CS6675: Study of Blockchain based Crowdsourcing System

ChangSeok Oh, Linh Hoang, Kedi Zheng
Georgia Institute of Technology

Abstract

Abstract is here.

1 Introduction

Crowdsourcing is one of sourcing models that describe how individuals or organizations supply necessary resources such as idea, goods, finances, human labors, etc. In this model, a huge job that looks nearly impossible to be achieved by an individual is split into multiple small tasks, and many participants who are interested in the job on internet perform them independently [10]. This idea sounds clever but not brand-new. We have observed many applications of the crowdsourcing model already (e.g., Wikipedia, LEGO ideas, Airbnb, etc.).

In 2005, Amazon introduced a crowdsourcing platform called Amazon Mechanical Turk. The key idea of the service is to provide a centralized marketplace with a rewarding system [1]. It is a brilliant idea, and seems to achieve certain success. However, we think the centralized labor marketplace still has some issues. First, the centripetal service charges expensive fees for maintenance (up to 40% depending on the amount of work). Second, Amazon Mechanical Turk does not provide a way to verify the quality of completed jobs. What job requesters can do is to just rely on a consensus among works who did the same work. Third, a generic interface is not provided, so job requesters should spend extra money to hire software developers. Lastly, Amazon's labor trading system only accepts a traditional banking system that prevents many of workers who do not want to reveal themselves from participating the crowdsourcing.

We will tackle these problems, using blockchain technology. Theoretically speaking, the blockchain should not suffer from aforementioned problems since it follows a distributed transaction model. There is no central authority that controls requests and works in the distributed model so that neither commision nor maintenance fee is required. Also, we can combine any banking system with the internal currency of blockchain. Some researchers and companies already did similar jobs [4, 8]. However, they are either proprietary or close-sourced so we could not access their work to analyze the blockchain-based crowdsourcing system.

This project aims to prototype our own crowdsourcing system running on blockchain technology, and identify its pros and cons. Clearly speaking, our goal is not to build a perfect and secure crowdsourcing system on the blockchain network, rather to explore possibilities and limitations of the technology when implementing crowdsourcing system on top of blockchain technology.

We present the design of our blockchain based crowdsourcing system in §2, and describe implementation details in §3. In §4.6, we deal with evaluation results of our crowdsourcing system with blockchain technology. Lastly, we summarize our findings and achievement in §5.

2 Design

We prototyped a blockchain-based crowdsourcing system in BMTURK, using Ethereum platform. Figure 1 shows overall system composition and design overview of BMTURK. There are three distinct roles in our system: a requester, a worker and a verifier. But they are performed by all participants on the crowdsourcing overlay network, and implemented as a distributed application for Ethereum platform. It follows four steps: (1) A requester creates a job (i.e., a contract), and posts it in the network; (2) Workers find the contract and see which task is available; (3) Workers conduct part of works for the contract and submit them at will; (4) Verifiers pick up the works done by workers, validate them, and notify results to task performers. We classified this procedure, realized each step into adequate protocol, and implemented the protocol on top of ethereum platform. In addition, to help participants expedite each step, we built user interface by using web technology.

