Resource Allocation Scheme for Community-Based Fog Computing Based on Reputation Mechanism

Ke Gu, *Member, IEEE*, Linyu Tang, Jiafu Jiang, and WeiJia Jia, *Fellow, IEEE*

***Abstract*— Fog computing needs to seamlessly integrate heterogeneous computing resources widely distributed in edge networks, and then, it can provide unified resources and services for users. However, due to a large number of fog nodes, there are still many security problems in the fog computing process. For example, when fog servers make resource allocation between users' tasks and fog nodes, some fog nodes can falsely claim that they have more resources than their actual ability, so as to get more task processing qualifications. Thus, it will greatly damage the interests of users and affect the quality of service. In this article, we propose a resource allocation scheme for community-based fog computing based on a reputation mechanism. When fog network provides computing services for users, we use a reputation mechanism to enable users to obtain reliable resources in fog computing. In our proposed scheme, a user first submits his/her task request to the community-based fog network, and then, the fog server makes a reliable resource allocation process based on multiple-layer communities and reputation calculation. Based on our experiments, our scheme enables users to obtain reliable resources and improves the service quality of computing resources in fog computing.**

***Index Terms*— Community, fog computing, reputation, resource allocation, security.**

I. Introduction

A. Background

CLOUD computing is a computing architecture, which integrates a series of heterogeneous resources and provides powerful computing power for users. Currently, it has been developed and applied more and more widely in recent years [1]. However, with the explosive growth of diversified terminal devices and user needs, it still needs to utilize data centers to process diverse requests from the edge network. While realizing the functions of high-speed computing and storage, cloud computing also brings about large operating costs, network delays, and other issues [2]. Thus, the Cisco company [3] proposed a new computing concept called fog computing. It moves computing, storage, and other functions of cloud computing from the center to the edge of the network, where all functions are closer to terminal users (as shown in Fig. 1). Nowadays, fog computing has gradually become a new concept that can provide resources for data storage, computing, and application services. Compared with traditional cloud computing, fog computing needs a wider distributed deployment, where the storage and processing of data mainly rely on devices in the edge network. Furthermore, it makes full use of resources closer to users to provide more effective services and obtain less service delay [4]. Fog computing needs to seamlessly integrate heterogeneous computing resources widely distributed in edge networks, and then, it provides unified resources and services for users. In fog computing, a large number of edge devices, such as personal computers, mobile phones, intelligent gateways, and base stations, can use part or all of their resources to complete certain computing, storage, and other tasks. Thus, it can achieve fast services that are closer to terminal users.

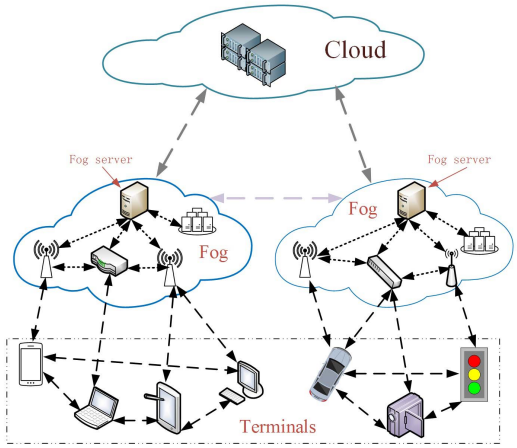


Fig. 1. Hierarchical structure of fog computing.

Currently, many researchers have proposed various application architectures of fog computing [5]–[7] in recent years, which can reduce service delay and network congestion. In fog computing, heterogeneous resource distribution based on multilayer location hierarchy is a common pattern of resource distribution. In this architecture, distributed fog nodes usually in many areas are aggregated layer by layer through different levels of network connecting devices, and certain scale and

Manuscript received January 23, 2020; revised May 31, 2020; accepted June 23, 2020. Date of publication July 10, 2020; date of current version November 10, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61402055 and Grant 61902040, in part by the National Natural Science Key Foundation of China under Grant 61532013 and Grant 61872239, in part by the Science and Technology Development Fund of SAR Macau under Grant 0007/2018/A1 and Grant 0060/2019/A1, in part by the University of Macau under Grant MYRG2018-00237-FST, and in part by the Natural Science Foundation of Hunan Province under Grant 2019JJ40314, Grant 2018JJ2445, and Grant 2016JJ3012. *(Corresponding author: Ke Gu.)*

Ke Gu, Linyu Tang, and Jiafu Jiang are with the School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410114, China (e-mail: gk4572@163.com).

WeiJia Jia is with the Department of Computer and Information Science, University of Macau, Macau 999078, China.

Digital Object Identifier 10.1109/TCSS.2020.3005761

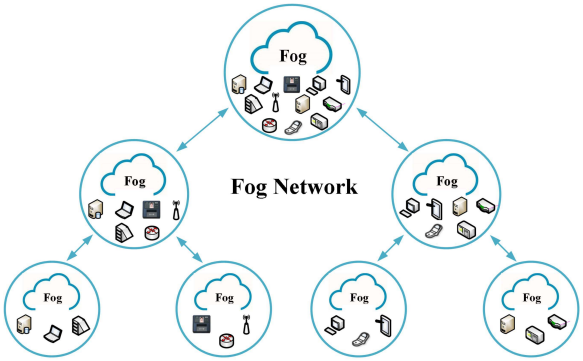


Fig. 2. Heterogeneous resource distribution based on multilayer location hierarchy in fog computing.

size of resource pool are formed, as shown in Fig. 2. These fog nodes can not only serve users as resource providers but also use the resources of other fog nodes as resource consumers [8]. Therefore, in order to quickly respond to users' requirements, the problem of resource allocation among many distributed fog nodes has actually become the problem of selecting more suitable fog nodes in a multilayer resource pool. Different from cloud computing, fog computing in the resource allocation problem further highlights the concepts of user and fog device location. The access method and location of users and fog nodes may influence the effectiveness and efficiency of resource allocation. At the same time, the resource allocation procedure involves not only whether users can efficiently get the resources they need but also how to make more effective use of all kinds of resources in fog network. Many traditional resource allocation schemes are usually considered only from one side. Therefore, the effective resource allocation scheme in fog computing usually needs to better balance the factors of users and fog resources, which provides effective services for users.

Furthermore, due to a large number of fog nodes, there are still many security problems in fog computing. For example, when fog servers make resource allocation between users' tasks and fog nodes, they can only identify the resources of fog nodes through parameter information submitted by fog nodes. However, there may be a lot of deception in this process. If payment is considered in fog computing, then a fog node may falsely claim that it has more resources than its actual ability, so as to get more task processing qualifications. In fact, its resources cannot meet the needs of users' tasks; thus, it will greatly damage the interests of users and affect the quality of service. Therefore, it is very important to make better use of the geographically distributed and heterogeneous resources in the edge network. Currently, reputation is an entity's opinion or evaluation to another entity through interaction. It establishes a trust relationship between entities through certain organizational and managerial methods and serves as a reference for possible interaction in the future. The establishment of the reputation model is based on the historical interaction information between entities, and it can reflect the behavior information of entities. To improve the overall availability and performance of the network, many scholars have introduced the reputation model into resource management. Many reputation models build trust relationships among entities through mutual evaluation or evaluation of a third party. For example, in some reputation models of resource management, the time and budget of a node on the corresponding task may be evaluated by reporting its CPU time, storage, or memory resources [9]. Therefore, the reputation mechanism is feasible for the security and reliability of resource allocation in fog computing. In this article, we mainly focus on how to make full use of the diverse and layered computing resources in fog computing, so as to provide users with reliable services based on improved reputation mechanism.

B. Our Contributions

In this article, we propose a resource allocation scheme for community-based fog computing based on a reputation mechanism. When fog network provides computing services to users, we use a reputation mechanism to enable users to obtain reliable resources in fog computing. In our proposed scheme, a user first submits his/her task request to the community-based fog network, and then, the fog server makes a reliable resource allocation process based on multiple-layer communities and reputation calculation. Thus, users can obtain their required resources (including reputation requirements) and can get reliable services to protect their own interests. In this article, our contributions are as follows.

1) We construct the reputation mechanism to calculate the reputation value of each fog node according to the historical behaviors of fog nodes. In our reputation mechanism, the fog server in each fog group needs to maintain the reputation values among communities and calculate the full reputation values of fog nodes according to the internal reputation, direct reputation, and indirect reputation of fog nodes.
2) We propose a resource allocation scheme for community-based fog computing based on reputation mechanism. In the proposed scheme, a community that meets the requirement of a submitted task is chosen, where the related fog server makes a reliable resource allocation process based on multiple-layer communities and reputation calculation. Furthermore, an improved stable matching algorithm is proposed to finish the assignment between subtasks and fog nodes in a community.
3) We make some experiments to test and analyze the performance of our proposed scheme. Through changing different proportions of nodes and different amount of tasks, we mainly test the reputation change of different nodes, the success rate of resource allocation, the average completion delay of tasks, and the average distance between users and service providers. Based on our experiments, our scheme enables users to obtain reliable resources and improves the service quality of fog computing resources.

C. Organization

The rest of this article is organized as follows. In Section II, we discuss the related works about resource allocation of

fog computing. In Section III, we show a proposed framework for fog networks based on a multilayer community structure. In Section IV, we propose a resource allocation scheme for community-based fog computing based on a reputation mechanism. In Section V, we make some experiments to test the efficiency and effectiveness of the proposed scheme. Finally, we draw our conclusion in Section VI.

II. Related Work

Currently, many scholars focus on fog computing and its applications. After the Cisco company proposed the notion of fog computing, Bonomi *et al.* [10] proposed the definition of fog computing in 2012. Yi *et al.* [11] summarized some applications of fog computing and discussed many problems in designing and implementing fog computing systems. Stojmenovic *et al.* [12] discussed that fog computing is applied to a smart grid and software-defined network (SDN) and pointed out its security and privacy problems. Aazam *et al.* [5] proposed an approach to preprocess the data that need to be sent to the cloud server by the fog-layer intelligent gateway. Their scheme may reduce the burden of the cloud server to provide more efficient services. Huang *et al.* [6] proposed a frame to combine heterogeneous wireless cloud access networks and fog network through appropriate collaboration. Datta *et al.* [13] proposed to deploy a fog platform by using roadside units and machine-to-machine gateways for vehicle networking. Their scheme can effectively establish vehicle network management and provide consumer-centric services. In 2018, the IEEE Standards Association formally adopted OpenFog architecture [7] as a formal standard, which is used as a general technical framework for fog computing. The formal standard is used to support data-intensive requirements for the Internet of Things (IoT), 5G, and AI applications in the future.

Fog computing is close to terminal users; there are some differences in resource management and utilization between fog computing and cloud computing [7]. In recent years, many different resource management and utilization models are proposed for different architectures and application scenarios. Nishio *et al.* [14] proposed a service-oriented heterogeneous resource sharing scheme, which can effectively reduce service waiting time. Euclides *et al.* [15] considered the specific multitenant requirements (latency and priority) and then used the multitenant load distribution algorithm to optimize load balancing in a fog environment. Kochar *et al.* [8] proposed a two-layer fog computing framework for resource sharing by taking into account the benefits of fog service providers. Their scheme can get higher system utilization rates. Zhang *et al.* [16] proposed a three-layer resource management framework for multiple data service operators, fog nodes, and multiple data service subscribers. Their framework is based on the Stackelberg game, moral hazard model, and matching game strategy. Girvan *et al.* [17] pointed out that some nodes in the networks are often connected by "groups" that are closely related, and the connections between these "groups" are relatively loose. Then, the "groups" are also called "communities." Filiposka *et al.* [18] proposed a community-based fog computing resource management and migration scheme, which can provide resource management and sharing with low latency and location-aware.

