symmetry

MDPI

Article

A Blockchain-Based Distributed Computational Resource Trading Strategy for Internet of Things Considering Multiple Preferences

Tonghe Wang [1], Songpu Ai [2], Junwei Cao [3] and Yuming Zhao [4,*]

[1] Guangzhou Institute of Energy Conversion, Chinese Academy of Sciences, Guangzhou 510640, China; wangth@ms.giec.ac.cn
[2] Beijing Teleinfo Network Technology Co., Ltd., Beijing 101399, China; aisongpu@teleinfo.cn
[3] Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China; jcao@tsinghua.edu.cn
[4] School of Computer Science and Software, Zhaoqing University, Zhaoqing 526061, China
* Correspondence: ymzhao@zqu.edu.cn

Abstract: The architecture of cloud–edge collaboration can improve the efficiency of Internet of Things (IoT) systems. Recent studies have pointed out that using IoT terminal devices as destinations for computing offloading can promote further optimized allocation of computational resources. However, in practice, this idea encounters the problem that participants might lack the motivation to take over computational tasks from others. Although the edge and the terminal are provided with symmetrical positions in collaborative offloading, their computational resources and capabilities are asymmetric. To mitigate this issue, this paper designs a distributed strategy for the trading of computational resources. The most prominent feature of our strategy is its multi-preference optimization objective that takes into account the overall satisfaction with task delay, energy cost, trading prices, and user reputation of participants. In addition, this paper proposes a system architecture based on the Blockchain-as-a-Service (BaaS) design to give full play to the good distributed technology features of blockchain, such as decentralization, traceability, immutability, and automation. In the simulation evaluation, we compare our trading strategy based on a matching mechanism called multi-preference matching (MPM) to trading using the classical double auction (DA) matching mechanism. The results show that our computational resource trading strategy is able to offload and execut more tasks, achieving a better throughput compared to the DA-based strategy.

Keywords: blockchain; computation offloading; edge computing; internet of things (IoT); resource trading

check for updates

Citation: Wang, T.; Ai, S.; Cao, J.; Zhao, Y. A Blockchain-Based Distributed Computational Resource Trading Strategy for Internet of Things Considering Multiple Preferences. *Symmetry* **2023**, *1*, 0. https://doi.org/

Academic Editor(s): Kuo-Hui Yeh

Received: 3 March 2023
Revised: 21 March 2023
Accepted: 23 March 2023
Published:

1. Introduction

The extensive application of sensor technologies in daily objects has given birth to the Internet of Things (IoT). With the increasing number of IoT devices being used, the significant volume of the data acquired from the environment has brought great challenges to data transmission, processing, and storage services provided by centralized cloud centers. In many time-sensitive scenarios, e.g., healthcare, vehicular networks, and smart grids, cloud centers may fail to respond in a timely manner, which can have serious consequences [1]. In response, the edge computing paradigm emerges as the times require. Computing offloading is the central topic in the study of edge computing. In an IoT system based on the cloud–edge architecture, terminal devices with relatively limited computational resources can transfer their computing tasks to the edge servers nearby. Furthermore, edge servers can choose to complete these offloaded tasks or defer their tasks to the cloud center [2]. In this case, local computational resources are mainly dedicated to the tasks with higher requirements for response time, therefore improving the response speed and alleviating the bottleneck in the cloud [3].

1.1. Collaborative Computation Task Offloading

To further improve the efficiency of edge computing, the concept of "collaborative offloading" has recently been proposed [4]. Unlike most related works, collaborative offloading here emphasizes the in-place task transfer. In other words, the task transfer in collaborative offloading takes place between two edge servers or from an edge server with more tasks than it can handle back to nearby terminal devices with surplus computational resources. In this setting, the edge and the terminal are given symmetrical positions in the sense that they can be both the departure and the destination of offloaded tasks. Users that provide resources for others are called *collaborators*, and users with offloading requirements are called *requesters*.

In collaborative offloading, it has been pointed out that collaborators may still lack the motivation to take over tasks from others. Since the computational resources and capabilities of the edge and the terminal are nevertheless asymmetric, the impact of the resources consumed by executing tasks from others is not the same. Recent works have started to introduce economic incentives to encourage collaborators to contribute surplus computational resources [5,6]. Due to the different concerns of different participants, their decisions of task offloading are nevertheless determined by multiple factors in addition to economic ones. For example, the quality of service, revenue and expenses, and credibility of the collaborators and requesters can simultaneously impact the experience of the participation [7]. Therefore, in order to encourage user participation and promote the practical application of the system, collaborative offloading needs to comprehensively consider the influence of multiple user preferences.

In this paper, we model collaborative task offloading as a resource trading process between collaborators and requesters. Its core stage, the transaction matching stage, is modeled by an optimization problem with multiple attributes (e.g., task delay, energy cost, price, and participant reputation) considered in the objectives. By satisfying more of the participants' personalized trading preferences, the matching strategy intuitively brings more incentives to its participants.

1.2. Blockchain for Internet of Things

The IoT faces a series of security challenges, and data security, user privacy, and service trustworthiness are the major concerns addressed in related studies [8]. In recent years, the blockchain has been widely applied in various fields due to its good distributed technology features such as transparency, privacy, immutability, and fault tolerance [9]. In essence, a blockchain is an aggregation of multiple distributed technologies. It stores data in the form of chained blocks, in which newly generated blocks must be verified and approved by the participants of the blockchain system through a distributed consensus algorithm before being stored in the chain. The distributed data storage scheme together with the data structure makes information transparent, traceable, and immutable. Moreover, blockchain-based systems can be automated with the support of smart contracts. The Blockchain-as-a-Service (BaaS) design greatly reduces the implementation difficulty of blockchain-based systems and promotes the penetration of the blockchain in various industries [10]. Instead of building an entire blockchain system, developers can use the ready-made blockchain interfaces and toolkits offered by the BaaS platform. Therefore, making full use of the convenient services provided by BaaS will better fit blockchain-based collaborative offloading schemes to the decentralized nature of the IoT.

In order to integrate the features of the blockchain more deeply in the IoT collaborative offloading scenario, this paper extends the BaaS application of [11–13]. Our work exploits many different features provided by BaaS for collaborative offloading in addition to using a blockchain as a secure database. In particular, this paper uses distributed ledger services to maintain reputation chains for participants, so as to provide incentives for the participation in collaborative transactions in a distributed manner.

1.3. Contributions

The main contributions of this paper are as follows:

1. This paper proposes a distributed computational resource trading strategy for IoT users, where the BaaS design is adopted in the architecture. Unlike most related works that simply use a blockchain as a secure database, this paper takes full advantage of the blockchain to promote the decentralization, reliability, and automation of resource trading.
2. This paper designs a multi-preference matching (MPM) mechanism for resource trading. The matching results between requesters and collaborators comprehensively consider the satisfaction with task delay, energy cost, trading prices, and the reputation scores of participants. As far as we know, few relevant studies have taken these factors into account all at once.
3. We compare our MPM mechanism with the matching strategy based on the classical double-auction (DA) matching mechanism [14]. We perform simulation experiments to show the advantages of MPM against DA.

The remaining content of this paper is arranged as follows: Section 2 provides a brief review on related works; Section 3 introduces the system architecture and the workflow of blockchain-based resource trading; Section 4 explains the MPM mechanism in detail; Section 5 conducts simulation experiments and numerical analysis by comparing the MPM mechanism with the DA-based matching mechanism; Section 6 concludes this paper.

2. Related Works

To explain the motivations of our work, we provide a brief summary of the related works on blockchain-based security for the IoT and on incentivizing collaborative offloading in this section. See Table 1 for the comparison of the related works in this Section.

Table 1. Highlights and Shortcomings of Related Works.

Topic	Ref.	Highlights	Shortcomings
Blockchain-Based IoT Security	[15]	Allows open participation with public blockchain	Additional Byzantine fault-tolerant consensus algorithm is not scalable for public blockchains
	[16]	Community detection is included to specify the scope of data with different privacy levels	Centralized community detection algorithm contradicts the decentralized privacy of the blockchain
	[17]	Data models are shared by federated learning instead of raw data	Frequent transmission of training models can bring large communication workloads
Collaborative Offloading	[18]	Social relationship of participants is considered	The use of DRL method in decision making increases the task delay
	[19]	DRL is adopted to optimize network utility of UAVs	
	[4]	Intelligence and selfishness of users are considered	Factors other than economic benefits are not considered
	[6]	Near-optimal competitive ratio is achieved	

2.1. Blockchain-Based Security for the IoT

Security issues are the major concerns of IoT-related studies, and the blockchain has become popular in providing security for IoT systems. For example, ref. [15] combines deep reinforcement learning with the blockchain to create a data collection and secure data sharing environment with improved system energy efficiency. Although it allows for open participation with the public Ethereum blockchain [20], it requires an additional Byzantine fault-tolerant consensus algorithm to prevent device failure, which is not very scalable to public blockchains. In [16], a blockchain layer is added to the IoT data sharing system to validate, sort, and store data trading records in a secure and reliable way. However, its centralized community detection subprotocol might have the possibility of privacy disclosure when calculating client similarity based on label data, which also contradicts the decentralization of the blockchain. Aiming to effectively protect user privacy, the data sharing architecture of [17] has a permissioned blockchain module to securely store and retrieve data and a federated learning module to share data models instead of raw data. In spite of the accuracy and efficiency of collaborative training, the consensus process requires the frequent transmission of training models between nodes outside the federated learning process, which could impose a greater workload on the communication network.