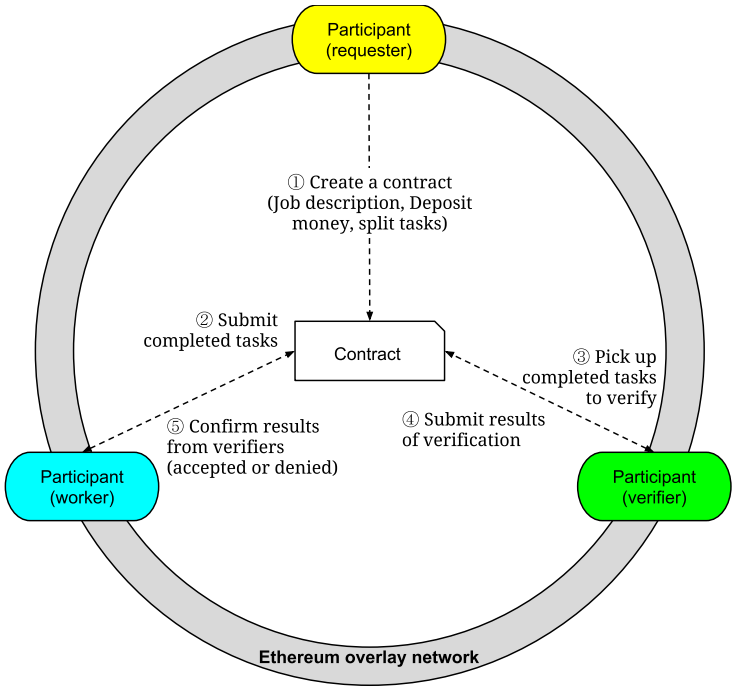

Figure 1: Design Overview of BMTURK

Figure 2 shows a simplified sequential workflow of BMTURK. The workflow starts with a participant who create a job to request. (i.e., a contract.) When creating a contract, the requester puts detailed information such as job description, success criteria, reward, the number of split tasks, and so on. The created contract is deployed to other participants in the network. When it is done, every participants can see what tasks are available in user interface we provide, conduct a task, and submit a completed task to the system. Then, the submitted tasks spread into the network again to get verifications from other participants (i.e., verifier). Once a completed task is inspected, the result is notified to the workers who completed the task. BMTURK displays the result as either 'accepted' or 'rejected'. For rejected tasks, a worker can choose either re-working or discarding the task. We designed that rewards are granted when all completed tasks are accepted, and the rewards are paid from the initial fund of the contract. Where the money runs out, the requester can put more money for the contract later. This is the basic workflow of BMTURK.

We admit this system might not handle all edge cases that can be exploited by a malicious participant.

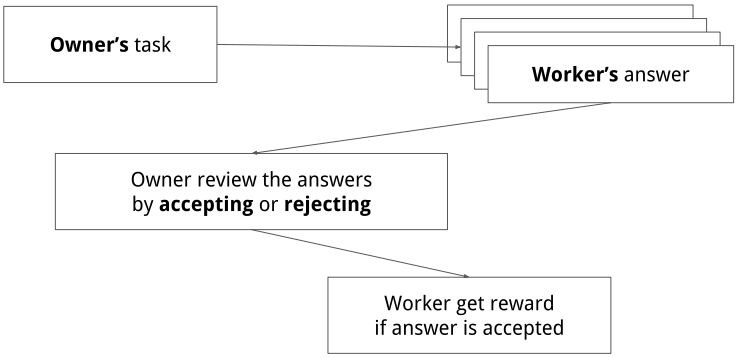

Figure 2: BMTURK workflow

For example, he can intentionally repeat incorrect verification jobs to seek verification fees. To prevent this, BMTURK may need a kind of reputation system. However, this is out of scope. Our goal is to analyze pros and cons of a blockchain based crowdsourcing system, not build a perfect crowdsourcing system. We leave the reputation system as a future work.

3 Implementation

For fast proof-of-concept, we simplified the whole design, but implemented essential protocols to make BMTURK similar to Amazon Mechanical Turk. The simplified workflow of BMTURK also begins from a job requester who creates a contract for crowdsourcing. In the contract, the requester can leave a job description, a compensation (e.g., amount of money), and the number of workers allowed. Once the job requester committed a contract, it is deployed across the Ethereum network and posted on a public bulettin. Workers can choose a suitable task according to their skill set, interests, and the reward. When they submit answers after finishing their tasks, the job requester can decide whether or not he accepts the answers. If he does, all workers who participated in the contract will receive the promised compensation. Otherwise, another or same worker can retry rejected jobs.