Although the resource management and sharing schemes in fog computing are rapidly developed, few researchers focus on the security of these schemes. Many existing schemes are based on the premise that fog nodes (or resources) are secure and reliable. Since fog computing is different from cloud computing, the security of fog computing is facing new challenges. Due to the variety and wide distribution of fog nodes, the resource allocation process in fog computing is vulnerable to the destruction and deception of malicious nodes, especially for the charge mode of "pay-as-you-go." Thus, a reputation mechanism is a feasible approach to effectively solve this problem. Josang *et al.* [19] summarized some trust and reputation systems that can be used to derive measures of trust and reputation for the Internet transactions. They further analyzed the current trends and developments in this area and then proposed a research agenda for trust and reputation systems. Wahab *et al.* [20] classified and compared the main findings related to trust and reputation in the context of web services. Furthermore, they discussed the challenging problem of having active malicious web services in the community-based architectures. Nagarathna *et al.* [21] proposed a service recommendation system for the grid based on trust, reputation, and quality of service. In their scheme, service providers may calculate the reliability of service by obtaining feedback information directly from service users and suggestions from other service providers. Tian *et al.* [22] proposed a reputation-based resource allocation model (RBAM) that introduces the reputation mechanism into a reverse auction. Their model establishes the trust and reliable relationship for resource trading participants to realize the rational allocation of resources. Wei *et al.* [23] introduced the concept of community into an opportunistic network and proposed an incentive scheme for opportunistic networks based on community and reputation. Their scheme can promote cooperation among selfish nodes in an opportunistic network. Shi *et al.* [24] proposed a standardized model of heterogeneous multirobot cooperation based on reputation, which greatly improves the success rate of task execution and reduces the time of task recovery and redistribution. Wang *et al.* [25] proposed an accurate and multilevel reputation evaluation scheme for cloud computing. Their scheme introduces two malicious rating detection methods to improve the accuracy of reputation calculation, which can help users evaluate and select cloud services from different perspectives. Lin *et al.* [26] proposed a cross-layer reputation mechanism (RP-CRM) based on reliable recommendation and privacy protection. Their scheme combines the reputation evaluation mechanism and the privacy protection method to identify and manage malicious internal nodes. Although the abovementioned schemes have been proposed, these schemes cannot still sufficiently meet the needs of resource management and allocation for fog computing with a wide range of network nodes. In this article, we mainly focus on how to make full use of the diverse and layered computing resources in fog computing, so as to provide users with reliable services based on improved reputation mechanism.

III. Proposed Framework

In fog computing, there are many characteristics, such as dense geographical distribution, mobility support, network, and data heterogeneity. Thus, due to the weak supervision and wide distribution of fog nodes, there is serious information asymmetry in the resource allocation process. For example, some users cannot directly know the resource number, operation status, reliability, and actual capability of fog nodes. Therefore, some unreliable fog nodes can easily use the asymmetry of information to provide false resources and low-quality services. In order to enable users to obtain reliable resources, we propose a resource allocation scheme for community-based fog computing. Based on the notion of community [17], the connection of fog nodes in the fog network may be defined as the multilayer community structure.

Definition 1 (Multilayer Community Tree Structure): Fog network is divided into a multilayer community tree structure according to the different granularity and connection of fog nodes. In the structure, each basic community is the smallest service unit consisting of multiple fog nodes directly connected to the same access point (AP). Each basic community is equivalent to a "fog group.[1]" For users, each community is equivalent to a resource pool consisting of different numbers and types of fog nodes, where some resources may be selected to respond to users' needs. According to the topological structure of the whole network, these bottom communities are further combined layer by layer to form larger community structures through network connecting devices (such as switches). Finally, a multilayer community tree structure consisting of multiple communities is formed. In the multilayer tree structure of community-based fog computing, the entities are classified as follows.

1) *Access Point (or Switch):* An access point used as the entrance and exit of one community can capture the information of fog nodes in a bottom community, where each bottom community consists of an access point, some fog nodes directly connected to the access point, and a fog server.
2) *Bottom Fog Server:* A bottom fog server communicates directly with an access point. It is responsible for maintaining related information and finishing reputation calculation and resource allocation. The bottom fog server maintains the parameter information of all fog nodes in the local community and makes reputation calculation and resource allocation according to their maintained parameters.
3) *Intermediary Fog Server:* One intermediary fog server in the multilayer community tree maintains the parameter information of the corresponding managed communities. The intermediary fog server is used as the "management node" of different levels of communities, which also can make the resource allocation process based on multiple-layer communities and reputation calculation.

[1]In this article, we call the basic community as the bottom community.

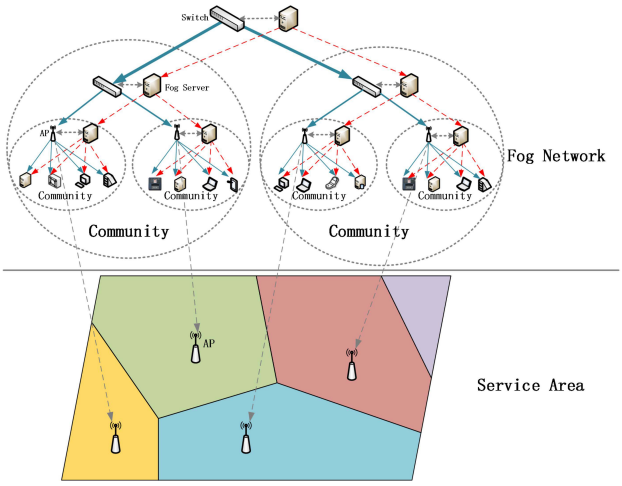


Fig. 3. Structure of the proposed framework.

4) *Fog Node:* A fog node must submit its own parameters to the corresponding fog server through AP when it joins one fog network.

In the multilayer community tree structure, the physical distribution can cover a larger geographical area when the structure is extended to the whole area of the fog network accordingly. The proposed framework is shown in Fig. 3. A multilayer tree structure of community-based fog computing can be constructed on a campus network, where some terminal devices connected to this fog network can provide their open computing resources for some legal campus users. In the community-based campus network, each fog server is a local network administrator to make reputation calculation and resource allocation according to the requirements of campus users. In addition, in this article, we assume that all the fog servers have certain security protection measures for damage and failure, where we further assume that the data storage and serving behaviors of the high-level fog servers (except for the bottom fog servers) must be secure and reliable.

IV. Proposed Resource Allocation Scheme

A. Reputation Mechanism for Community-Based Fog Computing

In the section, we construct the reputation mechanism to calculate the reputation value of each fog node according to the historical behaviors of fog nodes. According to the concept of community, each bottom community consists of an access point, some fog nodes, and a fog server. As a "management node", the bottom fog server is responsible for managing the internal reputation of the smallest community and maintaining the mutual evaluation among communities. The related symbols are shown in Table I.

1) Internal Reputation of Fog Nodes: In each bottom community, the corresponding fog server monitors the related fog nodes. When the fog nodes access the bottom community, they need to report their device information to the fog server, such as computing capability, memory size, network bandwidth, and other parameters, and then, the fog server checks and stores the related information. On the other hand, the fog server is

TABLE I
RELATED SYMBOLS

Symbols	Description
$LR^{(l)}(x)$	the internal reputation value of x after completing the l-th task
$U^{(l)}(x)$	the evaluation value of completing the l-th task on x
$LR(X)$	the internal reputation set of all fog nodes in the community
$URC^{(k)}(u, C_i)$	the direct reputation value of the community C_i generated by the user u
$E^{(k)}(u, C_i)$	the evaluation value of the community C_i generated by the user u on the k-th task
$URC(u)$	the direct reputation set of corresponding communities generated by the user u
$CRC^{(h)}(C_j, C_i)$	the full h-th evaluation value of C_j to C_i
$EC^{(h)}(C_j, C_i)$	the evaluation value of C_j to C_i on the completion of the h-th task generated by the corresponding user in the community C_j
$CRC(C_i)$	the set of evaluation values of C_i to each community
$UIRC(C_i)$	the indirect reputation of the community C_i
$RC(u, C_i)$	the comprehensive reputation value of the community C_i for u
$R(u, x)$	the reputation value of the fog node x for u

also responsible for evaluating the fog nodes and maintaining the internal reputation of each node in the bottom community.

Assuming that x is expressed as a fog node, its internal reputation in the corresponding community is defined as follows:

$$\mathrm{LR}^{(l)}(x) = \begin{cases} \mathrm{LR}_{\mathrm{init}} & l = 0 \\ (1-\alpha)\cdot \mathrm{LR}^{(l-1)}(x) + \alpha \cdot U^{(l)}(x) & l \neq 0. \end{cases} \quad (1)$$

In the abovementioned formula, $\mathrm{LR}_{\mathrm{init}}$ is the initial value of internal reputation of the new fog node; $\mathrm{LR}^{(l)}(x)$ is the internal reputation value of the fog node x in the community after the node completes the lth task; α is the weight of the last task evaluation ($0 \leq \alpha \leq 1$), which is used to balance the importance between the last task evaluation and the historical evaluation[2]; and $U^{(l)}(x)$ is the evaluation of completing the lth task on the node x. $U^{(l)}(x)$ is further defined as follows:

$$U^{(l)}(x) = 1 - \left[\beta \cdot \frac{tc^{(l)}_{\mathrm{actu}} - tc^{(l)}_{\mathrm{expect}}}{tc^{(l)}_{\mathrm{expect}}} + (1-\beta) \cdot \frac{tw^{(l)}_{\mathrm{actu}} - tw^{(l)}_{\mathrm{expect}} + 1}{tw^{(l)}_{\mathrm{expect}} + 1} \right]. \quad (2)$$

In the abovementioned formula, $tc^{(l)}_{\mathrm{actu}}$ represents the actual execution time of the lth task, $tc^{(l)}_{\mathrm{actu}} = t^{(l)}_{\mathrm{back}} - \mathbf{MAX}\{t^{(l)}_{\mathrm{send}}, t^{(l-1)}_{\mathrm{back}}\}$, where $t^{(l)}_{\mathrm{back}}$ represents the time that the node returns the calculation result of the lth task and $t^{(l)}_{\mathrm{send}}$ represents the time that the lth task is sent to the node; $tc^{(l)}_{\mathrm{expect}}$ represents the estimated time that the fog node performs the lth task, $tc^{(l)}_{\mathrm{expect}} = (L^{(l)}/\mathrm{Cap}C)$, where $L^{(l)}$ represents the number of instructions in the lth task and Cap$C$ represents the computing capacity of the node[3]; $tw^{(l)}_{\mathrm{actu}}$ represents the actual waiting time of the lth task before the lth task is executed on the node, $tw^{(l)}_{\mathrm{actu}} = \mathbf{MAX}\{t^{(l-1)}_{\mathrm{back}} - t^{(l)}_{\mathrm{send}}, 0\}$; $tw^{(l)}_{\mathrm{expect}}$ represents the estimated waiting time of the lth task, $tw^{(l)}_{\mathrm{expect}} = \mathbf{MAX}\{t^{(l-1)}_{\mathrm{start}} + (L^{(l)}/\mathrm{Cap}C) - t^{(l)}_{\mathrm{send}}, 0\}$, where $t^{(l-1)}_{\mathrm{start}}$ is the starting time of the $l-1$th task, $t^{(l-1)}_{\mathrm{start}} = \mathbf{MAX}\{t^{(l-1)}_{\mathrm{send}}, t^{(l-2)}_{\mathrm{back}}\}$[4]; and β is the evaluation weight of the execution time of this task, which is used to balance the importance between the execution time of the current task and the evaluation time of the previous tasks.[5] In addition, if $tc^{(l)}_{\mathrm{actu}} - tc^{(l)}_{\mathrm{expect}} < 0$, then we set $tc^{(l)}_{\mathrm{actu}} = tc^{(l)}_{\mathrm{expect}}$, and if $tw^{(l)}_{\mathrm{actu}} - tw^{(l)}_{\mathrm{expect}} \leq 0$, then we set $tw^{(l)}_{\mathrm{expect}} = tw^{(l)}_{\mathrm{actu}} + 1$; thus, the formula may obtain $U^{(l)}(x) = 1$. Obviously, the gap between the task execution and waiting times of fog node and the estimated values is the bigger, and the internal reputation value of fog node is the smaller. Then, the fog server stores and maintains the internal reputation of all fog nodes in the community as follows:

$$\mathrm{LR}(X) = [\mathrm{LR}(x_1), \mathrm{LR}(x_2) \ldots \mathrm{LR}(x_n)].$$

For the fog nodes whose internal reputation values are lower than a certain threshold, the fog server temporarily adds them to the blacklist and does not assign tasks to them. In fog computing, because a single node has relatively small resources, simply adding a fog node to the blacklist may lead to the gradual loss of more resources. Then, we introduce a "confinement" policy that allows repentance. The time when fog nodes are added to the blacklist is defined as follows:

$$\mathrm{BT}(l) = l^2 \cdot T.$$

In the abovementioned formula, $\mathrm{BT}(l)$ indicates the time when the node was blacklisted for the lth time, T is the unit time of confinement. As the number l is increased, the time of confinement is also increased with the square of l. On the other hand, we give each confined node the opportunity to correct. After the confinement time has arrived, the confined node is removed from the blacklist and reclassified as a new access node. If the number l exceeds a threshold T_{thres}, then the fog node is rejected to rejoin the community. Therefore, for the whole fog network, the changes in the available resources are dynamic. In addition, when a malicious fog node moves from one network community to another network community in our proposed scheme, the current fog server first needs to inquire about all other fog servers to obtain the historical data of the malicious fog node.

