Hindawi
Security and Communication Networks
Volume 2021, Article ID 5563312, 18 pages
https://doi.org/10.1155/2021/5563312

Research Article

Blockchain-Enhanced Fair Task Scheduling for Cloud-Fog-Edge Coordination Environments: Model and Algorithm

Wenjuan Li,[1] **Shihua Cao**,[1] **Keyong Hu**,[1] **Jian Cao**,[2] **and Rajkumar Buyya**[3]

[1]*Qianjiang College, Hangzhou Normal University, Hangzhou, China*
[2]*Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China*
[3]*The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, The University of Melbourne, Melbourne, Australia*

Correspondence should be addressed to Wenjuan Li; liellie@163.com

Received 6 January 2021; Revised 3 March 2021; Accepted 22 March 2021; Published 5 April 2021

Academic Editor: Honghao Gao

The cloud-fog-edge hybrid system is the evolution of the traditional centralized cloud computing model. Through the combination of different levels of resources, it is able to handle service requests from terminal users with a lower latency. However, it is accompanied by greater uncertainty, unreliability, and instability due to the decentralization and regionalization of service processing, as well as the unreasonable and unfairness in resource allocation, task scheduling, and coordination, caused by the autonomy of node distribution. Therefore, this paper introduces blockchain technology to construct a trust-enabled interaction framework in a cloud-fog-edge environment, and through a double-chain structure, it improves the reliability and verifiability of task processing without a big management overhead. Furthermore, in order to fully consider the reasonability and load balance in service coordination and task scheduling, Berger's model and the conception of service justice are introduced to perform reasonable matching of tasks and resources. We have developed a trust-based cloud-fog-edge service simulation system based on iFogsim, and through a large number of experiments, the performance of the proposed model is verified in terms of makespan, scheduling success rate, latency, and user satisfaction with some classical scheduling models.

1. Introduction

The traditional centralized cloud service mode, with all tasks being handled in the center, faces big challenges in practical applications, including high latency, network dependency, single point of failure, and failure scale effect and cannot adapt to the instant transaction scenarios. Therefore, researchers put forward the idea of distributing some time-sensitive and low-resource demand services to be processed at the edge of the network, which is the prototype of fog or edge computing.

Based on the idea, this paper designs a cloud-fog-edge hybrid computing architecture, in which the fog layer is actually a management middleware between edge and cloud, helping the scheduler decide where to deploy and implement a service, thus to better achieve the resource balancing and low service latency [1].

The cloud-fog-edge computing model provides a superior framework for the distributed coordination of resources and the timely process of tasks. However, due to the different level of heterogeneity and resource asymmetry, the possible cross-layer task offloading, and the dynamics and mobility of edge nodes, the three-tier hybrid architecture has been facing with a more severe crisis in terms of service credibility and rational resource allocation and scheduling [2–5]. On the other hand, the hybrid service provision environment, which contains the edge or IoT layer, has more abundant, random, and diversified application scenarios than traditional cloud computing and requires the task scheduling strategy to be more adaptive and robust.

At present, researchers have proposed many valuable solutions for the efficient task scheduling in the distributed systems (see [6–11]). However, the existing strategies cannot achieve full functions in a cloud-fog-edge environment due to the following reasons: (1) the centralized trust management model cannot handle the identity authentication and behavior management under the heterogeneous and

decentralized architecture, (2) the existing distributed or decentralized trust models cannot provide enough trust evidence with sufficient credibility to convince entities in the same domain or across domains, and (3) it is difficult to guarantee the relative fairness and global rationality of resource distribution in scheduling.

In response to these problems, this paper proposes a trust-enhanced fair scheduling model for the cloud-fog-edge environments. It firstly deploys the blockchain technology to build a decentralized trust framework to solve the problem of service credibility. And then, it introduces Berger's theory of wealth distribution to propose the concept and evaluation method of service fairness. Considering the mobile application scenarios, we divide scheduling into two levels. The first level is user scheduling, which deals with the matching of mobile terminals and access points, and the second one is task scheduling, which realizes the matching of tasks and resources in the local pool. Task scheduling in a cloud-fog-edge environment is treated as a multiobjective optimization problem, which comprehensively considers terminal mobility, QoS service requirements, and workload.

The main contributions of this paper are (1) it proposes a novel distributed decentralized trust management model based on blockchain technology; (2) it introduces Berger's fairness theory to design a preference-based fair task scheduling model for the cloud-fog-edge environments; and (3) it designs a new task scheduling algorithm that comprehensively considers user mobility, load balancing, and trust.

The rest of the paper is organized as follows. Section 2 briefly introduces the related work. Section 3 introduces the system architecture along with the design details of the double-blockchain structure-based service transaction framework. The trust-enhanced location-aware fair scheduling algorithm is proposed in Section 4. And performance evaluation is presented in Section 5. The last section concludes the paper.

2. Related Work

2.1. Task Scheduling Model. Task scheduling refers to the process of unified resource allocation among all resources and users based on a certain resource usage rules and user requirements in a specific service provision environment. Task scheduling is the core of cloud computing. The existing task scheduling models can generally be divided into three categories: performance-centric, user-centric, and hybrid.

The goal of the performance-centric models is to improve the performance of the systems, such as system throughput, makespan, and energy consumption. The classical task algorithms like min-min, max-min, greedy algorithm, swarm intelligence, and genetic algorithm all belong to this group. Aiming at solving the energy efficiency (EE) problem in fog computing, Y. Yang et al. proposed a collaborative task offloading algorithm named MEETS [12, 13]. In [14], they developed a delay and energy consumption balanced scheduling algorithm called DEBTS. H. Sun et al. designed a contract and cluster-based resource allocation model for fog-cloud hybrid platform [15]. N. Auluck et al. proposed a security aware task scheduling model for fog computing, through the classification of users and providers, achieving a better system performance [16].

Improving service QoS and enhancing user service experience are the primary tasks of the user-centric models [17–24]. In order to minimize the service delay, Z. Liu et al. proposed a dispersive stable task scheduling model named DATS [25]. M. Mukherjee et al. introduced a deadline-aware task scheduling algorithm for fog computing to complete as many tasks as possible before their deadlines [26]. G. Zhang et al. proposed a task offloading algorithm for fog computing systems to reduce the service delay [27]. G. Zhang et al. put forward a fog task offloading algorithm named DOTS to decrease latency with the help of the voluntary nodes (VNs) [28]. H. Apat et al. proposed a three-layered priority-based fog task scheduling model to meet the different deadline requirement of tasks [29].

In recent years, researchers have proposed many hybrid scheduling models considering multifactors, of which energy consumption and delay were their focus. For example, C. Tang et al. used genetic algorithms to design a hybrid task scheduling algorithm for mobile cloud computing taking into consideration multifactors such as energy consumption task requirement like deadline and cost [30]. J. Xu et al. proposed a task scheduling model combining laxity and ant colony algorithm to satisfy both energy consumption and latency [31]. In order to improve the makespan and execution cost, M. Yang et al. put forward an evolutionary heuristic-based multiobjective task scheduling model for fog environment [32]. Caching is an effective way to reduce the execution delay of tasks in edge computing, and thus some novel caching strategies have been proposed by researchers [33–35].

However, indicators considered by most current scheduling models are simply makespan, service QoS, load balancing, or economic principles, and the fairness of scheduling is usually ignored.

2.2. Trust Management in Distributed Systems. Trust is an efficient mechanism in dealing with the reputation and reliability issues in the distributed open network environment. A large number of highlight resulting in trust and trust-enabled interactions are achieved [36, 37].

W. Tian et al. made a survey on trust evaluation methods in sensor cloud systems [38]. J. Wang et al. proposed a recommendation trust evaluation method based on the cloud model and attribute weighted clustering [39]. P. Zhang et al. introduced a domain-based trust management model for the public cloud [40]. For the trust-base service management, Y. Li put forward an online learning aided service offloading model for mobile edge computing [41]. H. Yang et al. designed a context-aware trust prediction model for vehicle edge computing [42]. S. Jian et al. proposed a trust-based multiobjective task allocation model for cloud service transactions [43]. C. Hu et al. designed an e-commerce recommendation algorithm combining trust and distrust factors [44]. X. Meng et al. proposed a two-tier service selection method that matched

credibility and behavior patterns [45]. Z. Ma et al. introduced a trust-enabled edge data management model based on blockchain technology [46]. L. Cui et al. put forward an edge service configuration method combined with the decentralized trust [47]. We have also done some interesting work in this area, such as the service trust architecture for open cloud environment [48], trust decision strategy based on fuzzy clustering [49], and trust and preference learning-based service matching and combination models [50, 51].

However, the existing solutions cannot achieve full functions in mobile fog computing systems due to the following limitations: (1) the centralized trust framework cannot be accurately integrated with the cloud-fog-edge hybrid systems characterized by node dynamic distributed autonomy and topology loosely coupled, (2) trust crisis of the center node, which easily leads to single point of failure, (3) huge trust management overhead prevents it from being used in the instant trading scenario, and (4) trust evidence lacks transparency. Therefore, the decentralized trust management model and trust-enabled transaction mechanism for the cloud-fog-edge hybrid environments require further exploration.

3. System Overview

3.1. Trust-Enhanced Cloud-Fog-Edge Hybrid Framework. In a cloud-fog-edge hybrid framework, fog computing architects a management middleware between cloud servers and edge devices to coordinate resource allocation, thus improving system performance and fulfilling the customized demand of different devices and users. Figure 1 shows the hybrid scheduling framework of the proposed model. Blockchain technology is introduced to construct a decentralized trust model in the IoT device layer, thereby enhancing the interactive credibility of the entire cloud-fog-edge hybrid architecture.

The framework consists of three layers: (1) trust-enhanced edge/IoT layer, (2) fog layer, and (3) cloud layer. The trust-enhanced edge/IoT layer implements the peer-to-peer interconnections through the ubiquitous sensors and communication protocols over the traditional IoT infrastructure layer and constructs the distributed and decentralized trust management with a blockchain architecture. The fog layer is introduced to decide the allocation and management of resources and handle complex transactions such as cross-group/cross-cloud tasks. The fog layer consists of a large number of fog servers deployed at the network edge. Fog servers are much closer to users or terminals than the centralized cloud server and are more capable than the terminals. Therefore, they can ensure lower latency and meet the needs of cross-group or cross-domain interactions. Trust management becomes one part of the responsibility of edge/fog servers. The cloud layer is located at the top of the framework. It is mainly used to deal with some high-level and highly complex tasks, such as data mining and behavior or preference analysis, which impose high requirements on computing and storage capacity and relatively loose requirement on the response time.

