Towards Efficient and Deposit-Free Blockchain-Based Spatial Crowdsourcing

Mingzhe Li, *Student Member, IEEE*, Wei Wang, *Member, IEEE*, and Jin Zhang, *Member, IEEE*

***Abstract*—Spatial crowdsourcing emerges as a new computing paradigm that enables mobile users to accomplish spatio-temporal tasks in order to solve human-intrinsic problems. Existing crowdsourcing systems critically use centralized servers for interacting with workers and making task assignment decisions. These systems are hence susceptible to issues such as the single point of failure and the lack of operational transparency. Prior work, therefore, turns to blockchain-based decentralized crowdsourcing systems, yet still suffers from problems of *lacking efficient task assignment scheme, requiring a deposit to an untrusted system, low block generation speed, and high transaction fees.* To address these issues, we design a blockchain-based decentralized framework for spatial crowdsourcing, which we call SC-EOS. Our system does not rely on any trusted servers, while providing *efficient and user-customizable task assignment*, low monetary cost, and fast block generation. More importantly, it *frees users from making a deposit into an untrusted system*. Our framework can also be extended and applied to generic crowdsourcing systems. We implemented the proposed system on the EOS blockchain. Trace-driven evaluations involving real users show that our system attains the comparable task assignment performance against a clairvoyant scheme. It also achieves 10× cost savings than an Ethereum-based implementation.**

***Index Terms*—Blockchain, spatial crowdsourcing, task assignment, smart contract**

I. Introduction

CROWDSOURCING [15] is proposed as a new computing model that outsources problems (tasks) to an undefined group of internet users (workers) through an open call for solutions. Thanks to the ever-evolving smart phone technologies and the growing number of mobile phone users, more extensive types of tasks in different scenarios could be performed using mobile phones through spatial crowdsourcing [19]. In spatial crowdsourcing, the outsourced tasks contain spatial information, and they require workers to travel to some specific locations to accomplish the tasks. Many companies rely on spatial crowdsourcing as a primary means of solving real-world problems, ranging from infrastructure monitoring to smart transportation [29], [13], [18], [16], [26], [30]. There are numerous well-known spatial crowdsourcing applications such as Uber [7], Gigwalk [5] and Waze [8].

Traditional spatial crowdsourcing consists of three groups of entities, *requesters, workers* and a centralized *platform* (a.k.a. *server*) [19]. Requesters submit spatial- and temporal-related tasks that need to be performed in a spatial crowdsourcing platform. The platform then manages the tasks and assigns them to different workers. Workers, upon receiving the tasks, need to perform them at specified locations and return the results to the server. In return, the workers will receive some monetary rewards from the requesters through the platform.

Despite the popularity of the spatial crowdsourcing systems, their centralized computing models lead to several concerns. First, traditional spatial crowdsourcing systems are vulnerable to a single point of failure due to the centralized architecture [21]. Second, the centralized spatial crowdsoucing systems often suffer from the issue of lacking operational transparency, meaning that the operations made by the server cannot be verified and tracked by users. Therefore, a misbehaving centralized server could manipulate the operations and result in misbehaviors such as "collusion" and "false-reporting" [12]. Finally, spatial crowdsourcing companies charge non-trivial service fees to requesters [5], which, in turn, increases the costs for both workers and requesters.

Blockchain emerges as a promising *decentralized* technique that can address the aforementioned problems in traditional spatial crowdsourcing [11]. Many recent works, e.g., [21], [23], [31], [28], [20], explored to combine crowdsourcing and blockchain using smart contracts. The smart contracts can be seen as programs that are stored on blockchain and executed by blockchain nodes in a faithful manner. Yet, those efforts still fall short in mainly three perspectives. First, few of the existing blockchain-based crowdsourcing systems consider applying efficient task assignment mechanism design into blockchain, which is essential for any crowdsourcing systems [39], [14], [10]. Second, each user (e.g., worker, requester) is required to pay a deposit into the smart contracts so as to force users to behave properly (e.g. transfer correct amount of fees). However, the smart contracts are not bug-free and the contract developers may not be trustable. As a result, the users are taking the risk of losing their deposit [1]. Third, some previous work suffers from the problems of low block generation speed and high transaction fee [21]. The main reason is that those

M. Li is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, and with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China (email: mlibn@cse.ust.hk).

W. Wang is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong (email: weiwa@cse.ust.hk).

J. Zhang is with the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (email: zhangj4@sustech.edu.cn).

J. Zhang and W. Wang are the corresponding authors.

studies rely on Proof of Work (PoW) as their consensus protocol in which certain computing nodes (a.k.a. miners) calculate complicated puzzles and compete with each other to generate new blocks. As a result, the block generation speed is decreased and users need to pay extra fees to compensate the miners for their computing cost [9].

To address the above issues, we propose a novel blockchain-based framework for spatial crowdsourcing named SC-EOS. In SC-EOS, spatial crowdsourcing tasks of requesters and worker information are published onto the blockchain. By using various smart contracts we designed, the published tasks can be assigned efficiently to suitable workers. In addition, SC-EOS guards against malicious behaviors while eliminating the need for requesters and workers to make deposits into smart contracts, hence eliminating the risk of losing their deposit.

A well-performed system requires judicious designs. The main challenges are in three aspects. First, how to design a well-performed task assignment scheme. A good task assignment scheme should consider both the requirements of workers and requesters, and be able to assign a large number of tasks to workers. In practice, different workers/requesters may have variant requirements and preference on their interested tasks/workers. For instances, some workers like tasks with higher reward, some tasks need workers with higher expertise. Based on this observation, we propose a *user-customizable* task assignment scheme in SC-EOS in order *to perform task assignment while satisfying various requirements and preferences of workers and tasks.* The user-customizable task assignment scheme allows workers/requesters to customize their preference in choosing tasks/workers, publish their preference onto the blockchain, and the task assignment scheme matches proper tasks to workers based on the preference. *To assign more tasks to workers* hence improving task assignment efficiency, the proposed task assignment scheme is performed in a *batched* manner. Specifically, the task assignment mechanism is executed periodically, and multiple tasks are assigned to multiple workers at a time in a batch.

Second, how to enable such an efficient and user-customizable task assignment scheme in blockchain. It is hard to enable such a batched task assignment mechanism in blockchain. This is due to the fact that, in a distributed system such as blockchain, there does not exist a trusted centralized server to coordinate and handle the function of assigning multiple tasks to multiple workers. Therefore, *to accomplish the batched task assignment scheme in blockchain*, our design intuition is to periodically select a particular requester and require it to invoke the task assignment smart contract to match multiple tasks and workers. However, another problem occurs under such design. In blockchain, triggering the execution of smart contract requires monetary cost. Therefore, for fairness, the requester who triggers the task assignment smart contract execution should be compensated for its cost. Hence, *to achieve a fair compensation settlement*, we propose a compensation scheme to require the rest of the workers and tasks who are successfully assigned to compensate the cost spent by that requester. Finally, a task assignment scheme possesses lots of functions and complex logic. However, each block in blockchain only has limited space to deal with limited functions or logic, and thus cannot handle the whole task assignment process. To address that, we split the task assignment process into multiple small logic parts, each can be executed without exceeding the block limitation.