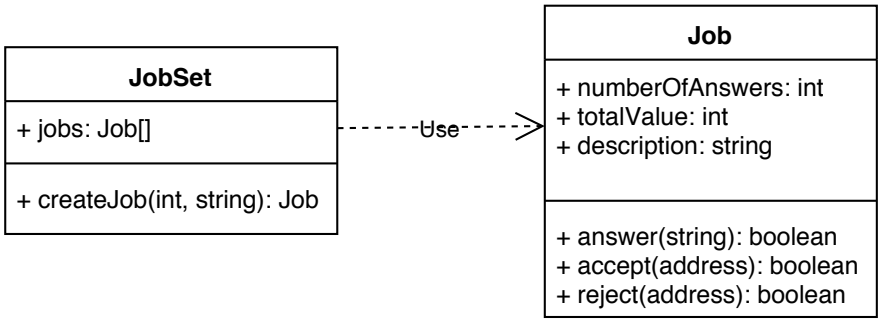

Figure 3: Smart contract design

BMTURK consists of two parts: a backend and a frontend. We implemented the backend, using IPFS [5] on the Rinkeby network which is a test net for Ethereum blockchain. It uses **proof-of-authority consensus algorithm** [6]. The IPFS is a peer-to-peer distributed file system that aims to connect all computing devices into one file system. We used it to lower the cost of deploying information on the blockchain as each task description can be represented by a 256-bit SHA-1 hash. Because this hash can be used to find the file on

IPFS nodes, it significantly lowers the cost of transfering data on the blockchain. To make the backend work on the Rinkeby, we implemented two smart contracts: (1) JobSet, (2) Job. Figure 3 shows relation between the two contracts. The JobSet represents a contract created by a requester so that it contains a full list of job instances. The two smart contracts provide basic but same functionalities of Amazon Mechanical Turk. We detailed each protocol in Table 1.

Protocol	Description
JobSet	
jobs: Job[]	The list of all crowdsourcing contracts
createJob(int, string): Job	The method to create a crowdsourcing contract
Job	
numberOfAnswers: int	The number of needed answers
totalValue: int	The total rewards of this contract
description: string	The description of this contract
answer(string): boolean	The method used by the worker to submit his work
accept(address): boolean	The method used by the owner to accept an answer
reject(address): boolean	The method used by the owner to reject an answer

Table 1: Details of BMTURK's smart contracts

On the other had, the frontend of BMTURK is implemented by using web technology (i.e., html, css, javascript). It can be either hosted on a web server or locally run on users' machine after download. Two browser extensions: MetaMask and IPFS are necessary to run BMTURK. MetaMask is a browser extension to connect to a blockchain node, and IPFS companion is used to connect with a IPFS node. BMTURK provides two views for users to facilitate crowdsourcing works. One is a browsing view that shows the full list of contracts, and the other is a contract view that shows details on each contract.