3.2. Service Transaction Model Based on a Double-Blockchain Structure. Generally speaking, a complete trust authentication contains two parts: identity authentication and behavior evaluation. And the first part is easy to obtain or evaluate because the identity information of a node is relatively statically stable even in a P2P network topology. In contrast, trading behavior is dynamic, requiring a lot of computing power to record and assess. Therefore, to improve the integrity and efficiency of the trust management in a real-time trading environment, a cloud service transaction model based on a double-blockchain structure is proposed, as shown in Figure 2.

The double-blockchain structure contains two blockchains, including a trust authentication blockchain and a trading behavior blockchain.

Definition 1. Trust Authentication Blockchain (TAB). TAB is a blockchain structure that stores the trust data of the cloud-fog-edge hybrid transaction system and assists trust transaction decisions. TAB adopts an alliance chain structure, in which blocks store the trust data of nodes in the transaction system.

TAB is responsible for managing trust data in the cloud service markets and provides trust evaluation results to other nodes. Each block in TAB contains two parts: identity trust data and behavior trust data. When a node initially joins, only the identity part is written; however, as time goes by with the transactions progressing, the behavior part is continuously written. Authentication is completed by a small number of supervisors, who can be the normal miners or some special nodes elected by the market authority. Miners are responsible for storing and authenticating trust data and ensuring the consistency of the data through some specifically designed consensus mechanisms. When nodes apply to enter the trading network, they must pay a fee to run a smart contract for the initial identity authentication. In addition, when they want to obtain the trust data of the other nodes, they also have to pay a fee. The funding provides the incentive fee for the miners. Figure 3 shows the basic content of a trust block of TAB.

Definition 2. Trading Behavior Blockchain (TBB). TBB is a blockchain structure that stores the transaction data in the cloud-fog-edge transaction system. A block in TBB contains both the transaction data and the evaluation data, which will assist in generating the behavioral trust data.

TBB is responsible for generating and storing the trading data. In TBB, the miners have two tasks, one is to generate the new transaction blocks based on the latest transaction results and the other is to evaluate the behavior trust, generate a trust block, and then forward it to TAB. The corresponding trust block will be confirmed and stored by the miners in TAB. Figure 4 shows the basic content of a trading block in TBB. The block structure of TBB is very similar to that of TAB, and the only difference is the storage content.

TAB and TBB jointly ensure the credibility of cloud-fog-edge interaction. When an entity registers for the first time, its identity trust will be written into TAB. As the transaction

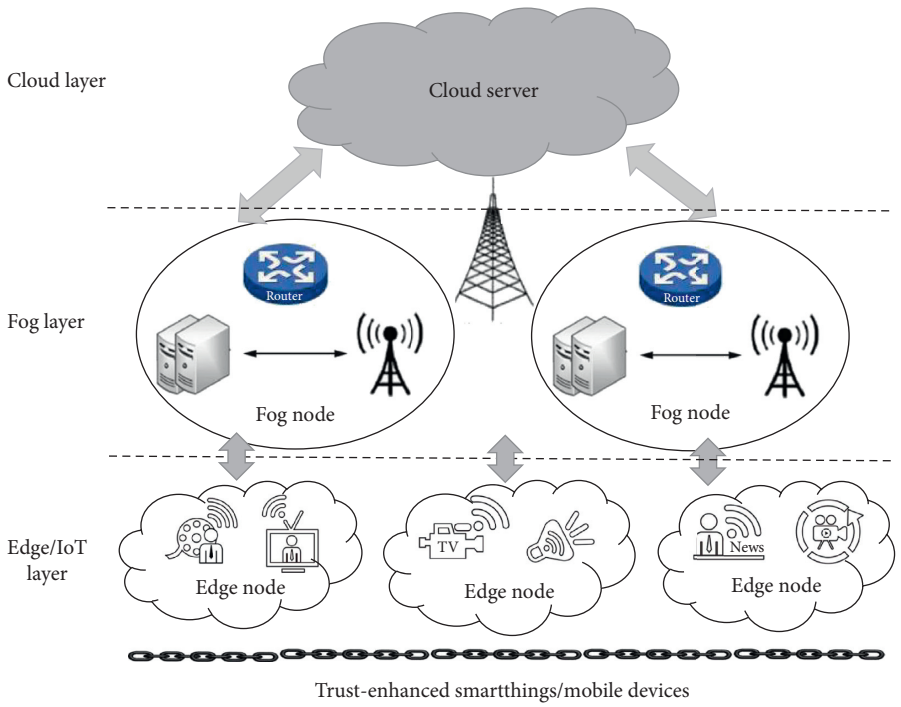


Figure 1: Trust-enhanced cloud-fog-edge hybrid framework.

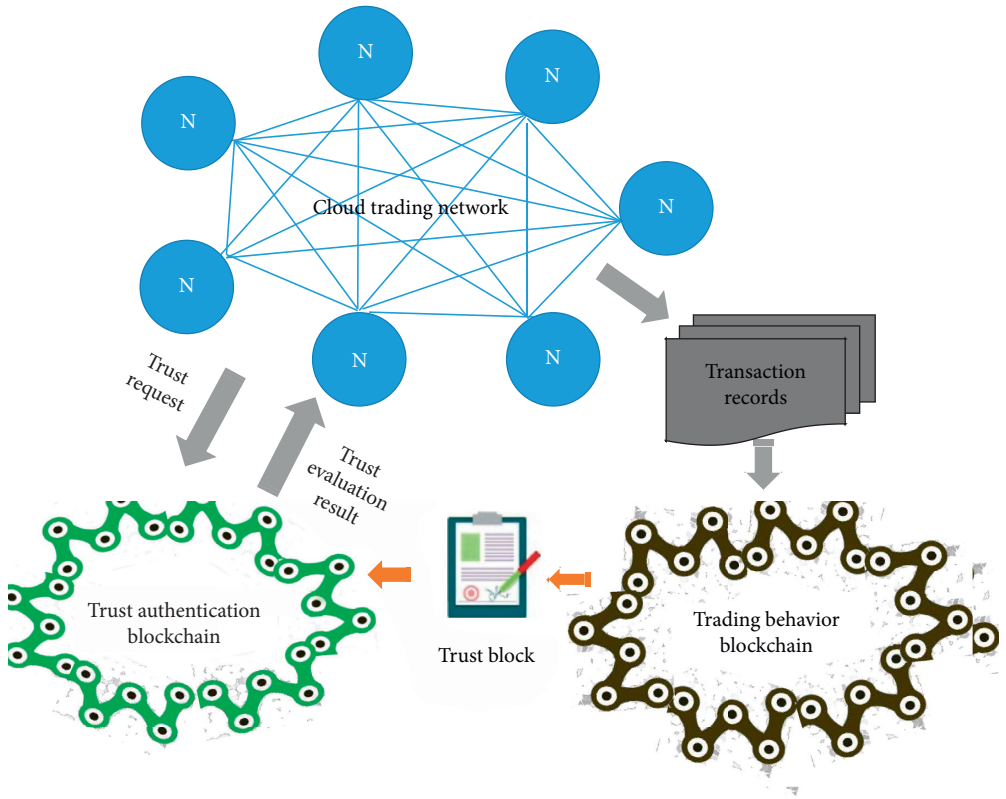


Figure 2: Service transaction model based on a double-blockchain structure.

progresses, the trading record between entities will be written into TBB one by one and then the behavior trust will be evaluated in TBB and be forwarded into TAB. When TAB gradually grasps enough trust data of the entities in the transaction chain, it can help entities make trust decisions more accurately, which improves the reliability and success

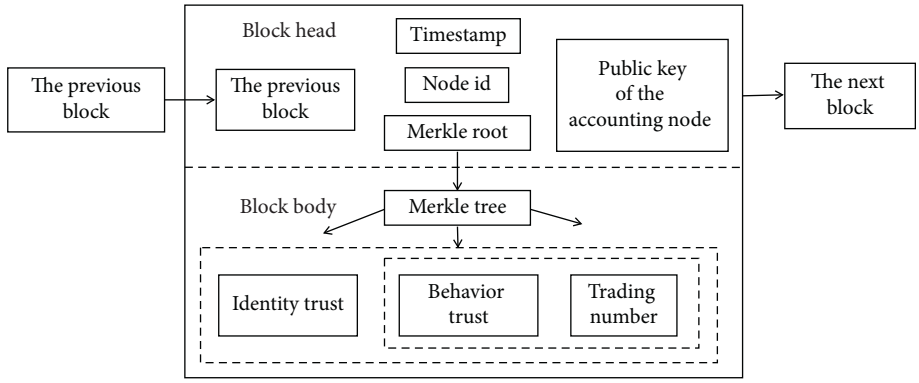


Figure 3: The basic content in a trust block of TAB.

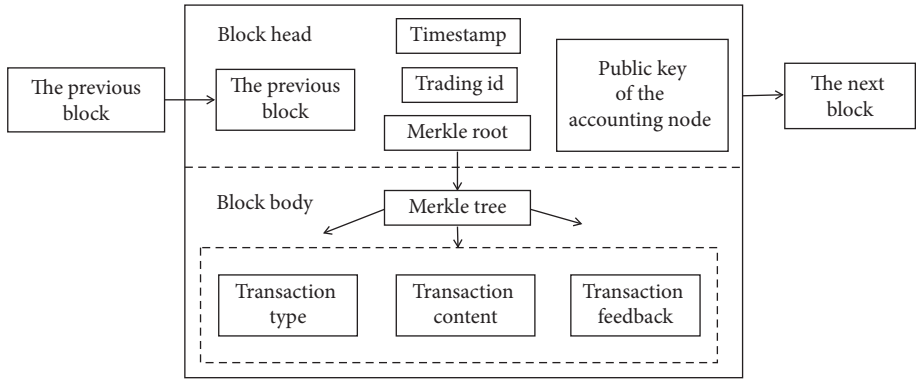


Figure 4: The basic content in a trading block of TBB.

rate of interactions. Furthermore, with the benefits of the double-blockchain structure, an efficient parallel computing is realized. Because trust value is provided by TAB, while the large-scale calculation or evaluation is done on the TBB side, this effectively reduces the latency caused by trust management and makes it possible for the application of blockchain in a real-time and high-reliability scenarios.