Third, how should the system prevent a user from misbehaving without requiring users to deposit into untrusted smart contracts. Without the bondage of the deposit, a user in the system may conduct misbehavior such as sending no reward or submitting no results. To address this issue, we design a series of linkage protocols, in which each user's actions are linked back and forth. Only if a user properly finishes the previous step, it can perform the next step. The linkage design thus incentive users to behave following the system's rules without requiring users to deposit into the untrusted smart contracts.

In addition, to accelerate the block generation speed and lower the cost, Delegated Proof of Stake (DPoS) is used as our consensus protocol where several delegated nodes collaborate together to generate blocks. Finally, *the proposed blockchain-based spatial crowdsoucing framework could be generalized to generic crowdsourcing systems* by some modifications.

In summary, our contributions are as follows.

- We propose a blockchain-based decentralized framework for spatial crowdsourcing named SC-EOS, and design the whole process by leveraging the power of smart contract. Our framework possesses the merits of efficient task assignment scheme, no deposit to untruthful system, low economic cost and fast block generation speed. In addition, it does not suffer from a single point of failure and possesses operation transparency.
- We propose a *user-customizable, blockchain-specialized batched task assignment mechanism* to achieve efficient and customizable task assignment for crowdsourcing atop of blockchain. In addition, a series of *linkage protocols are proposed to force users to make proper behaviors* in strict accordance with the system rules, obviating the need for making additional deposit into the spatial crowdsourcing system.
- We implement the proposed framework to verify the feasibility based on EOS with real users and real-world dataset. Experimental results show that the task assignment scheme leads to an efficient and effective result in terms of the number of the assigned tasks, which is close to the performance of a clairvoyant task assignment scheme. In addition, our framework achieves 10× lower economic cost for users compared with the same system we implement on Ethereum.

II. Background and Related Work

A. Blockchain and Smart Contract

Blockchain is a decentralized distributed ledger originally proposed by Nakamoto in the Bitcoin project [25]. In blockchain, a set of time-ordered *transactions* containing use-case specific data (e.g., currency transfers, or program codes) are recorded in files called *blocks*. Each block contains the hash value of the previous block, and they eventually form a hash chain named blockchain. The blockchain has the properties of decentralization, transparency, traceability and immutability. It thus has great potential to be exploited in various scenarios such as cryptocurrency, digital health, internet of things and crowdsourcing.

Smart contract is used for blockchain to provide various complex logic functions in a faithful manner. It is a collection of code and data stored in a blockchain, which can perform some pre-defined operations. The smart contract is non-tamperable and traceable by using the digital signature and the time stamp of the blockchain technology. Each blockchain nodes can observe the status and execution records of the contract through interactions with the blockchain. Therefore, we use smart contracts to perform task assignment process of spatial crowdsourcing, and hence, all the functions will get executed faithfully and no black-box operation exists in the system.

B. Ethereum

A straightforward way for the system implementation is to implement it based on Ethereum. Ethereum is a well-known blockchain-based platform that implements smart contracts [4]. The Ethereum uses PoW consensus protocol that requires miners to competitively do a compute-intensive verification to maintain the consensus for each block. In Ethereum, the execution for each transaction will consume a certain amount of fees (called "gas"), which is intended to limit the amount of operations to execute the transaction and pay the execution cost to miners. When executing a transaction, the gas will be gradually consumed according to certain rules.

However, there are some drawbacks of using Ethereum. First, the PoW-based consensus algorithm causes the waste of computing resources and limits the throughput of the network. At present, the throughput of Ethereum is only tens of transactions per second. The high latency of the network makes the performance of distributed applications built atop Ethereum incomparable to those of centralized services. Second, executing smart contracts on Ethereum consumes gas, and for highly interactive applications such as crowdsourcing, high-frequency transactions could result in high monetary cost. Therefore, Ethereum is not suitable for our crowdsourcing application scenario.

C. EOS

We choose EOS for our system implementation. EOS is a promising blockchain platform designed for commercial distributed applications that also adopts smart contracts [3]. The purpose is to solve the problems of low performance and high transaction cost of the existing blockchain platforms. EOS typically achieves a throughput of thousands of transactions per second [2]. We choose to build our system atop of EOS mainly because of its feature of high throughput and low monetary cost.

DPoS consensus protocol. DPoS consensus protocol enables the high throughput of EOS. Unlike Ethereum, EOS uses the DPoS (Delegated Proof of Stake) consensus mechanism to generate blocks. 21 super nodes are elected as block producers by the whole network token-holding nodes, and the super nodes take turns to confirm and record the transaction data. Since block producers do not need to compete for mining, EOS can accurately generate a block every 0.5 seconds, and the transaction can be irreversibly confirmed after 1 second, which is much faster than Ethereum.

RAM cost. The monetary cost in EOS is due to buying RAM, which is low according to our evaluation results. The RAM is used for a user to store data in an in-memory database. The database is called EOS database maintained by the super nodes. In practice, it is often the case to cope with multiple complex operations (e.g. the task assignment process in crowdsourcing) rather than a single monetary transfer between users. It is generally required for users to save data (e.g. task assignment results) onto the blockchain while executing sophisticated operations in smart contracts. Hence, users are required to purchase RAM space from EOS for data storage.

Action. A user needs to invoke the smart contracts to perform task assignment, such invocation is achieved by pushing an *action* into EOS network. An action is the most elementary component in EOS, which represents a single operation. One or multiple actions form a *transaction*, and the block producers will pack one or multiple transactions into a block and generate the block. Contracts and users in EOS communicate in the form of pushing actions to the blockchain network.

D. Prior Work and Its Inefficiency

Traditional spatial crowdsourcing systems. With the wide adoption of the smart mobile devices, Spatial crowdsourcing has attracted an increasing amount of attention. The main focus of spatial crowdsourcing includes: (a) task assignment scheme design [37], [14], [10], (b) incentive mechanism design [22], [35], [34], (c) data aggregation, quality control and evaluation [24], [17], [33], and (d) security and privacy problems in spatial crowdsourcing [36], [32], [38]. The design of the task assignment mechanism is crucial for a crowdsourcing system. An efficient task scheme should assign more tasks at faster speed with both the requesters and workers satisfying the assignment results. *Although many task assignment schemes have been proposed in traditional crowdsourcing systems, they cannot be directly applied in blockchain-based systems without a centralized server. Moreover, those centralized models suffer from issues such as a single point of failure and the lack of operation transparency, preventing the wide adoption of spatial crowdsourcing.*

Blockchain-based crowdsourcing system. With a growing interest in Blockchain technology, there are quite a few attempts to build crowdsourcing systems atop blockchain. For example, Li et al. [21] presented CrowdBC, a basic design of a blockchain-based framework for crowdsourcing systems. A private and anonymous blockchain-based crowdsourcing system called ZebraLancer was proposed in [23] in order to address the data leakage and identity breach problems. NF-Crowd [20] was proposed to reduce the lower bound of the total cost of a decentralized crowdsourcing project. Tan et al. [28] proposed a blockchain-empowered and decentralized trusted service mechanism for the crowdsourcing system in 5G-enabled smart cities. The most related work to our approach is an industry project named Moonlight [6], which is a platform to hire knowledgeable workers for different projects. However, the details about the design of its protocols are not provided. Furthermore, the platform is built on NEO blockchain, which is fundamentally different from the blockchain used in our system. *However, nearly none of the prior work considers to design an efficient task assignment scheme based on blockchain. Moreover, they require users to deposit into their system, which increases the risk for users to lose their deposit once facing malicious developers or encountering smart contract bugs* [27], [1].