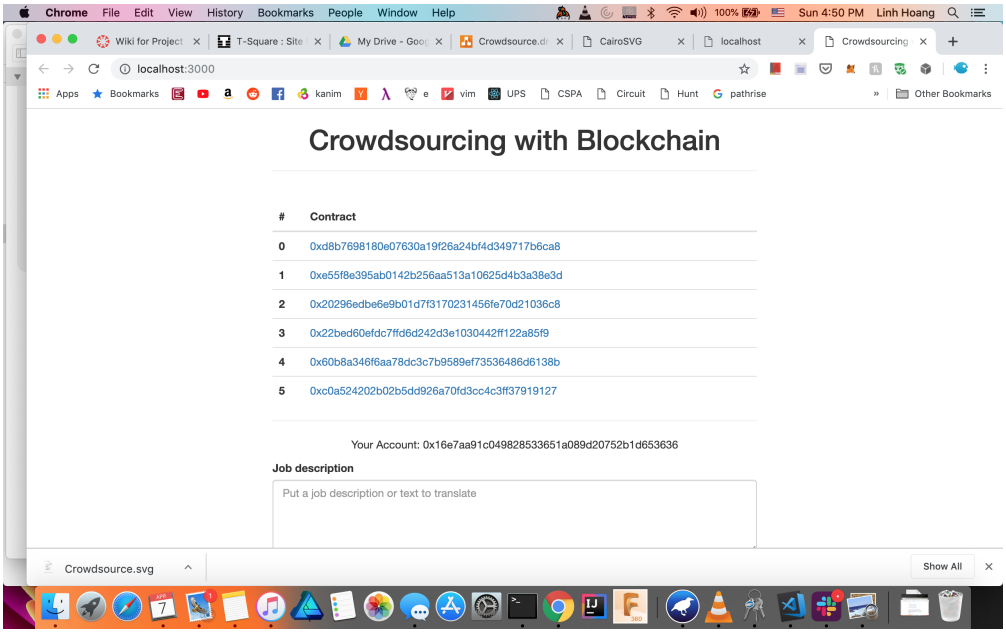

Figure 4: The browsing view

Figure 4 shows a snapshot of the browsing view. It contains the list of crowdsourcing contracts that are on the network and an interface to create the crowdsourcing contract on the blockchain. To create a crowdsourcing contract, one need to input the crowdsourcing task description, the reward to pay for each accepted answer, and the number of answers he wants into the text box.

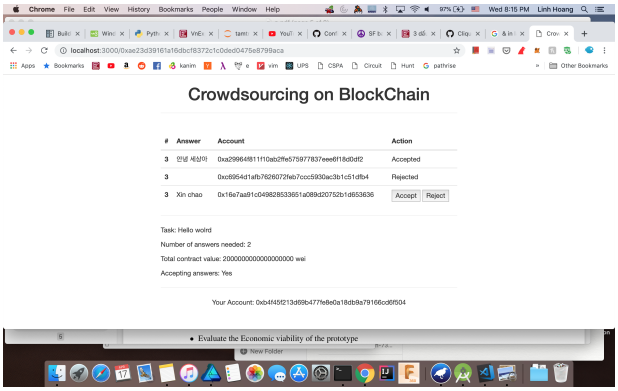

Figure 5: The owner's contract view

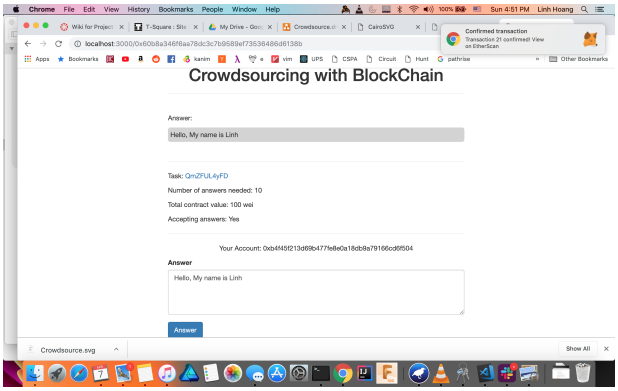

Figure 6: The worker's contract view

Figure 5 and Figure 6 show contract views for an owner and a worker, respectively. Both show the information (task description, reward,...) of the crowdsourcing contract and whether the contract is still accepting answers. The owner view contains the list of answers and the states of the answers. The owner reviews the answer by clicking on the Accept and Reject button next to the answer. The worker view contains an input box for the answer and his currently submitted answer. The worker view also displays whether his answer is accepted or rejected

4 Evaluation

This section will provide us with the experimental results of our prototype. Our experiment will use translation as a example problem for crowd sourcing. And our goals is the

- Evaluate the latency of the prototype
- Evaluate the Economic viability of the prototype

Since this paper mainly focus on functionality of this prototype, other measures for evaluations such as security, network's capacity will be topics for future development of this prototype.

4.1 Experiment setup

Our experiment machines are two computers connected to Rinkeby testnet, one as contract creater and another as worker via the MetaMask extension. Since computers serve as thin clients to our backend network, it is reasonable to assume that the performance for two clients in the network is smiliar to the performance of multiple clients using our prototype on the same network.

