Fluid: A Blockchain based Framework for Crowdsourcing

Siyuan Han, Zihuan Xu, Yuxiang Zeng, Lei Chen
The Hong Kong University of Science and Technology
{shanaj,zxuav,yzengal,leichen}@cse.ust.hk

ABSTRACT

Recently, crowdsourcing has emerged as a new computing paradigm to solve problems that need human intrinsic, such as image annotation. However, there are two limitations in existing crowdsourcing platforms, *i.e.*, *non-transparent incentive mechanism* and *isolated profiles of workers*, which harms the interests of both requesters and workers. Meanwhile, Blockchain technology introduces a solution to build a transparent, immutable data model in the Byzantine environment. Moreover, Blockchain systems (*e.g.*, Ethereum) can also support the Tuning-complete script called smart contracts. Thus, we are motivated to use the feature of transparent data model and smart contract in Blockchain to address the two limitations. Based on the proposed solutions, we have designed a Blockchain based framework which supports foundations of general crowdsourcing platforms. In addition, our framework also has following novel features: (1) it provides the transparent incentive mechanisms; (2) it supports a trusted worker's profile sharing in a cross-platform mode.

ACM Reference Format:
Siyuan Han, Zihuan Xu, Yuxiang Zeng, Lei Chen. 2019. Fluid: A Blockchain based Framework for Crowdsourcing. In *2019 International Conference on Management of Data (SIGMOD '19), June 30-July 5, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3299869.3320238

1 INTRODUCTION

In the last decades, with the development of internet and sharing economy, crowdsourcing has emerged as a new computing paradigm, which utilizes human intelligence to solve tasks that require human intrinsic. It has been widely studied in both academia (*e.g.*, database community [5, 8, 10]) and industry (*e.g.*, Amazon Mechanical Turk, CrowdFlower, Figure Eight Inc.). A crowdsourcing platform usually aims to attract a huge amount of crowd workers (*a.k.a.* workers) to perform all kinds of tasks, *e.g.*, image annotation, data labeling, etc. Thus, a foundational challenge in the crowdsourcing platform is *how to design incentive mechanism*, *i.e.*, how to attract both workers and requesters of the tasks?

To solve the challenge, existing works propose several different incentive mechanisms, *e.g.*, monetary based [4] and reputation based [1]. They usually design effective budget allocation method or reputation maintenance strategy. However, existing platforms [3, 7] usually do not support *transparent* incentive mechanism, *i.e.*, an explicit mechanism designed by platforms but visible to the workers. In practice, a *non-transparent* mechanism can harm the interests of participants. For instance, the requester, who pays a higher service fee to the platform, should have the authority to know the specific pricing strategy. Otherwise, he/she may eventually lose the trust and leave the platform.

Another limitation in existing platforms is that the profiles (*e.g.*, expertise, reputation) of the workers are not shared among different platforms. For example, a multi-skilled worker may want to make money from different crowdsourcing platforms (*e.g.*, Amazon Mechanical Turk and TopCoder). However, his high qualified profile in Amazon Mechanical Turk is not trusted by the other crowdsourcing platforms. The worker usually has no enough time to build up a new qualified profile, which makes him/her join in only one crowd labor markets at last. Hence the workers will make less money and some crowdsourcing platforms will have less human powers compared with the dominant companies.

With the demand of transparent and trustful collaborative data management, recently, a technology named Blockchain attracts much attention [2, 6, 11] due to the success of Bitcoin. The novel design of *Proof-of-Work* (PoW) consensus mechanism solves the traditional Byzantine problem in a non-deterministic way which can in turn build up trust among the participants without a third-party. Nodes within a Blockchain system can reach consensus of the system state in a decentralized way even with the existence of malicious nodes. Meanwhile, every node shares a chain-like append-only replicated data ledger. Since this ledger is maintained

SIGMOD '19, June 30-July 5, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5643-5/19/06…$15.00
https://doi.org/10.1145/3299869.3320238

by the crowd-powered peer to peer network, any single centralized party cannot control the whole network. Moreover, this ledger is transparent to every participants. Users will be aware of any modification on the Blockchain data.

