J Inf Process Syst, Vol.14, No.2, pp.377~395, April 2018
https://doi.org/10.3745/JIPS.03.0091

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Dynamic Cloud Resource Reservation Model Based on Trust

Jiao-Hong Qiang*,**, Ding-Wan Ning*, Tian-Jun Feng**, and Li-Wei Ping***

Abstract

Aiming at the problem of service reliability in resource reservation in cloud computing environments, a model of dynamic cloud resource reservation based on trust is proposed. A domain-specific cloud management architecture is designed in which resources are divided into different management domains according to the types of service for easier management. A dynamic resource reservation mechanism (DRRM) is used to test users' reservation requests and reserve resources for users. According to user preference, several resources are chosen to be candidate resources by fuzzy cluster analysis. The fuzzy evaluation method and a two-way trust evaluation mechanism are adopted to improve the availability and credibility of the model. An analysis and simulation experiments show that this model can increase the flexibility of resource reservation and improve user satisfaction.

Keywords

Cloud-Domain-Based Management Architecture, Decision of Candidate Resources, Dynamic Resource Reservation, Two-Way Trust Evaluation Mechanism

1. Introduction

Cloud computing [1,2] has begun to move from conception to application, and its convenience and economic advantages are attracting more and more individuals and businesses. However, cloud computing [3,4] still faces various problems with regard to the issue of security. Providing cloud services in a reliable way (such as high quality and high reliability) is one of the research hot spots in the field of cloud security [5]. In order to guarantee the quality of service, the current solution is mostly to optimize resource allocation and reservation management [6,7], but there is little consideration of the credibility of resources. Research into a trusted cloud resource reservation mechanism has important theoretical and practical significance for the effective organization of cloud resources and their utilization.

In the cloud computing environment, trust should be bidirectional. On the one hand, cloud users trust all types of services provided by cloud users. On the other hand, service providers trust users, that is, cloud service providers believe that users can use services without malicious behavior. Most of the

Manuscript received January 9, 2017; first revision Feburary 8, 2017; accepted April 10, 2017.
Corresponding Author: Li-Wei Ping (11065887@qq.com)
* College of Information Engineering, Handan College, Handan, China (jhq1983@163.com)
** School of Computer Science and Technology, Hebei University, Baoding, China (tianjunfeng@hbu.edu.cn)
*** School of Economics and Management, Beijing Jiaotong University, Beijing, China (11065887@qq.com)

current research studies focused on cloud service provider trust (mainly to the provided services the trust evaluation). However, these are not sufficient. Cloud users may hold a trusted identity, but their behavior may be unreliable. Thus, estimates of user credibility need to be carried out to effectively control the insecurity factors in cloud computing from users. High standards of service quality are required in application fields of the cloud. One way to satisfy this requirement is resource reservation, which indicates the ability of the cloud to ensure that users obtain the required resources. Therefore, it is of theoretical and practical significance to study the trust mechanism of cloud resource reservation for the effective organization of cloud resources.

The rest of this paper is organized as follows: Section 2 discusses the related work that has already been carried out on this subject. Section 3 introduces the logical architecture of cloud resource organization based on a management domain. The candidate service resource selection strategy, resource reservation strategy, and trust management strategy are examined in Sections 4, 5, and 6, respectively. Section 7 describes our experimental setup and presents a simulation and result analysis. Section 8 concludes the paper.

2. Related Work

Cloud computing is the evolution of parallel computing, distributed computing, and grid computing, social networks, search engines, Web 2.0 technology, and virtualization technology. Cloud computing is the further development of grid computing. From the characteristics of Internet applications, both are consistent. They support applications on the Internet and solve the problem of heterogeneous resource sharing and other issues. The current research about cloud resource reservation is not sufficient, whereas theories about the resources of the grid have resulted in a wealth of research results.