2) Indirect Reputation of Fog Nodes: In fog computing, a large number of fog nodes are widely distributed, and the probability of different nodes providing their resources for users is different. It is not reliable for only using the evaluation

[2]The greater the weight α, the more important the last task evaluation.

[3]It is quantified by the number of instructions that can be executed within the unit time.

[4]In this article, the formula $(tw^{(l)}_{\mathrm{actu}} - tw^{(l)}_{\mathrm{expect}} + 1/tw^{(l)}_{\mathrm{expect}} + 1)$ may be seen as the evaluation of completing the previous tasks on x.

[5]The greater the weight β, the more important the execution time of the current task.

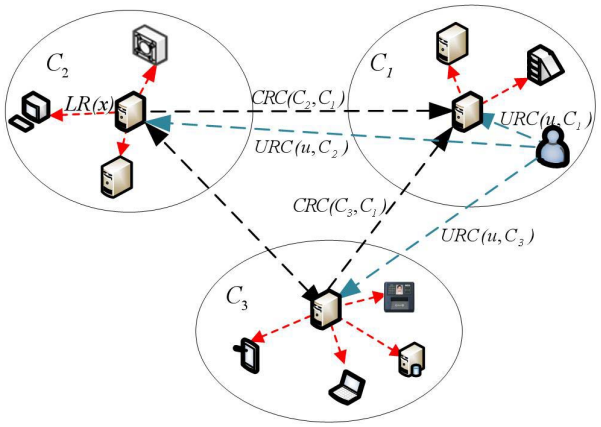


Fig. 4. Indirect reputation among communities.

of fog nodes (internal reputation) maintained by bottom fog servers. Thus, we construct the community reputation and use the fog server in each fog group to maintain the reputation among communities. Then, we may further obtain the full reputation of fog nodes by adding an indirect reputation of nodes, as shown in Fig. 4.

The community reputation varies with the completion of tasks by fog nodes. The comprehensive reputation value of the community C_i is defined as follows:

$$\mathrm{RC}(u, C_i) = \sigma \cdot \mathrm{URC}(u, C_i) + (1 - \sigma) \cdot \mathrm{UIRC}(C_i). \tag{3}$$

In the abovementioned formula, σ is the weight of the direct reputation value of the community on a user u ($0 \leq \sigma \leq 1$), which is used to balance the importance between the direct reputation $\mathrm{URC}(u, C_i)$ of u to the community C_i and the indirect reputation $\mathrm{UIRC}(C_i)$ of the community C_i.[6] Then, the detailed process is given as follows.

1) *Step 1:* $\mathrm{URC}(u, C_i)$ is the direct reputation value of the community C_i generated by the user u, which indicates the satisfaction of the user on the completion of the task through the community. The value of $\mathrm{URC}(u, C_i)$ is defined in $[0, 1]$, where the value 0 indicates that the user is completely dissatisfied with the completion of the task (the user may think that the related fog node is a malicious node); the value 1 indicates that the user is completely satisfied with the completion of the task (the user believes that the related node is credible). After each task is completed, the user must evaluate the completion of the task as an evaluation of the community that completed the task. In this article, the expected completion time $t_{\mathrm{req}}^{(k)}$ of the kth task is used as a measure of service quality to evaluate the fog nodes.[7] After a task is completed, the service difference can be obtained according to the requirement time of the task and its actual completion time through the formula, $\mathrm{Dev}^{(k)} = (t_{\mathrm{return}}^{(k)} - t_{\mathrm{req}}^{(k)}/t_{\mathrm{req}}^{(k)})$, where $t_{\mathrm{return}}^{(k)}$ indicates the time when the user receives the result of the kth task. If the service difference $Dev^{(k)}$ exceeds a certain threshold ε, the service of the fog node is considered to be unreliable. Thus, the evaluation value of the community C_i generated by the user u on the kth task $E^{(k)}(u, C_i)$ is defined as follows:

$$E^{(k)}(u, C_i) = \begin{cases} 1 - \dfrac{\mathrm{Dev}^{(k)}}{\varepsilon} & \mathrm{Dev}^{(k)} \leq \varepsilon \\ 0 & \mathrm{Dev}^{(k)} > \varepsilon. \end{cases} \tag{4}$$

The direct reputation value of the community C_i generated by the user u is calculated as follows:

$$\mathrm{URC}^{(k)}(u, C_i) = \begin{cases} \mathrm{URC}_{\mathrm{init}} & k = 0 \\ (1 - \gamma) \cdot \mathrm{URC}^{(k-1)}(u, C_i) & \\ \quad + \gamma \cdot E^{(k)}(u, C_i) & k \neq 0 \end{cases} \tag{5}$$

where $\mathrm{URC}_{\mathrm{init}}$ is the initial value of community's direct reputation; γ is the evaluation weight of u to C_i on the last task with $0 \leq \gamma \leq 1$, which is used to balance the importance between the evaluation of u to C_i on the last task and the evaluation of u to C_i on the previous tasks.[8] The related user u stores and maintains the direct reputation of corresponding communities as follows:

$$\mathrm{URC}(u) = [\mathrm{URC}(u, C_1), \mathrm{URC}(u, C_2) \ldots \mathrm{URC}(u, C_n)].$$

2) *Step 2:* $UIRC(C_i)$ is the indirect reputation of the community C_i. We set $C = \{C_1, C_2 \ldots C_n\}$, where the users in the community C_j ($C_j \in C$) used the resources in C_i ($C_i \in C$). Thus, after the fog node in C_i completes the hth task, the corresponding user in the community C_j should evaluate the completion of the hth task as an evaluation of C_j to C_i, which is defined as $EC^{(h)}(C_j, C_i)$. The value of $EC^{(h)}(C_j, C_i)$ is obtained by the score generated by the corresponding user.[9] Therefore, the full hth evaluation value of C_j to C_i is calculated as follows:

$$\mathrm{CRC}^{(h)}(C_j, C_i) = \begin{cases} \mathrm{CRC}_{\mathrm{init}} & h = 0 \\ (1-\delta) \cdot \mathrm{CRC}^{(h-1)}(C_j, C_i) & \\ \quad + \delta \cdot EC^{(h)}(C_j, C_i) & h \neq 0 \end{cases} \tag{6}$$

where $\mathrm{CRC}_{\mathrm{init}}$ is the initial evaluation value among communities; δ is the evaluation weight of C_j to C_i on the last task with $0 \leq \delta \leq 1$, which is used to balance the importance between the evaluation of C_j to C_i on the last task and the evaluation of C_j to C_i on the previous tasks.[10] Each related bottom fog server stores and maintains the corresponding evaluation values

[6]The greater the weight σ, the more important the direct reputation $\mathrm{URC}(u, C_i)$.

[7]In the work, we do not consider the accuracy of the completed tasks.

[8]The greater the weight γ, the more important the evaluation of u to C_i on the last task.

[9]In the work, the score belongs to [0,1].

[10]The greater the weight δ, the more important the evaluation of C_j to C_i on the last task.

among communities. For example, the community C_i stores and maintains the values as follows:

$$\mathrm{CRC}(C_i) = [\mathrm{CRC}(C_i, C_1), \mathrm{CRC}(C_i, C_2) \ldots \mathrm{CRC}(C_i, C_n)]$$

where we set $\mathrm{CRC}(C_i, C_i) = 1$. In addition, if a user maliciously defames or overpraises the fog node in a community, then it will have a direct impact on the reputation of the community. Thus, we use the formula $|EC^{(h)}(C_j, C_i) - \mathrm{CRC}^{(h-1)}(C_j, C_i)| < \rho$ to detect the validity of the user's evaluation on the task. If the difference between $EC^{(h)}(C_j, C_i)$ and $\mathrm{CRC}^{(h-1)}(C_j, C_i)$ exceeds the threshold value ρ, then the user's evaluation to the community C_i is invalid, so as to prevent from defaming or overpraising in a certain extent.

In fog computing, the physical distribution of fog nodes is related to geographical position. Thus, we consider that the frequency of cooperation among communities is different. As the communities are closer, they usually have more cooperative opportunities, and then, their mutual evaluations are more valuable [27]. Therefore, by taking into account the distance factor, when the other communities evaluate the community C_i, we further define the indirect reputation of the community C_i as follows:

$$\mathrm{UIRC}(C_i) = \sum_{j=1, j\neq i}^{n} \omega_j \cdot \mathrm{CRC}(C_j, C_i)$$

$$\omega_j = 1 - \frac{\mathrm{Dist}_{j,i}}{\sum_{j=1, j\neq i}^{n} \mathrm{Dist}_{j,i}}.$$

where $\mathrm{Dist}_{j,i}$ denotes the distance between the community C_j and the community C_i[11]; ω_j indicates the distance weight of C_j to C_i on the whole, which is used to set the distance importance between any two communities C_j and C_i. Thus, two communities are more distant; their distance weight is smaller.

3) *Step 3:* The comprehensive reputation value of the community C_i is calculated as follows:

$$\mathrm{RC}(u, C_i) = \sigma \cdot URC(u, C_i) + (1 - \sigma) \cdot UIRC(C_i).$$

The direct reputation $URC(u, C_i)$ of u to the community C_i is actually a cumulative evaluation of u to the different fog nodes in C_i. The indirect reputation $UIRC(C_i)$ of the community is the cumulative evaluation of the different users in the different communities to the different fog nodes in C_i.

4) *Step 4:* For widely distributed fog nodes, it is difficult for users to evaluate each fog node. Therefore, we construct the full reputation value of a single fog node, which is based on the comprehensive reputation of the community and the internal reputation of the fog node. The formula is described as follows:

$$R(u, x) = \sqrt{\mathrm{RC}(u, C_i) \cdot \mathrm{LR}(x)}$$

where $R(u, x)$ is the reputation value of the fog node x for the user u, and the fog node x belongs to the community C_i. We may know that the reputation $R(u, x)$ of the fog node x for the user u is based on $URC(u, C_i)$, $UIRC(C_i)$, and $\mathrm{LR}(x)$. That is to say, our proposed reputation mechanism synthesizes the evaluation of all relevant users. Then, the computed reputation of a fog node is different for different users, namely, the reputation of a fog node may be seen as specific-user-oriented.

We assume that the whole network is divided into n bottom communities. Then, the procedure of calculating $R(u, x)$ is shown in Algorithm 1, where Algorithm 1 is executed on the corresponding fog server.

Algorithm 1 Fog Nodes' Reputation Algorithm

Input:$\{Dist_{j,i}\}$
Output: $R(u, x)$

Computing-Reputation-Algorithm($\{Dist_{j,i}\}$)
Begin
  Inquire $URC(u)$ from u;
  $SumDist = 0$;
  $j = 1$;
  while $j \in [1, n] \& j \neq i$ **do**
    $SumDist = +Dist_{j,i}$;
    $j++$;
  End while
  $UIRC(C_i) = 0$;
  $j = 1$;
  while $j \in [1, n] \& j \neq i$ **do**
    Inquire $CRC(C_j, C_i)$ from the fog server of C_j;
    $UIRC(C_i) = +(1 - \frac{Dist_{j,i}}{SumDist}) \cdot CRC(C_j, C_i)$;
    $j++$;
  End while
  $RC(u, C_i) = \sigma \cdot URC(u, C_i) + (1 - \sigma) \cdot UIRC(C_i)$;
  Inquire $LR(x)$ from the fog server of C_i;
  $R(u, x) = \sqrt{RC(u, C_i) \cdot LR(x)}$;
  return $R(u, x)$;
End

B. Resource Allocation Scheme Based on Reputation Mechanism

Based on the structure of community-based fog network, we use the proposed reputation mechanism to construct a reliable resource allocation scheme. In fog computing, choosing computing resources closer to users is more conducive to reducing the delay of tasks. Thus, under the condition of satisfying user's reputation requirement, our proposed scheme chooses the nearest communities to complete users' tasks. In the section, the related symbols are shown in Table II.