These works, as well as many other related works, simply use the blockchain for secure data storage, and the advantages of the blockchain are not fully exploited [9]. The reason for thus is that system designers often face huge workloads when developing a sophisticated blockchain system. BaaS can greatly reduce the difficulty of implementing blockchain-based systems by providing various basic blockchain functions. The BaaS design in [11] is deployed into an edge computing platform to support distributed resource trading for task offloading based on smart contracts. In [12,13], BaaS is integrated to undertake energy supply–demand matching in fully decentralized electric power systems. They both use the smart contract service to provide automation, which greatly enhances the performance of energy trading.
In view of its great convenience, BaaS is adopted in this paper in the design of a decentralized, secure, reliable, incentivizing, and automated computational resource trading system for the IoT.

2.2. Incentivizing Collaborative Offloading

Recently, the edge computing paradigm has been extensively applied in IoT systems [21]. To mitigate the response latency issue in edge servers, some works recommend collaborative offloading to allow terminal users with additional computational resources to take over the tasks of edge servers [18,19]. For example, the concept of hybrid offloading is proposed in [18], which extends traditional edge computing offloading to hybrid offloading that combines both edge computing offloading and device-to-device offloading. The offloading scheme takes social relationships into consideration and tries to reduce overall execution delay while enhancing its data caching service. In addition, the authors of [19] designed a cooperative offloading structure that allows unmanned aerial vehicles (UAV) to execute the computation tasks of the others; here, a deep reinforcement learning (DRL) method is adopted to optimize the long-term utility of the mobile edge computing network.

However, the promotion of the above collaborative offloading in practice has encountered some obstacles because users lack incentive mechanisms to complete the computing tasks offloaded by others [22]. As a result, some studies include economic measures and transform collaborative offloading into a computational resource trading problem. In [4], the intelligence and selfishness of terminal users are considered when making trading decisions. The offloading strategy tries to maximize social welfare by considering the cooperation between edge devices and terminal users as resource trading. Similarly, the social welfare maximization problem in computation offloading is also studied in [6]. The authors show that their mechanism has a near-optimal competitive ratio and is able to guarantee individual rationality, truthfulness, and computational traceability. The problem is nevertheless that in addition to economic factors, response time, energy consumption, reputation, and many other factors may also affect the decision of computing task allocation. The influence of multiple factors in collaborative offloading is rarely addressed in the related research. Therefore, the comprehensive consideration of various factors in our MPM-based computational resource trading method is more in line with practical requirements.

3. Computational Resource Trading System Architecture and Workflow

The IoT system considered in this paper is based on the classical three-layered architecture of edge computing and further extends the application scope of BaaS described in [11–13]. As shown by Figure 1, this architecture consists of five major components:

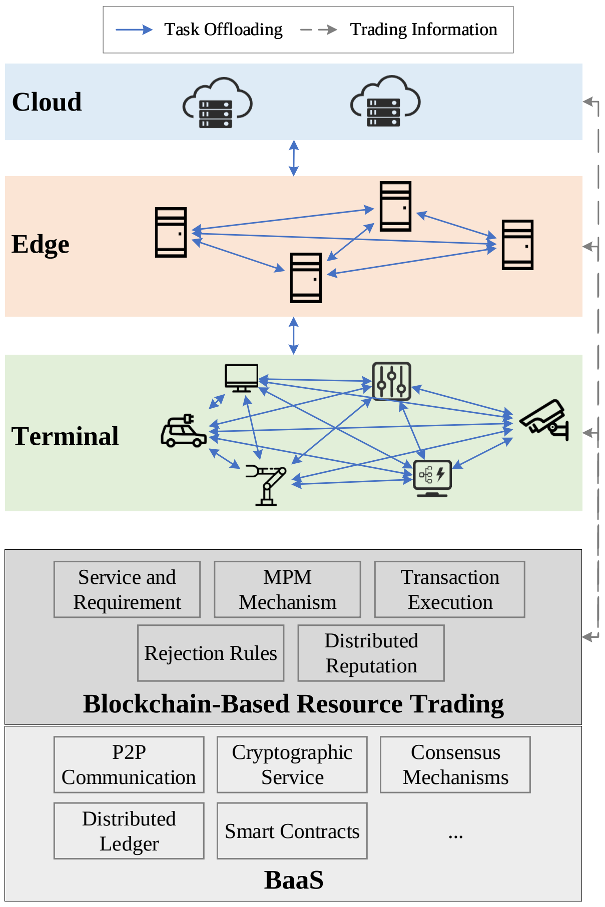


Figure 1. System architecture.

Terminal. The terminal layer is made up by IoT terminal devices embedded with sensors for data perception. We assume that all IoT terminals are lightweight in the sense that their computing capabilities are rather limited compared to that of edge servers.

Edge. The edge layer contains edge servers deployed near terminals. These servers have some computational resources so that they can take over the tasks offloaded from terminals. However, for economic concerns, edge servers usually have less computational resources compared to cloud servers. Although the transmission time between terminals and edge servers can be largely reduced, the latency caused by task queuing in edge servers cannot be neglected.

Cloud. The cloud layer has cloud servers with powerful computational capabilities and is far away from the edge servers. It is generally assumed that the cloud can process any number of tasks at the same time, but with significant transmission latency.

BaaS. The BaaS platform is fundamental to achieve distributed computational resource trading for collaborative offloading. The system takes advantage of the following blockchain services:

- **P2P communication.** Messages and data are shared and retrieved transparently without the need of a central server. This makes the system more robust against single-point failures. Moreover, it allows participants to join and leave the network freely.
- **Cryptographic service.** Message transmission channels and data storage are secured by various cryptographic algorithms, and user privacy is also guaranteed.
- **Consensus mechanisms.** As the central component of the blockchain, consensus is used to validate newly generated blocks that contain transaction information or reputation updates. Any change that occurs to smart contracts also needs to be validated by participants through consensus.
- **Distributed ledger.** The system uses three distributed ledgers, namely, the transaction chain, the collaborator reputation chain, and the requester reputation chain, to record transaction data and reputation scores. This makes all the changes in transaction and reputation information traceable.
- **Smart contract.** The execution of resource trading and the application of reputation update rules can be automated through smart contracts, programmable scripts that are triggered and executed automatically when predefined conditions are met.

Blockchain-Based Resource Trading. This module is developed upon the BaaS platform. It contains all the functionalities that support resource trading:

- **Service and requirements.** This submodule defines and regulates the service information submitted by collaborators and the requirement information submitted by requesters. More details are provided later in this section.
- **MPM mechanism.** Its purpose is to find matches between services of collaborators and requirements of requesters with multiple preferences of users considered. As the core of the resource trading strategy, the MPM mechanism is introduced in detail in Section 4.
- **Rejection rules.** It is possible that the matching result of one round fails to satisfy all participants. If the individual satisfaction scores of participants are not high enough, they can choose to reject the matching results. More details are given in Section 4.5.
- **Transaction execution.** If participants do not refuse the matching result, they are required to fulfill the transaction according to the matching result. The execution results will affect their reputation scores.
- **Distributed reputation.** Reputation scores are assigned to all participants to evaluate their credibility in resource trading. This submodule is simplified from the blockchain-based reputation system in [23]. More details are given in Section 4.6.

In this paper, task offloading can take place between two terminals, between two edge servers, or between a terminal and a server. During computational resource trading, collaborators can reveal the information of the resources they are willing to offer, and requesters with computational tasks can then choose the collaborators to whom they will offload tasks in exchange for payment.

The distributed resource trading workflow is as follows:

Step 1: Collaborator j with surplus computational resources publishes its service information including:

- C_j: size of the cache offered;
- f_j: central processing unit (CPU) frequency offered;
- r_j: transmission rate offered;
- ϵ_j: maximum acceptable energy consumption per CPU cycle;
- op_j: offering price, i.e., (the lowest) price of task execution offered per CPU cycle;
- R^c_j: collaborator reputation score.