The authors [8,9] presented the general process of grid resource reservation, namely the resource management system of user reservation requests for admission tests. The reservation task is created with the appropriate resources if the request is passed; otherwise, the server will reject the user's reservation request. At present, most of the resource reservation mechanisms in the grid environment function as follows: if there are some resources to meet certain requirements, then the application will be inserted into the reservation task queue according to a certain strategy. That is, the allocation of resources is completed before the user actually uses the resource. This is called static resource reservation. When the reservation queue changes, it is difficult to control the impact of the utilization of resources. In a cloud computing environment, the dynamic joining or withdrawal of service providers and the uncertainty of the arrival time of a user's request make the static reservation mechanism appear to be inadequate. Thus, a dynamic reservation mechanism is a necessary choice. In [10], a dynamic grid resource reservation mechanism is proposed. This mechanism is based on an acceptance test to determine whether to accept the reservation request, and to use the resource allocation. However, it does not have a detailed description of how to select the candidate resource, and also has no evaluation of the trust of both the supply and demand sides. Wu and He [11] proposed a hierarchical trust model based on a domain according to the characteristic of grid technology. Using this model to deal with the trust relationship between entities in the grid, the trust in the management domain is divided into a relationship in the domain and among the domains. The update mechanism of the trust value is improved. However, this model does not explain the basis of the division of the domain, and the trust

value of the nodes in the domain is not initialized.

Tang and Chen [12] presented a quantitative description, type, and evaluation mechanism of trust by using the fuzzy set theory for modeling trust management and a formal derivation of the trust relationship. In the reference, the tree structure is used to describe the type of trust, which is called a concept tree. When the height of the concept tree is 1, the process of evaluation is a simple process of fuzzy comprehensive evaluation. Cheng and Pan [13] presented a dynamic resource allocation strategy mechanism based on continuous double auction (CDA) and node trust in the cloud computing environment. The bidding strategy for the supply and demand sides of the resources, and the failure rules of the resource, are given. The dynamic resource allocation model achieved only a balance of supply and demand according to the pricing strategy of buyers and sellers. It did not take into account the preference issue of the resource demander for resources. There exists a resource waiting deadlock under the fair scheduler when the resource requisition of applications is beyond the amount that the cluster can provide. Thus, Yao et al. [14] developed a new admission control mechanism that dynamically reserves resources for processing tasks in order to avoid resource waiting deadlocks.

Ghribi and Zeghlache [15] presented a new graph-coloring model for advance resource reservation with minimum energy consumption in heterogeneous IaaS cloud data centers. Shi et al. [16] used the Linear Predicting Method (LPM) and the Flat Period Reservation-Reduced Method (FPRRM) to obtain useful information from the resource utilization log. Their research reduced the response time and energy consumption of the M/M/1 queuing theory predicting method for developing efficient energy-saving methods to reduce the huge energy consumption in the cloud data center. However, the authors [14–16] did not take into account the trust factor.

Based on the above analysis, we proposed a model of dynamic cloud resource reservation based on a trust evaluation strategy. The work includes the following aspects:

(1) Cloud computing resources are divided into different administrative domains according to function for easy service organization distribution and centralized management. It can also reduce the user search service cost and configure the domain volume of resources by using the Bernoulli large-number theorem.

(2) The use of a dynamic resource reservation strategy of user reservation requests for admission tests and reservations effectively buffers the impact of a cloud user's dynamic accesses.

(3) The service will be classified according to user preferences. We choose a service with a higher reputation according to users' preferences as the candidate, and shunt the users. At the same time, we can reduce resource competition and improve resource utilization as much as possible.

(4) The trust is subjective, and subjective trust itself has fuzzy characteristics. We adopt a fuzzy evaluation method for trust evaluation.

3. Logical Architecture of Cloud Resource Organization Based on Management Domain

Cloud computing amasses various computing resources, storage resources, and hardware and software resources to form a large-scale resource pool. Cloud computing can provide users with unprecedented computing power and highly reliable services. Users can use its IT services anytime and anywhere.

In order to reduce the complexity of resource management and trust management in a cloud computing system, in this study, the service of a cloud computing network environment is divided into management domains according to the application function. We take this logical architecture as the logic basis of trust management, service entity management, and resource reservation. The logical architecture is shown in Fig. 1.

The framework includes an access control center and management domain. The management domain contains a trust agent, service entity agent, and service entity.

Access control center: The identity management for accessing users, such as user registration, cancellation, and identity authentication. In order to highlight the research work of this paper, the access control center functions are guaranteed through the use of appropriate technology.

Management domain: Different services are divided into different logical domains according to their functions. The services in each management domain have the same service type, but there are some differences between the attributes of service. The same type of service will be gathered in the same management domain. This has two advantages:

(1) It is convenient for centralized management services for the same function, and meanwhile it reduces management costs;

(2) From the perspective of the user, it reduces not only the difficulty of users to buy services but also the search costs.