4.2 Translation as an example

We use translation as an example crowd sourcing problem. Translation has been one of the first problem that has been addressed by crowd sourcing in the internet age. [11]. Translation has several properities that makes

it an example of our prototype: (1) Translation task can be split up into sub tasks without too much loss of information. (2): Translation tasks don't require too much technical expertises. (3): The pecularity and informality of the human languages makes translations hard for machines(for now). These three properities make crowd-sourcing a good starting problem for our prototype.

For our evaluation, we used Wikipedia articles in English with various sizes as the contract's task and the translated Chinese as the answer.

4.3 Evaluation data set

We used text excerpt from Wikipedia [9] and picked paragraphs of size ranging from 500 words to 1000 words. We will use plain text, instead of IPFS, as transmission method. Plain text is much bigger than IPFS link, and therefore allows us to measure how the network perform with heavier payload. Also IPFS utilizes P2P technology and is therefore hard to find stable connections in some network, such as coporate and university network.

The translated text will be in Chinese with similar word counts.

4.4 Latency

Latency is measured as **contract propagation time - contract create time**. It will vary depend on the payload in our contract.

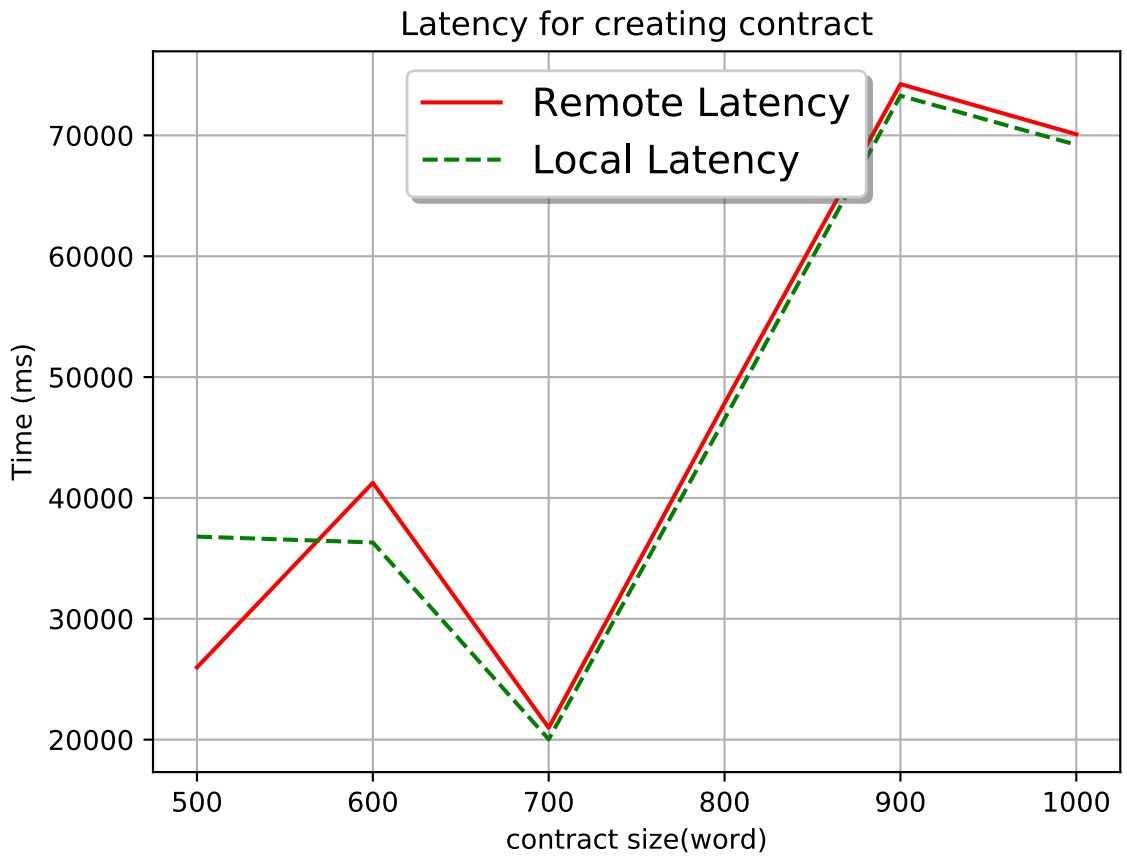

Figure 7: Latency

Figure 7 two line, the green line is local latency, how long does it take for the contractor to see the contract on the block chain. The red line is remote latency: how long does the worker can see the contract on the block chain. As the payload increase, so does the latency. It ranges from 40 seconds to 70 seconds.

We also noticed that latency dipped for payload of 700 words. Because we are use MetaMask, a public test network, instead of a private block chain network, so the performance would vary depends on how busy the network is.

These two lines are closely coupled with each other because for both the worker and the contract creator, they both connect to the MetaMask network and the network will notify them only when the payload is propagated to all nodes in the network.

A latency of 40 to 70 seconds might seem a lot in a performance perspective. But since we are using plain text, the size of the payload will be in the kilobytes. If use IFPS, the payload will only be at most 100 bytes, reducing the payload size by at least a magnitude.

4.5 Cost

Since we are using Ethereum network as our backend, we need to pay the miners for their work that verify our transactions. In Ethereum terminalogy, this cost of using the network is called **gas**. Just like gas for automobiles, gas in Ethereum will be used to propell payloads and the heavier the payload, the more gas

it will take. Cost is measured in ETH(Ethereum) per transaction. It will vary depend on the payload in the contract.