1) Community Resource Recommendation: We define a fog node FN and its resource list $(r_1, r_2, \ldots, r_K)$, where r_τ represents the current available number of the resource τ in the node FN with $1 \leq \tau \leq K$, and K represents the maximal number of types of resource. Also, we use the list $(fr_1, fr_2, \ldots, fr_K)$ to represent the maximal amounts of

[11] In this article, we set that the physical distance in the network is measured by network hop.

TABLE II
RELATED SYMBOLS

Symbols	Description
CR	the total amounts of available resources in a community
TR	the total resource needs of a task
$StaT_{i,j}$	the estimated completion time of the subtask t_i executed on the fog node FN_j
$d_{i,j}$	the physical network distance from the user to the fog node FN_j
L_i	the number of instructions in the subtask t_i
$CapC_j$	the calculation capability of the fog node FN_j
$u_{i,j,\tau}$	the utilization rate of the resource τ on FN_j after the subtask t_i is executed
$PreT_{i,j}$	the preference of the subtask t_i to the fog node FN_j
$PreF_{i,j}$	the preference of the fog node FN_j to the subtask t_i

all resources in the node FN with $0 \le r_\tau \le fr_\tau$, where fr_τ represents the maximal number of the resource τ in the node FN. Therefore, fog nodes in a certain geographical area may form a community to provide resources for users. We assume that the high-level community Com'_x consists of a group of bottom (underlying) communities $\{com_1, com_2, \ldots, com_m\}$ and that an aggregation function Aggregation$A()$ is used to represent the total amounts of available resources in the corresponding community. Then, the total amount of available resources in the high-level community Com'_x is described as follows:

$$\begin{aligned}\mathrm{CR} &= \mathrm{Aggregation}A(\mathrm{Com}'_x)\\ &= \sum_{i=1}^{m} \mathrm{Aggregatio}A(\mathrm{com}_i)\\ &= \sum_{i=1}^{m}\sum_{j=1}^{num_i}[\mathrm{FN}_j :: (r_{j,1}, r_{j,2}, \ldots, r_{j,K})]\\ &= (Rr_1, Rr_2, \ldots, Rr_K)\end{aligned}$$

where num_i indicates the number of fog nodes that satisfying the user's reputation requirement in the bottom community com_i; $\mathrm{FN}_j :: (r_{j,1}, r_{j,2}, \ldots, r_{j,K})$ indicates the total numbers of current available resources in the fog node FN_j, which satisfies the user's reputation requirement in com_i; and Rr_τ indicates the sum of the resource τ in Com'_x. In addition, when $m = 1$, it denotes that the high-level community Com'_x is only the bottom community.

Similarly, we define a user task consisting of N subtasks,[12] namely, $T = (t_1, t_2, \ldots, t_N)$. The resource needs of each subtask t_p are represented as $(re_1, re_2, \ldots, re_K)$, where re_τ denotes the resource need of the subtask t_p for the resource τ with $1 \le p \le N$ and $1 \le \tau \le K$. In order to meet the requirements of the user's tasks, each resource in fog node allocated to each subtask needs to satisfy the constraint condition $0 < re_\tau \le r_\tau$. Furthermore, if all the resources of the node can satisfy the constraint conditions, then the subtask can be executed on the node. At the same time, when a subtask is executed on the fog node, all available resources of the node should be updated accordingly by $r_\tau = r_\tau - re_\tau$; on the contrary, after a subtask is executed, all available resources should also be updated by $r_\tau = r_\tau + re_\tau$. Therefore, an aggregation function $AggregationR()$ is used to represent the total amounts of required resources on the corresponding task. The total resource needs of the task T are described as follows:

$$\begin{aligned}\mathrm{TR} &= \mathrm{Aggregation}R(T)\\ &= \sum_{p=1}^{N}[t_p :: (re_{p,1}, re_{p,2}, \ldots, re_{p,K})]\\ &= (Re_1, Re_2, \ldots, Re_K)\end{aligned}$$

where $t_p :: (re_{p,1}, re_{p,2}, \ldots, re_{p,K})$ indicates the resource needs of the subtask t_p, and Re_τ indicates the sum of the required resource τ in T. Thus, the main goal of the resource allocation scheme is to find the nearest community that can meet the total resource and reputation requirements of the user's tasks. When a fog server receives task requests from multiple users, the users' tasks are put into a queue to be allocated, in turn, according to their processed deadlines. Each user's task is to proceed as follows (as shown in Fig. 5).

1) *Step 1:* A user u first locates the directly connected community and then submits the task request T and the corresponding reputation requirement R to the fog server of the current bottom community. The fog server aggregates the resource requirements for the task request T to obtain a total resource needs TR.
2) *Step 2:* The current layer fog server inquires the user u for the direct reputation $\mathrm{URC}(u)$ about all the related communities. It inquires the internal reputation $\mathrm{LR}(X)$ from each bottom fog server managed by itself. It inquires all bottom fog servers for the evaluation values related to the bottom communities $\{C_i\}$ as follows:

$$(\{\mathrm{CRC}(C_1, C_i), \mathrm{CRC}(C_2, C_i), \ldots, \mathrm{CRC}(C_n, C_i)\})$$

where the bottom communities $\{C_i\}$ are within the scope of the current fog server's management with $1 \le i \le m$; m is the number of the bottom communities managed by the current fog server; and n is the number of the bottom communities in the whole fog network. For example, if $m = 1$, it denotes that the current fog server is only the bottom fog server, which only manages a community. Then, the reputation values $(R_1, R_2, \ldots, R_Q)$ of all available fog nodes in the current layer community are calculated according to Algorithm 1, where Q is the number of fog nodes in the current layer community.
3) *Step 3:* The fog nodes satisfying the condition $R_p > R$ with $p \in [1, Q]$ are selected from the corresponding community, and the resource aggregation is performed to obtain a total amount CR of available resources. For CR and TR, if all the resources can satisfy the corresponding

[12]In the work, we mainly consider the task can be parallel processed.

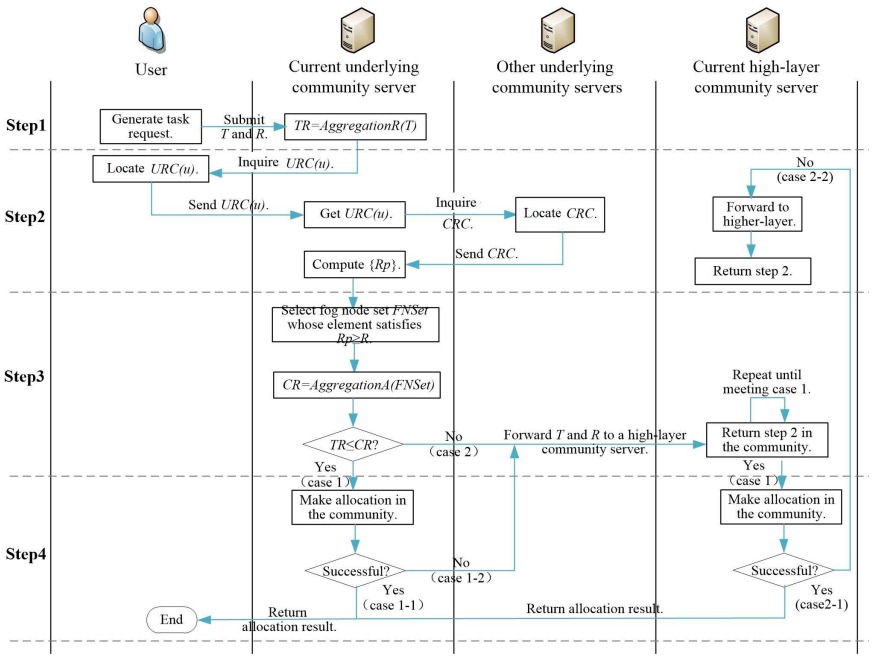


Fig. 5. Community resource recommendation process on the user's task.

conditions $0 < Re_\tau \leq Rr_\tau$ for all τs and then go to Step 4; otherwise, the current fog server transfers the user's task request to the upper layer fog server (namely the parent of the current fog server) and then go to Step 2 until a high-level community can meet the user's needs.

4) *Step 4:* The current fog server makes resource allocation within the corresponding community. If the allocation fails, then go to Step 2 until a successful resource allocation can be performed in a high-level community.

Since the total resources of a community are aggregated by multiple fog nodes, there are likely to be some resource fragments that cannot be utilized. Therefore, the resource allocation process within a community may fail, and a detailed approach of resource allocation within a community is further described in the next section.

We assume that the community tree of a fog network is constructed, where C_Com is the structure of the current community, C_id is the community index, Fs_id is the index of fog server in the current community, $FNum$ is the number of fog nodes in the current community, PC_pr is the pointer of parent community of the current community, and $FNList$ is the list of fog nodes in the current community. The task of a user u is $T = (t_1, t_2, \ldots, t_N)$, the reputation requirement is R, and each fog node has K kinds of resources. The recommendation process of community resources is shown in Algorithm 2.

2) Resource Allocation Within Community: In the section, an improved stable matching algorithm is proposed to finish the assignment between subtasks and fog nodes in a community. We assume that there are M fog nodes whose reputation satisfies the requirement of user in the community. The M fog nodes are expressed as $\text{FN\_List} = \{\text{FN}_1, \text{FN}_2, \ldots, \text{FN}_M\}$. Also, the task T of a user consists of N subtasks, namely, $T = (t_1, t_2, \ldots, t_N)$. For example, we set the processor, memory, and bandwidth of a fog node j as three different kinds of resource ($1 \leq j \leq M$). Then, its available resource list may be expressed as $F\_List_j = (r_{j,1}, r_{j,2}, r_{j,3})$, and the resource needs of a subtask t_i ($1 \leq i \leq N$) are represented as $T\_List_i = (re_{i,1}, re_{i,2}, re_{i,3})$. Thus, N subtasks and M fog nodes need to be matched through meeting the resource requirements of all the subtasks.

For users, the dependence and completion time of each subtask on different resources are different. We set the weight vector $\Phi_i = (\varphi_{i,1}, \varphi_{i,2}, \varphi_{i,3})$ of the subtask t_i to represent the different dependence of t_i on the abovementioned three kinds of resource. Also, we set $\varphi_{i,4}$ to indicate the requirement degree of the subtask t_i to completion time, where $\varphi_{i,1} + \varphi_{i,2} + \varphi_{i,3} + \varphi_{i,4} = 1$. Since the completion time of a task is usually the most intuitive measure for the service quality of the task, the weight $\varphi_{i,4}$ is usually chosen to be greater than 0.5. Therefore, we use the following formulas to measure the preference of t_i to FN_j:

$$\text{Pre}T_{i,j} = \frac{\Phi_i \cdot T\_\text{List}_i}{\varphi_{i,4} \cdot \text{StaT}_{\text{i,j}} \cdot d_{i,j}} \tag{7}$$

$$\text{Sta}T_{i,j} = \text{Last}T_j + \frac{L_i}{\text{Cap}C_j} \tag{8}$$

where $StaT_{i,j}$ is the estimated completion time of the subtask t_i executed on the fog node FN_j, $LastT_j$ is the estimated completion time of the last task allocated to the fog node FN_j, $CapC_j$ is the calculation capability of the fog node FN_j, L_i is the number of instructions in the subtask t_i, and $d_{i,j}$ is the physical network distance from the user to the fog node FN_j. For the subtask t_i, all the fog nodes are sorted according to $PreT_{i,j}$ from large to small. Thus, we may obtain the preference list of the subtask t_i to the fog nodes.