III. System Overview

In this section, we present the basic system architecture for our blockchain-based spatial crowdsourcing system. Based on the system architecture, we explain the intuitions and challenges behind several core designs of our system, i.e., the blockchain-based efficient and user-customizable task assignment mechanism design and the design of the linkage protocol to prevent users from misbehaving with no deposit.

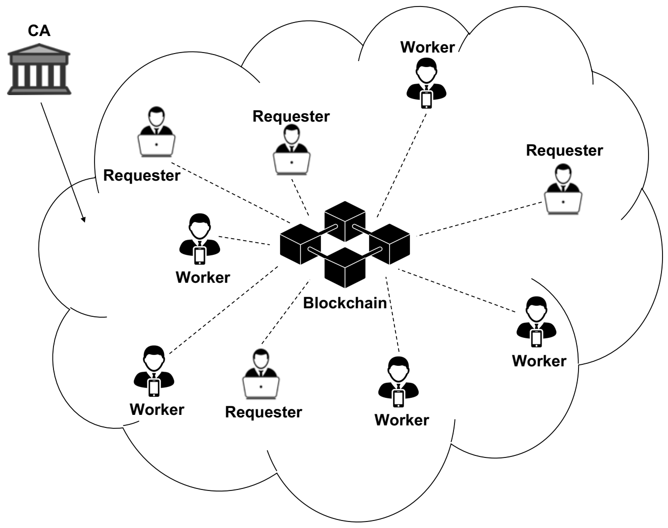


Fig. 1: SC-EOS system architecture.

A. System Architecture

There are four entities in our system model: **Requester, Worker, Blockchain Platform,** and **Certification Authority (CA)**. The basic system architecture is shown in Fig. 1. In particular, *requesters*, denoted as $R = \{r_1, ..., r_j, ..., r_m\}$, with location-related tasks $T = \{t_1, ..., t_k, ..., t_p\}$ could release their task information and gather the task results through the blockchain. *Workers*, identified by $W = \{w_1, ..., w_i, ..., w_n\}$, also publish their information and could claim rewards via the blockchain. The *blockchain* serves as a hub to bridge workers and requesters as well as manages the spatial crowdsourcing process by leveraging multiple smart contracts. The *CA* verifies and manages unique identities of workers and requesters before they join into our system, by binding each identity to a unique credential (e.g. a digital certificate).

The CA conducts user authentication and access control for the system, which is a necessary demand for many real-world crowdsourcing systems. For example, MTurk and Waze need the CA to prevent malicious participants. In addition, the authentication is done offline only once for each worker and requester and hence can be seen as an initialization process of the whole procedure in our system.

B. Design Intuitions and Challenges

It is with significant importance in a crowdsourcing system to design a task assignment mechanism with high efficiency. However, existing blockchain-based crowdsourcing systems only apply naive task assignment scheme where requesters publish their tasks atop blockchain and workers select the ones they are interested in. Hence, we design a blockchain-based efficient task assignment mechanism, where the functionalities of the mechanism are implemented in smart contracts. In our design, requeseters and workers can publish their customized requirements onto the blockchain, the blockchain then uses smart contracts to assign multiple tasks to workers based on their variant requirements periodically. Therefore, more tasks will be assigned in shorter time and efficiency is achieved. However, **it is challenging to design a blockchain-based efficient task assignment scheme.** We list several main challenges as follows.

First, how to efficiently assign tasks in a *decentralized* blockchain system. The users join in and leave the system dynamically, hence, individually conducting task assignment for each user would be inefficient. Therefore, an efficient task assignment scheme should be periodically performed in different time points, and group users to assign multiple tasks to workers at once. However, such a method is challenging to be implemented in a decentralized blockchain framework, since there exists no centralized server to control the time points, perform the task assignment, and bear the computational cost. In blockchain, it is the distributed users who invoke the smart contracts to perform task assignment, and the user who invokes a smart contract should bear the cost for executing it. Therefore, we design a protocol that the smart contract periodically delegates a requester to perform task assignment for multiple workers and tasks. In addition, the assigned workers and tasks give back a compensation to the delegated requester to compensate its cost for executing task assignment process.

Second, how to implement the task assignment scheme with numerous operations in the blochchain when each block has limited space. Each block in blockchain only allows limited operations (i.e., transactions) to be executed. The exceeded operations will cause the failure for a block to be packed. However, a task assignment scheme contains vast number of operations, making the combination of the task assignment scheme and the blockchain even challenging. To address this challenge, we split the whole task assignment process into multiple modules, each with limited operations, so that the whole task assignment process could be handled in multiple blocks without failure.

Third, how to consider both workers' and requesters' variant requirements and preference, and assign tasks to proper workers. The results of the task assignment should be satisfactory to both the requesters and the workers. Existing task assignment schemes usually consider distance as the solely metric and assign tasks to those workers who are near to them. However, different workers/requesters may have different requirements and preference in choosing tasks/workers in practice. For example, some of the workers want to find tasks with higher rewards, while some other workers prefer tasks that are easy to finish. Therefore, we propose a user-customizable task assignment scheme in which both requesters and workers could choose different preference requirements on workers/tasks. Based on the preferences, a stable matching is performed to assign tasks to the proper workers.

It is essential to prevent users from misbehaviors in any spatial crowdsourcing systems, where a misbehaving user could evade paying money or paying an incorrect amount of money. To ensure proper behavior by users, existing blockchain-based crowdsourcing systems force the users to deposit certain fees into the smart contract they developed before they join into the system. However, such a deposit scheme could raise significant issues if a malicious system developer, for instance, withdraws the money deposited in the smart contract or the deposited money gets stolen by a hacker. Thereby, we design a series of protocols to motivate users to behave following the rules in the system without requiring a deposit. The intuition of the protocol design is to link a user's previous step to its subsequent step. Only a user when determined by the smart contract as behaving properly in the former step, could the user perform the next step.

IV. System Design

In this section, we present the proposed blockchain-based spatial crowdsourcing system, SC-EOS. We first describe the overview of the system, followed by a brief description of the user information content. Then, we explain the two core designs of our system, which are the user-customizable and blockchain-specialized task assignment mechanism, and the misbehavior-preventing and deposit-free protocol.

A. System Overview

We now briefly overview our system. The certificated requesters/workers can publish their task/worker information onto the blockchain, containing their various requirements and preference. Those information will be used for task assignment. Time is divided into slots in our system, to enable efficient task assignment in blockchain, only one requester will be delegated to perform task assignment in one slot. Specifically, the first arrived requester in a slot is elected and perform task assignment for multiple online workers and tasks. To make proper assignment, the task assignment scheme will consider the tasks' and worker' variant preference, and tries to assign tasks to workers who are more suitable for them. Since the task assignment is performed by certain requester, to compensate its cost for invoking smart contract, an amount of compensation is given back to it by the other tasks/workers who get assigned. Through the whole procedure, requesters/workers invoke multiple smart contracts to achieve all the functionalities. To enable the complex functionalities to be successfully executed, the smart contracts are split into finer parts, so that each part can be executed in a block. Finally, to prevent users from misbehavior, we link smart contracts with each other. Only if a user properly finished the last step, can it start to move on to the next step.