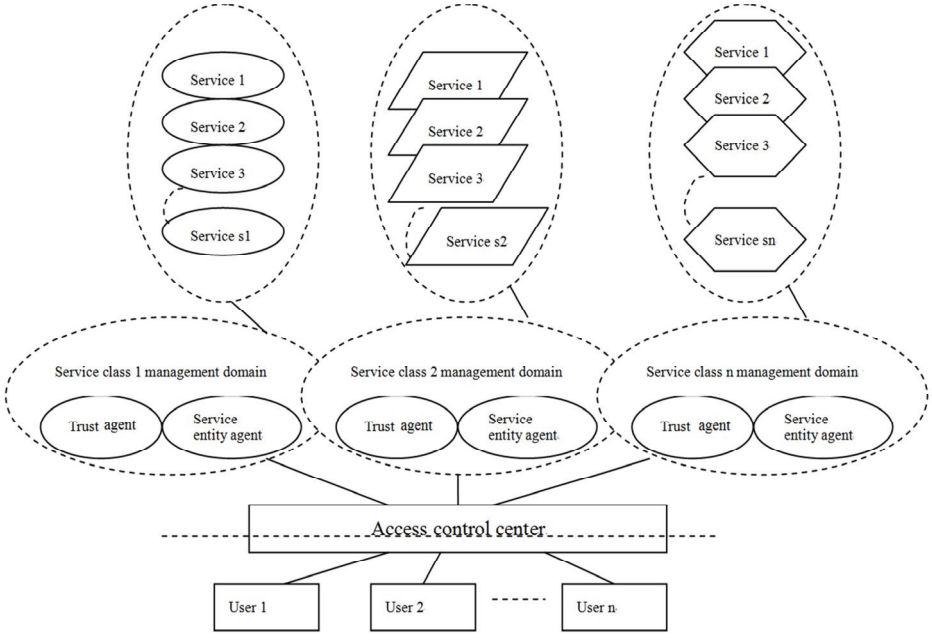


Fig. 1. Cloud-domain-based management logical architecture.

Set two agents as a management domain-trust management agent and service entity agent:

(1) Trust management agent: collects, calculates, and stores the credit information of users and service entities. It maintains a trust table of the user and service entity. Dynamically updates the trust value.

(2) Service entity management agent: accepts and dynamically preprocesses user reservation requests, and manages the entrance and exit of service entities.

Service entity: provides a variety of services for users of resources as a service entity. A service entity corresponds to a resource.

In the paper, the proposed model is based on the following assumptions:

(1) The user can request a plurality of service entities, but each of the user's tasks is independent of the others;

(2) User tasks are executed sequentially on the service entity, one at a time;

(3) A non-reservation task can start from scratch when needed after an interruption.

The cloud of each service is classified by a function domain that meets the needs of users and guarantees the premise of the resource utilization rate. Bernoulli's large-number theorem is used to determine the number of service entities in each domain.

Bernoulli's large-number theorem shows that the probability of occurrence of an event in terms of probability converges to the event. This theorem expresses the frequency stability in a strictly mathematical form. That is, when the number of repeated trials is large, the probability of occurrence of the event with a large probability is very small. From the actual inference principle, in practice, when the number of tests is large, we can use the frequency of events instead of the probability of the event.

User demand behavior in real life often has some type of regularity. We use a discrete random variable X to indicate the total amount of user needs. Based on historical information, with the time cycle of T, the total value of the user demand is collected. Assuming that the X value is x_k (k = 1, 2, …), the distribution of X is P(X = k) = p_k, k = 1, 2, … p_k. According to the Bernoulli large-number theorem, when the number of tests is large, the occurrence frequency of the event can be used to replace the probability of the event. Therefore, the distribution of X values can be tested and determined by Bernoulli's large-number theorem. The distribution law of X is as follows:

X	x_1	x_2	x_3	...
p_k	p_1	p_2	p_3	…

We can obtain the mathematical expectation of the total demand of users in each cycle T as $E(X) = \sum_{k=1} x_k p_k$ and the variance as $D(X) = \sum_{k=1} [x_k - E(X)]^2 p_k$. We can estimate the total demand of users in each cycle T.