4. Trust-Enhanced Location-Aware Fair Scheduling Model in a Cloud-Fog-Edge Computing System

4.1. Service Justice. The theory of distribution justice (Berger's theory) proposed by Berger et al., is a theory of the distribution of social wealth. Berger et al. believed that justice is a subjective behavior, of which the conclusion is drawn from the comparison with the comparable objects. In the real world, people usually refer to the surrounding social information, such as the status and compensation of others, to generate their own expectations, which are used to judge whether they are fairly treated or not. Furthermore, the satisfaction of these expectations affects their future behaviors. Figure 5 is the diagram of Berger's model.

Here, c or C is the attribute set and go or GO is the expectation set; go represents the expectation value, while GO means the actually obtained value. In Berger's theory, the distribution justice holds only when the local self-structure and the reference structure are significantly related.

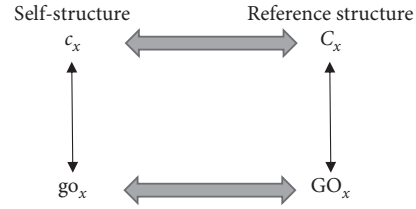


Figure 5: Berger's model.

In a cloud-fog-edge hybrid scheduling framework, fog provides services to users by coordinating resources from edge or cloud and users pay a service fee according to the usage of resources or time. Therefore, service fairness of users is reflected in the following: the scheduling system is able to provide services on user demand and preferences, ensuring a high user satisfaction and experience.

Definition 3. Service Justice (SJ). It represents the degree of agreement between the QoS of user actually obtained and the expected. How to evaluate SJ_i is given in the following formula:

$$SJ_i = \theta \ln \frac{OS_i}{ES_i}, \tag{1}$$

where θ represents the equilibrium coefficient, OS_i means the actual obtained service quality vector of $user_i$, and ES_i is the demand vector of $user_i$.

Obviously, the overall service justice of the scheduling system is the average of all system users' SJ, which is illustrated in the following formula:

$$SJ = \sum_{i=1}^{n} \varphi_i SJ_i. \tag{2}$$

The overall system balance is achieved by constraining the overall system service justice, namely, min SJ.

Suppose the capability vector of a resource can be described by $CR = (C_{cpu}, C_{ram}, C_{bd})$, where C_{cpu}, C_{ram}, and C_{bd} represent the CPU, memory, and bandwidth capability of the resource/vm. The resource expectation of a user to the resource is described by $ER = (E_{cpu}, E_{ram}, E_{bd})$, where E_{cpu}, E_{ram}, and E_{bd} represent the expected CPU, memory, and bandwidth capability of the resource. In order to eliminate the influence of dimensions of different capability, it uses the following formula for normalization:

$$\frac{C_i - C_{min}}{C_{max} - C_{min}} \text{ or } \frac{E_i - E_{min}}{E_{max} - E_{min}}. \tag{3}$$

After normalization, performance or expectation parameters of the virtual machine are mapped to the [0, 1]. However, different users or tasks have different service preferences, for example, some are CPU or computing power enthusiasts and some are bandwidth pursuers. Let $P = (P_{cpu}, P_{ram}, P_{bd})$ describe the service preference or weight of the attribute. It uses modified cosine function to compare the similarity and distance between expectation and obtained. Let SJ_{ij} denote the service justice when $task_i$ is scheduled to $resource_j$, and SJ_{ij} can be calculated by the following formula:

$$SJ_{ij} = \frac{CR_iP_i \cdot ER_iP_i}{\|CR_iP_i\| \|ER_iP_i\|}. \tag{4}$$

Accordingly, the SJ of the task set can be obtained by the following formula:

$$SJ = \sum_{i=1}^{n} \frac{CR_iP_i \cdot ER_iP_i}{CR_iP_iER_iP_i}. \tag{5}$$

Here, n denotes the number of tasks in the scheduling system.

4.2. Trust-Based Location-Aware Fair Scheduling Model

4.2.1. User/Smart Thing Model. A user or a smart thing is an intelligent entity who requests for mobile services. Let $ST = \{st_0, st_1, \ldots, st_{k-1}\}$ represents a user or a smart thing set. The i^{th} user st_i ($i \in [0, k-1]$) can be further described as $st_i = \{stI_D, st_{Name}, st_{Location}, st_{TaskSet}, st_{Trust}\}$. The meaning of each attribute is as follows:

(i) st_{ID} represents the unique system identity of a user/smart thing

(ii) st_{Name} represents the name of the user/smart thing

(iii) $st_{Location}$ represents the location of the user/smart thing

(iv) $st_{TaskSet}$ represents the task set submitted by the user/smart thing

(v) st_{Trust} represents the trust requirement of the user/smart thing

4.2.2. Task Model. $T = \{t_0, t_1, \ldots, t_{n-1}\}$ represents a task set, where the i^{th} task ti ($i \in [0, n-1]$) can be further described as $ti = \{t_{ID}, t_{State}, t_{RRes}, t_{ORSet}, t_{DeadLine}\}$. The meaning of each attribute is as follows:

(i) t_{ID} represents the identity of a task

(ii) t_{State} represents the state of the task

(iii) t_{RRes} represents the resource requirements of the task, t_{RRes} can be further described as $t_{RRes} = \{t_{Comp}, t_{BW}, t_{Stor}\}$, where t_{Comp}, t_{BW}, and t_{Stor} represent the computation, network, and storage requirement of the task

(iv) t_{ORSet} represents the obtained resource set of the task

(v) $t_{DeadLine}$ represents the latest completion time of the task

4.2.3. Provider Model. $AP = \{ap_0, ap_1, \ldots, ap_{k-1}\}$ represents the provider set. In this paper, service provider refers to the access point or gateway server. In a cloud-fog-edge hybrid platform, a service provider can be a cloud server, a fog server, or even an edge server.

The i^{th} provider ap_i ($i \in [0, k-1]$) can be further described as $api = \{ap_{ID}, ap_{Name}, ap_{ResouceSet}, ap_{Load}, ap_{Location}\}$. The meaning of each attribute is as follows:

(i) ap_{ID} represents the unique system identity of a provider

(ii) ap_{Name} represents the name of the provider

(iii) $ap_{ResourceSet}$ represents the resource set managed by the provider

(iv) ap_{Load} represents the load of the provider

(v) $ap_{Location}$ represents the location of the provider

4.2.4. Resource Model. $R = \{r_0, r_1, \ldots, r_{m-1}\}$ represents the resource set. The j^{th} resource rj ($j \in [0, m-1]$) can be further described as $rj = \{r_{ID}, r_{Name}, r_{Provider}, r_{Cap}\}$. The meaning of each attribute is as follows:

(i) r_{ID} represents the unique system identity of a resource

(ii) r_{Name} represents the name of the resource

(iii) $r_{Provider}$ represents the manage provider of the resource

(iv) r_{Cap} represents the capability of the resource, $rCap = \{rComp, rBW, rStor\}$, where r_{Comp}, r_{BW}, and r_{Stor} represent the computation, network, and storage capability of the resource.

4.2.5. Two-Level Service Scheduling Model. This paper divides scheduling in a cloud-fog-edge hybrid environment into two levels, user scheduling and task scheduling. User scheduling realizes the binding of the smart thing (mobile user or terminal) to the credible nearest access point, while task scheduling realizes the matching of tasks issued by the user with resources/vms managed by the access point.

The user scheduling model is defined as USM = (ST, AP, ACS), which selects the most suitable gateway or access point for the user from the list of providers:

(i) ST means the smart thing set, as defined in 4.2.1.

(ii) AP represents the gateway or access point set, as defined in 4.2.3.

(iii) ACS: AP->AP*i* represents the selection of the access point *ap* for smart thing *st* using a certain strategy. In a cloud-fog-edge environment, since the location of a user or terminal is constantly changing and a fog or edge server also has a limited service capability, distance between *st* and *ap* must be considered. Besides, for a trustworthy trading, credibility is another important factor. Therefore, the primary principles of ACS include distance and trust.

The task scheduling model is defined as TSM = (*T*, *R*, RCS), which matches the most suitable resources for the task set of the user.

(i) *T* means the task set, as defined in 4.2.2.

(ii) *R* means the resource set managed by the service provider, as defined in 4.2.4.

(iii) RCS: *R*- > *Ri* represents the selection of the resource *r* for task *t* using a certain strategy. The main selection principle of RCS in this paper is service justice defined in 4.1 and work load.

Figure 6 shows the general steps of the proposed model in this paper.

(1) The user scheduler searches appropriate access points for a specific user/smart thing according to the current location and makes recommendation according to distance and trust;

(2) When the recommendation arrives, the user/smart thing requests a trust service from the TAB and makes decision after obtaining the trust data. If it agrees to trade, then proceed to step (3); otherwise, it asks the scheduler to resume recommendation and returns to step (1);

(3) The access point (gateway) starts the task scheduler to select specific resources for the user;

(4) The task scheduler selects the appropriate virtual machines to perform tasks according to the principle of service justice and load balance;

(5) The user obtains the service and makes service evaluation. The transaction data enter the TBB;

(6) TBB regularly performs trust evaluation and forwards trust blocks into TAB.

4.3. User Scheduling Algorithm. The user scheduler chooses the most suitable access point for user or smart thing as shown in Algorithm 1. The basic principle is distance and trust.

In Algorithm 1, coverage means the service coverage of an *ap*, and *trust_threshold* is the minimum required trust value of the smart thing.