Moreover, Ethereum applies a near Turing-complete language in its Blockchain system. It makes the idea of *smart contract* come into reality, which refers to a piece of code representing the business logic. By pre-defining and consensing a smart contract, the same operations could be automatically executed to update the system state when the pre-defined conditions are fulfilled. Smart contract expands the power of Blockchain and provides more possibility to develop decentralized applications (DApp) on top of Blockchain.

In this demo, we implement a Blockchain based framework called Fluid which provides transparent incentive mechanism for crowdsourcing platform. We also design the complete workflow which integrate typical crowdsourcing process with Blockchain. In addition, we design three smart contract templates to combine existing crowdsourcing algorithms. Moreover, we develop several APIs to manage task and Blockchain data based on Ethereum. Finally, we develop an Android client for users to utilize our framework.

In the rest of the paper, we present Fluid in Sec. 2, our implementation in Sec. 3, and demonstration plan in Sec. 4.

2 FRAMEWORK OVERVIEW

2.1 Components and Architecture

We first introduce the four main components in the Fluid: *task* (*i.e.*, *requester*), *worker*, *platform*, and *Blockchain*.

Task. Tasks are usually submitted to the crowdsourcing platform by the requesters with their pre-set requirements, *e.g.*, deadline, expertise, reputation, etc. To answer a task, a worker has to satisfy all its requirements. After receiving the aggregated answers from platform, the requesters can eventually confirm the payments to the workers.

Worker. Workers, with unique accounts in Blockchain, can register at different crowdsourcing platforms with their identities in Blockchain. Via the smart contract of Blockchain, they can browse and select the available tasks. After accepting a task, a worker has to answer the task within its deadline specified by the requester.

Platform. The platforms connect both requesters and workers. After requesters publish their tasks, the platforms will carefully design the tasks and transform them into the smart contracts of Blockchain, in order to make them visible to the registered workers in Blockchain. Besides, the platform implements the incentive mechanisms in smart contracts, which are transparent to both workers and requesters.

Blockchain. Blockchain provides a Byzantine-fault tolerate environment and an immutable ledger. Via the smart contracts, it provides the underlying services for the requesters, workers, and platforms. Specifically, since smart contracts support Turing-complete language, we can implement *credit contract* for each participants, *task contents* for each task, as well as embedded incentive mechanism (*e.g.*, cost control method), latency control method in the smart contracts.

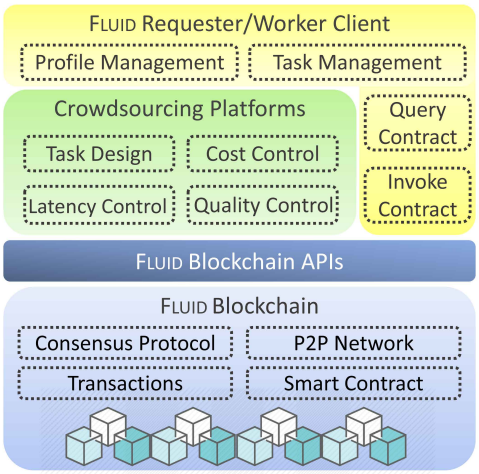


Figure 1: The system architecture of Fluid

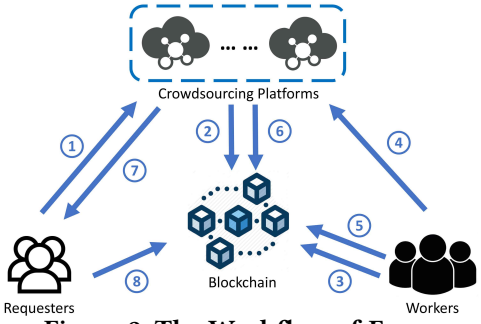


Figure 2: The Workflow of Fluid

The architecture of the Fluid is illustrated in Fig. 1. We use Blockchain to connect the crowdsourcing platforms, requesters and workers through the P2P network. Fluid also provides APIs to manage tasks and users' profiles, and support basic crowdsourced data operations (*e.g.*, query), etc.