According to the total demand of users in each cycle T, in the assurance of a resource utilization situation, we determine the number of required service entities. Set a certain threshold for the utilization rate of resources. Only when the rate is greater than or equal to the threshold do we put it into the scope of the analysis.

We use a discrete random variable Y to indicate the number of service entities. The possible values for Y are y_t (t = 1, 2, …). According to Bernoulli's large-number theorem, the distribution of Y is P(Y = t) = p_t, and the values of t = 1, 2, … p_k are determined by a Bernoulli's large-number theorem experiment.

The distribution law of X is as follows:

Y	y_1	y_2	y_3	...
P_t	p_1	p_2	p_3	...

Then, the mathematical expectation of the number of service entities satisfying a certain resource utilization threshold is $E(Y) = \sum_{k=1} y_t p_t$, and the variance is $D(Y) = \sum_{k=1} [y_k - E(Y)]^2 p_k$.

According to this, the number of service entities that satisfy the total demand of each cycle T of the user is estimated.

4. Candidate Service Resource Selection Strategy

DEFINITION 1. Direct trust: subject A (user or service entity) and subject B made a transaction. After the trust evaluation, A gets a trust value about B.

DEFINITION 2. Recommendation trust: subject A accepts the direct trust of the other subjects (such as subject C) provided by B.

Different users have their own preferences (such as reliability, quality of service, and price), and these preferences will affect the user's trust in the service.

A clustering analysis can be used to select a service entity that is close to the user's preference. On this basis, we can select the service entity that satisfies a certain threshold of trust as the candidate resource of the user by combining the direct trust and the recommend trust. This enhances the ability of meeting the needs of the user.

When the user has a reservation request, the user first sends a reservation request to the cloud. The service entity agent in the corresponding management domain receives the user request. Then, it processes the user request and determines the candidate resource that meets the needs of the user. The process is shown in Fig. 2.

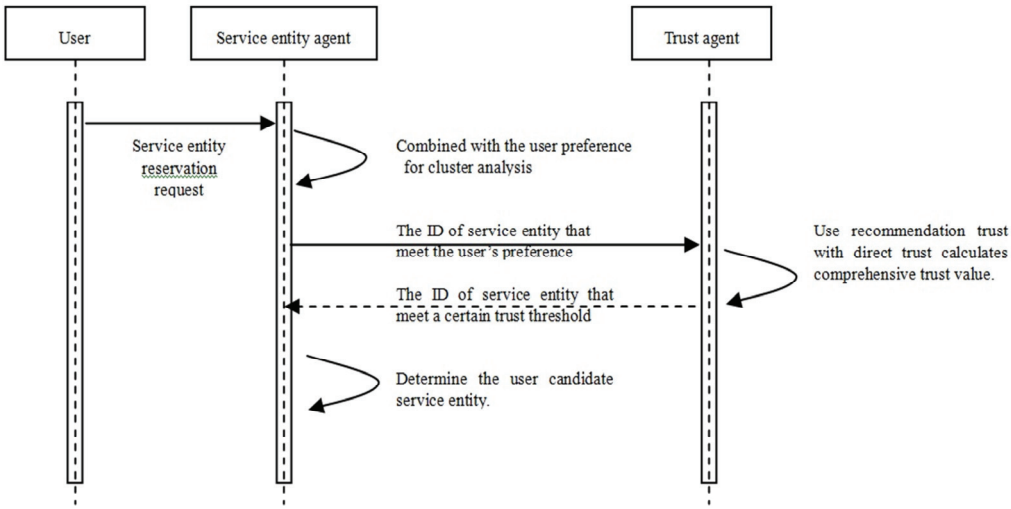


Fig. 2. Sequence diagram of determine the user's candidate resources.

4.1 Resource Clustering Based on User Preferences

In real life, we often classify things according to their properties or characteristics. A cluster analysis is a specific requirement to classify the mathematical methods of things. This study considers that different users may have different preferences for their choice of services. For example, some users prefer high-quality services, and some users prefer low-priced services. It is necessary to classify the service according to the user's preference, and to select the service closest to the user's preference as the candidate resource.