Meanwhile, requester i with computational tasks to offload submits their offloading requirement information including:

- s_i: size of tasks;
- Q_i: CPU cycles required by tasks;

- τ_i: maximum tolerable delay of tasks;
- bp_i: bidding price, i.e., (the highest) acceptable price of task execution per CPU cycle;
- R_i^{r}: requester reputation score.

Step 2: The service of collaborator j and the requirement of requester i are stored in the transaction ledger.

Step 3: The smart contract of MPM, a matching mechanism considering multiple preferences, is triggered, and corresponding pre-matching results will be returned to requesters for confirmation. More details about the MPM mechanism are provided in Section 4.

Step 4: Participants can choose to reject the matching results if they are dissatisfied (see more details in Section 4.5). Matching results that are not rejected are seen as accepted. Once accepted, each pre-transaction result will generate a transaction contract with the trading price calculated and will be stored into a distributed ledger in the form of smart contract provided by BaaS. Then, requesters make the payment according to their confirmed transactions.

Step 5: On the execution time, transaction contracts will be triggered automatically, and the corresponding task offloading will take place. The reputation scores of both the requester and the collaborator will be updated according to the execution results and predefined reputation rules. Note that the submitter of an unmatched requirement can choose to either execute the tasks locally or offload the tasks further to the cloud, while the submitter of an unmatched service can choose to keep the service and wait for another round of matching.

Reputation scores of requesters and collaborators, R_i^{r} and R_j^{c}, are defined to evaluate and regulate the behavior of requesters and collaborators. We will provide more details about our reputation system in Section 4.6.

4. Multi-Preference Matching Mechanism

The core of our resource trading strategy is the MPM mechanism that aims to maximize the overall satisfaction of both requesters and collaborators considering their respective preferences. The computational power of IoT terminal devices is much weaker than that of cloud centers and edge servers. In addition, the diversity of terminal devices makes the computing tasks they need to complete and the computing services they can provide vary a lot. Task delay and offering price determine the requesters' satisfaction with a service, while energy consumption and bidding price affect the collaborator's experience of undertaking a requirement. At the same time, reputation can be used to evaluate the credibility of participants according to their historical behavior. In this case, expressing multiple preferences can better meet the matching requirements of both parties, so as to provide more personalized services.

The operation of our MPM mechanism also relies on the services provided by BaaS. First, matching results are programmed into smart contracts and encapsulated into blocks. These blocks will be stored in a distributed ledger, called the *transaction chain*, once they are validated through distributed consensus. Second, reputation scores for collaborators and requesters are also stored in ledgers, called the *collaborator reputation chain* and the *requester reputation chain*, respectively. Participants can query each other's latest reputation scores and track the corresponding update history. Furthermore, the rules for reputation updates are also programmed into smart contracts, and any addition, deletion, and modification of these rules needs to be validated by participants through distributed consensus.

Suppose there are m requesters and n collaborators in a matching round. For the sake of simplicity, we assume that no participant is both a collaborator and a requester at the same time. Moreover, we assume that each requester submits one requirement, and each collaborator submits one service. These assumptions are for mathematical convenience and can be easily removed by assigning unique identifiers to different roles, services, and requirements of a participant if otherwise.

We use matrix $X = (x_{ij})_{m \times n}$ to represent the result of one round of matching, where $x_{ij} \in [0,1]$ represents the proportion of the tasks in requirement i to be offloaded to collaborator j. The MPM mechanism to decide X works as follows:

Step 1: Fetch the information of the services of collaborators and the requirements of requesters from the transaction ledger. We require that bidding price bp_i and offering price op_j should fall in $[p_{\min}, p_{\max}]$, where $p_{\min}$ and $p_{\max}$ are the lowest and the highest prices allowed. In practice, $p_{\min}$ and $p_{\max}$ are usually published in market rules and policies, and the trading service will periodically update and synchronize these values. All the requirements and services whose prices are out of the range will be forcibly removed (similar mechanisms can also be found in [24,25]).

Step 2: Calculate the service preference score (SPS) of each collaborator service for requester $i \in \{1, 2, \ldots, m\}$, and calculate the requirement preference score (RPS) of each requester requirement for each collaborator $j \in \{1, 2, \ldots, n\}$, with respect to different preferences. We will provide more details about the calculation of these two scores later on in Sections 4.1 and 4.2.

Step 3: Calculate the average requester satisfaction (ARS) for all requesters and the average collaborator satisfaction (ACS) for all collaborators. The calculation will be introduced in Section 4.3.

Step 4: Model the matching as an optimization problem and find the solution. This step will be further explained in Section 4.4.

The above steps include the calculation of several scores, and we will introduce the calculation method of these scores in the next section. As in related works such as [4,26], the energy cost and time delay caused by collaborators when sending back computational results from requesters are considered. This approach is based on the consideration that the data sizes of the computation results in practice are usually very small. In addition, with the aid of the following definition of characteristic function, we can make our notations more succinct:

$$\mathbb{I}[X] = \begin{cases} 1 & \text{Event } X \text{ is true;} \\ 0 & \text{Otherwise.} \end{cases} \tag{1}$$

4.1. Service Preference Score Calculation

First, let $SPS(x_{ij})$ be the SPS of service j for requester i. It evaluates i's comprehensive satisfaction with j considering i's preference in task delay, offering price, and collaborator reputation, which can be calculated by:

$$SPS(x_{ij}) = \left(\sum_{k=1}^{3} \phi_k \cdot sps_{i,j,k} \right) \cdot \mathbb{I}[x_{ij} \neq 0], \tag{2}$$

where ϕ_k $(k = 1, 2, 3)$ are significance factors, and $sps_{i,j,k} \in [0,1]$ $(k = 1, 2, 3)$ will be explained in the next section. Significance factors are specified by requesters, which indicates the importance of a certain service preference to the requesters. For example, if the requester pays more attention to the offering price, it could increase the corresponding significance factor ϕ_2 and reduce ϕ_1 and ϕ_3.

4.1.1. Task Delay

Task delay is the main factor that influences the quality of service (QoS) of the requesters, which composes transmission delay and computation delay:

$$t_{ij} = x_{ij} s_i / r_j + x_{ij} Q_i / f_j. \tag{3}$$

Then, $sps_{i,j,1}$, the task delay SPS of service j for requester i, is calculated by:

$$sps_{i,j,1} = (1 - t_{ij}/\tau_i) \cdot \mathbb{I}[t_{ij} \leq \tau_i]. \tag{4}$$

The shorter the task delay is, the better the QoS is, and the higher $sps_{i,j,1}$ will be.

4.1.2. Offering Price

The offering price SPS of service j for requester i, denoted by $sps_{i,j,2}$, is calculated by:

$$sps_{i,j,2} = e^{op_j - bp_i} \cdot \mathbb{I}[op_j \leq bp_i]. \tag{5}$$

The closer op_j and bp_i are, the more satisfied the requester will be with the matching result.

4.1.3. Collaborator Reputation

Collaborator reputation score R^c_j reflects the credibility of service j and directly serves as $sps_{i,j,3}$ in the MPM mechanism:

$$sps_{i,j,3} = R^c_j. \tag{6}$$

4.2. Requirement Preference Score Calculation

Then, let $RPS(x_{ij})$ be the RPS of requirement i for collaborator j, which can be calculated by:

$$RPS(x_{ij}) = \left(\sum_{l=1}^{3} \psi_l \cdot rps_{j,i,l} \right) \cdot \mathbb{I}[x_{ij} \neq 0], \tag{7}$$

where ψ_l $(l = 1, 2, 3)$ are significance factors, and $rps_{j,i,l}$ $(l = 1, 2, 3)$ will be explained in the next section. $RPS(x_{ij})$ evaluates j's comprehensive satisfaction with i considering j's preference in energy consumption, bidding price, and requester reputation, corresponding to the three terms of (7). Like ϕ_k, significance factors ψ_l are specified by collaborators and indicate the importance of a certain requirement preference to the collaborators.

4.2.1. Energy Consumption in Collaborator

Compared to task delay, collaborators care more about their energy consumption when taking over the tasks from requesters. The energy consumption in collaborator i can be calculated by:

$$E_{ji} = e^{\text{com}}_j x_{ij} s_i / r_j + e^{\text{exe}}_j x_{ij} Q_i / f_j, \tag{8}$$

where e^{com}_j and e^{exe}_j are the energy consumption of communication and task execution per second. The two terms of (8) correspond to the energy cost by receiving the data of the offloaded tasks and executing them. Considering ϵ_j, the maximum energy cost by executing the tasks offloaded to collaborator j, the RPS of requirement i for collaborator j, denoted by $rps_{j,i,1}$, is calculated by:

$$rps_{j,i,1} = (1 - E_{ji}/\epsilon_j) \cdot \mathbb{I}[E_{ji} \leq \epsilon_j]. \tag{9}$$

We can see that the closer to ϵ_j the value of E_{ji} is, the smaller $rps_{j,i,1}$ becomes. Moreover, $rps_{j,i,1}$ will become zero if the energy consumption of taking the task exceeds the bearing limit of collaborator i.