B. Task/Worker Information Content

Requesters and workers need to publish their information atop of the blockchain, which contains the required information for the task assignment. Each user in our system needs to publish the required information onto the blockchain to be online through either a *TaskReleasing* smart contract for a requester or a *WorkerReleasing* smart contract for a worker. The *TaskReleasing/WorkerReleasing* smart contract will record the task/worker information into the EOS database.

The basic task information for a requester includes: the task ID, the requester ID, the location of the task, the waiting time of the task, the payment of the task, the preference rule of the task, other property information of the task, and the description of the task.

The basic information for a worker contains: the worker ID, the location of the worker, the waiting time of the worker, the preference rule of the worker, along with other property information of that worker.

We further explain several fields in the information. First, the waiting time, which will be used in the batched task assignment, represents the maximum amount of time that a worker or task is willing to wait before it gets assigned. The task/worker can not be assigned after the waiting time is expired. Second, the preference rule means what preference should a task/worker follow to sort and choose worker/task (e.g. based on distance). Third, the field of other property information for a task/worker contains more options in order to allow a worker/task to sort (e.g. the difficulty of a task). The

TABLE I: Functionality of Smart Contract

Smart Contract Name	Main Functionality
TaskReleasing	Record the task information and invoke the task assignment execution
WorkerReleasing	Record the worker information
TaskMatching	Perform the task assignment and invoke the result notification
ResultInform	Notify users about the task assignment results and the compensation
CompensationVerify	Transfer and verify the compensation
TaskSubmission	Record the task results
Reward	Grant rewards to workers

above two fields will be used to achieve the user-customizable task assignment.

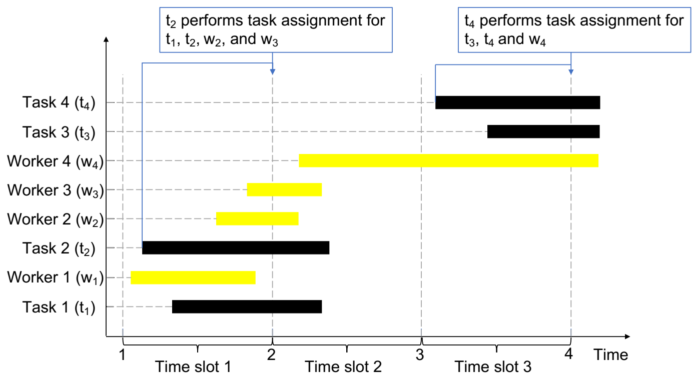


Fig. 2: An Illustration of Batched Task Assignment.

C. *Task Assignment Mechanism Design*

A naive consideration to make task assignments based on blockchain is to empower workers choose the published tasks, which is adopted in most of the previous blockchain-based crowdsourcing systems. However, this could result in unsatisfactory task assignment results since one's preferred worker/task may prefer others. Furthermore, it will be inefficient if the mechanism allows workers and requesters to negotiate with each other and choose the satisfied opposites, since the negotiation may last for several rounds or even may not terminate. Hence, we propose to leverage the smart contract to help workers and requesters perform task assignments.

The proposed task assignment mechanism is implemented in the *TaskMatching* smart contract and will be automatically executed after the execution of the *TaskReleasing* smart contract. The *TaskMatching* smart contract will first judge whether to make a task assignment and only assign tasks when it is needed. The tasks will be assigned according to both the preference rules of workers and requesters through a *stable matching* approach. The assignment results will be sent to the matched requesters and workers through a *ResultInform* smart contract after the task assignment is performed.

We will now explicitly explain the proposed task assignment scheme in terms of how it improves the efficiency through a batched manner, and how it performs the task assignment based on users' preferences.

Task assignment in batch. *Our task assignment scheme is with a batched manner, in which a group of users get assigned together*, to improve its efficiency. Specifically, *the first arrived requester in one time slot will trigger the execution of the task assignment at the next time checkpoint, and the task assignment mechanism will assign all the waiting tasks to waiting workers in a batch.* We will now elaborate on the design aspects we made to achieve the above purpose.

To enable periodic task assignment, we divide time into multiple time slots and create time checkpoints at the beginning of each slot. This functionality is achieved through implementing a global clock which records the latest block generation time. In our framework, each time slot is set to be with equal interval (e.g. 10 minutes). Once a requester r_j has a task t_k, it can push an action (similar as sending a transaction) to blockchain and trigger the *TaskReleasing* smart contract to record the task information. The *TaskReleasing* smart contract will also check the global clock, record it as the arrival time of r_j, and compare the arrival time with the last time checkpoint. Only if r_j is the first arrived requester in one time slot (if multiple requesters arrive at the same time, randomly pick one), the *TaskReleasing* smart contract triggered by r_j will invoke *TaskMatching* to perform task assignments automatically at the next time checkpoint.

The task assignment would get delayed and automatic execution at certain time checkpoint. As aforementioned, the requester arrival and the task assignment are not at the same time. Specifically, the task assignment mechanism will get automatically performed at the time checkpoint in the future. To empower this functionality, we use the action which can perform delayed invocation named deferred action. Through a deferred action, the tasks will not get assigned immediately when invoking *TaskMatching*, yet will be allocated automatically at the next time checkpoint.

Fig. 2 presents an illustration of our batched task assignment scheme. In which the owner of t_2 will perform task assignment at time checkpoint 2 for the waiting workers w_2, w_3 and waiting tasks t_1, t_2, as t_2 is the first arrived task during time slot 1. The same reason for the task assignment at checkpoint 4. The task assignment will not get performed at time checkpoint 3 since no task arrives during time slot 2.

User-customizable task assignment. *The proposed task assignment scheme conduct a stable matching between the waiting workers and tasks, based on their preference rules.* Since different workers/tasks may have different preferences when choosing tasks/workers. The stable matching is used for improving the users' satisfaction of the task assignment results.

The *user-customizable* task assignment scheme is implemented in the *TaskMatching* smart contract. The mechanism will firstly build a specific preference list for each waiting worker/task according to its selected preference rule. For example, if a worker wants to select tasks based on the distance, the mechanism will generate a preference list for it which contains the waiting tasks ordered from the nearest to the farthest. Table II lists the preference rules pre-defined in our system for workers and requesters to choose. More other preference rules could be supplemented to the system as well.

Based on the preference lists of tasks and workers, a *stable matching* algorithm will get performed between waiting workers and tasks, and generate the task assignment results. A worker/task may get one of the two possible states after the matching: either it is stable or not matched by anyone, since the number of waiting workers and the number of waiting tasks may not be the same. In addition, here we consider the situation that one task can be assigned to one worker, and one worker can be assigned one task at a time. This is a practical demand in real world such as car sharing like Uber. If a requester wants to publish more tasks or a worker seeks for more tasks to perform, one will be matched in the future time checkpoints.

Transaction split. *We split the transactions containing the task assignment mechanism and increase the running time limitation in each block to enable a successful execution for each transaction. Since a task assignment scheme contains multiple operations (regarded as transactions in blockchain) sometimes will exceed the limitation of a block.* Specifically, there is a running time limitation for each transaction as well as block in EOS. Hence, we split the transactions with long running time into multiple micro transactions and increase the transaction running time limitation to ensure that each transaction can be finished within the transaction running time limitation, resulting a larger scale task assignment.