The user scheduler chooses the nearest and reliable access point for a user. The general steps of user scheduling are as follows: (1) the user scheduler selects the candidate subset $AP_{candidate}$ from AP that can cover st_i and meet its minimum trust requirement, (2) it calculates the distance between ap and st_i and chooses the nearest ap, namely, ap_{chosen} from $AP_{candidate}$, and (3) it connects st_i with ap_{chosen}.

4.4. Task Scheduling Algorithm. The task scheduling algorithm references Berger's model to ensure the overall system justice.

Algorithm 2 shows how task scheduler (running on an access point) chooses the suitable resources/servers for mobile tasks triggered by a user or a smart thing.

In Algorithm 2, *load_safe_range* represents the safe load limit of a server or a resource. The general steps of task scheduling are as follows: (1) it puts the load-safe virtual resources into the candidate resource set $R_{\text{candidate}}$, (2) it calculates the service fairness of each resource r_k in $R_{\text{candidate}}$ according to the demand of *ti* and the capability of r_k, and selects the most suitable resource r_{chosen}, which achieves the maximum *SJ*, and (3) it schedules *ti* to r_{chosen} and updates the workload of r_{chosen}.

4.5. Blockchain-Architected Lightweight Trust Algorithm. Trust runs through the entire process of scheduling. The node initiates the transaction requests trust service from the trust blockchain TAB. After paying a certain fee, the trustworthiness of the candidate provider (the access server or the application server) is obtained, thereby helping it to make trustworthy decisions. After each transaction, transaction-related data (transaction type, content, and evaluation data) will be used to generate transaction blocks and be added to the transaction blockchain TBB after consensus. Thereafter, TBB initiates a trust evaluation transaction at regular intervals and pushes the new trust block to TAB.

This paper uses a lightweight method to compute the trust of a specific trading node as shown in the following formula:

$$T_i = \sum_{k=1}^{n} \theta V_{ki}, \tag{6}$$

where T_i is the trust value of node i, θ is the evaluation weight of trustor node k, and V_{ki} represents the trust evaluation value from k to i.

In the proposed mode, in order to deploy blockchain-based framework, we defined some transaction-related classes and trust-related classes. The transaction-related classes are used to maintain the transaction data and keep it traceable and tamper-proof. The trust-related classes are

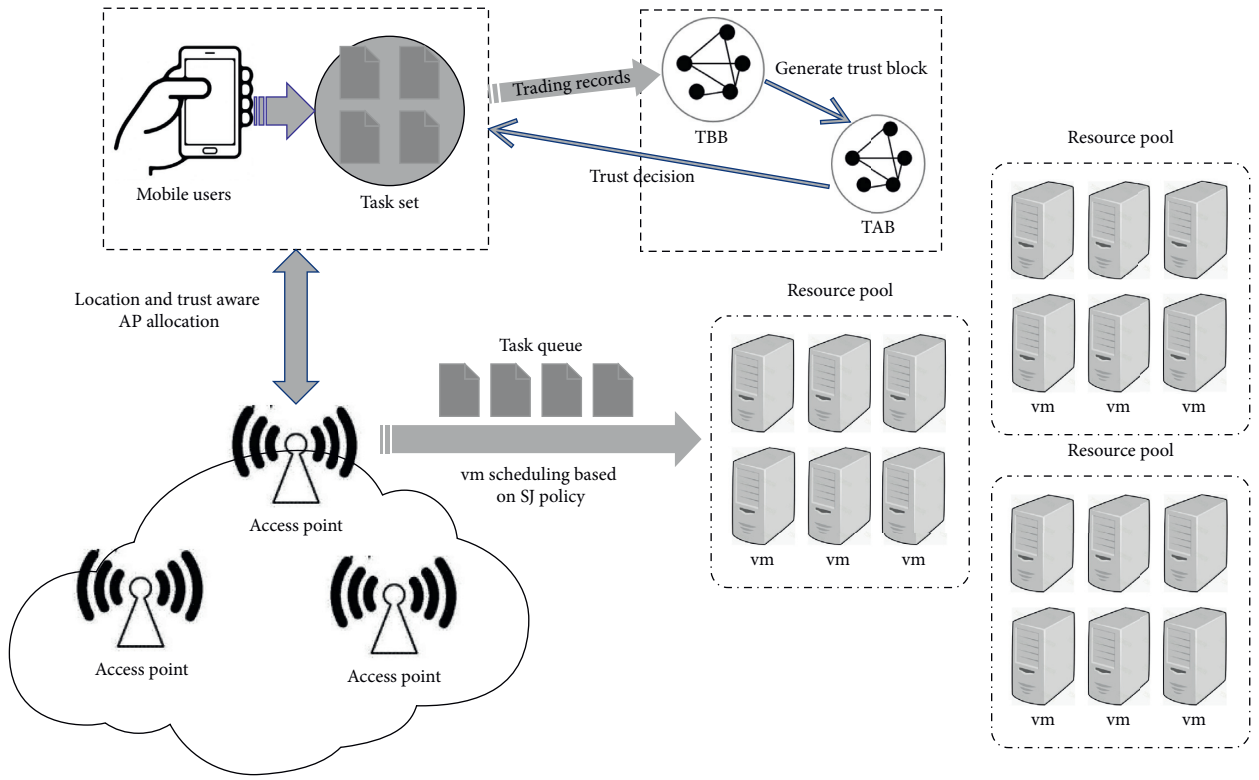


Figure 6: Trust-based location-aware fair scheduling model.

Input: AP, st_i, coverage, trust_threshold
Output: the matching of st_i to an appropriate ap

(1) **for** each ap_j in AP **do**
(2) Calculate the distance from st_i to ap_j;
(3) Calculate trust from st_i to ap_j;
(4) **if** ($distance_{sti\_apj} \leq coverage$) && ($trust_{sti\_apj} \geq trust\_threshold$) **then**
(5) $ap_j -> AP_{candidate}$
(6) **end if**
(7) **done**
(8) **for** each ap_k in $AP_{candidate}$ **do**
(9) **if** ($distance_{sti\_apk} < min\_distance$) **then**
(10) $min\_distance = distance_{sti\_apk}$;
(11) $ap_{chosen} = ap_k$;
(12) **end if**
(13) **done**
(14) **if** $ap_{chosen} != NULL$ **then**
(15) connect st_i to ap_{chosen};
(16) **end if**

Algorithm 1: Connect st_i with an appropriate access point ap.

used to maintain trust data. They are mainly generated offline and can support trust decisions in the real-time transactions. Following is the definition of the critical data structures. The class "transaction" defined in Algorithm 3 stores transaction data of the trading system, through which trust value from trustor to trustee can be obtained. And the class "TrustBlock" defined in Algorithm 4 is used to store the trust value of a certain node. Each block in TAB or TBB has the normal content of a blockchain block (timestamp, hash value, and the pointer to the previous block) and some special functions (mineBlock, calculateHash, and addTransaction). (Algorithms 3 and 4)

5. Performance Evaluation

5.1. Experimental Design. This paper designed a task scheduling prototype for performance test in a cloud-fog-edge environment based on iFogsim [52]. In iFogsim, FogDevice is

(i) **Input:** R, ti, $load\_safe\_range$
(ii) **Output:** the matching of ti to an appropriate r
(1) **for** each r_j in R **do**
(2) observe the workload of r_j;
(3) **if** ($workload_{rj} \leq load\_safe\_range$) **then**
(4) r_j - > $R_{candidate}$
(5) **end if**
(6) **done**
(7) **for** each r_k in $R_{candidate}$ **do**
(8) Calculate service justice $SJ_{ti\ \_rk}$ of ti in r_j according to formula (4);
(9) **If** ($SJ_{ti\ \_rk} < min\_sj$) **then**
(10) $min\_sk = SJ_{ti\ \_rk}$;
(11) $r_{chosen} = r_k$;
(12) **end if**
(13) **done**
(14) **if** r_{chosen}! = $NULL$ **then**
(15) schedule t_i to r_{chosen};
(16) update the workload of r_{chosen};
(17) **end if**

Algorithm 2:. Matching mobile task ti with the suitable resource r.

used to represent a common device from terminal to fog server. In order to reflect the mobility and distinguish the functions of different devices, new entities are added: MobileDevice, AccessPointDevice, MobileSensor, and MobileActuator. Inheriting from parent class FogDevice, MobileDevice handles mobile tasks, making trust decision and service selection. A mobile device randomly appears in a specific location and has its own specific resource preferences. MobileSensor inherits from sensor, acting as a mobile sensor, and is able to generate and transmit tasks. While MobileActuator inherits from Actuator to deal with the execution and output of tasks. AccessPoint also inherits from FogDevice, which is an intermediary for resource allocation and service scheduling according to the location of the mobile devices. It helps a mobile device to choose the most cost-effective service resources and handles real-time task migration. The resource pool that actually executes tasks is a collection of traditional fog devices, and each fog device is configured with certain capabilities. Figure 7 shows the key classes and their inherent relationships in the scheduling model.

This paper draws on the idea of blockchain to construct a trust-enhanced task scheduling model. The proposed model contains a dual-blockchain structure, including a transaction blockchain and a trust blockchain. Thus, the new classes including the class of TransactionBlock and TrustBlock class are derived. The transaction block records the actual transactions of the system in a traceable and nontamperable manner. And the trust block records the trust value of the nodes, where the identity trust depends on the authenticity of the node, while the behavior trust is continuously updated by the transaction evaluation results. The generation of the trust block mainly adopts the offline mode, and a trust block is generated according to the feedback data obtained from the transaction blockchain. Trust management contains three parts: trust initialization, trust decision, and trust maintenance. Trust decision helps entities to choose the credible trading partners. Figure 8 is the UML diagram of the trust-related classes and their relationships.

In order to serve mobile devices, we designed the mapping and migration rules for tasks in virtual machines, which were implemented in the class MigrationPolicy. The main strategies include load balance, shortest distance between smartThing and AccessPoint or between smartThing and foglets (vms), lowest latency, and trust-based. Figure 9 shows the migration-related classes and their relationships.

5.2. Simulation Indicator and Benchmarks. In the experiments, we use various evaluation indicators for performance test of the different scheduling models, such as makespan, tuple loss rate, distribution justice, latency, and transaction success rate. Here, makespan represents the overall execution time of the task set. Tuple loss rate refers to the percentage of tasks dropped due to the limited processing capacity of the edge server or task migration. And the system justice is defined in formula (5).