2.2 Workflow

We next present the workflow of Fluid. Different from existing systems where requesters and workers only interact with the platforms, we separate parts of the processing logic and put them on Blockchain. Thus, our workflow consists of eight main steps in Fig. 2. Intuitively, we divide them into three phases: *task creation*, *task processing*, and *task completion*.

Task Creation. In Fluid, a task is represented by a *task contract*. Although a platform can store the tasks in its own database, their copies are also recorded in Blockchain by the smart contracts. Steps 1-2 represent the detailed procedure.

Step 1. Requesters submit the raw tasks to a crowdsourcing platform with additional requirements (*e.g.*, expertise, deadline). Most importantly, requesters should transmit some credit (*i.e.*, token) that they are willing to pay from their credit contract to the platform's credit contract.

Step 2. The platform designs and decomposes the raw task into several atomic tasks [10]. Based on requesters' requirements and the quality control method, a *worker selection* function is generated. In addition, the *incentive* function is chosen by the platform. Finally, task content, worker selection function and incentive function are transformed into the smart contract which will be published on Blockchain.

Task Processing. In this phase, workers will browse, select and answer their available tasks. Steps 3-5 represent the detailed procedure.

Step 3. A worker accepts the task by transmitting required credit in the task contract. The worker selection function is automatically invoked to lock the credit or reject unqualified worker. This step is confirmed by consensus protocol.

Step 4. Workers submit their answers to the platform.

Step 5. A transaction (including digest of the answer, etc.), which represents the allocation between a task and its worker, will be automatically generated and sent to Blockchain.

Task Completion. After receiving the answers, the platform will aggregate the final result. Meanwhile, based on the incentive function, rewards will be paid to allocated workers.

Step 6. After platform confirms the final results, a completion record will be sent to the task contract. Meanwhile, platform can choose to active *zero-knowledge proof* (Zk) function for others to verify the completion of the task without exposing the actual answers. This step guarantees the trust and protects the privacy of users.

Step 7. Platform returns the final result to the requester.

Step 8 (optional). The requester record the feedback in the task contract.

2.3 Smart Contract

To generate the smart contracts at different phases, we design three contract templates, *identity contract*, *credit contract*, and *task contract*, which include the compulsory logic functions.

Identity contract is used to record the roles of network participants. In Fluid, each user has a unique Blockchain node address. This contract maintains tuples of <*address*, *role*>. It is created when Fluid Blockchain is established.

Credit contract records the credit and expertise of the user. The credit is a kind of digital token (*i.e.*, ERC-20 token in Ethereum). It could represent reputation in different platforms, money, etc. Credit could be transferred, mortgaged, extracted between authorized users.

Task contract is built by the crowdsourcing platform based on the raw task provided by requesters. Task contract contains three basic elements: *(1)* **Task content** consists of the basics of a task, *e.g.*, task type, task description, deadline, requirements, etc. *(2)* **Global status** includes the current status of a task, the corresponding requester, worker and crowdsourcing platform. *(3)* **Incentive function** represents the specific incentive mechanisms like task pricing, reputation maintaining function.

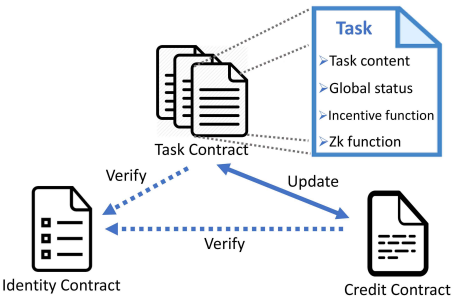


Figure 3: The smart contract designs in Fluid

Additionally, task contract can include the **zero-knowledge proof function** (Zk-function) as well. Specifically, to build a trustful workflow, a user must have the ability to verify a task is actually done by selected workers without exposing the results for privacy protection. We design a Zk-function layer to verify the answers to the tasks confirmed by the platform. By using zero-knowledge proof algorithms (*e.g.*, zk-SNARKs [9]), everyone can check whether workers give valid results to a task without knowing the details.