Compensation claim. *The matched workers and tasks need to compensate the cost spent by the requester who triggers the task assignment execution. Since the execution of our proposed task assignment mechanism will consume the RAM space of the requester who triggers it (e.g. requester r_j), leading to a result that r_j spends more monetary cost.* Such unfairness would hinder users from joining into our system. To improve the fairness among users in terms of monetary cost, other matched workers/requesters with the same batch should transfer compensation to r_j.

To inform each matched worker and requester about the compensation it should send and the matching results, the *ResultInform* smart contract will get invoked by *TaskMatching* after the task assignment. The notification messages are then sent via blockchain from the requester r_j to the matched workers and requesters.

The amount of compensation is decided and calculated in the *TaskMatching* smart contract. The compensation calculation rule is that each of the other matched workers and tasks should pay requester r_j $1/N$ of the total consumed RAM cost, where N is the number of matched workers and tasks during this task assignment process.

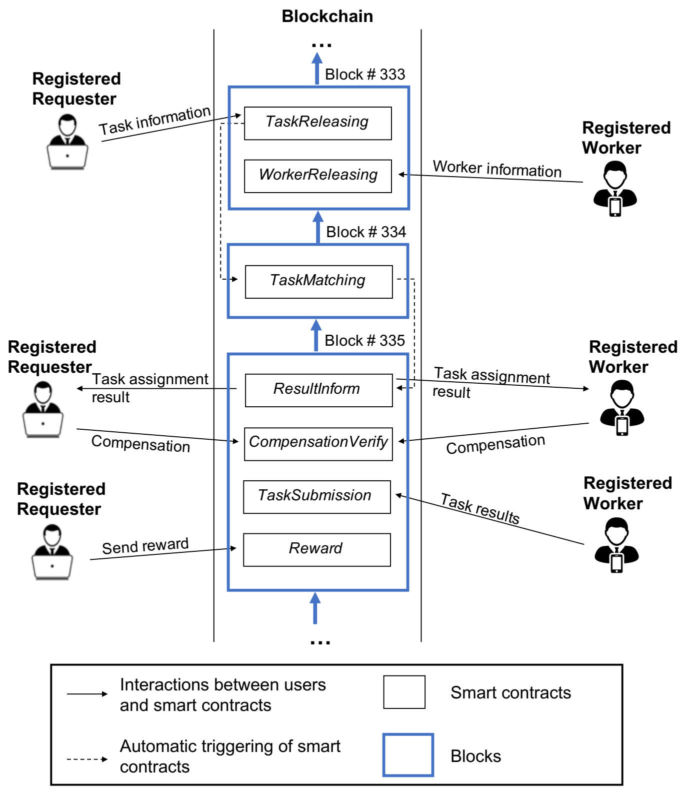


Fig. 3: Basic workflow.

D. Summarized Workflow

In this section, we give a summary for the basic workflow of SC-EOS, which is shown in Fig. 3. In our design, we implement main functionalities in a spatial crowdsourcing system into different smart contracts, and the smart contracts invoke each other to accomplish the spatial crowdsourcing procedure. The main functionalities for different smart contracts in our framework are enumerated in Table I.

Our system mainly contains following stages:

Stage 0: the requesters and the workers get certifications and register into our system through CA. **Stage 1:** each certificated requester/worker can publish its task/worker information onto the blockchain through a *TaskReleasing/WorkerReleasing* smart contract. **Stage 2:** when it is necessary to perform the task assignment for workers and tasks, the *TaskMatching* smart contract will be invoked by *TaskReleasing* to execute the task assignment scheme in a batched and user-customizable manner. **Stage 3:** after the task assignment is executed, the *TaskMatching* smart contract will invoke *ResultInform* to send assignment results to the assigned workers and requesters. **Stage 4:** since the task assignment process is in deed triggered by a certain requester (i.e. *TaskReleasing*, who invokes *TaskMatching*, is triggered by a requester), an amount of fee is required to compensate the cost of that requester, for performing the task assignment, through the *CompensationVerify* smart contract. **Stage 5:** the task results will be sent to the blockchain via the *TaskSubmission* smart contract by the workers who performed the assigned tasks, and **Stage 6:** the requesters after receiving the task results will send rewards to the workers through the *Reward* smart contract.

E. Misbehavior-preventing and Deposit-free Protocol Design

A user has incentives to misbehave deviating from the system rules without the bondage of a deposit, while depositing to the smart contracts of an untrusted system will cause other issues. Therefore, we inventively design a series protocols to prevent users from misbehaving with no need for depositing into our smart contracts. The misbehaviors in our system contain: not sending compensation by users, not performing tasks by workers, not sending rewards by requesters, and creating redundant user information. The main idea of the protocol is to *link several steps together to ensure that only if users behave properly in the previous steps can they perform the following steps.*

The linkage between *TaskMatching* and *CompensationVerify*. *To prevent users from not compensating to the requesters who performed the task assignments,* the *CompensationVerify* smart contract will verify the compensation status. Specifically, when a worker/requester receives the task assignment result and the claimed compensation, it needs to send the corresponding compensation to requester r_j through a transfer. This transfer will trigger a smart contract called *CompensationVerify*, and the smart contract will search the data, in EOS database, which containing the amount of required compensation and generated by *TaskMatching*. *CompensationVerify* will then check whether the compensation in the transfer is equal to the required compensation. If and only if a worker/requester indeed sends the transfer and the compensation equation holds, it will be marked as valid in *CompensationVerify* and the validation data will be saved in the EOS database, as well one can perform the following steps.

After a worker receives the task assignment result and sends the compensation to requester r_j, it can physically move to the location of the assigned task and perform it. After the task is performed, the worker needs to submit the task result along with its location onto the blockchain through the *TaskSubmission* smart contract. The location, which can be obtained through the GPS on the smartphone, is necessary to be reported since it can verify whether the worker is indeed at the task location.

The linkage between *CompensationVerify* and *TaskSubmission*. Once a worker triggers the execution of the *TaskSubmission* smart contract, it will first check whether the worker is valid by searching the validation data generated by *CompensationVerify*. If the worker is invalid, the *TaskSubmission* smart contract will reject the worker's task submission request and hence the worker cannot receive any task reward in the following steps. Otherwise, the smart contract will record the task result and the location of the worker into the blockchain. In addition, the *TaskSubmission* smart contract will also mark the worker for this task as offline by removing the corresponding worker information from the EOS database. If multiple worker information coexists for a worker, the smart contract will only remove the one related to performing this specific task.

The linkage between *CompensationVerify* and *Reward*. After the worker successfully submit the task result, the requester who owns the task need to trigger the *Reward* smart contract to perform task evaluation and send reward to the worker who performed the task. The *Reward* smart contract will first search and check the validation data of the requester to scrutinize whether the requester has submitted the correct compensation to requester r_j. If the compensation has not been sent or the amount of compensation is incorrect, representing the requester is invalid, the task evaluation request will get rejected by *Reward* and the requester cannot conduct further steps. If the requester is valid, the *Reward* smart contract will perform the task evaluation according to an evaluation function to judge the quality of the task performing results. In addition, the smart contract will calculate the amount of reward and send proper reward to the specific worker. At the end, the *Reward* smart contract will mark the task as offline and wipe up the corresponding task information from the EOS database.