The mentioned strategies were implemented and tested in the EEG Tractor Beam Game [53]. It is a latency-critical game requiring each player to wear an EEG headset to process the EEG signals and obtain his brain state. In EEG Tractor Beam Game, there are seven types of tuples carried between the different modules of the application, as shown in Table 1. The capabilities of cloud, fog, and edge devices are shown in Table 2, and the task sending interval of two different EEG headsets is 10 ms and 5 ms, respectively.

The speed of MobileDevice is set to be 20 m/s, including nine different moving directions: NONE, EAST, NORTH, NORTH EAST, NORTHWEST, WEST, SOUTHWEST,

```
    class Transaction {
(1) String transactionId;
(2) PublicKey trustor;//the public key of trustor
(3) PublicKey trustee;//the public key of trustee
(4) double trust;//trust value from trustor to trustee
(5) byte[] signature;
(6) ArrayList < Transaction > transactions;
(7) boolean processTransaction() {
(8)             verifySignature();
(9)             transactionId = calulateHash();
(10)            generateTrust();//generate trust according to transaction details
(11)            TransactionOutput(this.recipient, trust, transactionId));
(12)            return true;
(13) } //add transactions into transactionChain TBB
(14) boolean addTransaction(Transaction transaction) {
(15)            verifyTransaction();
(16)            processTransaction();
(17)            transactions.add(transaction);
(18)            return true;
(19) }
    }
```

Algorithm 3: Class of transaction.

```
    TrustBlock {
(1) String hash;
(2) String previousHash;
(3) String merkleRoot;
(4) long timeStamp;
(5) int nonce;//the proof of miners
(6) double trust;//the integrated trust of a certain node
(7) TrustBlock(String previousHash) {//the basic structure of a trust block
(8)         this.previousHash = previousHash;
(9)         this.timeStamp = new Date().getTime();
(10)        this.hash = calculateHash(); }
(11) void mineBlock() {
(12)        merkleRoot = StringUtil.getMerkleRoot(transactions);
(13)        while(!hash.substring(0, difficulty).equals(target)) {
(14)            nonce ++;
(15)            hash = calculateHash();
(16) }
(17) }
    }
```

Algorithm 4: Class of TrustBlock.

SOUTH, and SOUTHEAST. The maximum switch radius of vm to a certain access point is set to be 40 meters. Various fog devices (resources and access points) are randomly and evenly distributed on the map.

This paper aims at improving the performance of resource scheduling in a cloud-fog-edge hybrid computing architecture, including the total execution time, service delay, the fairness, and reliability. Therefore, we chose the min-min and Berger's model as the benchmark comparison models. At the same time, in order to measure the influence of the different factors in the proposed model, we also compared it with the subtraction models, namely, locate-ware and load-balance. The locate-ware is the subtraction model which only remains the location switching strategy of

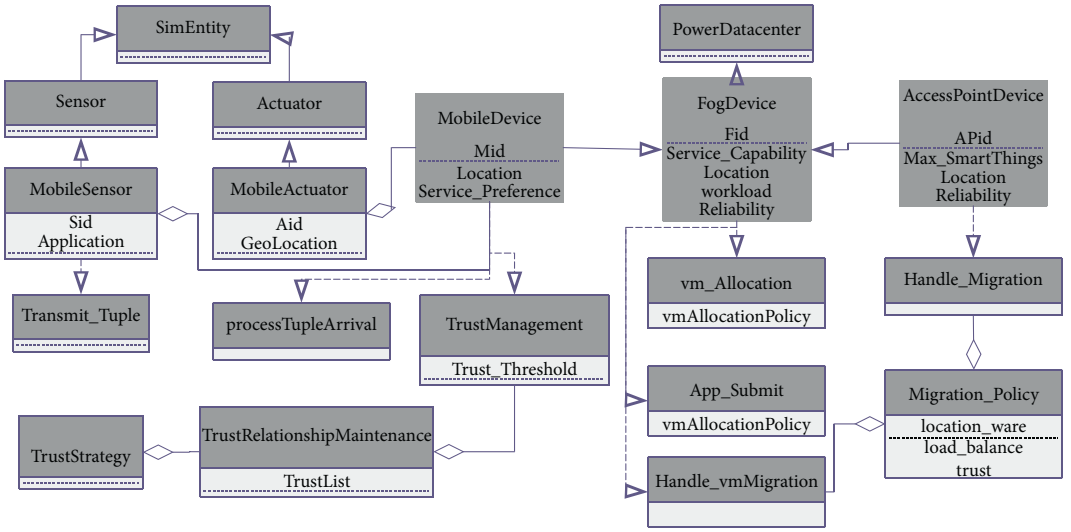


Figure 7: The key classes in scheduling procedure.

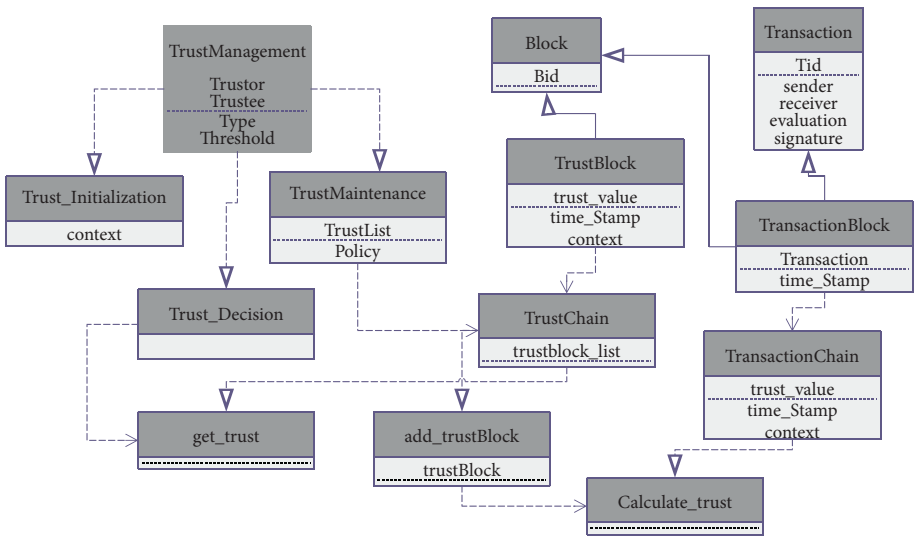


Figure 8: Blockchain-enhanced trust architecture.

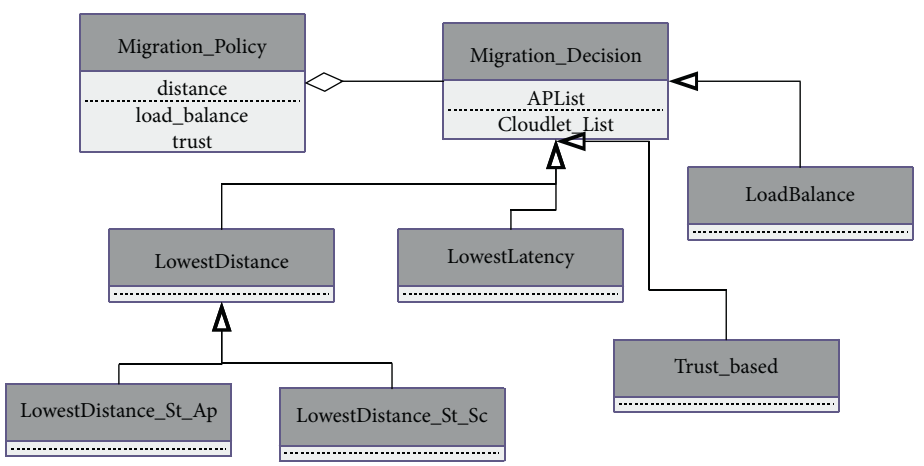


Figure 9: Types of migration rules.

Table 1: The parameters of the intermodule tuples in the EEG tractor game.

Tuple type	CPU length (MIPS)	N/W length
EEG	2000(A)/2500(B)	500
_SENSOR	3500	500
PLAY_GAME_STATE	1000	1000
CONCENTRATION	14	500
GLOBAL_GAME_STATE	1000	1000
GLOBAL_STATE_UPDATE	1000	500
SELF_STATE_UPDATE	1000	500

Table 2: The configuration of the device.

Tuple type	CPU (GHz)	RAM (GB)	Power (W)
EEG	2000(A)/2500(B)	500	107.339(M) 83.433(I)
_SENSOR	3500	500	107.339(M) 83.433(I)
PLAY_GAME_STATE	1000	1000	83.53(M) 82.44(I)
CONCENTRATION	14	500	107.339(M) 83.433(I)

the proposed model in scheduling, and the load-balance is the subtraction model that only maintains the load-sensitive strategy.

The task scheduling algorithms including min-min, Berger's Model, proposed model, location-aware, and load-balance model are compared from various perspectives.

5.3. Simulation Result. The experimental results are shown in Figures 10–17. From the results, we can see that the proposed model has achieved better effect than the other models on almost all the experimental metrics. And two subtraction models also gained better results than the other benchmark models.

5.3.1. Comparison on System Performance. For the test of system performance, we compared the mentioned models in terms of makespan, service latency, and tuple loss rate.

The proposed model and the location-aware model work well in the total completion time and service delay. Because the built-in location-sensitive scheduling strategy takes into account the mobility of nodes to match it with the closer access point, which is able to effectively reduce the transmission overhead, the service delay, and the total execution time. The load-balance model gains a good performance in the index of the tuple loss rate because it considers the load status of resources in the task scheduling stage, which can effectively reduce the waiting time in queue and avoid tuple loss caused by overload.

The overall system performance of the min-min model is not bad. However, since it does not consider the resource preferences, mobility, and load, it may cause tuple loss and redistribution, which ultimately affects the overall execution time. Berger's model does not work well because it focuses only on the fairness of scheduling, which tends to satisfy the needs of users rather than the overall performance of the system. In a cloud-fog-edge hybrid environment with nodes moving frequently, although the global fairness algorithm can find the most suitable resource for nodes, the scheduling result may not be implemented due to the distance, which increases the possibility of tuple loss and redistribution, thus eventually increasing in the total execution time. In particular, the overhead of Berger's model is relatively large. When the total number of tasks increases, its execution time is very long, which may affect its deployment in the actual applications.