4.2.2. Bidding Price

The bidding price RPS of requirement i for collaborator j, denoted by $rps_{j,i,2}$, is calculated by:

$$rps_{j,i,2} = e^{-op_j / bp_i} \cdot \mathbb{I}[bp_i \geq op_j], \tag{10}$$

which is positively related to the bidding price bp_i and negatively related to the offering price op_j, and it becomes zero if the bidding price fails to exceed the offering price. That is to say, the larger bp_i is, the higher the requester is willing to pay, and the more the collaborator can benefit.

4.2.3. Requester Reputation

Similarly, requester reputation reflects the credibility of requirement, which is directly regarded as $rps_{j,i,3}$:

$$rps_{j,i,3} = R_i^{\mathrm{r}}. \tag{11}$$

4.3. Average Requester/Collaborator Satisfaction Score Calculation

The requester and collaborator average satisfaction scores, denoted by $ARS(X)$ and $ACS(X)$, respectively, evaluate the overall degree of satisfaction of all requesters and collaborators with their matching result X. The two scores are calculated by:

$$ARS(X) = 1/m \cdot \sum_{i=1}^{m} \sum_{j=1}^{n} SPS(x_{ij})x_{ij}, \tag{12}$$

$$ACS(X) = 1/n \cdot \sum_{j=1}^{n} \sum_{i=1}^{m} RPS(x_{ij})x_{ij}. \tag{13}$$

By requiring $\sum_{k=1}^{3} \phi_k = \sum_{l=1}^{3} \psi_l = 1$, $ARS(X)$ and $ACS(X)$ are also in the range of $[0,1]$.

4.4. Modeling and Solving the Optimization Problem

The objective of the MPM mechanism is to find the optimal $X = X^*$ that maximize $\mathcal{J}(X)$, the overall satisfaction of all participants:

$$\max_{X \in [0,1]^{m \times n}} \quad \mathcal{J}(X) \tag{14}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} \leq 1, \;\; 1 \leq i \leq m, \tag{15}$$

$$\sum_{i=1}^{m} s_i x_{ij} \leq C_j \;\; 1 \leq j \leq n, \tag{16}$$

$$0 \leq x_{ij} \leq 1, \;\; 1 \leq i \leq m, \;\; 1 \leq j \leq n, \tag{17}$$

where

$$\mathcal{J}(X) = w_1 ARS(X) + w_2 ACS(X), \tag{18}$$

and w_1 and w_2 are the weights that indicate the significance of requesters and collaborators. Constraint (15) means that the total tasks offloaded by requester i cannot exceed what is submitted in requirement i, and constraint (16) means that the total size of the tasks offloaded to collaborator j cannot exceed the cache size offered by service j.

Since $SPS(x_{ij})$ is a linear function to x_{ij} because of the calculation of t_{ij} by (3), the optimization problem represented by (14) is a quadratic programming problem. Note that although the modeling and solving of this quadratic programming problem is centralized, it does not make our MPM strategy centralized. The tasks of the other steps, including service and requirement submission, matching rejection, transaction execution, and reputation evaluation, need the cooperation of all requesters and collaborators. The completion of these tasks cannot be delegated to any centralized individual in the system. Therefore, according to the theory of "decentralization scope and relativity" [27], the MPM strategy is distributed if our scope covers all the necessary steps of the MPM.

4.5. Rejection Rules and Transaction Execution

Since the MPM mechanism tries to maximize the overall satisfaction of all participants, it might be possible that some individuals are not willing to compromise their interests. In this case, they can choose not to accept the transaction based on their rejection rules. In more detail, users can also build smart contracts that specify their criteria for rejection, e.g., the lowest acceptable RPS or SPS in their mind, fixed requester/collaborator ID, or requester/collaborator type (i.e., whether the request/service is from a terminal or an edge). Note that these smart contracts also need to be validated by participants via distributed consensus. Any matching result that fails to meet these criteria will be automatically rejected. As a result, unmatched requirements and services will enter the next round of matching together with new ones. Otherwise, the submitter of unmatched requirements can also choose to offload their tasks to the cloud center.

On the other hand, all matching transactions that are not rejected are seen as accepted. They will be saved in the transaction ledger as smart contracts and will come into effect immediately, automatically, and mandatorily. It is worth noting that transaction execution may fail when the requester fails to pay the price, or the collaborator fails to provide the resource as promised. Whether it is successful or not, the transaction execution results will be fed back to the reputation system, which could then affect the reputation scores of participants involved.

4.6. Distributed Requester/Collaborator Reputation

As mentioned before, requester reputation R_i^{r} and collaborator reputation R_j^{c} evaluate the credibility of requester i and collaborator j respectively. This is a kind of distributed reputation mechanism where reputation scores rely on the mutual evaluation between requesters and collaborators, which can promote the collaborative regulation of the resource trading behavior. In this paper, we adopt a similar idea as [23] in the design of the distributed reputation mechanism.

The design of the blockchain-based distributed reputation mechanism of this paper is shown in Figure 2. Two blockchains are maintained, one for requester reputation and the other for collaborator reputation. These reputation scores can be queried by any participant in the system. Reputation rules are smart contracts that specify the methods for reputation updates and the conditions under which these updates are triggered. All requesters and collaborators make the decision on the addition, deletion, and modification of reputation rules together by running consensus.

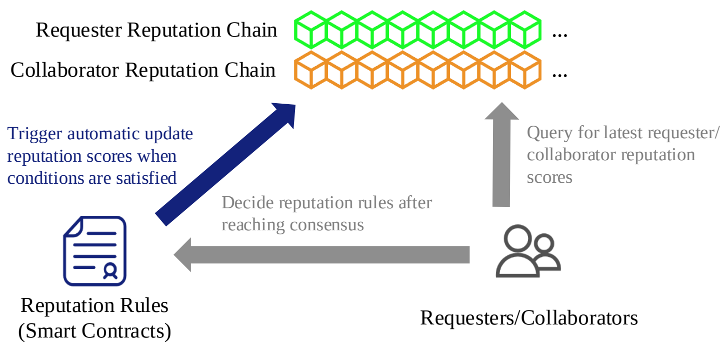

Requester Reputation Chain
...
Collaborator Reputation Chain
...
Trigger automatic update reputation scores when conditions are satisfied
Query for latest requester/ collaborator reputation scores
Decide reputation rules after reaching consensus
Reputation Rules (Smart Contracts)
Requesters/Collaborators

Figure 2. Blockchain-based distributed reputation.

According to [23], blockchain can provide many favorable features for our distributed reputation mechanism. First, reputation rules are implemented in the form of smart contracts. Any addition, deletion, or modification of these rules cannot come into effect until they are approved by all participants by running distributed consensus. This not only increases the transparency and traceability of reputation records, but it also encourages the participation of requesters and collaborators in system regulation. Second, these smart contracts containing reputation rules can automatically run once their predefined conditions are triggered. This circumvents the time cost and possible error of human labor in reputation update. Furthermore, reputation records are stored in reputation chains using the linked list data structure in chronological order, and different replicas of the chains are shared among participants. To tamper with the data in any block, the adversary also needs to modify all the data in the subsequent blocks in all replicas. This makes it prohibitively difficult to tamper with reputation data.

To guarantee the correctness of smart contracts, formal verification of smart contracts is necessary. There are different methods and goals of formal verification, including model checking, theorem proving, symbolic execution, runtime verification, and so on [28]. Reference [29] also provides a list of tools for smart contract verification. One convenient way is to convert smart contracts into Solidity files and verify them through the Remix tool provided by Ethereum [30].

4.6.1. Reputation-Based Trading Price

The traditional double auction method usually sets the final trading price as the bidding price submitted by the buyer. In order to increase fairness, we adopt the reputation-based α-double auction [23], which calculates the trading price as follows:

$$tp_{ij} = \alpha \cdot bp_i + (1 - \alpha) \cdot op_j, \tag{19}$$

where $\alpha = R^{\mathrm{c}}_j / \left(R^{\mathrm{r}}_i + R^{\mathrm{c}}_j \right)$. The resulting trading price will be closer to the price given by the party with the lower reputation, which is more beneficial to the party with the higher reputation.

4.6.2. Reputation Rules

Reputation rules are the core of the distributed reputation mechanism. To reduce the system complexity, we adopt the following simple rules:

- A new participant is assigned with an initial reputation score of 0.6.
- On a successful resource transaction, the reputation scores of both the requester and collaborator increase by 0.01.
- For every failed resource transaction: if requester i fails to pay the trading price, then R^{r}_i is decreased by 0.1; if collaborator j fails to provide the claimed service, then R^{c}_j is decreased by 0.1.
- The final reputation scores should be restricted in $[0, 1]$.