The linkage between *TaskSubmission/Reward* and *WorkerReleasing/TaskReleasing*. *We manage the whole process as a* circle *to prevent more improper behaviors.* Specifically, a user could misbehave to *create multiple accounts*, publish multiple redundant user information to the blockchain. As a result, a misbehaving worker may not perform the assigned tasks, meanwhile a misbehaving requester may refuse to pay for the rewards. We tackle this issue *using the power of the certification authority (CA).* Each user should first register and get certificated at CA before joining in, to bind its real identity to a unique account in our system.

A misbehaving user could also *publish multiple redundant user information even with a single account.* To solve this problem, *we set a* quota *for each worker and requester.* The quota is a system-defined value set in the *TaskReleasing* and *WorkerReleasing* smart contracts, representing the maximum number of task/worker information that can be simultaneously coexisted for one requester/worker. If the number of the online task/worker information for a given requester/worker does not exceed the quota value, the requester/worker can further publish their information onto the blockchain, otherwise the *TaskReleasing/WorkerReleasing* smart contract will reject the requester's/worker's request.

Therefore, if a worker refuse to perform the assigned tasks or a requester refuse to pay for the rewards for multiple times, the number of its online task/worker information will eventually reach its quota, leading to a result that the user cannot publish any information anymore.

Discussion of the protocol. We assume the users in the system want to use the system in long-term. Therefore, the linkage protocol will prevent users from misbehaving since it link each step together, and if a user misbehaves, it will stuck and cannot do any further actions. However, our design has

TABLE II: Preference Rules for Workers and Requesters

For Workers (Choose Tasks According to)	For Tasks (Choose Workers According to)
Distance	Distance
Payment	Reputation
Difficulty	Expertise
Deadline	Time to reach the task

limited prevention for the misbehaving users who only want to use the system for few times. To alleviate this issue, we use the power of the certification authority (CA). The CA will evaluate the credit of a user in real world, and only certificates the users who are credible, and bind their real identity to a unique account in our system (i.e., it prevents Sybil Attacks). When a user perform misbehavior in the system, its real-world credit will also get downgraded. Therefore, rational users will reduce their misbehavior and behave properly in our system.

It is worth noting that EOS requires users to stake tokens for the usage of CPU and NET and the tokens will be returned after the usage of CPU and NET. Such kind of deposit does not belong to the category that "deposit to the smart contract of an untrusted system". The reason is that the tokens to exchange CPU and NET is actually deposited in EOS itself, while the tokens that need to be deposited in the system is deposited to the application built atop of the blockchain. A practical thinking is that the blockchain has a higher degree of credibility than the applications built upon it. Hence we make the assumption that the blockchain itself can be trusted while the systems built on the blockchain has lower credibility.

Isolate the permission of the data modification. As mentioned in Sec. II, the data is saved in the EOS database. In addition, the database can be public seen by anyone and the data in it could, by default, be accessed by any user. However, a misbehaving user can modify other users' data if there is no restriction on the data modification permission. Hence, it is essential to thoroughly manage the data modification permission control in our system.

In our SC-EOS system, *each user only has the permission to modify the data belongs to itself.* However, this permission limitation makes the system design more sophisticated. For example, after the task assignment mechanism is performed, the status of whether a user is successfully assigned should be updated. A most convenient way to achieve this is to directly change the status for every user. However, in our design, the requester r_j who performs the task assignment is not permitted to modify any data that belongs to others. Thereby, we design to record the user states in its own data field, and the one who needs to get these states can search for the fields that store the data and read it.

V. Evaluation

In this section, we will show the results of our proposed SC-EOS to see its economic cost and task assignment performance.

TABLE III: Monetary Cost Evaluation for Single User

Requester/Worker ID	#1	#2	#3	#4	#5
Monetary cost for requesters ($)	0.26	0.23	0.23	0.24	0.25
Monetary cost for workers ($)	0.11	0.10	0.11	0.11	0.10

A. Evaluation Setup

We implemented a prototype of our system based on EOS blockchain to test our framework and scheme on a test network, since the underlying architecture of EOS has been modified to better match our framework. The test network consists of three PCs to play the role of super nodes, they are Lenovo with Ubuntu 16.04 LTS installed, equipped with Intel Core i5-8400 CPU and 16GB main memory, along with five laptops mimic requesters and five laptops act as workers.

We first evaluated the monetary cost caused by the RAM consumption for single user in our SC-EOS system. We then evaluated the economic cost of our SC-EOS and also implemented a simplified version of our spatial crowdsourcing framework on Ethereum as a comparison. Finally, the evaluation also includes the performance of our proposed user-customizable blockchain-specialized task assignment scheme.

The last two experiments are performed on the NYC TLC Trips dataset. We used the data from trips completed in Yellow taxis in NYC in the year of 2016. We assumed that the workers' locations were at the drop-off locations while tasks were at the pick-up locations. The tasks' arrival times are set as the pick-up times in the dataset while the arrival of workers are set as obeying the Poisson process with a default arrival rate $\lambda = 1/2$ per min, which means the arriving interval $\alpha = 2min$. As for the waiting times of workers and tasks, we use a widely accepted assumption that they obey Pareto distribution, in which the default values are set as minimum waiting time $x_{min} = 10min$, and the shape parameter $k = 3$.

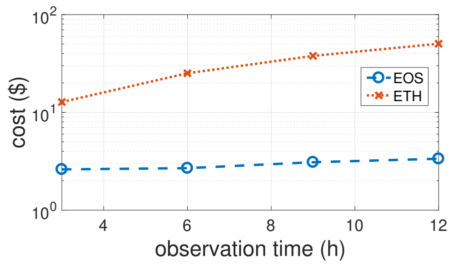
B. Monetary Cost Evaluation for Single User

We employed 5 requesters and 5 workers in different locations to imitate the spatial crowdsourcing process in SC-EOS, where each requester had one food take-away request, and each worker delivered the food for the assigned requester. We mainly evaluated the monetary cost for each user due to the RAM consumption. The evaluation results are illustrated in Table III.

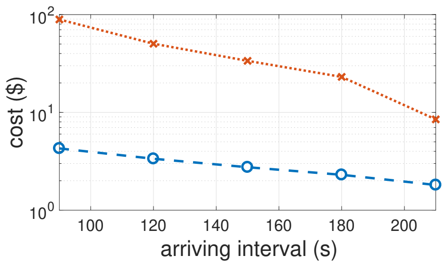
The results show that both workers and requesters only cost few in our system. As a comparison, the well-known spatial crowdsourcing platform Gigwalk will typically charge the requester for several dollars for publishing one task [5]. The cost for a requester is usually higher than a worker mainly because of more information contained in a task.

C. Overall Economic Cost Evaluation

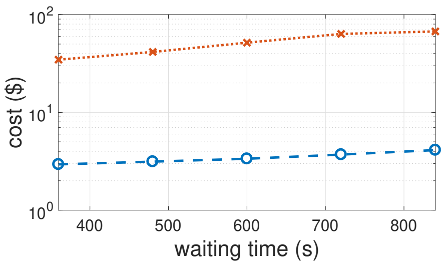
We conducted the evaluations to measure the system overall economic cost. Especially, we implemented a prototype of our



(a) Observation time vs. cost.



(b) Arriving interval vs. cost.



(c) Waiting time vs. cost.

Fig. 4: Overall Cost Evaluation.

spatial crowdsourcing system atop of EOS (SC-EOS) and also a simplified version on Ethereum, and compare them to see how much SC-EOS can save money for users.