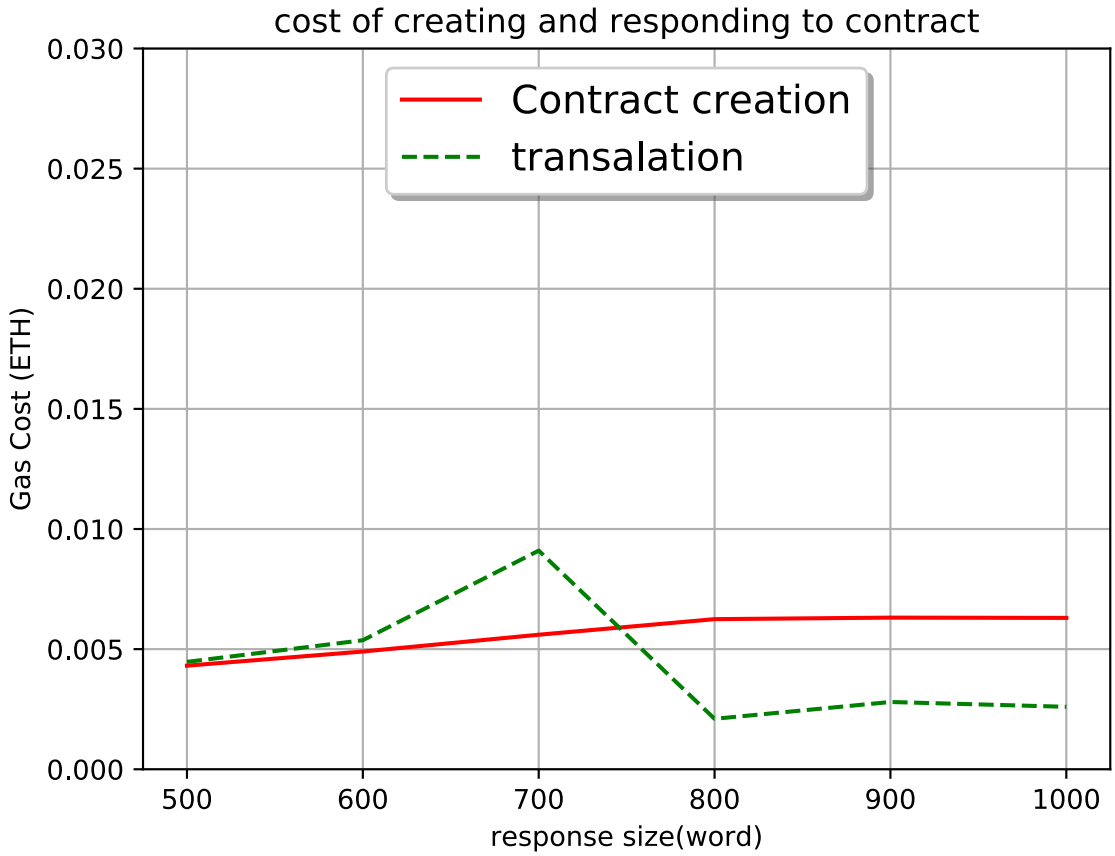

Figure 8: Cost in terms of ETH

We can see the price didn't flutuate as much as the payload varies. A payload of 500 words and a payload of 1000 words belong to the same magnitude so the cost didn't vary as much. And If we switched to IPFS payload, the cost will drop down 50 times to around 0.0001 ETH per transaction. At the time of this writing, 1 ETH coin is worth $\$166.62$ [3]. That will bring our transaction to $\$0.83$ for plain text and $\$0.016$ for IPFS.

4.6 Economic viability compared with other crowd sourcing products

In the crowd sourcing market place, the centralized vendors provide a platform for contract creator and workers and in exchange, the vendors will take a commission out of the contract. Amazon Mechnical Turk charges $20\% - 40\%$ depending on the type of tasks [2]. For comparision, Amazon Mechnical Turk price for translating a sentence is about $\$0.10$ each [7] and an sentence averages to 10 words. The price for translating will be $\$10$ and the commision charged will be $\$2$. On the other hand, using our prototype with IPFS, the cost would be $\$0.016$.

The upside becomes bigger as the task becomes bigger. Since we are using IPFS, the cost will almost always remain constant. At the same time, the cost of using Amazon MTurk will go up with the size of the

translation job.

5 Conclusion