The parameters in these rules are determined by repeated simulation tests. Every new participant is assumed to be benign and is assigned with a "pass" reputation score of 0.6. Moreover, the penalty of 0.1 for a failed transaction is much higher than the reward of 0.01 for a successful transaction. This can intuitively encourage participants to effectively complete each transaction. These reputation rules are implemented as smart contracts. Once the current trading period is over, these rules will automatically trigger reputation updates once the predefined conditions are satisfied.

5. Evaluation

This section evaluates our system through simulation. In the evaluation, we mainly compare our MPM mechanism with the classical DA matching mechanism [14].

5.1. Simulation Setup

All simulation programs are written using Python 3.8 (64 bit) and are implemented on a laptop computer with an Intel® Core™ i7-6500U CPU 2.50GHz 2.59GHz, and the size of the random access memory (RAM) is 8GB. The optimization problem (14) is solved via the GEKKO Python library, which can automatically choose the most suitable solver without explicitly specifying which type of optimization problem needs to be solved (i.e., linear, quadratic, nonlinear, and mixed integer programming) [31]. In addition, we use "reputation for blockchain-based trading (RBT)" as our BaaS in our simulation, which uses a delegated version of practical Byzantine Fault Tolerance (PBFT) as its consensus mechanism (readers can refer to [23] for more details). We built a distributed environment, and we use independent and asynchronous threads to simulate both edge servers and IoT nodes. We use a Poisson process to simulate the generation of services and requirements.

The ranges of parameters in the simulation are shown by Table 2, and all parameters are selected in their ranges uniformly at random. The selection of these ranges is based on the works of [4,5,11,26,32]. We set $m = n$, and the ratio of the numbers of edge servers and IoT terminals is set as 1 to 30. In addition, by repeated adjustment and verification, we choose $w_1 = w_2 = 0.5$, $\phi_1 = \phi_3 = \psi_1 = \psi_3 = 0.36$, and $\phi_2 = \psi_2 = 0.28$.

Table 2. Selection Ranges of Parameters for Simulation.

Parameter	Explanation	Edge	Terminal
s_i	Size of tasks (GB)	$[0.06, 10]$	$[0.06, 10]$
Q_i	CPU cycles required (Gcycle)	$[0.6, 90]$	$[0.6, 90]$
τ_i	Maximum tolerable delay (s)	$[10, 30]$	$[5, 15]$
C_j	Cache size offered (GB)	$[5, 10]$	$[1, 5]$
f_j	CPU frequency offered (GHz)	$[3, 15]$	$[1, 5]$
r_j	Transmission rate offered (Gbps)	$[0.5, 2.5]$	$[0.1, 0.9]$
ϵ_j	Maximum tolerable energy consumption (J)	$[150, 250]$	$[5, 15]$
e_j^{com}	Unit energy consumption for transmission (J/s)	$[0.2, 0.5]$	$[0.1, 0.3]$
e_j^{exe}	Unit energy consumption for execution (J/s)	$[0.75, 1.25]$	$[0.3, 0.6]$
bp_i	Bidding price (USD/Gcycle)	$[0.1, 10]$	$[0.1, 10]$
op_j	Offering price (USD/Gcycle)	$[0.1, 10]$	$[0.1, 10]$
$R_i^{\text{c}}, R_j^{\text{r}}$	Requester/Collaborator reputation	$[40, 100]$	$[40, 100]$

5.2. Double Auction

DA is one of the most popular matching strategies in market designs of various settings, e.g., computational resource allocation [33], energy trading [34], data trading [35], and asset markets [36]. We believe that it is worthwhile to compare our method with the DA method because it is such a widely used matching mechanism. Specifically, DA sorts the requirement list and the service list in different orders. The requirement list is sorted based on the ascending order of bidding prices, and the service list is sorted based on the descending order of offering prices [14]. It then traverses each list from the top and finds a match when it encounters an offering price in the service list that is lower than the bidding price at the current location of the requirement list. The trading price of this match will be the bidding price provided by the requester.

For each group of requirements and services generated, we execute MPM and DA separately and compare their performance indices under the same conditions. We use solid lines to represent MPM and dashed lines to represent DA in all the figures that follow.

5.3. Satisfaction Scores

Figure 3 compares the ARS, ACS, and objective values of the MPM and DA mechanisms. We can see that there are significant gaps between the two methods in these scores: the values of $ARS(X)$, $ACS(X)$, and $\mathcal{J}(X)$ of MPM are more than twice that of DA. Note that these three scores evaluate the average satisfaction of requesters, collaborators, and all participants towards matching result X. It can also be interpreted that, on average, participants are more likely to accept the matching results of MPM than DA. This intuitively provides an incentive for collaborative with offloading participants when the MPM mechanism is adopted.

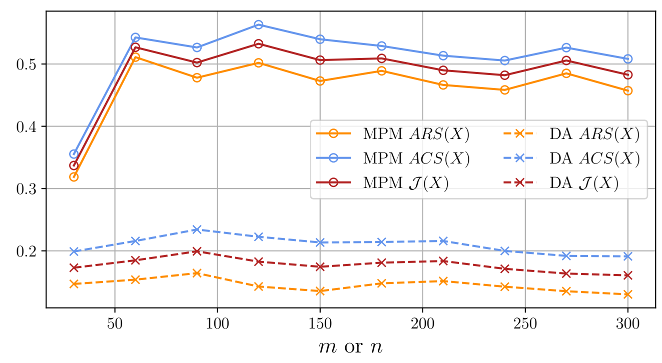


Figure 3. Comparisons of ARS, ACS, and objective values.

To further analyze the differences between the matching results of MPM and DA, we extract the average scores of SPS and RPS that rank the top 10% and the bottom 10% from each group of data for nonzero matching results. We can see from the two graphs in Figure 4 that the high scores of SPS and RPS of MPM are slightly higher than that of DA. Moreover, the low scores of SPS and RPS of MPM fall between 0.5 and 0.6, but the low scores of DA remain below 0.3. This indicates that more than 10% of the matching results of DA will be rejected if the participants apply the same rejection rules as in MPM. Detailed analysis of the rejection rates is deferred to Appendix A.

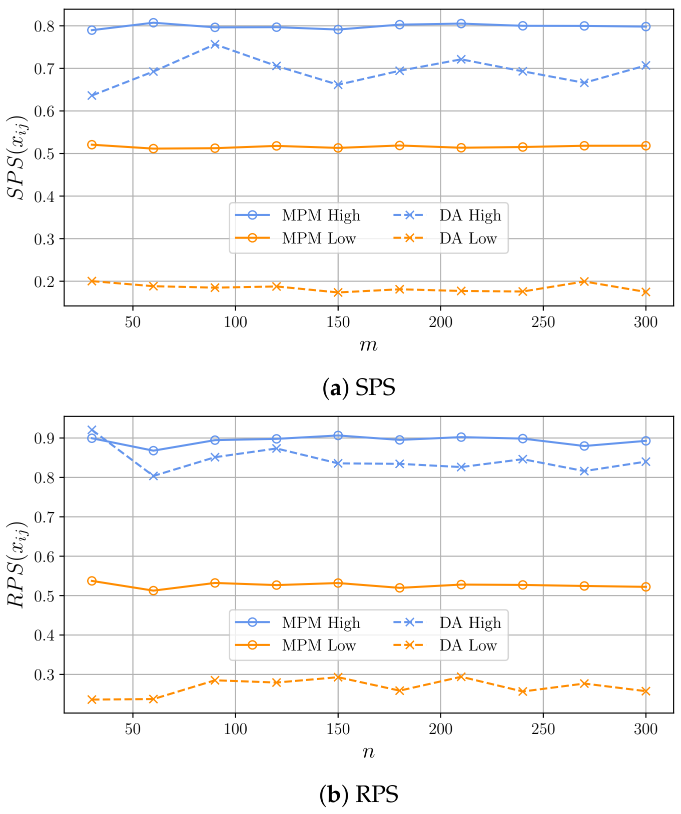
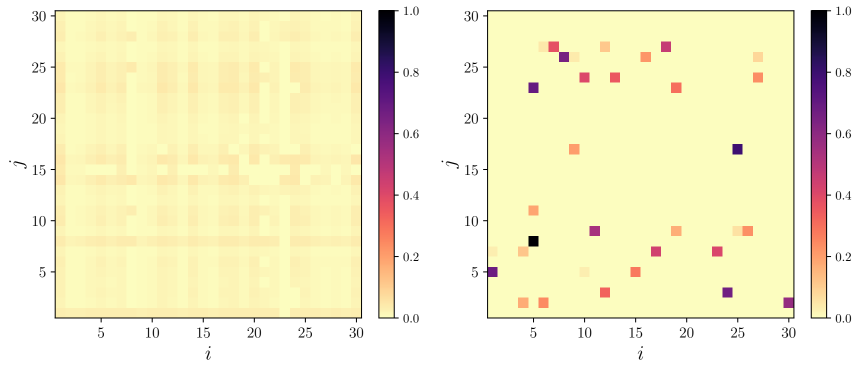


Figure 4. Comparisons of high/low SPS and RPS.