As shown in Fig. 3, both task contacts and credit contracts will be used to verify the personal information of users on Blockchain. Besides, they are both updated during the eight steps of our workflow.

3 IMPLEMENTATION

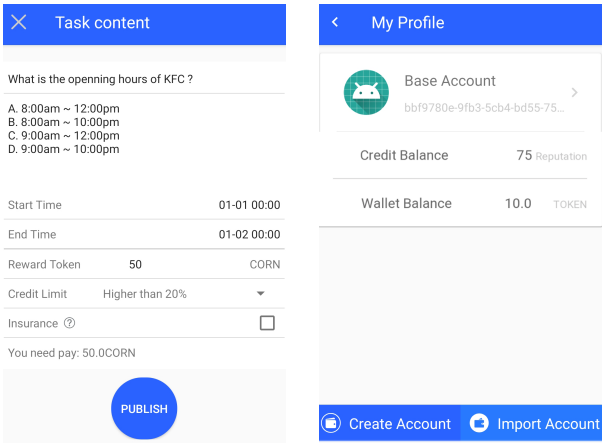
In this section, we describe the details of Fluid implementation. We forked the go-Ethereum version 1.8.19 as our Blockchain solution. Based on it we developed the aforementioned APIs. In addition, we use the Solidity (version 0.4.22) programming language to implement the contract templates.

To simulate the function of the crowdsourcing platform, we set up a back-end server and implement several the state-of-the-art algorithms in existing studies. Specifically, we use the OPQ-based method to decompose a task into several atomic tasks [10]. We implement the incentive mechanism in [4] to determine the price of task. Finally, we use the greedy-based method in [12] to speed up the task completion.

For demonstration, we developed an Android application which utilizes the APIs of both simulated back-end crowdsourcing platform and Fluid Blockchain. Requester can publish tasks as shown in (Fig. 4 (a)). Workers can browse and do the tasks as well. Users can view their own profile (*i.e.*, credit and expertise record) on the Blockchain (Fig. 4 (b)). Moreover, they can check the states of entire Fluid Blockchain (Fig. 5).

4 DEMONSTRATION PROPOSAL

In this demonstration, we will describe how Fluid works and how Fluid framework integrates with typical crowdsourcing platforms. We will present two kinds of scenarios, *task lifecycle simulation* and *credit contract checking*, of the Fluid in details, and explain the purposes of our demonstration.



(a) Submission interface (b) User profile

Figure 4: The client of requester in Fluid

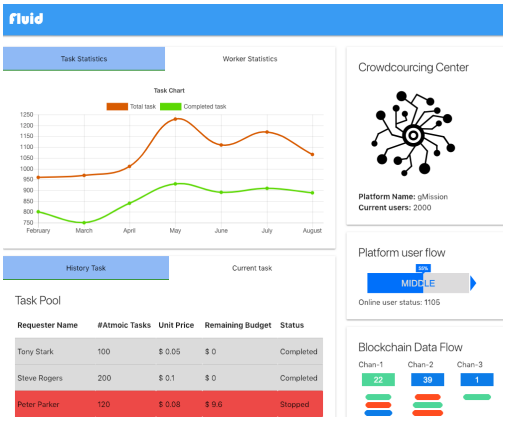


Figure 5: The platform dashboard of Fluid

Scenario 1: task life-cycle simulation. In this scenario, we exhibit the entire workflow in Fluid to the audiences. In the phase of **task creation**, a registered requester submits a set of raw tasks with descriptions and his/her requirements (*e.g.*, deadline, reputation, etc.) to the crowdsourcing platform as shown in Fig. 4(a). The platform will decompose the tasks into more atomic tasks [10] and transform them into smart contracts with built-in incentive mechanisms based on our contract template. In the phase of **task processing**, workers can select these tasks on Blockchain and mortgage the credit from the credit contract into the task contract. While workers submit their answers to the platform. the corresponding transactions will be automatically sent to Blockchain. In the last phase (*i.e.*, **task completion**), the platform will locally aggregate the answers and provide them to the requesters. After the confirmation of requesters, the platform will give the payments to workers and update their reputations based on the incentive function in the task contract.