We use a service entity vector that meets the basic conditions and denote it as $A = [a_1, a_2, \ldots a_n]$. Each entity's corresponding M service reputation evaluation attributes are denoted as $a_i = [a_{i1}, a_{i2}, \ldots a_{im}]$, ($i = 1, 2, \ldots m$). We can obtain a reputation evaluation primitive vector:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}$$

Among them, a_{nm} denotes the original data of the n_{th} service entity m_{th}'s reputation evaluation attribute. Because the units and dimensions of the reputation evaluation attributes of service entities may not be the same, this leads to a situation in which the attribute with a large absolute value occupies the leading role, and a small absolute value does not work. In order to avoid this situation, we use the method of linear range transformation to standardize the original data.

$$a'_{ij} = \frac{a_{ij} - \min\limits_{1 \le k \le n}(a_{kj})}{\max\limits_{1 \le k \le n}(a_{kj}) - \min\limits_{1 \le i \le n}(a_{kj})} (i = 1,2,\ldots, n; j = 1,2,\ldots, m) \tag{1}$$

This method can eliminate the effects of the dimensions. Among them, $a'ij \in [0,1]$, and we can obtain the standardized evaluation matrix R_s:

$$R_s = \begin{bmatrix} a'_{11} & a'_{12} & \cdots & a'_{1m} \\ a'_{21} & a'_{22} & \cdots & a'_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ a'_{n1} & a'_{n2} & \cdots & a'_{nm} \end{bmatrix}$$

In order to facilitate the expression of the different preferences of the user with regard to the service entities' reputation attributes, we introduce a user preference vector, denoted as $w = [w_1, w_2, \ldots w_m]$, $w_i \in [0,1]$, and $\sum_{i=1}^{m} w_i = 1$.

Using the weighted absolute value method to compute a fuzzy similar matrix:

$$r_{ij} = 1 - c\sum_{k=1}^{m} w_k \left| a'_{ik} - a'_{jk} \right| (i, j \in [1, n], k \in [1, m]) \tag{2}$$

A fuzzy similar matrix is obtained:

$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nn} \end{bmatrix}$$

According to Theorem 3.3 in [18], we prove that R is a fuzzy equivalent matrix.

Cluster the matrix R with the transitive closure method. Take $ë$ from 1 to 0, and cut the equivalence relationship $R_{ë}$. Thus, all service entities A are classified with $ë$ from 1 to 0. The resulting classification is gradually merged to form a clustering graph.

The clustering figure presents a classification of different $ë$ values and forms a dynamic clustering. For a reasonable threshold, we use the F statistic for selecting the optimum $ë$.

After the classification, we choose a class similar to that of the user's preferences. We use the average value method to make a judgment. The value corresponding to the k_{th} class C_k is represented by Q_k.

$$Q_k = AVG(C_k) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}\omega_j \tag{3}$$

The character n denotes the number of k_{th} service entity classifications. w_j denotes the user preference weight of the j_{th} service reputation evaluation attribute, and $a_{ij} \in C_k$ denotes that the reputation evaluation attribute value of the j_{th} service provided by the i_{th} service entity in k_{th} classifications. The larger the value of Q_k, the closer the classification is to the user's preferences, and the better the services in this class satisfy the user's preferences.

4.2 Candidate Resource Choice Based on Trust Value

In the management domain, the trust value of a resource is denoted by $T_u^s \in [0,1]$. For unused resources, the initial trust value is set to $T_u^s = 0.5$.

Through a cluster analysis, we obtain the service entity that satisfies the user's preferences. First, the trust value of the service entity is calculated, and the service entity satisfying a certain trust threshold is used as the candidate resource.

With regard to the direct trust of user i to service entity j, taking into account the time decay, the number of interactions and other factors, the calculation formula is as follows:

$$T_{ud}^s(i,j) = \begin{cases} 0.5 & m=0 \\ \eta \times f(t) \times T_d(i,j) & m>0 \end{cases} \tag{4}$$

T_d is the trust of user i to service entity j in the last transaction, $T_{ud}^s(i,j)$ is the updated trust value, η is the number of interactions of the influence factor, and $\eta=\sqrt{n/m+1}$. n indicates that the entity service of the trust value does not decrease in the interaction process of m times. m is the total number

of transactions. $f(t)$ it a time decay function that uses the exponential decay $f(t) = e^{-[(t-t_0)/T_0]}$. If the trust value is lower than 0.5 after attenuation, we set it back to 0.5. Among them, t is the transaction time, t_0 is the last trading time, and T_0 is the interval. The closer the transaction is to the time of the transaction, the greater the impact on the credibility.