In fog computing, the requirements of different tasks on different resources are also different. Thus, it is easy to generate a large amount of resource fragments when tasks are

Algorithm 2 Community Resource Recommendation Algorithm

Input: C_Com, u, T
Output: $IsSuccess$
Community_Structure{
  int C_id;
  int Fs_id;
  int $FNum$;
  Community_Structure* PC_pr;
  List<FNode> $FNList$; } C_Com;
Com-Res-Recom-Algorithm(C_Com, u, T)
Begin
  $IsSuccess = 0$;
  $TR = AggregationR(T)$;
  if $C\_Com == NULL$ **then**
    return $IsSuccess$;
  End if
  $C\_length \leftarrow C\_Com.FNList.length$;
  $C\_List \leftarrow C\_Com.FNList$;
  $R\_List \leftarrow \emptyset$;
  Call **Algorithm** 1 to compute $R_j = R(u, x_j)$ for $x_j \in C\_List \& j \in [1, C\_length]$;
  Add x_j into $C\_availability$ for $x_j \in C\_List$ if $R_j \geq R$;
  $CR = AggregationA(C\_availability)$;
  $Satis = 1$;
  $j = 1$;
  while $Re_j \in TR$ & $Rr_j \in CR$ & $j \in [1, K]$ **do**
    if $Re_j > Rr_j$ **then**
      $Satis = 0$;
      Break;
    End if
    $j++$;
  End while
  if $Satis == 1$ **then**
    $IsSuccess \leftarrow$ Call **Algorithm 3**;
    if $IsSuccess == 1$ **then**
      return $IsSuccess$;
    End if
  End if
  Com-Res-Recom-Algorithm($C\_Com.PC\_pr$, u, T);
End

placed on fog nodes, resulting in waste of resources. As far as possible, balancing the utilization ratio of different resources on the same fog node can reduce resource fragments and improve the utilization ratio of fog resources. In this article, the concept of skewness [28] is introduced as a measure of the preference of the fog node FN_j to the subtasks. Then, our proposed method uses the variance of K different resource occupancy rates on the fog node to measure the balance of resource utilization ratios. The formula is described as follows:

$$PreF_{i,j} = \frac{1}{K} \cdot \sum_{\tau=1}^{K} \left(u_{i,j,\tau} - \overline{u_{i,j}}\right)^2 = \frac{1}{K} \cdot \left(\sum_{\tau=1}^{K} \left(u_{i,j,\tau}^2\right) - \sum_{\tau=1}^{K} \left(\overline{u_{i,j}}^2\right)\right) \tag{9}$$

$$u_{i,j,\tau} = 1 - \frac{r_{j,\tau} - re_{i,\tau}}{fr_{j,\tau}} \tag{10}$$

$$\overline{u_{i,j}} = \frac{1}{K} \sum_{\tau=1}^{K} u_{i,j,\tau}. \tag{11}$$

In the abovementioned formula, $r_{j,\tau}$ represents the number of the current available resource τ in the fog node FN_j, $fr_{j,\tau}$ represents the maximal amount of the resource τ in FN_j, and $re_{i,\tau}$ represents the need number of the subtask t_i to the resource τ. Thus, $u_{i,j,\tau}$ indicates the utilization rate of the resource τ on FN_j after the subtask t_i is executed; $\overline{u_{i,j}}$ indicates the average utilization rate of all the kinds of resources on FN_j. For the fog node FN_j, all the subtasks are sorted according to $PreF_{i,j}$ from small to large. Thus, we may obtain the preference list of the fog node FN_j to the subtasks.

All the subtasks from the user are added to the no allocated subtask list. The resource allocation procedure is described as follows.

1) *Step 1:* For the selected subtasks from the no allocated subtask list, the algorithm respectively constructs the first "task–fog node" pair (t_i, FN_j) according to the preference list of each subtask. For each "task–fog node" pair (t_i, FN_j), if the fog node FN_j only appears once in the current round of selection, then the algorithm checks whether the fog node satisfies the constraint condition $r_{j,\tau} \geq re_{i,\tau}$ for each resource τ; if the conditions are satisfied, then the corresponding "task–fog node" pair (t_i, FN_j) is added to the paired table; if the fog node FN_j appears multiple times, then the current most preferred "task–fog node" pair (t_i, FN_j) is selected based on the preference list of FN_j, and the algorithm checks whether the corresponding fog node satisfies the constraint condition $r_{j,\tau} \geq re_{i,\tau}$ for each resource τ; and if the conditions are satisfied, then the corresponding "task–fog node" pair is added to the paired table. The saved "task–fog node" pairs in the paired table are called to be a successful match of subtasks and computing resources (fog nodes).
2) *Step 2:* For the no matched subtasks, the algorithm, respectively, continues to construct the next (second) "task–fog node" pairs in their corresponding subtask preference lists. For each "task-fog node" pair (t_i, FN_j), the following steps are finished.
   a) If the fog node FN_j only appears once in the current round of selection, then the following holds.
      i) If the fog node FN_j is not paired in the last round of selection (namely, the fog node FN_j is not found in the paired table), then the algorithm checks whether the fog node satisfies the constraint condition $r_{j,\tau} \geq re_{i,\tau}$ for each resource τ; if the conditions are satisfied, the corresponding "task–fog node" pair (t_i, FN_j) is added to the paired table.
      ii) If the fog node FN_j may be found in the paired table, the current "task–fog node" pair (t_i, FN_j) needs to compare with the saved corresponding "task–fog node" pair according to the preference list of FN_j; if the current pair

is more preferred, then the algorithm checks whether the fog node satisfies the constraint condition $r_{j,\tau} \geq re_{i,\tau}$ for each resource τ; and if the conditions are satisfied, the current pair is to replace the saved pair in the paired table, and the corresponding subtask from the saved pair is entered to the list of the no allocated subtasks; otherwise, the corresponding subtask from the current pair continues to be entered to the list of the no matched subtasks.

b) If the fog node FN_j appears multiple times, then the following holds.

i) If, namely, the fog node FN_j is not found in the paired table, then the current most preferred "task–fog node" pair (t_i, FN_j) is selected based on the preference list of FN_j, and the algorithm checks whether the corresponding fog node satisfies the constraint condition $r_{j,\tau} \geq re_{i,\tau}$ for each resource τ; if the conditions are satisfied, then the corresponding "task–fog node" pair is added to the paired table.

ii) If the fog node FN_j may be found in the paired table, then the related "task–fog node" pairs need to compare with the saved corresponding "task–fog node" pair according to the preference list of FN_j; if one of the related pairs is more preferred, then the algorithm checks whether the fog node satisfies the constraint condition $r_{j,\tau} \geq re_{i,\tau}$ for each resource τ; and if the conditions are satisfied, then the corresponding pair is to replace the saved pair in the paired table, and the subtask from the saved pair and the other subtasks related to FN_j are entered to the list of the no allocated subtasks; otherwise, all the subtasks from the related "task–fog node" pairs continue to be entered to the list of the no allocated subtasks.

3) *Step 3:* Repeat Step 2 until the related subtasks are matched to the fog nodes or all the subtask preference lists of no matched subtasks had been accessed.

4) *Step 4:* If all the subtasks are allocated to the appropriate fog nodes, then this allocation is completed; otherwise, this allocation fails in the community, and the task request of the user is forwarded to the upper level fog server.

We assume that the resource allocation is performed in a community for a subtask set $T = (t_1, t_2, \ldots, t_N)$, where all available fog nodes in the community are represented as a set $FN\_List = \{FN_1, FN_2, \ldots, FN_M\}$. We set that $PreTS = \{Pt_i\}$ is a set of the preference lists of subtask, where Pt_i is the preference list of the subtask t_i with $1 \leq i \leq N$; $PreFS = \{Pf_j\}$ is a set of the preference lists of fog node, where Pf_j is the preference list of the fog node FN_j with $1 \leq j \leq M$; Ψ denotes a "task–fog node" pair set whose element is the "task–fog node" pair (t_i, FN_j) with $i \in \{1, 2, \ldots, N\}$ and $j \in \{1, 2, \ldots, M\}$; Ω denotes a subtask set whose element is the subtask that is temporarily failed to be assigned to the related fog node; and Υ denotes a subtask set whose element is the subtask that is failed to be assigned to the fog node. The whole allocation procedure is described as Algorithm 3.

Algorithm 3 Resource Allocation Algorithm Within a Community

Input: T, FN_List
Output: Ψ
Begin
$\Omega \leftarrow T$;
Get the set $PreTS = \{Pt_i\}$ according to Formula 7;
Get the set $PreFS = \{Pf_j\}$ according to Formula 9;
$\Psi \leftarrow \emptyset$;
$\Upsilon \leftarrow \emptyset$;
while $\Omega \neq \emptyset$ **do**
Remove t_i from Ω and add t_i into Υ for $t_i \in \Omega$ if $Pt_i = \emptyset$ with $Pt_i \in PreTS$;
Remove the related fog node FN_j from the Pt_i of t_i for $t_i \in \Omega$ if $r_{j,\tau} < re_{i,\tau}$ for each resource τ in FN_j;
All the most preferred "task–fog node" pair $t_i - FN_j$s are constructed according to the Pt_i of t_i for $t_i \in \Omega$;
The pair $t_i - FN_j$ is saved to Ψ for all $t_i - FN_j$s if $r_{j,\tau} \geq re_{i,\tau}$ for each resource τ in FN_j;
while $FN_j \in FN\_List \& \Psi \neq \emptyset$ **do**
($T\_pair \leftarrow t_i - FN_j$)$\leftarrow$**FindFirstPair**($\Psi$,$FN_j$);
//The first pair $t_i - FN_j$ related to FN_j in Ψ is found to T_pair
while $t_i - FN_j \leftarrow$**FindPair**(Ψ,FN_j)$\& T\_pair \neq \emptyset$ **do**
//Every pair $t_i - FN_j$ related to FN_j in Ψ is accessed until the accessed pair is null
if Preference(T_pair)$\geq$**Preference**($t_i - FN_j$) **then**
//The preference of the sub-task from T_pair is prior to the preference of t_i in Pf_j according to the Pf_j of FN_j
Remove $t_i - FN_j$ from Ψ;
Add t_i into Ω;
Delete FN_j from the Pt_i of t_i;
Else if Preference(T_pair)$<$**Preference**($t_i - FN_j$)
Remove T_pair from Ψ;
Add the related sub-task in T_pair into Ω;
Delete the related fog node in T_pair from the preference list of the corresponding sub-task in T_pair;
$T\_pair \leftarrow t_i - FN_j$;
End if
End while
if $T\_pair \neq \emptyset$ **then**
Remove the related sub-task in T_pair from Ω;
End if
End while
End while
if $\Upsilon \neq NULL$ **then**
$\Psi = NULL$;
End if
Output(Ψ);
End

V. Experimental Analysis

A. Experiment Environment

We make some experiments to test the performance of our proposed scheme through the CloudSim [29] simulation

platform, where we use JAVA to code our proposed algorithms.[13] In the experiments, the test environment is under Win7, Intel i5 CPU, and 8-GB RAM. We use the extended CloudSim platform to simulate a fog computing environment under a hierarchical community structure. We set 80 simulated fog nodes in the fog network, where each simulated node can provide computing services as a fog node or as a user node, and the resources in each fog node are represented as a tuple (core, memory, and bandwidth). Also, the whole fog network structure is set to a four-layer tree structure, where every ten fog nodes are connected to an access point to form a bottom community; every two access points are connected to one access-layer switch; every two access-layer switches are connected to one convergence-layer switch; and every two convergence-layer switches are connected to one core switch. Thus, in the whole hierarchical community structure, 15 communities may be formed with different granularity through eight access points, four access-layer switches, two convergence-layer switches, and one core switch.