5.4. Matching Results

Figure 5 visualizes an example of the matching result X of both mechanisms when $m = n = 30$. In Figure 5a, the color difference of the blocks is not very obvious, but the distribution is quite uniform. This is because matrix X of MPM does not have many zero entries, but all nonzero entries are relatively small. In other words, most requirements will be matched, but each matching collaborator receives a fairly small portion of these tasks. On the other hand, Figure 5b has several dark-colored blocks, meaning that matrix X of DA has only a small number of nonzero entries. This suggests that the number of matches DA generates is much smaller, but some of the matched collaborators may need to undertake a large proportion of the offloaded tasks. The same pattern also holds when there are more than 30 requesters and 30 services. Due to the limited space, the details of these cases will be omitted here.



(**a**) X of MPM (**b**) X of DA

Figure 5. Comparison of matching results X with $m = n = 30$.

5.5. Task Completion of Requesters

Here, we observe the completion of the requesters' tasks. Figure 6 compares the total sizes of the tasks executed using both mechanisms. The figure shows that the total size of tasks completed by using MPM is about 2.5 times of that of DA. On the other hand, compared with DA, Figure 7 exhibits a reduction of more than 60 times in the maximum task delay of MPM. The huge gaps in Figures 6 and 7 are caused by MPM's more equal distribution of tasks among collaborators. This is also supported by the example in Figure 5. We also deferred detailed efficiency analysis to Appendix B.

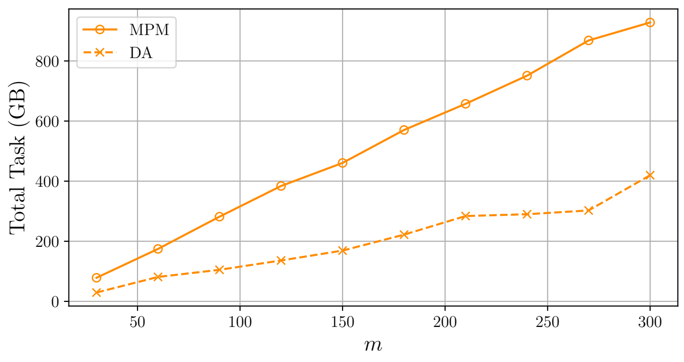


Figure 6. Comparison of total sizes of tasks executed.

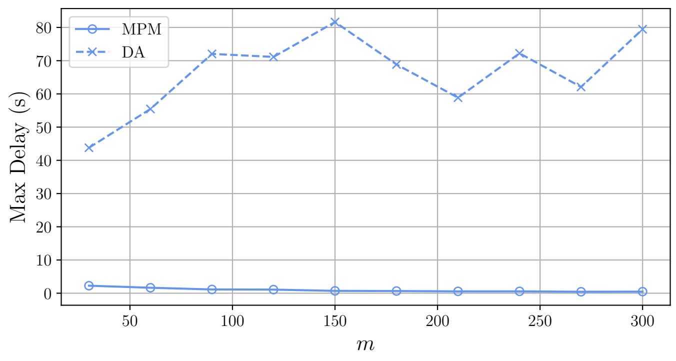
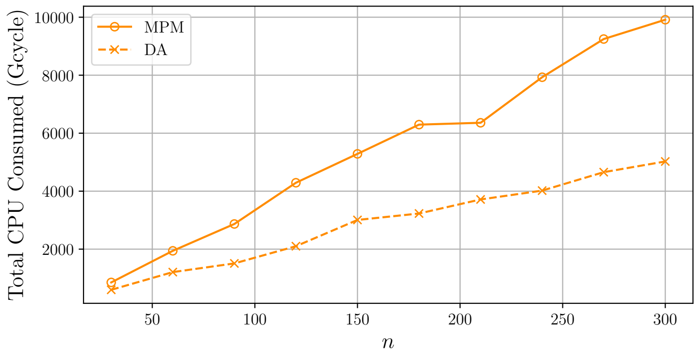


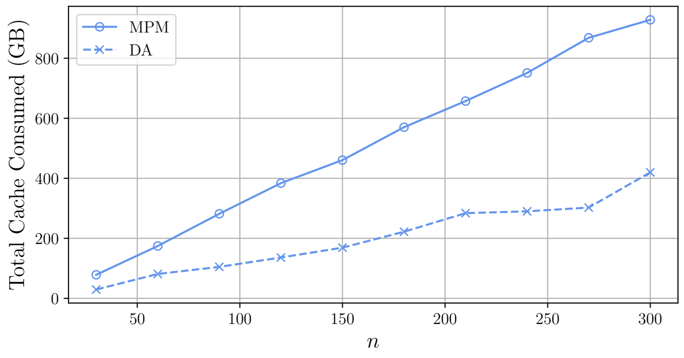
Figure 7. Comparison of maximum task delays.

5.6. Resource Consumption of Collaborators

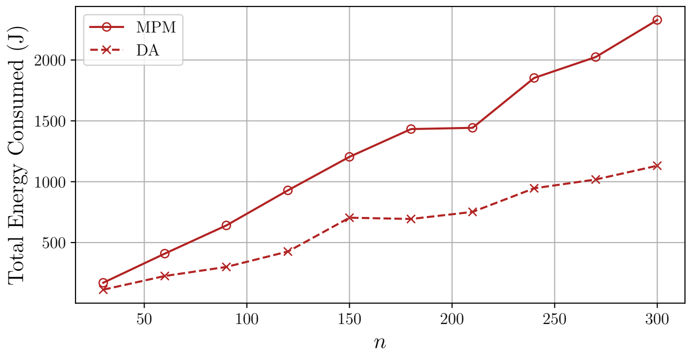
Next, we evaluate the resource consumption of collaborators. Figure 8 compares the consumption of total CPU cycles, cache sizes, and energy of collaborators between two mechanisms. By calculation, we find that compared with DA, MPM increases the consumption of these three resources by 89%, 152%, and 97%, respectively. This significant increase in resource consumption of collaborators is because more tasks can be executed by adopting the matching results of MPM.



(**a**) CPU Cycles



(**b**) Cache Sizes



(**c**) Energy

Figure 8. Comparisons of resource consumption of collaborators.

5.7. Trade Price

Figure 9 compares the average trade prices of two matching mechanisms. The trade prices of MPM are about 37.99% lower than those of DA on average. This drop is because MPM adopts α-double auction in (19), where the reputation scores of both the collaborators and requesters are taken into account while calculating the trade price. The analysis of the influence of reputation scores on trading prices is deferred to Appendix C.

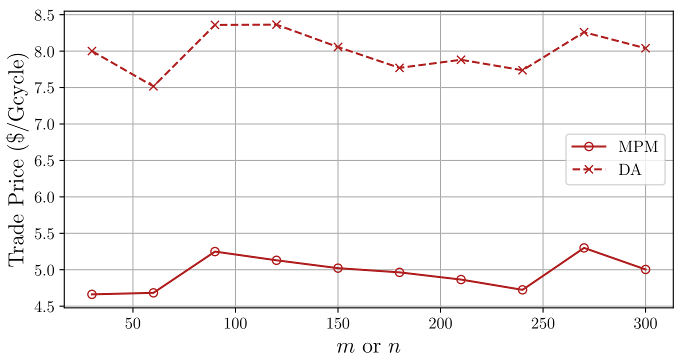


Figure 9. Comparison of average trade prices.

6. Conclusions

In this paper, we design a distributed computational resource trading strategy for IoT systems. The trading adopts a matching mechanism that takes into account the multiple preferences of requesters and collaborators. With the help of the blockchain, the decentralization of resource trading is achieved; the security, traceability, and immutability of transaction records are guaranteed; and the automation of distributed matching and reputation mechanisms is enabled. Compared with the classical DA matching mechanism, our MPM mechanism has more tasks offloaded and executed and can encourage the participation of collaborators with higher reputation scores. The efficiency and scalability of our simulation program can be further improved by adopting heuristic algorithms as subsitutions for traditional GEKKO solvers.