In this project, we have created a crowd sourcing protocol and built a proof-of-concept prototype that implemented parts of the protocol. The prototype can run on any Etherium network can be used as a distributed platform for crowd sourcing tasks such as translation. We evaluatd our prototype using translation as a real world problem and have shown that though the speed of creating contract is not ideal, but it is almost negligible compared with the time it takes a human to do translation. We have also shown that the cost benefit of using our product: It doesn't charge commission and will only incur a flat rate for propagating the payload to the entire network. If using IPFS and other file transfer system to transmit the actual payload, the price paid for creating contract is very cheap, around $\$0.01$ per contract.

References

[1] Amazon. MTurk, 2019. https://www.mturk.com.

[2] Amazon. mechnicalTurk-pricing, 2019. https://www.mturk.com/pricing.

[3] C. Base. ETH-price, 2019. https://www.coinbase.com/price/ethereum.

[4] icrodrops.com. Gems, 2019. https://icodrops.com/gems.

[5] IPFS. IPFS, 2019. http://ipfs.io.

[6] karalabe. Clique PoA protocol & Rinkeby PoA testnet, 2019. https://github.com/ethereum/EIPs/issues/225.

[7] C. C.-B. Michael Bloodgood. Using Mechanical Turk to Build Machine Translation Evaluation Sets, 2012. http://cis.upenn.edu/~ccb/publications/using-mechanical-turk-to-build-machine-translation-evaluation-sets.pdf.

[8] Microwork. Microwork, 2019. http://www.microwork.io.

[9] Wikipedia. translated-article, 2019. https://en.wikipedia.org/wiki/Vietnam.

[10] Wikipedia. Crowdsourcing, 2019. https://en.wikipedia.org/wiki/Crowdsourcing.

[11] Wikipedia. wiki-translation, 2019. https://en.wikipedia.org/wiki/Wikipedia:Translate_us.