In the hierarchical community structure, the positional relationship between the user and the service (fog) nodes may appear as follows: 1) in the bottom (fourth) layer, the user and all the service nodes connect to the same access point, namely, they are located in the same bottom community; 2) in the third layer, the user needs to communicate with some service nodes through an access-layer switch, namely, all the service nodes are distributed in the third-layer community; 3) in the second layer, the user needs to communicate with some service nodes through the access-layer switches and the convergence-layer switch, namely, all the service nodes are distributed in the second-layer community, and 4) in the top (first) layer, the user needs to communicate with some service nodes through the access-layer switches, the convergence-layer switches, and the core switch, namely, all the service nodes are distributed in the top community. In addition, we set the communication delay between network devices to 0.04 s and the communication delay between network devices and simulated cloud server to 0.6 s [30]. We set that task requesters are randomly generated from 80 fog nodes, where each user's task consists of ten subtasks and the completion time of each task depends on the completion time of its last subtask. Other related experimental parameters are set as in Table III.

In our proposed experiments, there are four different kinds of fog computing nodes.

1) *Honest Nodes:* These nodes provide users with authentic and reliable resources, and they also can honestly and objectively evaluate other nodes when they are as users.
2) *Collusion Nodes With Servers:* These nodes may continue to obtain good internal evaluations from their colluding fog servers.[14]

[13]The CloudSim simulation platform is an open source, where users can extend the platform according to their own research content. Based on the open platform, users may add their own code to the platform and then recompile and release the platform. In our experiments, we first configure the experimental environment according to the interfaces provided by the platform; then, we code these algorithms (Algorithms 1–3) by JAVA and add the code to the open platform.

[14]These fog nodes can only collude with the bottom fog servers.

TABLE III
RELATED EXPERIMENTAL PARAMETERS

Related parameters	values
Core (processor) of node	2-4
Bandwidth of node	100-500M
RAM of node	2-4G
Computing capability of node	10MIPS
Length of subtask (the number of instructions)	80-100MI
Computing capability of cloud server	1000MIPS
Core need of subtask	1-4
Bandwidth need of subtask	100-500M
RAM need of subtask	2-4G
Weight of completion time	0.6
Weight of three kinds of resource	0.1-0.4

TABLE IV
RELATED REPUTATION PARAMETERS

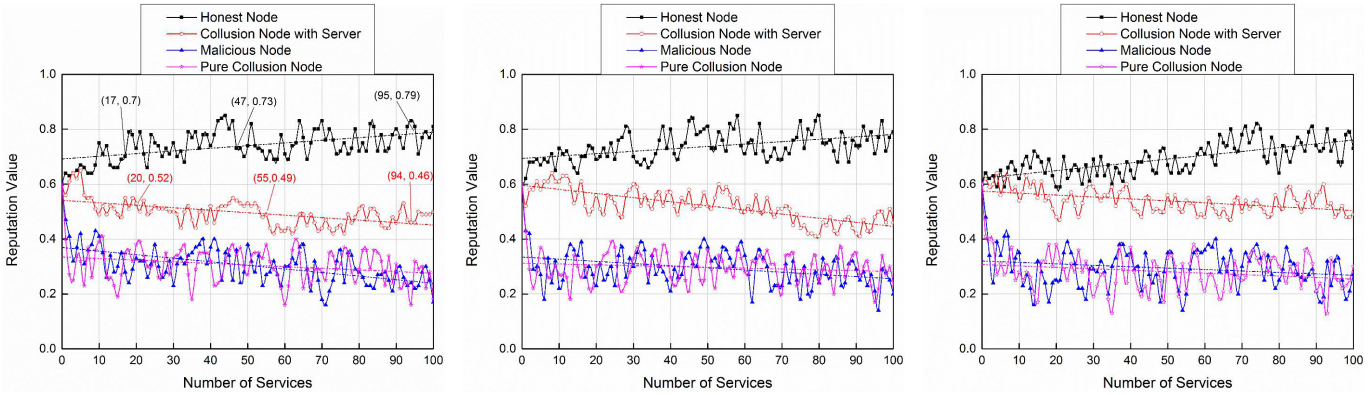
Reputation parameters	values
Initial reputation of node	0.6
Historical factor	0.6
Timeout threshold	0.1
Direct reputation factor	0.6
Reputation for denial of service	0.5

3) *Pure Collusion Nodes:* These nodes can collide with each other to provide good evaluation for collusion members and bad feedback for other no colluding nodes.
4) *Malicious Nodes:* These nodes provide users with fake resources (or dissatisfied resources).

The latter three kinds of nodes are called dishonest nodes. The related reputation parameters of the node are set as in Table IV.

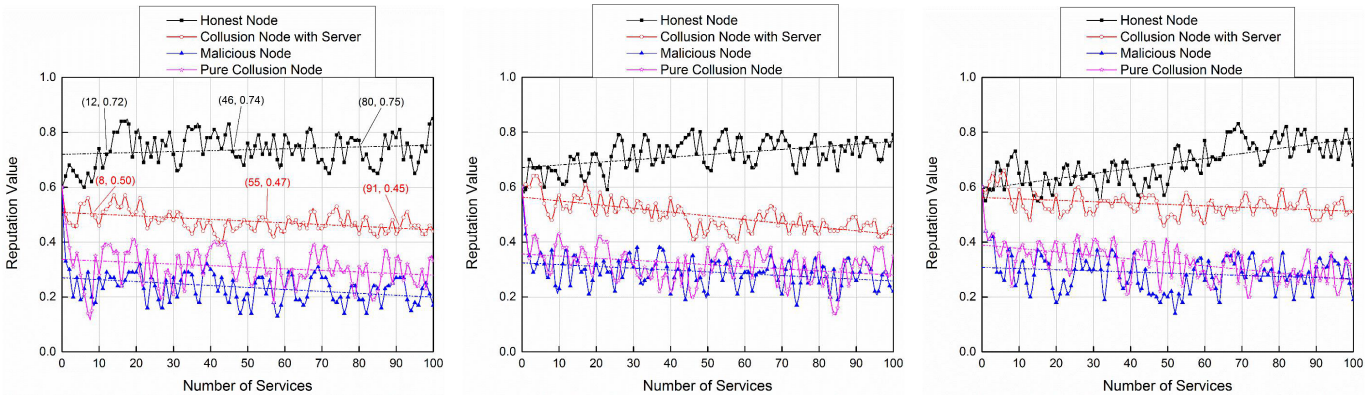
Also, for the setting of the parameters, we have the following explanation.

1) In some traditional reputation models, the initial value of reputation is often set to 0.5. Considering that users initially have some trust in fog service providers, we set the initial reputation value of fog node is higher than 0.5.
2) In terms of time, the closer the time, the greater the reference value of data. Therefore, the historical factors (such as α, β, γ, and δ) are usually set to be greater than 0.5.
3) The timeout threshold ε is set to 0.1. After a task is completed, the service difference can be obtained according to the requirement time of the task and its actual completion time through the formula: $Dev^{(k)} = (t_{\text{return}}^{(k)} - t_{\text{req}}^{(k)}/t_{\text{req}}^{(k)})$. If the service difference $Dev^{(k)}$ exceeds the timeout threshold ε, then the service of the fog node is considered to be unreliable. The larger the timeout threshold is set, the greater the user's tolerance of service timeout is.
4) The direct reputation factor σ is set to 0.6. σ is the weight of the direct reputation value of the community generated by a user. The larger the direct reputation factor, the greater the impact of the direct reputation on the full reputation.
5) The reputation for denial of service is the reputation requirement R submitted by a user. The user refuses to



(a) (b) (c)

Fig. 6. Reputation values of different nodes when the internal proportion of dishonest nodes is 2:1:1. (a) Proportion of dishonest nodes is 25%. (b) Proportion of dishonest nodes is 50%. (c) Proportion of dishonest nodes is 75%.



(a) (b) (c)

Fig. 7. Reputation values of different nodes when the internal proportion of dishonest nodes is 1:1:2. (a) Proportion of dishonest nodes is 25%. (b) Proportion of dishonest nodes is 50%. (c) Proportion of dishonest nodes is 75%.

use the resources provided by fog nodes whose reputation values are lower than R. The larger the reputation requirement R, the higher the reputation requirement of the user for computing resources.

In order to evaluate the performance of our proposed scheme, we make some experiments to test and analyze the reputation change of different nodes, the success rate of resource allocation, the average completion delay of tasks, and the average distance between users and service providers when we change different proportions of nodes and different amounts of tasks.

B. Simulation Tests and Analysis

1) Reputation Change of Nodes: In the test, we simulate the change of reputation for each node,[15] where we assume that all dishonest nodes may also provide fake resources. With the increase in service times, Figs. 6 and 7 show the changing trend of reputations of four kinds of nodes under the settings of 25%, 50%, and 75% proportions of dishonest nodes. The proportion of malicious nodes, collusion nodes with servers, and pure collusion nodes in all dishonest nodes is 2:1:1 in Fig. 6; the proportion of pure collusion nodes is increased, and the total proportion is set to 1:1:2 in Fig. 7. As can be seen from Figs. 6 and 7, the reputation of honest nodes will gradually increase with the increase in service times, while the reputation of dishonest nodes will gradually decrease. According to the trend lines, it can be seen that with the increase in the number of services, the reputation value of honest nodes will gradually increase, and the reputation value of dishonest nodes will gradually decrease. In Fig. 6(a), when the number of services increases from 17 to 95, the reputation value of honest nodes gradually increases from 0.7 to about 0.79. Also, when the number of services increases from 20 to 94, the reputation value of collusion nodes with servers decreases from 0.52 to about 0.46. In Fig. 7, the proportion of collusion nodes is increased. Therefore, in Fig. 7(d), when the number of services increases from 12 to 80, the reputation value of honest nodes gradually increases from 0.72 to about 0.75. Also, when the number of services increases from 8 to 91, the reputation value of collusion nodes with servers decreases from 0.50 to about 0.45.

For malicious nodes, the reputation of such nodes will gradually decrease due to the certain falsity of the resources that they provide. As the number of services increases, fewer users are willing to use their resources. It makes the downward

[15]In our proposed experiments, we set that the reputation requirement R of a user is 0.5, where the reputation requirement R is used to test the success rate of resource allocation. Thus, in the test, we permit the reputation of a fog node is not restricted, so as to show the full reputation change procedure of fog nodes.

trend of the reputation of such nodes gradually and becomes stable and only fluctuates within a lower reputation interval. As shown in Fig. 7(f), the reputation of malicious nodes gradually decreases and is mostly maintained at about 0.3 in the later stage. Comparing Figs. 6 and 7, since the malicious nodes do not interact with each other, the proportion change of malicious nodes has little effect on the reputation of a single malicious node.

Also, in the figures, the declining trend of reputations of collusion nodes with servers is slower, and their reputations are higher than those of pure collusion nodes and malicious nodes on the whole, which is because the fog server for internal reputation evaluation of node is an important factor when it calculates the node's reputation, which can influence the true evaluation of collusion nodes with servers in our proposed scheme. However, with the increase in the number of services, the mutual evaluation between communities can gradually balance the false evaluation of the nodes from the collusive fog servers. Thus, the reputation of collusion nodes with fog servers gradually decreases and, finally, maintains a certain range. As shown in Fig. 7(d), the reputation of the collusion nodes with servers gradually decreases and is basically maintained below 0.5 in the later stage. Due to the importance of fog servers in fog computing, more secure and reliable devices are often used; thus, the likelihood of fraud and deceptive behaviors from fog servers is relatively small.

In addition, due to the existence of "accomplices," the fluctuation of reputations of pure collusion nodes is relatively severe compared with other types of dishonest nodes. Compared with malicious nodes, collusion nodes may obtain better evaluations because of "accomplices"; thus, their reputations may be relatively high for a period of time. However, due to the lower evaluations given by other users, their reputations decrease and gradually maintain in a certain range. At the same time, comparing Figs. 6 and 7, we can find that when the proportion of collusion nodes is small, the difference in variation tendencies of reputations between collusion nodes and malicious nodes is not particularly obvious. In Fig. 7, when the proportion of collusion nodes increases, the reputation change of collusion nodes is more obvious, and the overall decline trend of the reputation value slows down, the value slightly increases in the later stage. As shown in Fig. 6(c), the late reputation of collusion nodes is basically maintained at about 0.25; in Fig. 7(f), the late reputation of collusion nodes is basically maintained at about 0.3. That is because the more the number of collusion nodes, the greater the chance of giving false evaluations between collusion nodes. Thus, the reputation value of the node fluctuates greatly, and its decline trend slows down.