5.3.2. Comparison on User Satisfaction. User satisfaction, measured by service justice (SJ) in this paper, is an important index to evaluate the quality of scheduling algorithms. Berger's model performed the best in this respect because scheduling fairness was its core concern. However, using the traditional Berger's model to perform global matching and fairness calculations during task scheduling often leads to high computational overhead, which may cause the paralysis of the resource allocation system when the task scale is large. The experimental results indicate that when the number of tuples is greater than 125, Berger's model cannot give a resource allocation decision within the tolerable time.

The proposed model still performs well in terms of user satisfaction because it adopts a two-level scheduling mode, and through the fairness factors, it takes into consideration the specific requirements of tasks in the process of task scheduling.

5.3.3. Influence of the Decentralized Trust Mechanism. This section tested the performance of the double-blockchain-based trust mechanism proposed in this paper.

Figure 17 shows the result of the transaction success rate. It can be seen that the decentralized trust can maintain a high degree of the transaction success rate despite the increase in the malicious nodes, indicating that it can effectively assist a reliable transaction decisions. In sharp contrast, in the random transaction scenario, the transaction success rate drops sharply as the number of malicious nodes increases.

This paper also used NetLogo [54] to test the efficiency of the proposed trust mechanism. Table 3 shows the parameters of the simulations.

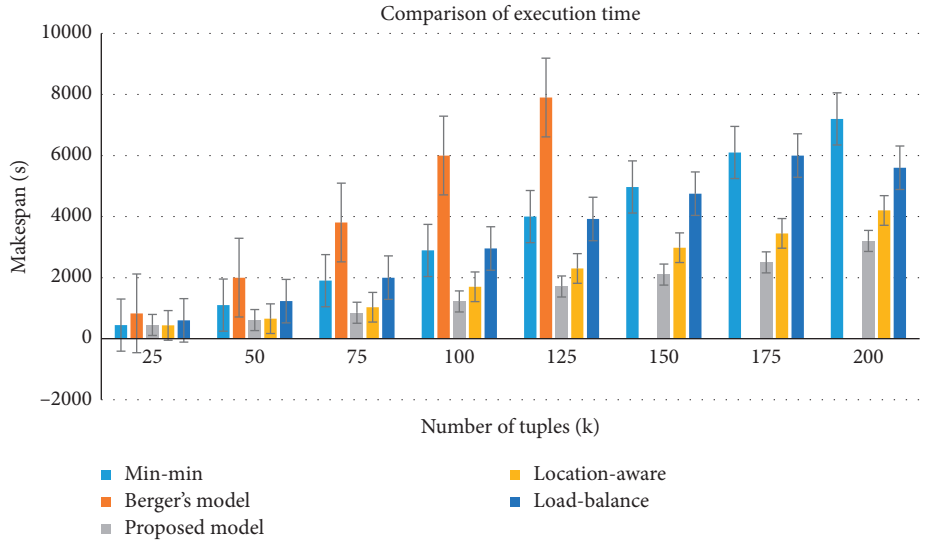


Figure 10: Comparison of makespan.

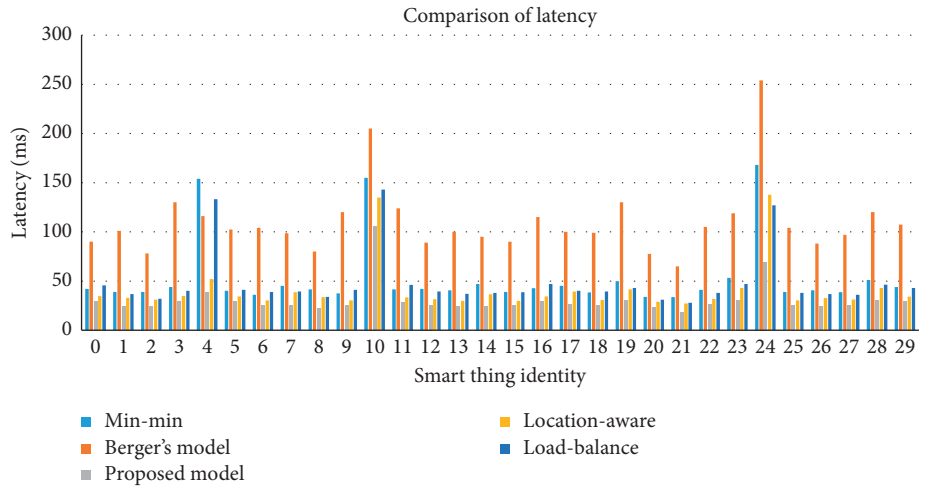


Figure 11: Comparison of service latency.

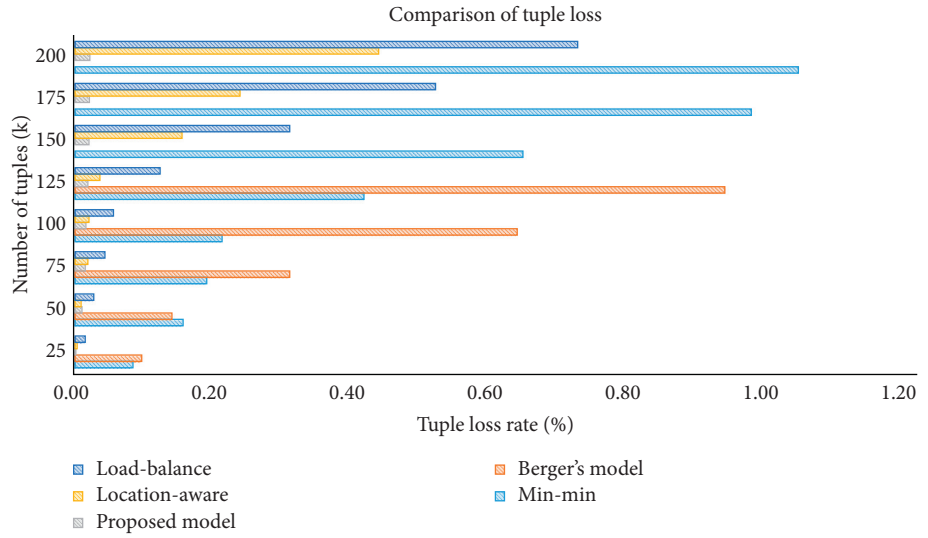


Figure 12: Comparison of tuple loss rate.

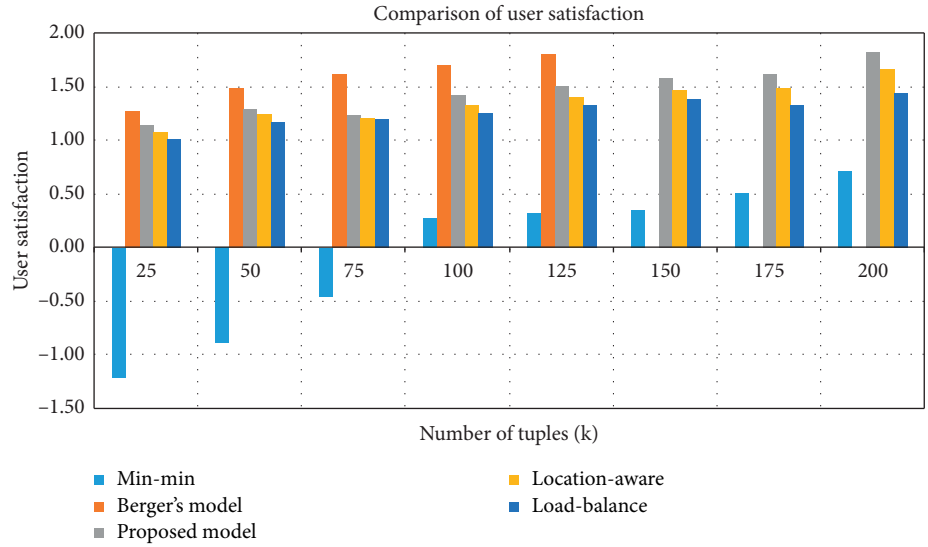


Figure 13: Comparison of user satisfaction.

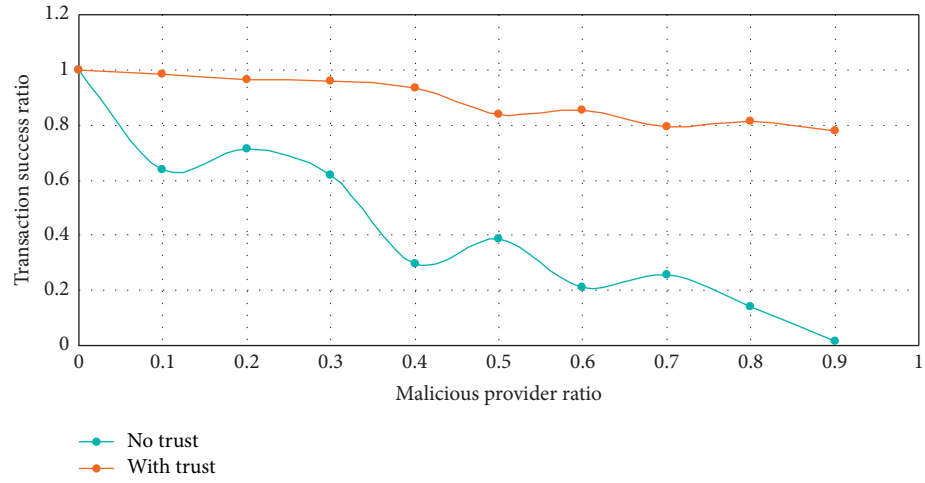


Figure 14: Comparison of transaction success ratio.