Scenario 2: credit contract checking. Given a task, platform X could check qualified workers not only from their own workers but also from other platforms via Fluid. Specifically, after X sends a task contract to the Fluid, some workers may apply for answering this task. Then the platform will check the identity contract to find the identity information of the workers and browse the historical records from credit contracts. As shown in Fig. 4(b), these information are visible to both platforms and workers themselves in the client. Since this platform X can know the historical credits and reputations of workers from other platforms, X can use its own quality control method to filter qualified workers. If the platform X still doubts about these records, it can invoke those Zk-functions from all the historical task contracts to check the truthfulness and correctness of those credits.

ACKNOWLEDGMENTS

This work is partially supported by the following grants: Hong Kong RGC GRF Project 16202218; the National Science Foundation of China (NSFC) under Grant No. 61729201; Science and Technology Planning Project of Guangdong Province, China, No. 2015B010110006; Hong Kong ITC ITF grants ITS/470/18FX and ITS/212/16FP; Didi-HKUST joint research lab project; Huawei PhD Fellowship; Microsoft Research Asia Collaborative Research Grant; Wechat Research Grant

REFERENCES

[1] Caleb Chen Cao, Yongxin Tong, Lei Chen, and H. V. Jagadish. 2013. WiseMarket: a new paradigm for managing wisdom of online social users. In *KDD*.
[2] Tien Tuan Anh Dinh, Ji Wang, Gang Chen, Rui Liu, Beng Chin Ooi, and Kian-Lee Tan. 2017. BLOCKBENCH: A Framework for Analyzing Private Blockchains. In *SIGMOD*.
[3] Amber Feng, Michael J. Franklin, Donald Kossmann, Tim Kraska, Samuel Madden, Sukriti Ramesh, Andrew Wang, and Reynold Xin. 2011. CrowdDB: Query Processing with the VLDB Crowd. In *PVLDB*.
[4] Yihan Gao and Aditya G. Parameswaran. 2014. Finish Them!: Pricing Algorithms for Human Computation. In *PVLDB*.
[5] Hector Garcia-Molina, Manas Joglekar, Adam Marcus, Aditya G. Parameswaran, and Vasilis Verroios. 2016. Challenges in Data Crowdsourcing. *IEEE Trans. Knowl. Data Eng.* 28, 4 (2016), 901–911.
[6] Siyuan Han, Zihuan Xu, and Lei Chen. 2018. Jupiter: A Blockchain Platform for Mobile Devices. In *ICDE*.
[7] Guoliang Li, Chengliang Chai, Ju Fan, Xueping Weng, Jian Li, Yudian Zheng, Yuanbing Li, Xiang Yu, Xiaohang Zhang, and Haitao Yuan. 2018. CDB: A Crowd-Powered Database System. In *PVLDB*.
[8] Guoliang Li, Jiannan Wang, Yudian Zheng, and Michael J. Franklin. 2016. Crowdsourced Data Management: A Survey. *IEEE Trans. Knowl. Data Eng.* 28, 9 (2016), 2296–2319.
[9] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized anonymous payments from bitcoin. In *IEEE S & P*.
[10] Yongxin Tong, Lei Chen, Zimu Zhou, H. V. Jagadish, Lidan Shou, and Weifeng Lv. 2018. SLADE: A Smart Large-Scale Task Decomposer in Crowdsourcing. *IEEE Trans. Knowl. Data Eng.* 30, 8 (2018), 1588–1601.
[11] Zihuan Xu, Siyuan Han, and Lei Chen. 2018. CUB, a Consensus Unit-Based Storage Scheme for Blockchain System. In *ICDE*.
[12] Yuxiang Zeng, Yongxin Tong, Lei Chen, and Zimu Zhou. 2018. Latency-Oriented Task Completion via Spatial Crowdsourcing. In *ICDE*.