Furthermore, comparing the proportions of three kinds of dishonest nodes when the proportion of dishonest nodes is low (about 25%), the reputation of honest nodes grows faster and soon reaches about 0.8. When the proportions of dishonest nodes are 50% and 75%, respectively, the reputation of honest nodes fluctuates greatly during the rising process, and the number of services is bigger when the reputation reaches 0.8. That is because when the proportion of dishonest nodes is high, more collusion nodes may produce false evaluations, which may interfere with the reputation of honest nodes. For collusion nodes with servers, we find that when the proportion of dishonest nodes increases, the reputation decline speed of collusion nodes with servers decreases. As shown in Fig. 7(d), the reputation of collusion nodes with servers is gradually maintained at about 0.45; in Fig. 7(f), the reputation of collusion nodes with servers is gradually maintained at about 0.55. That is, because, when the proportion of dishonest nodes increases, the proportion of collusion nodes with servers increases, the possibility of using such dishonest nodes increases accordingly. In this process, since the internal evaluation of the node maintained by fog server has a relatively large impact on final reputation calculation, other nodes have a relatively small influence on the evaluation. Thus, the reputation of collusion nodes with servers decreases slowly. For malicious nodes and pure collusion nodes, when the proportion of dishonest nodes increases, the decrease rate of the reputation value also slows down, and the maintenance range of the reputation value increases. It can be seen from Figs. 6 and 7 that the reputation changes of the four kinds of nodes substantially conform to their expected trends. As the proportion of dishonest nodes increases, the reputation changes of various kinds of nodes gradually slow down (especially when the fog network is in the high proportion of dishonest nodes).

2) Success Rate of Resource Allocation: In the test, we use "deception rate" to denote the proportion of nodes providing false resources in all pure collusion nodes or collusion nodes with servers. Based on the completion of 100 tasks (namely, 1000 subtasks), we test the effects of the proportion of malicious nodes and the deception rate of other dishonest nodes on the success rate of resource allocation. We further set that the success rate of resource allocation is the proportion that the subtasks are allocated on time according to their needs. Under our proposed scheme and the no-reputation model, Fig. 8 shows the effect of the proportion of malicious nodes on the success rate of resource allocation. It can be seen from Fig. 8 that when the proportion of malicious nodes is small, the success rates of the two schemes are not different. However, as the proportion of malicious nodes increases, the success rate decreases rapidly in the no-reputation model because it does not take any measures against malicious nodes. When the proportion of malicious nodes is 50%, the success rate generated by the nonreputation model is only 50.7%. On the contrary, our proposed scheme can effectively identify malicious nodes. Thus, with the increasing proportion of malicious nodes, the success rate generated by our proposed scheme decreases slowly. When the proportion of malicious nodes is 50%, the success rate generated by our proposed scheme can reach 78%.

Fig. 9(a) shows the effect of the deception rate of pure collusion nodes on the success rate when the proportion of collusion nodes is 50%. Since collusion nodes may be more easily allocated to tasks through good evaluations from their accomplices, the no-reputation model cannot distinguish such deceptions. Thus, it results in a rapid decline in the success rate. At the same time, we find that when the deception rate of collusion nodes is less than 60%, our proposed scheme

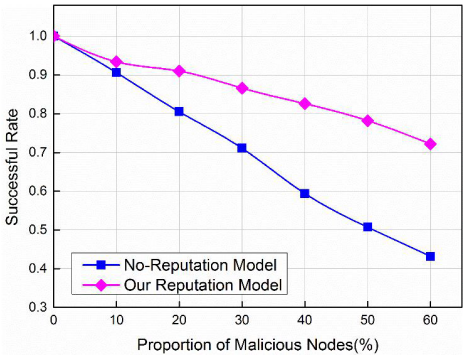


Fig. 8. Effect of the proportion of malicious nodes on service success rate.

identifies the deception behaviors of collusion nodes slowly, and the success rate is still declining rapidly. However, with the increase in the deception rate of collusion nodes, the decline trend of the success rate is rapidly slowed down. Because our proposed scheme can effectively identify the collusion nodes and their deceptive behaviors, it has an obvious restraining effect on the deception behaviors of collusion nodes.

Fig. 9(b) shows the effect of the deception rate of collusion nodes with servers on the success rate when the proportion of collusion nodes is 50%. Similar to Figs. 8 and 9(a), the success rate generated by the no-reputation model decreases rapidly. Since the internal reputation of the node maintained by fog server has a great influence on the final reputation of the node, it is difficult to distinguish the deception behavior of collusion with fog server when the deception rate of collusion nodes with servers is less than 60%. Thus, the success rates of the two schemes drop to about 70% rapidly. However, with the increase in the deception rate, the success rate generated by our proposed scheme is gradually maintained above 65%. Therefore, our proposed scheme has a certain restraining effect on the deception behavior of collusion with the fog server.

When the number of tasks changes from 10 to 100 and the proportions of dishonest nodes are 40% and 60% respectively, Fig. 10 shows the change of success rates under our proposed scheme and the no-reputation model. As shown in Fig. 10, when the number of tasks increases, the success rate generated by our proposed scheme is increasing. However, the success rate generated by the no-reputation model is constantly fluctuating because the no-reputation model cannot identify the dishonest behavior of nodes. In our scheme, since the reputation values of dishonest nodes gradually decline, more users will reject services from the nodes with lower reputation values. Thus, the success rate generated by our scheme will gradually increase. Furthermore, comparing the two groups of figures with different proportions of dishonest nodes, when the proportion of dishonest nodes is high (about 60%), the success rate decreases slightly. Also, the decrease in success rate is least affected by malicious nodes, secondly by pure collusion nodes, and relatively greater by collusion nodes with servers. When the number of completed tasks increases, this decrease can be gradually eased. For example, when the number of tasks is 40, the success rate generated by our proposed scheme in Fig. 10(e) is 16% lower than that in Fig. 10(b); when the number of tasks is 100, the success

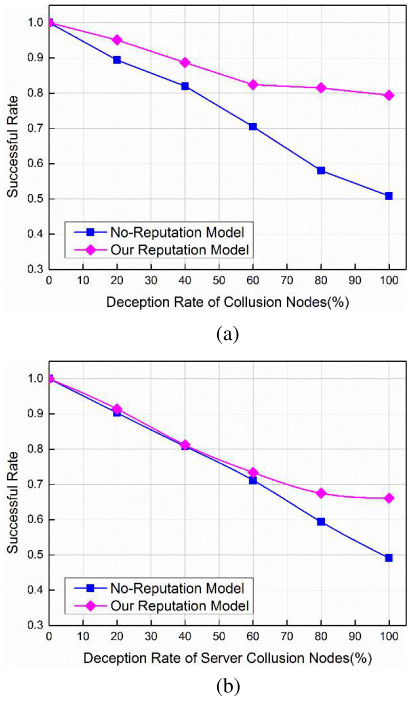


Fig. 9. Effect of deception rate on service success rate. (a) Deception rate of pure collusion nodes. (b) Deception rate of collusion nodes with servers.

rate generated by our proposed scheme in Fig. 10(e) is only 9% lower than that in Fig. 10(b). Therefore, it can be seen that our proposed scheme can effectively resist the deception behaviors of dishonest nodes.

3) Service Delay: In the section, we test the average task delay and the average distance between users and service nodes under different sizes of tasks and different ratios of malicious nodes. First, we test the average task completion delay according to our proposed scheme, the random allocation scheme, the traditional RRSA polling scheme, and the simulated cloud computing scheme. Fig. 11 shows that when all fog nodes are honest nodes, our proposed allocation scheme has obvious advantages over the random allocation scheme, the RRSA allocation scheme, and the simulated cloud computing scheme on the average completion delay of tasks. Compared with the random allocation and RRSA allocation schemes, this advantage becomes more obvious with the increase in the number of tasks. Compared with the simulated cloud computing scheme, this advantage becomes small with the increase in the number of tasks, which is because, with the increase in the number of tasks, the cloud server effectively reduces service delay by virtue of its powerful computing capability. Furthermore, we analyze the average completion delay of the tasks when the tasks are composed of 5, 10, and 15 subtasks, respectively. It can be clearly seen from Fig. 12 that the more the number of subtasks in a task, the greater the completion delay of the task. From another point, we can see that when the total number of subtasks is the same, the more the number of subtasks in each task, the shorter the average completion delay of the tasks. For example, in Fig. 12, when the number of subtasks in each task is 10 and the number of the tasks is 100 (the total number of subtasks is 1000), the average task completion delay is about

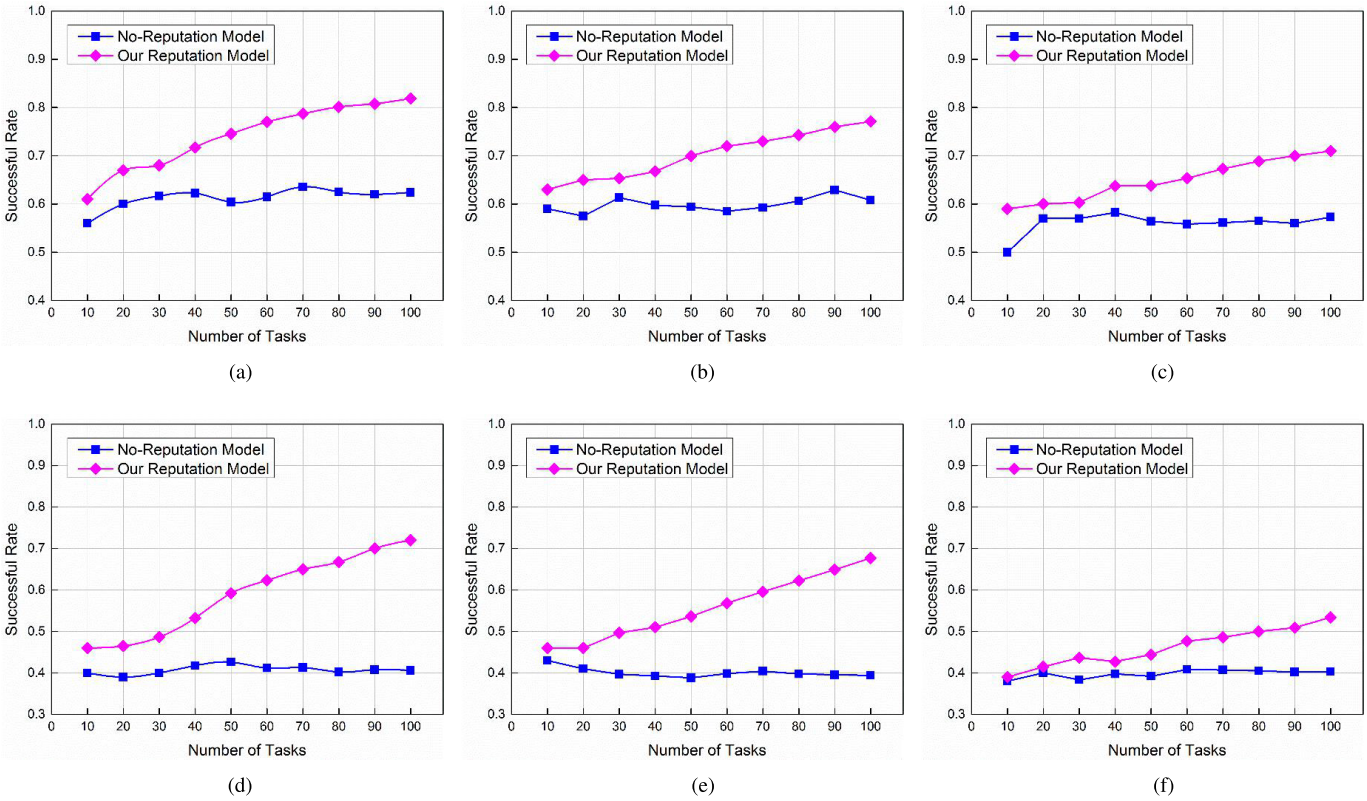


Fig. 10. Effect of proportions of dishonest nodes on service success rate. (a) Proportion of malicious nodes is 40%. (b) Proportion of pure collusion nodes is 40%. (c) Proportion of collusion nodes with servers is 40%. (d) Proportion of malicious nodes is 60%. (e) Proportion of pure collusion nodes is 60%. (f) Proportion of collusion nodes with servers is 60%.