It is worth noting that the reputation mechanism in our system is simplified for illustrative purposes. A practical reputation system can be more comprehensive and complicated. For one thing, a complete reputation system needs to cover all possible behavior of participants and specify corresponding reputation update rules. For another, the design, deployment, and maintenance of a reputation system depends on either an authority or the distributed trust. How to design a reasonable reputation mechanism for IoT systems in a distributed way is a direction worthy of further study. In addition, the evaluation of our resource trading strategy is mainly based on simulation experiments. Furthermore, experiments that consider the impact of the heterogeneity in processing time, task load, communication media, and blockchain platform in the real environment can be more helpful to test the practicability of our work. This will take place in our future work. Finally, our strategy needs multiple negotiation iterations, which risks delaying the execution of urgent computing tasks. We use the cloud or local execution as the backup plan, but practical implementation may require a more detailed real-time scheduling method. Therefore, combining our resource trading strategy with a real-time task scheduling method will be another direction of our future work.

Author Contributions: T.W.: Conceptualization, Investigation, Software, Data Curation, Visualization, Writing—Original Draft; S.A.: Methodology, Formal Analysis, Validation; J.C.: Supervision, Project administration; Y.Z.: Resources, Funding Acquisition, Writing—Review and Editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusion of this article will be made available by the authors with reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ACS	Average Collaborator Satisfaction
ARS	Average Requester Satisfaction
BaaS	Blockchain-as-a-Service
CPU	Central Processing Unit
DA	Double Auction
DRL	Deep Reinforcement Learning
IoT	Internet of Things
MPM	Multi-Preference Matching
P2P	Peer-to-Peer
PBFT	Practical Byzantine Fault Tolerance
QoS	Quality of Service
RAM	Random Access Memory
RBT	Reputation for Blockchain-based Trading
RPS	Requirement Preference Score
SPS	Service Preference Score
UAV	Unmanned Aerial Vehicle

Appendix A. Rejection Rate Analysis

As we mentioned in Section 4.5, the reasons for rejecting a matching result can vary a lot according to the individual interests of different participants. For the sake of simplicity, we chose the lowest acceptable RPS or SPS as the rejection rule for our simulation (other rules are rather subjective and have lower analytical value). In more detail, we set the lowest acceptable RPS and the lowest acceptable SPS to 0.5. For matching result $x_{ij} > 0$, if $SPS(x_{ij}) < 0.5$ or $RPS(x_{ij}) < 0.5$, then the match between requester i and collaborator j will be rejected, and x_{ij} is set to 0. According to Figure A1, we can find that the rejection rate of MPM decreases dramatically when the number of participants increases. In other words, the absolutely dominant majority of the participants will be satisfied with the matching results.

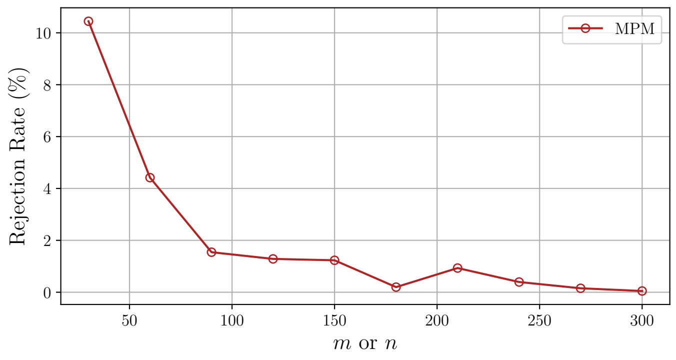


Figure A1. Rejection rate of MPM.

Appendix B. Computational Efficiency Analysis

Figure A2 compares the computational efficiency of MPM and DA. We can see from the figure that the time cost of MPM rises dramatically as the number of participants increases, while DA completes within 0.2 s and its time cost barely changes. This is because that MPM needs to solve quadratic programming problem (14). Calling traditional solvers in GEKKO is the main reason for the large time consumption of MPM. Moreover, the procedures of evaluating matching results, applying rejection rules, and generating the final matching results are also time consuming. On the other hand, DA does not need to solve any optimization problem or updating matching results. Its computational cost is mainly caused by traversing requirement lists and service lists.

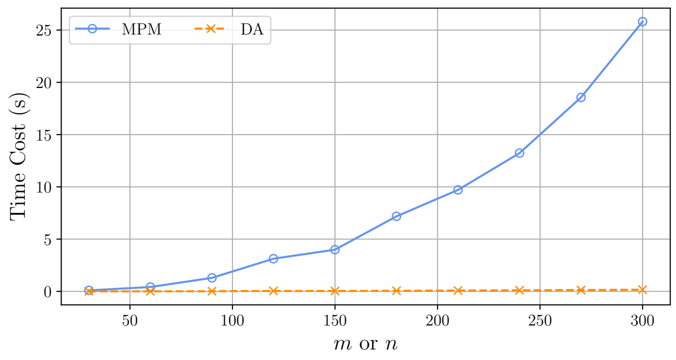
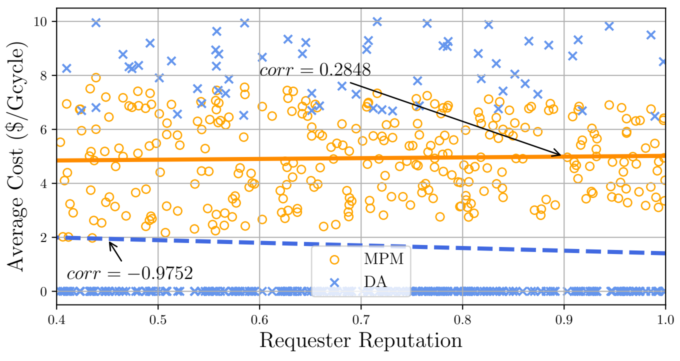


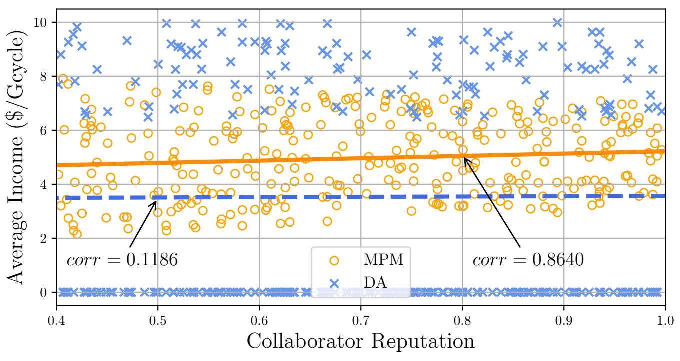
Figure A2. Comparison of computational efficiency.

Appendix C. Reputation Analysis

We look into the relationship between requester/collaborator reputation and average cost/income. Figure A3 looks into the matching results of the case of $m = n = 300$. In Figure A3b, we plot the scatter chart of average cost versus requester reputation and then draw the corresponding linear fitting line. The slope of the linear fitting line, denoted by *corr*, is the correlation coefficient between average cost and requester reputation. Since the *corr* of DA is negative but has a greater absolute value, the DA mechanism is friendlier to requesters with higher reputation scores. Similarly, Figure A3a shows that the *corr* of MPM is positive and has a slightly greater absolute value, it indicates that our MPM mechanism is more advantageous for collaborators with higher reputation. This also indirectly proves that our resource trading system with MPM can better incentivize the participation of collaborators through the distributed reputation system.



(**a**)



(**b**)

Figure A3. Comparisons of average costs of requesters and average incomes of collaborators. (**a**) Average Cost vs. Requester Reputation. (**b**) Average Income vs. Collaborator Reputation.