We assume that every worker/requester will sell its RAM after using, and will buy RAM when it is needed. We first evaluated the overall cost change over time, Fig. 4a illustrates the result. We then changed worker arriving interval and evaluated for 12 hours to observe its influence to the overall cost and shows the result in Fig. 4b. We also performed the evaluation to see the influence to the overall cost in 12 hours by changing task and worker waiting time, where the result is shown in Fig. 4c.

From the results we can see that the overall cost in SC-EOS is always much lower (more than 10× lower) than the spatial crowdsourcing system built in Ethereum in any situation above no matter how the parameters change. The main reason is that the economic cost for SC-EOS is mainly produced by buying RAM. The RAM cost is a one time cost and one can sell it after using, and only need to be charged for 0.5% transaction fee by the EOS blockchain. However, in Ethereum, each operation will consume gas, which will cause transaction fees for users. The consumption of the gas is related to the operations and cannot be sell like the RAM in EOS.

D. *Task Assignment Scheme Evaluation*

We presented the task assignment mechanism performance in this subsection. We mainly considered two metrics to illustrate the performance of our task assignment mechanism. The first one is the number of workers/tasks that are successfully matched. The second one is named "rank percent". Since each worker or requester has its own preference rule (e.g. r_j likes to order workers by distances, while w_i prefer to order tasks by payments), it is nonsense to only choose one rule (e.g. distance) to illustrate the mechanism performance. Thus, we used rank percent to represent the top percent of ranking in the total preference list for a worker/task who is matched to another task/worker. For example, if the length of a preference list of a worker w_i is 10 and the matched task t_k is the second preferred task for that worker, the rank percent is hence 20%.

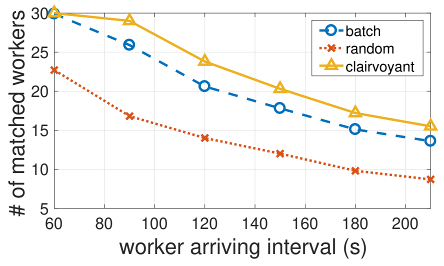
We compared our task assignment scheme with a naive random matching scheme and a clairvoyant matching scheme.

- Naive random matching. Each task, once it is online, will search all the workers and tasks that are waiting at that time and perform a random matching. However, only the matching result related to itself is valid.
- Clairvoyant matching. The system knows all the future tasks and workers and can do stable matching at any time point with the combination of the future information.

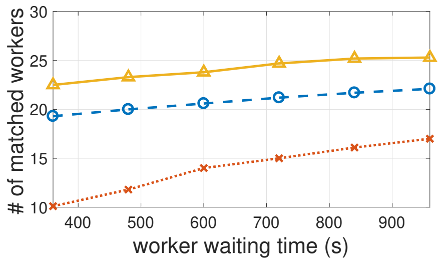
In this evaluation, we randomly chose 30 tasks from the dataset and the workers are keep arriving. We conducted a series of evaluations to see the number of matched workers/tasks and rank percent results by changing the worker arriving interval, worker waiting time, the shape parameter k value, and the task waiting time. The results are shown as Fig. 5. From the results we can conclude that our mechanism achieves sub-optimal performance compared with the clairvoyant mechanism, in both the number of matched workers/tasks and rank percent. The main reason for the gap is that the clairvoyant scheme can predict the future and do matching at any time point, yet this clairvoyant scheme is impractical in real world systems. The random matching scheme performs the worst in matched number, since it doesn't apply batched algorithm and only let each task to do matching for itself. The random matching also has the highest rank percent, which is also the worst, since it only use random matching and hence the user may not satisfied with its matching result.

VI. Conclusions and Future Directions

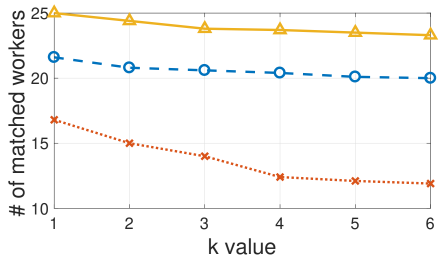
In this paper, we presented the design of SC-EOS, an EOS-based framework for spatial crowdsourcing. We analyzed that the traditional centralized spatial crowdsourcing system suffers from the problems such as lack of operation transparency, single point of failure and high services fee, while previous blockchain-based crowdsourcing frameworks hold the issues of depositing to an untrusted system, naive task assignment scheme, low throughput and high transaction fee. We formalized SC-EOS to handle these problems. We carefully designed our spatial crowdsourcing paradigm and modified EOS to make them a better fit. Specifically, we proposed a user-customizable and batched task assignment mechanism specially built for blockchain, along with a series of protocols based on smart contract to prevent users from misbehaving without the need for a deposit. Besides, we



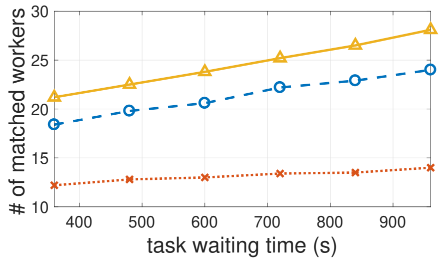
(a) Worker arriving interval vs. # of matched workers/tasks.



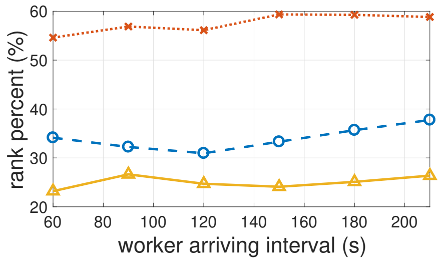
(b) Worker waiting time vs. # of matched workers/tasks.



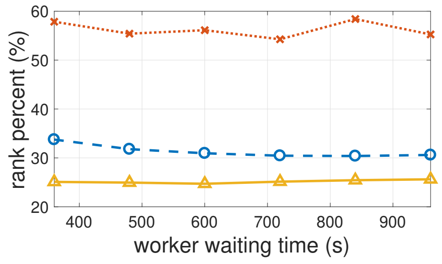
(c) k value vs. # of matched workers/tasks.



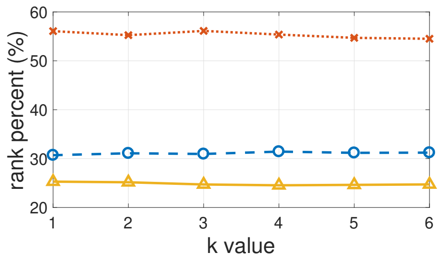
(d) Task waiting time vs. # of matched workers/tasks.



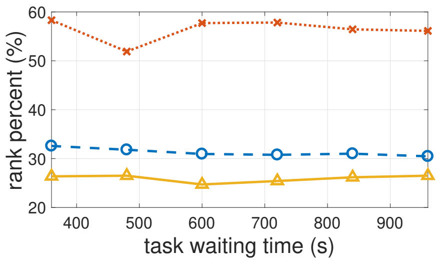
(e) Worker arriving interval vs. rank percent.



(f) Worker waiting time vs. rank percent.



(g) k value vs. rank percent.



(h) Task waiting time vs. rank percent.

Fig. 5: Task Assignment Scheme Evaluation.

evaluated our approach on EOS based on real users and real-world dataset. Evaluation results shown that our SC-EOS can achieve efficient task assignment with low economic cost.