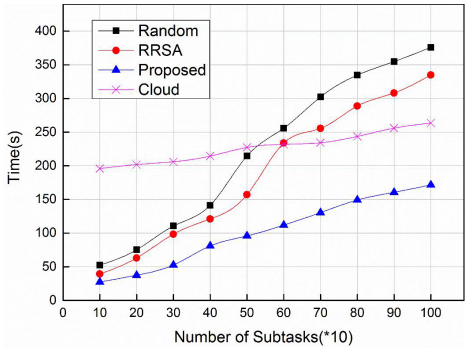


Fig. 11. Average time comparison of four schemes.

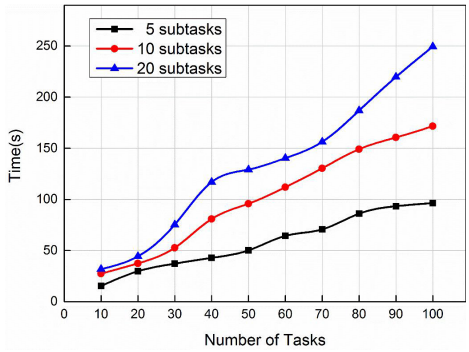


Fig. 12. Average time under different sizes of subtasks in each task.

171 s; when the number of subtasks in each task is 20 and the number of the tasks is 50 (the total number of subtasks is also 1000), the average task completion delay is only about 129 s. That is because the larger the size of subtasks in each task is, the more the subtasks may be allocated at one time. Therefore, our proposed scheme is more beneficial to the tasks that are composed of more subtasks.

When the proportions of dishonest nodes are set as 0%, 20%, and 40%, Figs. 13 and 14 show the average completion delay of tasks and the average distance between users and service nodes, respectively. Fig. 13 shows that when the proportion of malicious nodes increases, the average completion delay of tasks increases gradually. With the increase in the number of completed tasks, the ability of our proposed scheme to distinguish dishonest nodes is correspondingly increased. Thus, while more users can reject the service of the nodes with small reputation values, our proposed scheme can alleviate the increase in average task completion delay caused by the increase in malicious nodes. For example, in Fig. 13, when the proportions of malicious nodes are 20% and 40% respectively, the average service times are similar because our proposed scheme can distinguish dishonest nodes. Fig. 14 shows the changing trend of the average distance between users and service nodes. As the number of tasks increases, the average distance between users and service nodes will also increase. This may be because the increase in the number of tasks may lead to that the tasks need to be allocated to farther communities. On the other hand, when the proportion of malicious nodes increases, resource nodes in a community that can satisfy resource requirements and reputation requirements

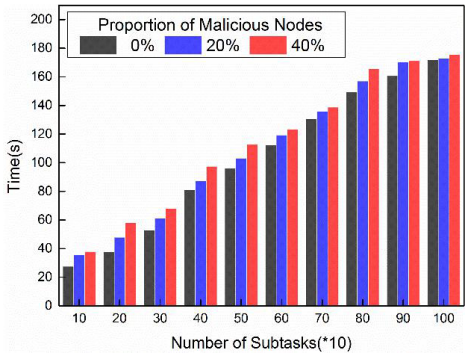


Fig. 13. Average time under different proportions of malicious nodes.

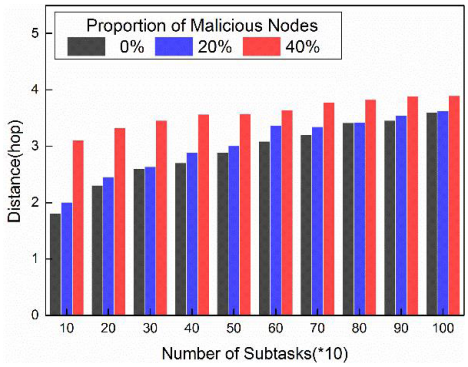


Fig. 14. Average service distance under different proportions of malicious nodes.

decrease; thus, the tasks also need to be allocated to farther communities. In addition, when the proportion of malicious nodes reaches 40%, it is difficult to complete the resource allocation of a task in two bottom communities. Thus, the average distance between users and service nodes is basically maintained at more than three hops. It can be seen from the abovementioned experiments that our proposed scheme can effectively resist the attacks of malicious nodes and collusion nodes. Therefore, our proposed scheme can enable users to obtain reliable resources and improves the service quality of fog computing resources.

VI. Conclusion

In fog computing, due to a large number of fog nodes, there are still many security problems in the computing process. When fog servers make resource allocation between users' tasks and fog nodes, some fog nodes may falsely claim that they have more resources than their actual ability, so as to get more task processing qualifications. Thus, it will greatly damage the interests of users and affect the quality of service. In this article, we propose a resource allocation scheme for community-based fog computing based on a reputation mechanism. When fog network provides computing services to users, we use a reputation mechanism to enable users to obtain reliable resources. In our proposed scheme, a user first submits his/her task request to the community-based fog network, and then, the fog server makes a reliable resource allocation process based on multiple-layer communities and reputation calculation. Based on our proposed experiments, our scheme enables users to obtain reliable computing resources. In our current work, we mainly provide a reputation mechanism to manage computing resources in fog nodes. Thus, for the malicious behaviors of a user, we only propose a simple approach to restrain the influence of malicious behaviors. If there are some malicious users defaming fog nodes in fog computing, then the malicious behaviors of the users may destroy the system. Therefore, it is also necessary to take certain measures to avoid more malicious behaviors of users. In future work, we consider making the reputation model for users.

References

[1] S. Bera, S. Misra, and J. J. P. C. Rodrigues, "Cloud computing applications for smart grid: A survey," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 5, pp. 1477–1494, May 2015.

[2] M. Mukherjee *et al.*, "Security and privacy in fog computing: Challenges," *IEEE Access*, vol. 5, pp. 19293–19304, 2017.

[3] F. Bonomi, "Connected vehicles, the Internet of Things, and fog computing," in *Proc. 8th ACM Int. Workshop Veh. Inter-Netw. (VANET)*, Las Vegas, VA, USA, 2011, pp. 13–15.

[4] I. Stojmenovic, "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks," in *Proc. Australas. Telecommun. Netw. Appl. Conf. (ATNAC)*, Nov. 2014, pp. 117–122.

[5] M. Aazam and E.-N. Huh, "Fog computing and smart gateway based communication for cloud of things," in *Proc. Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2014, pp. 464–470.

[6] S.-C. Hung, H. Hsu, S.-Y. Lien, and K.-C. Chen, "Architecture harmonization between cloud radio access networks and fog networks," *IEEE Access*, vol. 3, pp. 3019–3034, 2015.

[7] *OpenFog Reference Architecture for fog Computing*, document OPFRA001, OpenFog Consortium Archit. Working Group, 2017, vol. 20817, p. 162.

[8] V. Kochar and A. Sarkar, "Real time resource allocation on a dynamic two level symbiotic fog architecture," in *Proc. 6th Int. Symp. IEEE Embedded Comput. Syst. Design (ISED)*, Dec. 2016, pp. 49–55.

[9] J. Echaiz, J. R. Ardenghi, and G. R. Simari, "A novel algorithm for indirect reputation-based grid resource management," in *Proc. 19th Int. Symp. Comput. Archit. High Perform. Comput. (SBAC-PAD)*, Oct. 2007, pp. 151–158.

[10] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput. (MCC)*, 2012, pp. 13–16.

[11] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. Workshop Mobile Big Data*, 2015, pp. 37–42.

[12] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *Proc. Federated Conf. Comput. Sci. Inf. Syst. (FedCSIS)*, Sep. 2014, pp. 1–8.

[13] S. K. Datta, C. Bonnet, and J. Haerri, "Fog computing architecture to enable consumer centric Internet of Things services," in *Proc. Int. Symp. Consum. Electron. (ISCE)*, Jun. 2015, pp. 1–2.

[14] T. Nishio, R. Shinkuma, T. Takahashi, and N. B. Mandayam, "Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud," in *Proc. 1st Int. Workshop Mobile cloud Comput. Netw. (MobileCloud)*, 2013, pp. 19–26.

[15] E. C. Pinto Neto, G. Callou, and F. Aires, "An algorithm to optimise the load distribution of fog environments," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2017, pp. 1292–1297.

[16] H. Zhang, Y. Zhang, Y. Gu, D. Niyato, and Z. Han, "A hierarchical game framework for resource management in fog computing," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 52–57, Aug. 2017.

[17] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 12, pp. 7821–7826, Jun. 2002.

[18] S. Filiposka, A. Mishev, and K. Gilly, "Community-based allocation and migration strategies for fog computing," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6.

[19] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decis. Support Syst.*, vol. 43, no. 2, pp. 618–644, Mar. 2007.

[20] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "A survey on trust and reputation models for Web services: Single, composite, and communities," *Decis. Support Syst.*, vol. 74, pp. 121–134, Jun. 2015.

[21] N. Nagarathna and M. Indiramma, "Recommendation framework for service-oriented grid," in *Proc. 4th Int. Conf. Adv. Comput. (ICoAC)*, Dec. 2012, pp. 1–6.
[22] J. Tian, P. Yuan, and Y. Lu, "Security for resource allocation based on trust and reputation in computational economy model for grid," in *Proc. 4th Int. Conf. IEEE Frontier Comput. Sci. Technol. (FCST)*, Dec. 2009, pp. 339–345.
[23] H. Wei, Y. Zhang, D. Guo, and X. Wei, "CARISON: A community and reputation based incentive scheme for opportunistic networks," in *Proc. 5th Int. Conf. Instrum. Meas., Comput., Commun. Control (IMCCC)*, Sep. 2015, pp. 1398–1403.
[24] Z. Shi, J. Wei, X. Wei, K. Tan, and Zhilian, "The task allocation model based on reputation for the heterogeneous multi-robot collaboration system," in *Proc. 8th World Congr. Intell. Control Autom.*, Jul. 2010, pp. 6642–6647.
[25] M. Wang, G. Wang, J. Tian, H. Zhang, and Y. Zhang, "An accurate and multi-faceted reputation scheme for cloud computing," *Procedia Comput. Sci.*, vol. 34, pp. 466–473, Jan. 2014.
[26] H. Lin, L. Xu, Y. Mu, and W. Wu, "A reliable recommendation and privacy-preserving based cross-layer reputation mechanism for mobile cloud computing," *Future Gener. Comput. Syst.*, vol. 52, pp. 125–136, Nov. 2015.
[27] S. Wang, L. Huang, C.-H. Hsu, and F. Yang, "Collaboration reputation for trustworthy Web service selection in social networks," *J. Comput. Syst. Sci.*, vol. 82, no. 1, pp. 130–143, Feb. 2016.
[28] X. Xu and H. Yu, "A game theory approach to fair and efficient resource allocation in cloud computing," *Math. Problems Eng.*, vol. 2014, pp. 1–14, Apr. 2014.
[29] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Exper.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.
[30] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *Proc. 3rd IEEE Workshop Hot Topics Web Syst. Technol. (HotWeb)*, Nov. 2015, pp. 73–78.

Ke Gu (Member, IEEE) received the Ph.D. degree from the School of Information Science and Engineering, Central South University, Changsha, China, in 2012.

In 2013, he joined the School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha, where he is currently an Associate Professor. He has authored or coauthored more than 70 research articles in journals or conferences. His research interests include cryptography, and network and information security.

Linyu Tang received the master's degree from the School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha, China, in 2019.

Her research interest includes network and information security.

Jiafu Jiang is currently a Full Professor with the Changsha University of Science and Technology, Changsha, China.

He has authored or coauthored more than 50 research articles in journals or conferences. His research interests include data mining, image process, network, and information security.

Weijia Jia (Fellow, IEEE) received the Ph.D. degree in computer science from the Polytechnic Faculty of Mons, Mons, Belgium, in 1993. He is currently a Full Professor with the Department of Computer and Information Science, University of Macau, Macau, China.

He has authored or coauthored more than 100 research articles in famous journals or conferences. His research interests include next-generation wireless communication, distributed systems, and multicast and anycast routing protocols.