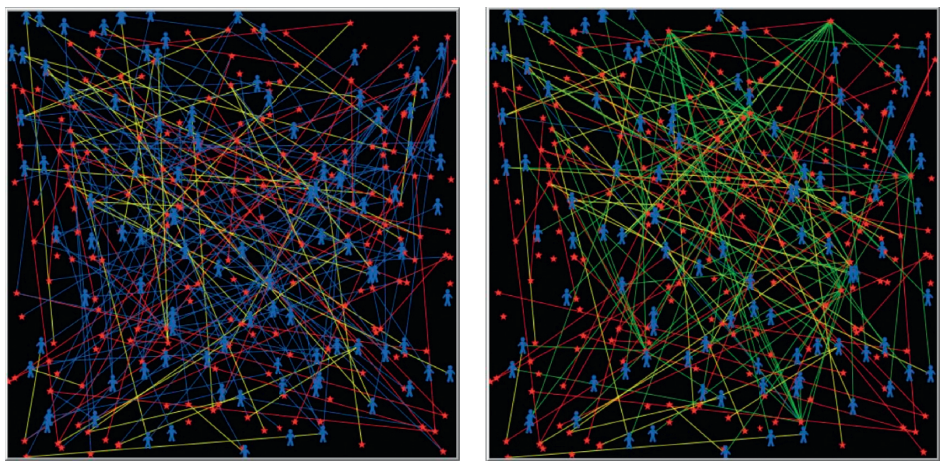
Several experiments were done to observe the impact of trust on the success rate of transactions when the proportion of malicious providers was fixed to 30%. Figure 15 is the experimental result when trust mechanism was loaded, and Figure 16 is the experimental data without any trust mechanisms (in the case of random transactions).

In the simulation chart, users are represented by the blue color person-shaped turtles, and the fog providers (access point or service providers) are the red color star-shaped turtles. The links in the charts represent the relationship between the different entities, including the trading relationship (blue edges), the recommendation relationship (yellow edges), and the cooperation between service providers (red edges).

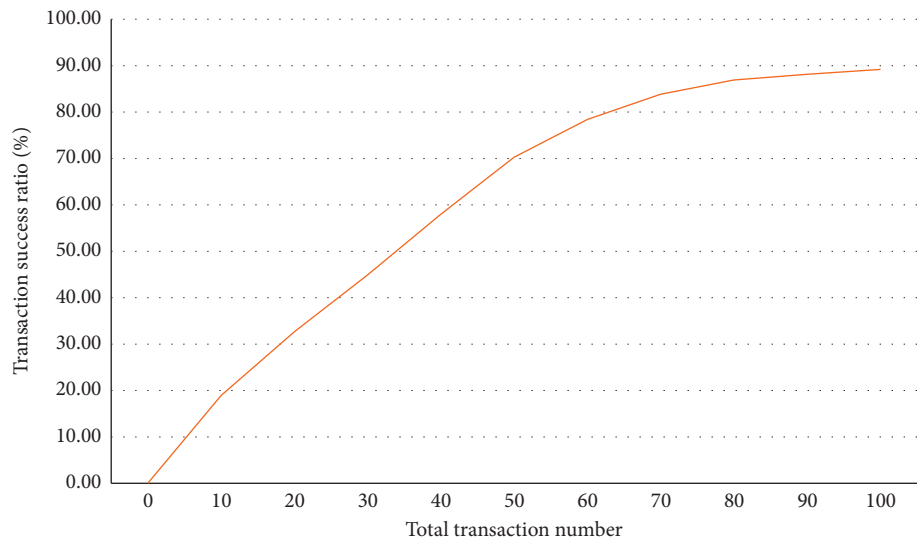
Figures 15(a) and 16(a) are the initial state of the simulation, Figures 15(b) and 16(b) are the final state of the trading cycle, and Figures 15(c) and 16(c) are the records of transaction success ratio output from NetLogo reporter. It can be seen from the figures that, when the trust mechanism is loaded, the transaction success rate gradually increases with the increasing of the number of transactions, which remains almost the same when no trust is loaded.

In order to measure the overhead of trust, we also evaluated the influence of trust on the total execution time. Figure 17 shows the additional time overhead brought by the loading of trust, from which we can see that trust does not have much influence on the total execution time, with even a few nodes showing the shorter execution time. The reason is that after trust has been running for a period of time, when entities in the system are able to correctly select the credible trading partner, the reselection or transaction failure in the random transactions can be effectively avoided, thus, to a certain extent, reducing the total execution time.

In summary, in the cloud-fog-edge hybrid environment, in order to improve the effective matching of terminals, access points, and their expected resources, it is necessary to fully consider the location, the workload, and also the

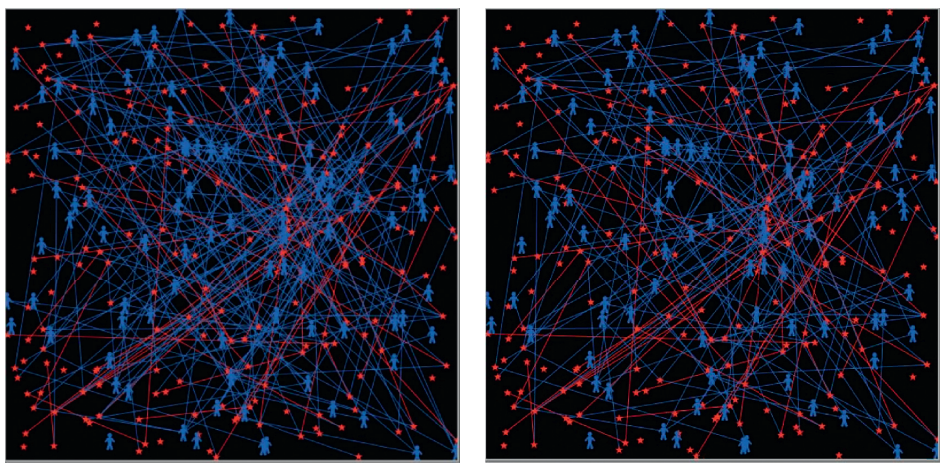


(a) (b)



(c)

Figure 15: (a) The initial state (with trust). (b) The final state (with trust). (c) The transaction success rate (with trust).



(a) (b)

Figure 16: Continued.

Figure 16: (a) The initial state (without trust). (b) The final state (without trust). (c) The transaction success rate (without trust).

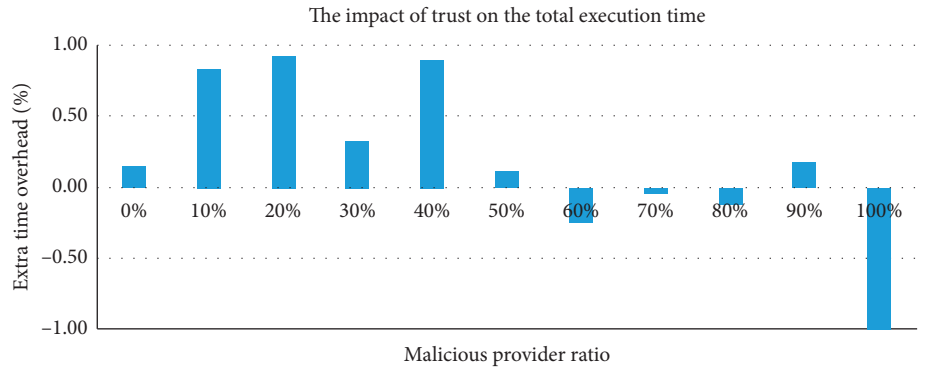


Figure 17: Percentage of increased time overhead by trust.

Table 3: The parameters in the trust performance test.

Number of providers	Number of users	Link probability	Ratio of malicious providers (%)
200	100	0.20	30

credibility of nodes. The proposed decentralized trust strategy is able to guarantee the safety and the reliability of interaction.

6. Conclusion and Future Work

This paper proposes a trust-enhanced location-aware fair task scheduling model for the cloud-fog-edge hybrid environment. The new model contains a three-layer architecture of IoT, fog, and cloud. The fog layer is utilized to achieve the cloud-fog or fog-edge resource coordination and unified scheduling. The proposed task scheduling algorithm comprehensively considers user mobility, system justice, load balance, and trust requirement. Berger's theory is introduced to solve the fairness problem in task scheduling. In resource allocation, it comprehensively considers the location, task's QoS requirements, the capability, and the load of the resources. In addition, to improve the credibility of service interaction, it draws on the idea of blockchain to build a decentralized trust framework. The performance of the new model and the related strategies was evaluated by a series of related experiments.

However, the proposed model still has some imperfections. For example, it adopts a two-level scheduling mode to solve the problem of mobile access. However, the location of the mobile terminals is constantly changing, requiring the timely replacement of the access point. Therefore, it is necessary to determine where and when to switch and also the resource reservation algorithms. Things like the moving direction, speed, and preferences can be used to predict the

next access point and make resource reservations for reducing the handover delay. In addition, the dual-blockchain-based trust management, with all the transactions recorded in the trading behavior blockchain, needs to find the most appropriate time to generate the trust blocks, whether in a fix time period or a certain number of transactions. All these problems are our future work.

Data Availability

Data is available upon request to the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under grant 61702151, 61702320, 61772334, the National Key Research and Development Plan under grant 2018YFB1003800, and the Joint Funds of the Zhejiang Provincial Natural Science Foundation of China under grant LHY21E090004.

References

[1] C. Chang, S. Srirama, R. Buyya, and I. Fog, "An efficient fog-computing infrastructure for the internet of things," *Computer*, vol. 50, no. 9, pp. 92–98, 2017.

[2] C. Huang, X. Mei, G. Zhao, J. Wu et al., "Transaction modelling and execution analysis of uncertainty composition service in mobility computing environments," *Science China: Information Science*, vol. 45, no. 1, pp. 70–96, 2015.

[3] X. Deng, P. Guan, E. Liu et al., "Integrated trust based resource cooperation in edge computing," *Journal of Computer Research and Development*, vol. 55, no. 3, pp. 449–477, 2018.

[4] J. Zhang, Y. Zhao, B. Chen et al., "Survey on data security and privacy-preserving for the research of edge computing," *Journal on Communications*, vol. 39, no. 3, pp. 1–21, 2018.

[5] M. Mukherjee, R. Matam, L. Shu et al., "Security and privacy in fog computing:challenges," *IEEE Access*, vol. 5, pp. 19293–19304, 2017.

[6] L. Bittencourt, J. Diaz-Montes, R. Buyya et al., "Mobility-aware application scheduling in fog computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 26–35, 2017.

