price (the amount money he is willing to pay at most). In our previous works [4][5], we proposed a *Bayesian Nash equilibrium* of service providers. The equilibrium can motivate the cost-efficient service providers to offer the high QoS at low price. That can help the service requester to design his requests. The service providers read the requests from the blockchain network, and then writes their *responses* into the blockchain (Line 8). A response includes a QoS offer and a bid price (the minimum price he accepts) According to the selection & pricing rules, the smart contract automatically selects a service provider, creates an *agreement* with the selected service provider and stores the agreement into the blockchain (Line 11); If there are no unassigned tasks, the service requester obtains a composite service. When the service requester runs the composite service, the smart contract *automatically execute the agreements* (Line 18).

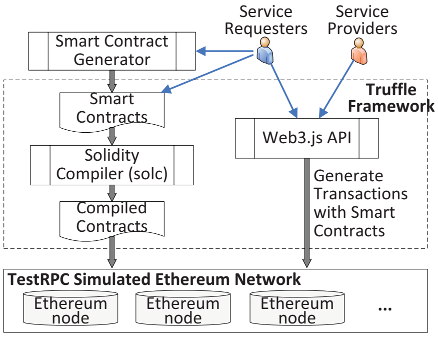

Figure 2: Smart Contract Creation and Inovcation

We created a simulated *Ethereum* blockchain network by *TestRPC* and implemented the algorithm on the simulated network. Figure 2 shows the creation and invocation of the *Ethereum* smart contracts. In our algorithm, the service requesters create the smart contracts written in the language *Solidity*, use the development framework *Truffle* to compile the smart contracts and deploy the smart contracts in the simulated *Ethereum* network. The service requesters and the service providers use the *Web3.js* API to invoke the smart contracts to write requests/responses into the *Ethereum* network.

3 CONCLUSION

In this work, we propose a smart-contract based algorithm for the QoS-aware service composition. There are three merits of our approach: ***first***, the smart contracts are automatically triggered and executed. This reduces the transaction costs associated with negotiation; ***second***, the smart contracts ensure that the transactions are automatically, reliably performed as agreed upon between the service requesters and the service providers; ***third***, the blockchain-based smart contracts guarantee it is almost impossible for the service requesters and the service providers to falsify their data.

REFERENCES

[1] R. Chinnici, J.-J Moreau, A.Ryman, and S. Weerawarana. 2007. Web Service Description Language (WSDL) version 2.0.

[2] Shuiguang Deng, Bin Wu, and Jianwei Yin. 2013. Efficient Planning for Top-k Web Service Composition. *Knolwedge and Information Systems* 36, 3 (2013), 579–605.

[3] Qiang He, Jun Yan, Hai Jin, and Yun Yang. 2014. Quality-Aware Service Selection for Service-Based Systems Based on Iterative Multi-Attribute Combinatorial Auction. *IEEE Transactions on Software Engineering* 40, 2 (2014), 192–215.

[4] Puwei Wang, Tao Liu, Ying Zhan, and Xiaoyong Du. 2017. A Bayesian Nash Equilibrium of QoS-aware Web Service Composition. In *International Conference on Web Services (ICWS)*. 676–683.

[5] Puwei Wang, Ying Zhan, Tao Liu, and Xiaoyong Du. 2017. QoS-Aware Service Composition for Service-Based Systems Using Multi-Round Vickery Auction. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2891–2896.

[6] Gavin Wood. 2017. Ethereum: A secure decentralized generalized transaction ledger Ethereum: A secure decentralized generalized transaction ledger. (2017).

[7] Tao Yu, Yue Zhang, and Kwei-Jay Lin. 2007. Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints. *ACM Transactions on the Web* 1, 1 (2007).