The time attenuation and the number of interactions can also affect the result; thus, we must pay attention to them when we measure the direct trust of user i to user r. The calculation formula $T_{ud}^{u}(i,j)$ can be the same as $T_{ud}^{s}(i,j)$.

In addition to direct trust, users should consider the trust of other users. When selecting service entities, the user will refer to the evaluation information of other users with regard to the service entities.

In the initial case, a newly added user does not have any trust interaction with other existing users, so there is no trust relationship with other users. The initial trust value for the user is set to 0.5, and the trust for the user in other domains is also referenced. We can calculate the trust value of this user in the domain.

User i references the trust of the service entity of user r ($r \neq i$). We calculate the recommendation trust of user i to service entity j:

$$T_{ur}^{s}(i,j) = \frac{\sum_r T_{ud}^{u}(i,r) \times T_{ud}^{s}(r,j)}{\sum_r T_{ud}^{u}(i,r)} \tag{5}$$

The final trust of user i to service entity j is

$$T_f(i,j) = \zeta T_{ud}^{s}(i,j) + (1-\zeta) T_{ur}^{s}(i,j) \tag{6}$$

The character æ is a confidence factor. Its value is related to time. $\zeta = 1 - \mu^m$ $\mu \in [0,1]$ and μ denote the trust weight of users for their own history, while m is the number of transactions.

The service entity satisfying the user's preference is obtained by a cluster analysis. After the calculation of the trust value, the service entity satisfying a certain threshold is taken as the candidate resource of the user.

5. Resource Reservation Strategy

The resource reservation strategy is based on assumptions made in Section 3. The user may appear on a shortage of estimates of resources in reserving resources, and then allows the user to dynamically apply the resources in the use process. The prerequisite is that there are free resources to meet the user's requirements. Otherwise, the dynamic applications will fail. Dynamic application is also conducted based on the fact that "a task can only invoke a single resource once".

A 3-tuple G<S, Q, E> denotes the service entities-reservation requests graph, where S denotes the set of all service entity nodes and corresponds to all service entity in a management domain; Q represents a collection of preserved nodes, the corresponding management in the domain of service entities of a reservation requests a service entity. If a service entity node s_i can satisfy the request of q_j, it constitute an edge of e_{ij}, and $e_{ij} \in E$.

The range of candidate resources is defined for each reservation request, and the side resources are indicated as candidate resources satisfying the user's task. In this way, the change in a service entity and reservation request in a management domain corresponds to the change of nodes and edges in the graph, and the relation between the service entity and reserved tasks is shown in this figure.

A bipartite graph between the resource and reservation request is shown in Fig. 3.

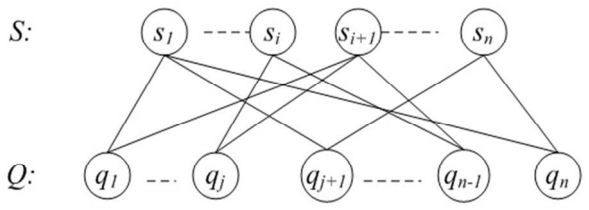


Fig. 3. Service entities: reservation requests.

When a reservation request for new users is issued, this request shall be added to the request list in accordance with the time it is issued. If two requests occur at the same time, the one with a higher credit value goes first, and then the one with a shorter execution time takes precedence. A new node is added to the reservation chart, which means a new reservation is accepted. This may affect previously accepted reservation requests and cause resource allocation conflicts.

Let us say that in the reserved graph, the position of the new joined node is k, namely the reservation request q_k has a certain candidate resource $\ddot{e}(q_k)$. If one of its resource s_j is also one of candidate resource q_i of preserved nodes in its former intersection set at the same time, s_j will be called a conflict resource. The former intersection set is a set of reserved nodes that intersects the q_k in the reserved window and the node number is before k, it is written as $\alpha(q_k)$. We call this case a conflict between q_k and q_i on s_j.