[7] M. Lopes, W. Higashino, M. Capretz et al., "MyiFogSim: a simulator for virtual machine migration in fog computing," in *Proceedings of UCC Companion 17, the 10th International Conference. ACM*, pp. 47–52, Zurich, Switzerland, December 2017.

[8] Y. Yin, F. Yu, Y. Xu, L. Yu, and J. Mu, "Network location-aware service recommendation with random walk in cyber-physical systems," *Sensors*, vol. 17, no. 9, p. 2059, 2017.

[9] Y. Yin, J. Xia, Y. Li, Y. Xu, W. Xu, and L. Yu, "Group-wise itinerary planning in temporary mobile social network," *IEEE Access*, vol. 7, pp. 83682–83693, 2019.

[10] B. Xu, C. Zhao, E. Hu et al., "Job scheduling algorithm based on Berger model in cloud environment," *Advances in Engineering Software*, vol. 42, no. 7, pp. 419–425, 2011.

[11] C. Zhao, "Research and realization of job scheduling algorithm in cloud environment," Master's thesis, Beijing Jiaotong University, Beijing, China, 2009.

[12] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo, and M. Zhou, "Maximal energy efficient task scheduling for homogeneous fog networks," in *Proceedings of the IEEE INFOCOM 2018—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 274–279, Honolulu, HI, USA, April 2018.

[13] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo, and M. Zhou, "MEETS: maximal energy efficient task scheduling in homogeneous fog networks," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4076–4087, 2018.

[14] Y. Yang, S. Zhao, W. Zhang, Y. Chen, X. Luo, and J. Wang, "DEBTS: delay energy balanced task scheduling in homogeneous fog networks," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2094–2106, 2018.

[15] H. Sun, H. Yu, and G. Fan, "Contract-based resource sharing for time effective task scheduling in fog-cloud environment," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, 2020.

[16] N. Auluck, O. Rana, S. Nepal, A. Jones, and A. Singh, "Scheduling real time security aware tasks in fog networks," *IEEE Transactions on Services Computing*, 2019.

[17] H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, and X. Wang, "Context-aware QoS prediction with neural collaborative filtering for internet-of-things services," *IEEE Internet of Things Journal*, vol. 7, 2019.

[18] H. Gao, Y. Duan, L. Shao, and X. Sun, "Transformation-based processing of typed resources for multimedia sources in the IoT environment," *Wireless Networks*, 2019.

[19] J. Yu, J. Li, Z. Yu, and Q. Huang, "Multimodal transformer with multi-view visual representation for image captioning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 8, 2019.

[20] J. Yu, M. Tan, H. Zhang, D. Tao, and Y. Rui, "Hierarchical deep click feature prediction for fine-grained image recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[21] H. Gao, C. Liu, Y. Li, and X. Yang, "V2VR: reliable hybrid-network-oriented V2V data transmission and routing considering RSUs and connectivity probability," *IEEE Transactions on Intelligent Transportation Systems(T-ITS)*, 2020.

[22] H. Gao, L. Kuang, Y. Yin, B. Guo, and K. Dou, "Mining consuming behaviors with temporal evolution for personalized recommendation in mobile marketing apps," *ACM/Springer Mobile Networks and Applications (MONET)*, vol. 25, no. 4, pp. 1233–1248, 2020.

[23] Y. Yin, Z. Cao, Y. Xu et al., "QoS prediction for service recommendation with features learning in mobile edge computing environment," *IEEE Transactions on Cognitive Communications and Networking*, 2020.

[24] X. Yang, S. Zhou, and C. Min, "An approach to alleviate the sparsity problem of hybrid collaborative filtering based recommendations: the product-attribute perspective from user reviews," *Mobile Networks & Applications*, vol. 25, no. 2, pp. 376–390, 2020.

[25] Z. Liu, X. Yang, Y. Yang, K. Wang, and G. Mao, "DATS: dispersive stable task scheduling in heterogeneous fog networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3423–3436, 2019.

[26] M. Mukherjee, M. Guo, J. Lloret, R. Iqbal, and Q. Zhang, "Deadline-aware fair scheduling for offloaded tasks in fog computing with inter-fog dependency," *IEEE Communications Letters*, vol. 24, no. 2, pp. 307–311, 2020.

[27] G. Zhang, F. Shen, Y. Zhang, R. Yang, Y. Yang, and E. A. Jorswieck, "Delay minimized task scheduling in fog-enabled IoT networks," in *Proceedings of the 2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1–6, Hangzhou, China, October 2018.

[28] G. Zhang, F. Shen, N. Chen, P. Zhu, X. Dai, and Y. Yang, "DOTS: delay-optimal task scheduling among voluntary nodes in fog networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3533–3544, 2019.

[29] H. K. Apat, "An optimal task scheduling towards minimized cost and response time in fog computing infrastructure," in *Proceedings of the 2019 International Conference on Information Technology (ICIT)*, pp. 160–165, Bhubaneswar, India, 2019.

[30] C. Tang, S. Xiao, X. Wei, M. Hao, and W. Chen, "Energy efficient and deadline satisfied task scheduling in mobile cloud computing," in *Proceedings of the 2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 198–205, Shanghai, China, 2018.

[31] J. Xu, Z. Hao, R. Zhang, and X. Sun, "A method based on the combination of laxity and ant colony system for cloud-fog task scheduling," *IEEE Access*, vol. 7, pp. 116218–116226, 2019.

[32] M. Yang, H. Ma, S. Wei, Y. Zeng, Y. Chen, and Y. Hu, "A multi-objective task scheduling method for fog computing in cyber-physical-social services," *IEEE Access*, vol. 8, 2020.

[33] X. Wei, J. Liu, Y. Wang, C. Tang, and Y. Hu, "Wireless edge caching based on content similarity in dynamic environments," *Journal of Systems Architecture*, vol. 115, pp. 1–8, 2021.

[34] J. Zhang, X. Hu, Z. Ning et al., "Joint resource allocation for latency-sensitive services over mobile edge computing networks with caching," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4283–4294, 2019.

[35] W. Wen, Y. Cui, T. Q. S. Quek, F.-C. Zheng, and S. Jin, "Joint optimal software caching, computation offloading and communications resource allocation for mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7879–7894, 2019.

[36] J. Jiang, G. Han, L. Shu, S. Chan, and K. Wang, "A trust model based on cloud theory in underwater acoustic sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 1, pp. 342–350, 2017.

[37] P. Zhang, Y. Kong, and M. Zhou, "A domain partition-based trust model for unreliable clouds," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 9, pp. 2167–2178, 2018.

[38] T. Wang, G. Zhang, S. Cai et al., "Survey on trust evaluation mechanism in sensor-cloud," *Journal on Communications*, vol. 39, no. 6, pp. 37–51, 2018.

[39] J. Wang, Z. Yu, H. Zhang et al., "Service recommended trust algorithm based on cloud model attributes weighted clustering," *Journal of System Simulation*, vol. 30, no. 11, pp. 275–289, 2018.

[40] P. Zhang, Y. Kong, and M. Zhou, "A novel trust model for unreliable public clouds based on domain partition," in *Proceedings of the of IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)*, pp. 275–280, IEEE, Calabria, Italy, May 2017.

[41] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, and X. Wang, "Learning-aided computation offloading for trusted collaborative mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 8, 2019.

[42] H. Yang, J. Cho, H. Son, and D. Lee, "Context-aware trust estimation for realtime crowdsensing services in vehicular edge networks," in *Proceedings of the 17th Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–6, IEEE, Las Vegas, NV, USA, January 2020.

[43] J. Shu, C. Liang, and J. Xu, "Trust-based multi-objectives task assignment model in cloud service system," *Journal of Computer Research and Development*, vol. 55, no. 6, pp. 1167–1179, 2018.

[44] C. Hu, X. Tong, and W. Liang, "The real-value restricted Boltzmann machine recommendation algorithm based on trust-distrust relationship," *Systems Engineering-Theory & Practice*, vol. 39, no. 7, pp. 1817–1830, 2019.

[45] X. Meng, J. Ma, D. Lu, and Y. Wang, "Trust and behavioral modeling based two layer service selection," *Journal of Xidian University (Natural Science)*, vol. 41, no. 4, pp. 198–204, 2014.

[46] Z. Ma, X. Wang, D. Jain, H. Khan, H. Gao, and Z. Wang, "A blockchain-based trusted data management scheme in edge computing," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2013–2021, 2020.

[47] L. Cui, S. Yang, Z. Chen, Y. Pan, Z. Ming, and M. Xu, "A decentralized and trusted edge computing platform for Internet of Things," *IEEE Internet of Things Journal*, vol. 11, 2019.

[48] W. Li, L. Ping, and X. Pan, "Trust model to enhance security and interoperability of cloud environment," in *Proceedings of CloudCom'09*, pp. 69–79, Springer, Bangalore, India, September 2009.

[49] W. Li, L. Ping, Q. Qiu, and Q. Zhang, "Research on trust management strategies in cloud computing environment," *Journal of Computational Information Systems*, vol. 8, no. 4, pp. 1757–1763, 2012.

[50] W. Li, J. Cao, S. Qian, and R. Buyya, "TSLAM: a trust-enabled self-learning agent model for service matching in the cloud market," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 13, no. 4, pp. 1–41, 2019.

[51] W. Li, J. Cao, K. Hu, J. Xu, and R. Buyya, "A trust-based agent learning model for service composition in mobile cloud computing environments," *IEEE Access*, vol. 7, pp. 34207–34226, 2019.

[52] H. Gupta, A. Dastjerdi, S. Ghosh, and R. Buyya, "iFogSim: a toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," *Software Practice & Experience*, vol. 47, pp. 1275–1296, 2017.

[53] J. K. Zao, T. T. Gan, C. K. You et al., "Augmented brain computer interaction based on fog computing and linked data," in *Proceedings of the International Conference on Intelligent Environments (IE)*, pp. 374–377, IEEE, Shanghai, China, June-July 2014.

[54] NetLogo, "NetLogo user manual version 6.0.3," 2020, https://ccl.northwestern.edu/netlogo/docs/.