References

1. Wu, H.; Yan, Y.; Sun, D.; Wu, H.; Liu, P. Multibuffers Multiobjects Optimal Matching Scheme for Edge Devices in IIoT. *IEEE Internet Things J.* **2021**, *8*, 11514–11525. https://doi.org/10.1109/JIOT.2021.3053017.
2. Vakilian, S.; Fanian, A.; Falsafain, H.; Gulliver, T.A. Node cooperation for workload offloading in a fog computing network via multi-objective optimization. *J. Netw. Comput. Appl.* **2022**, *205*, 103428. https://doi.org/10.1016/j.jnca.2022.103428.
3. Liyanage, M.; Porambage, P.; Ding, A.Y.; Kalla, A. Driving forces for Multi-Access Edge Computing (MEC) IoT integration in 5G. *ICT Express* **2021**, *7*, 127–137. https://doi.org/10.1016/j.icte.2021.05.007.
4. Li, G.; Cai, J. An Online Incentive Mechanism for Collaborative Task Offloading in Mobile Edge Computing. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 624–636. https://doi.org/10.1109/TWC.2019.2947046.
5. Ng, J.S.; Lim, W.Y.B.; Garg, S.; Xiong, Z.; Niyato, D.; Guizani, M.; Leung, C. Collaborative Coded Computation Offloading: An All-pay Auction Approach. *arXiv* **2020**, arXiv:2012.04854.
6. He, J.; Zhang, D.; Zhou, Y.; Zhang, Y. A Truthful Online Mechanism for Collaborative Computation Offloading in Mobile Edge Computing. *IEEE Trans. Ind. Informatics* **2020**, *16*, 4832–4841. https://doi.org/10.1109/TII.2019.2960127.
7. Ai, S.; Hu, D.; Guo, J.; Jiang, Y.; Rong, C.; Cao, J. Distributed Multi-Factor Electricity Transaction Match Mechanism based on Blockchain. In Proceedings of the IEEE International Conference on Energy Internet (ICEI), Sydney, Australia, 24–28 August 2020; pp. 121–127. https://doi.org/10.1109/ICEI49372.2020.00030.
8. Yousefpoor, M.S.; Yousefpoor, E.; Barati, H.; Barati, A.; Movaghar, A.; Hosseinzadeh, M. Secure data aggregation methods and countermeasures against various attacks in wireless sensor networks: A comprehensive review. *J. Netw. Comput. Appl.* **2021**, *190*, 103118. https://doi.org/10.1016/j.jnca.2021.103118.
9. Wang, T.; Hua, H.; Wei, Z.; Cao, J. Challenges of blockchain in new generation energy systems and future outlooks. *Int. J. Electr. Power Energy Syst.* **2022**, *135*, 107499. https://doi.org/10.1016/j.ijepes.2021.107499.
10. Samaniego, M.; Jamsrandorj, U.; Deters, R. Blockchain as a Service for IoT. In Proceedings of the 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Chengdu, China, 15–18 December 2016. https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2016.102.
11. Nguyen, D.C.; Pathirana, P.N.; Ding, M.; Seneviratne, A. Blockchain as a Service for Multi-Access Edge Computing: A Deep Reinforcement Learning Approach. *arXiv* **2019**, arXiv:2001.08165.
12. Ai, S.; Hu, D.; Zhang, T.; Jiang, Y.; Rong, C.; Cao, J. Blockchain based Power Transaction Asynchronous Settlement System. In Proceedings of the 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), Virtual, 25 May–31 July 2020; pp. 1–6. https://doi.org/10.1109/VTC2020-Spring48590.2020.9129593.
13. Ai, S.; Hu, D.; Guo, J.; Jiang, Y.; Rong, C.; Cao, J. A Blockchain-based Distributed Controllable Electricity Transaction Match System. In Proceedings of the IEEE International Conference on Energy Internet (ICEI), Sydney, Australia, 24–28 August 2020; pp. 56–62. https://doi.org/10.1109/ICEI49372.2020.00019.
14. Bandara, K.Y.; Thakur, S.; Breslin, J.G. Flocking-based decentralised double auction for P2P energy trading within neighbourhoods. *Int. J. Electr. Power Energy Syst.* **2021**, *129*, 106766. https://doi.org/10.1016/j.ijepes.2021.106766.
15. Liu, C.H.; Lin, Q.; Wen, S. Blockchain-Enabled Data Collection and Sharing for Industrial IoT With Deep Reinforcement Learning. *IEEE Trans. Ind. Informatics* **2019**, *15*, 3516–3526. https://doi.org/10.1109/TII.2018.2890203.
16. Chi, J.; Li, Y.; Huang, J.; Liu, J.; Jin, Y.; Chen, C.; Qiu, T. A secure and efficient data sharing scheme based on blockchain in industrial Internet of Things. *J. Netw. Comput. Appl.* **2020**, *167*, 102710. https://doi.org/10.1016/j.jnca.2020.102710.
17. Lu, Y.; Huang, X.; Dai, Y.; Maharjan, S.; Zhang, Y. Blockchain and Federated Learning for Privacy-Preserved Data Sharing in Industrial IoT. *IEEE Trans. Ind. Informatics* **2020**, *16*, 4177–4186. https://doi.org/10.1109/TII.2019.2942190.
18. Yu, S.; Langar, R. Collaborative Computation Offloading for Multi-access Edge Computing. In Proceedings of the 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Washington, DC, USA, 8–12 April 2019; pp. 689–694.
19. Liu, Y.; Xie, S.; Zhang, Y. Cooperative Offloading and Resource Management for UAV-Enabled Mobile Edge Computing in Power IoT System. *IEEE Trans. Veh. Technol.* **2020**, *69*, 12229–12239. https://doi.org/10.1109/TVT.2020.3016840.
20. Liu, L.; Tsai, W.T.; Bhuiyan, M.Z.A.; Peng, H.; Liu, M. Blockchain-enabled fraud discovery through abnormal smart contract detection on Ethereum. *Future Gener. Comput. Syst.* **2022**, *128*, 158–166. https://doi.org/10.1016/j.future.2021.08.023.
21. Porambage, P.; Okwuibe, J.; Liyanage, M.; Ylianttila, M.; Taleb, T. Survey on Multi-Access Edge Computing for Internet of Things Realization. *IEEE Commun. Surv. Tutorials* **2018**, *20*, 2961–2991. https://doi.org/10.1109/COMST.2018.2849509.
22. Wang, S.; Huang, X.; Tan, B.; Yu, R. A Contract-Based Incentive Mechanism for Resource Sharing and Task Allocation in Container-Based Vehicular Edge Computing. In *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 116–129.
23. Wang, T.; Guo, J.; Ai, S.; Cao, J. RBT: A distributed reputation system for blockchain-based peer-to-peer energy trading with fairness consideration. *Appl. Energy* **2021**, *295*, 117056. https://doi.org/10.1016/j.apenergy.2021.117056.
24. Li, J.; Zhou, Z.; Wu, J.; Li, J.; Mumtaz, S.; Lin, X.; Gacanin, H.; Alotaibi, S. Decentralized On-Demand Energy Supply for Blockchain in Internet of Things: A Microgrids Approach. *IEEE Trans. Comput. Soc. Syst.* **2019**, *6*, 1395–1406. https://doi.org/10.1109/TCSS.2019.2917335.
25. Yao, H.; Mai, T.; Wang, J.; Ji, Z.; Jiang, C.; Qian, Y. Resource Trading in Blockchain-Based Industrial Internet of Things. *IEEE Trans. Ind. Informatics* **2019**, *15*, 3602–3609. https://doi.org/10.1109/TII.2019.2902563.

26. Wang, J.; Wu, W.; Liao, Z.; Sangaiah, A.K.; Sherratt, R.S. An Energy-Efficient Off-Loading Scheme for Low Latency in Collaborative Edge Computing. *IEEE Access* **2019**, *7*, 149182–149190. https://doi.org/10.1109/ACCESS.2019.2946683.
27. Slepak, G.; Petrova, A. The DCS Theorem. *arXiv* **2018**, arXiv:1801.04335.
28. Krichen, M.; Lahami, M.; Al-Haija, Q.A. Formal Methods for the Verification of Smart Contracts: A Review. In Proceedings of the 2022 15th International Conference on Security of Information and Networks (SIN), Sousse, Tunisia, 11–13 November 2022. https://doi.org/10.1109/sin56466.2022.9970534.
29. Almakhour, M.; Sliman, L.; Samhat, A.E.; Mellouk, A. Verification of smart contracts: A survey. *Pervasive Mob. Comput.* **2020**, *67*, 101227. https://doi.org/10.1016/j.pmcj.2020.101227.
30. Remix—Ethereum IDE. Available online: https://remix.ethereum.org/ (accessed on 17 March 2023).
31. Beal, L.; Hill, D.; Martin, R.; Hedengren, J. GEKKO Optimization Suite. *Processes* **2018**, *6*, 106.
32. Li, Z.; Zhou, X.; Liu, Y.; Fan, C.; Wang, W. A Computation Offloading Model over Collaborative cloud–edge Networks with Optimal Transport Theory. In Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China, 29 December 2020–1 January 2021. https://doi.org/10.1109/TrustCom50675.2020.00134.
33. Li, L.; Li, Y.; Li, R. Double Auction-Based Two-Level Resource Allocation Mechanism for Computation Offloading in Mobile Blockchain Application. *Mob. Inf. Syst.* **2021**, *2021*, 1–15. https://doi.org/10.1155/2021/8821583.
34. Haggi, H.; Sun, W. Multi-Round Double Auction-Enabled Peer-to-Peer Energy Exchange in Active Distribution Networks. *IEEE Trans. Smart Grid* **2021**, *12*, 4403–4414. https://doi.org/10.1109/TSG.2021.3088309.
35. Mao, J.; Tian, L.; Zhang, J.; Duan, G.; Wang, C. Many-to-Many Data Trading Algorithm Based on Double Auction Theory. *Proc. Comput. Sci.* **2020**, *174*, 200–209. https://doi.org/10.1016/j.procs.2020.06.075.
36. Miklánek, T.; Zajicek, M. Personal Traits and Trading in an Experimental Asset Market. *J. Behav. Exp. Econ.* **2020**, *86*, 101538