We are still in the early stage of blockchain technology, let alone the blockchain applications in spatial crowdsourcing. We now briefly summarize some potential meaningful future work. First, the task matching algorithm is executed on blockchain by utilizing smart contract, yet the logical scheme can be considered to run in somewhere else to save computational resource. Second, privacy protection can be considered in the future blockchain based spatial crowdsourcing system design as privacy is a significant concern for users. Third, an efficient evaluation mechanism is crucial in any spatial crowdsourcing system, while how to design an efficient evaluation scheme based on blockchain is still worth researching. Finally, spatial crowdsourcing requires workers to physically travel to the task locations to perform tasks. However, how to verify whether the workers are indeed at the specified locations is an urgent problem, especially for blockchain since blockchain is based on a trustless model.

References

[1] https://www.cryptocompare.com/coins/guides/the-dao-the-hack-the-soft-fork-and-the-hard-fork/.
[2] https://medium.com/futurepia/fastest-transaction-speed-mainnet-2eb3799bbed2.
[3] Eos. https://eos.io/.
[4] Ethereum. https://www.ethereum.org/.
[5] Gigwalk. http://www.gigwalk.com.
[6] Moonlight. https://moonlight.io.
[7] Uber. https://www.uber.com.
[8] Waze. https://www.waze.com.
[9] V. Buterin et al. A next-generation smart contract and decentralized application platform.
[10] Z. Chen, P. Cheng, L. Chen, X. Lin, and C. Shahabi. Fair task assignment in spatial crowdsourcing. *Proceedings of the VLDB Endowment*, 13(12), 2020.
[11] M. Crosby, P. Pattanayak, S. Verma, V. Kalyanaraman, et al. Blockchain technology: Beyond bitcoin. *Applied Innovation*, 2(6-10):71, 2016.
[12] W. Feng, Z. Yan, H. Zhang, K. Zeng, Y. Xiao, and Y. T. Hou. A survey on security, privacy, and trust in mobile crowdsourcing. *IEEE Internet of Things Journal*, 5(4):2971–2992, 2018.
[13] S. R. B. Gummidi, X. Xie, and T. B. Pedersen. A survey of spatial crowdsourcing. *ACM Transactions on Database Systems (TODS)*, 44(2):1–46, 2019.
[14] B. Guo, Y. Liu, L. Wang, V. O. Li, J. C. Lam, and Z. Yu. Task allocation in spatial crowdsourcing: Current state and future directions. *IEEE Internet of Things Journal*, 5(3):1749–1764, 2018.
[15] J. Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.

[16] S. Hu, L. Su, H. Liu, H. Wang, and T. F. Abdelzaher. Smartroad: Smartphone-based crowd sensing for traffic regulator detection and identification. *ACM Transactions on Sensor Networks (TOSN)*, 11(4):55, 2015.

[17] H. Jin, L. Su, and K. Nahrstedt. Theseus: Incentivizing truth discovery in mobile crowd sensing systems. In *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, page 1. ACM, 2017.

[18] B. Kantarci and H. T. Mouftah. Trustworthy sensing for public safety in cloud-centric internet of things. *IEEE Internet of Things Journal*, 1(4):360–368, 2014.

[19] L. Kazemi and C. Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *Proceedings of the 20th international conference on advances in geographic information systems*, pages 189–198. ACM, 2012.

[20] C. Li, B. Palanisamy, R. Xu, J. Wang, and J. Liu. Nf-crowd: Nearly-free blockchain-based crowdsourcing. In *2020 International Symposium on Reliable Distributed Systems (SRDS)*, pages 41–50. IEEE, 2020.

[21] M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, J.-N. Liu, Y. Xiang, and R. Deng. Crowdbc: A blockchain-based decentralized framework for crowdsourcing. *IEEE Transactions on Parallel and Distributed Systems*, 2018.

[22] J. Lin, M. Li, D. Yang, G. Xue, and J. Tang. Sybil-proof incentive mechanisms for crowdsensing. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[23] Y. Lu, Q. Tang, and G. Wang. Zebralancer: Private and anonymous crowdsourcing system atop open blockchain. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 853–865. IEEE, 2018.

[24] C. Miao, L. Su, W. Jiang, Y. Li, and M. Tian. A lightweight privacy-preserving truth discovery framework for mobile crowd sensing systems. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[25] S. Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. 2008.

[26] V. Pankratius, F. Lind, A. Coster, P. Erickson, and J. Semeter. Mobile crowd sensing in space weather monitoring: the mahali project. *IEEE Communications Magazine*, 52(8):22–28, 2014.

[27] P. Praitheeshan, L. Pan, J. Yu, J. Liu, and R. Doss. Security analysis methods on ethereum smart contract vulnerabilities: a survey. *arXiv preprint arXiv:1908.08605*, 2019.

[28] L. Tan, H. Xiao, K. Yu, M. Aloqaily, and Y. Jararweh. A blockchain-empowered crowdsourcing system for 5g-enabled smart cities. *Computer Standards & Interfaces*, 76:103517, 2021.

[29] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, and C. Shahabi. Spatial crowdsourcing: a survey. *The VLDB Journal*, 29(1):217–250, 2020.

[30] C. Wang, H. Liu, K.-L. Wright, B. Krishnamachari, and M. Annavaram. A privacy mechanism for mobile-based urban traffic monitoring. *Pervasive and Mobile Computing*, 20:1–12, 2015.

[31] J. Wang, M. Li, Y. He, H. Li, K. Xiao, and C. Wang. A blockchain based privacy-preserving incentive mechanism in crowdsensing applications. *IEEE Access*, 6:17545–17556, 2018.

[32] L. Wang, D. Yang, X. Han, T. Wang, D. Zhang, and X. Ma. Location privacy-preserving task allocation for mobile crowdsensing with differential geo-obfuscation. In *Proceedings of the 26th International Conference on World Wide Web*, pages 627–636. International World Wide Web Conferences Steering Committee, 2017.

[33] H. Wu, L. Wang, and G. Xue. Privacy-aware task allocation and data aggregation in fog-assisted spatial crowdsourcing. *IEEE Transactions on Network Science and Engineering*, 7(1):589–602, 2019.

[34] M. Xiao, K. Ma, A. Liu, H. Zhao, Z. Li, K. Zheng, and X. Zhou. Sra: Secure reverse auction for task assignment in spatial crowdsourcing.

[38] D. Yuan, Q. Li, G. Li, Q. Wang, and K. Ren. Priradar: A privacy-preserving framework for spatial crowdsourcing. *IEEE transactions on information forensics and security*, 15:299–314, 2019.

IEEE Transactions on Knowledge and Data Engineering, 32(4):782–796, 2019.

[35] P. Xiong, L. Zhang, and T. Zhu. Reward-based spatial crowdsourcing with differential privacy preservation. *Enterprise Information Systems*, 11(10):1500–1517, 2017.

[36] K. Yang, K. Zhang, J. Ren, and X. Shen. Security and privacy in mobile crowdsourcing networks: challenges and opportunities. *IEEE communications magazine*, 53(8):75–81, 2015.

[37] S. Yang, K. Han, Z. Zheng, S. Tang, and F. Wu. Towards personalized task matching in mobile crowdsensing via fine-grained user profiling. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 2411–2419. IEEE, 2018.

[39] Y. Zhao and Q. Han. Spatial crowdsourcing: current state and future directions. *IEEE communications magazine*, 54(7):102–107, 2016.