Because the premise sets a user task execution order of each resource, and q_i starts earlier, it will have the priority to occupy s_j, thus depriving q_k's use of the resources. Every time a reservation conflict occurs, the available resource of q_k will decrease by 1. In the former intersection of q_k, only one reservation task will be the actual scheduling of a conflict of resources. Thus, the actual maximum number of collisions with respect to the same conflict resource is 1. Similarly, for a request q_k and a reserved node with multiple resource conflicts, the maximum number of conflicts is also 1.

The actual maximum conflict number of q_k's candidate resources is

$$C_r = \sum_{s_j \in \lambda(q_k)} sng(\sum_{q_i \in \alpha(q_k)} c_{ij}) \tag{7}$$

The actual maximum conflict number between q_k and its former intersection is

$$C_i = \sum_{q_i \in \alpha(q_k)} sng(\sum_{s_j \in \lambda(q_k)} c_{ij}) \tag{8}$$

The freedom degree of q_k is

$$freeDeg(q_k) = \deg ree(q_k) - \min(C_r, C_i) \tag{9}$$

c_{ij} is the value of e_{ij}, and

$$e_{ij} = \begin{cases} 1, & if\ c_{ij}\ exists \\ 0, & if\ c_{ij}\ does\ not\ exist \end{cases}$$

$degree(q_k) = \Sigma N\ j = 1.\ c_{ij}$ is the degree of the node, and $\ddot{e}(q_k)$ is a candidate resource set that meets entity node q_k's resource requirements.

If $freeDeg\ (q_k) \cdot \prod_{q_m \in \beta(q_k)} freeDeg\ (q_m) > 0$, $\beta(q_k)$ is the set that intersects with the request q_k in the reserved window, and the node number is after k of the intersection set, then we accept the reservation request. The request q_k will be added to the graph, and the graph is updated with the reservation. Otherwise, we reject the reservation request and delete the corresponding content of the request q_k in the reservation list.

The user reservation request processing procedure is as follows:

1. IF the user's reputation value ≧ reputation threshold && user requests the reserved time $\in [T_C + T_1, T_C + T_2]$ && reservation time section of the user request reservation work ≦ the pre-evaluation time of the task execution
2. {
3. Determine the candidate user resource reservation requests;
4. Use service entities-reservation requests graph to conduct acceptance test;
5. *IF the test passed*
6. Accept the user's request;
7. *ELSE*
8. Reject the user's request;
9. }
10. *ELSE* Reject the user's request;

In step 1, using the current time T_C for reference, the user can reserve resources in the $[T_1, T_2]$ time section.

In step 1, because some users may not be able to accurately determine the time required for their own resources, we need to pre-evaluate the execution time of users' tasks. This will avoid users' prolonged use time when they are using resources, which can result in later users waiting too long.

There are two ways to define how a user uses services: (1) all-inclusive, where users submit only the required data processing and data-processing-required conditions (such as software, hardware, and platform) by the execution after the cloud, and the cloud returns the results to the user; and (2) half-inclusive, where users own some of the facilities and use some of the services provided by the cloud (such as platforms) to perform their tasks.

For these two different modes, the method to estimate the running time is different. The cloud uses its own software to run user tasks, and the user needs only to submit the scale attributes related to the tasks. The time and space required by the user tasks are calculated through execution algorithms, which are denoted as $O(n)*t$ and $S(n)*p$. The parameters t and p represent time and space units, respectively, and n is the task scale. Because the users use the software to run their own tasks, users need to submit tasks related to scale properties. This also requires the user to submit a software implementation of the algorithm's time and space complexity to calculate the time and space required to perform user tasks.

The pre-evaluation data of the running time of a user task are compared with the time reserved by the user application. If the user reservation request is rejected or the user application is returned for the reservation request at higher than the evaluation running time, the user is required to increase the reservation time.

6. Candidate Service Resource Selection Strategy

6.1 Trust Evaluation of User to Service Entity

Users use the fuzzy evaluation method to evaluate the trust of a resource after using the resource. First, the membership degree of the fuzzy set is determined. The methods of determining whether the membership function can be used are the fuzzy statistical method, example method, and expert experiences.

On the comprehensive evaluation of trust, we need to consider the following four factors: service reputation evaluation attribute set $E = \{re1, re2 \ldots re_n\}$, for example, E = {price, quality, timeliness}; the service entity evaluation set $D = \{d_1, d_2, \ldots d_m\}$, for example, D = {not credible, critical credible, generally credible, very credible, absolutely credible}; reputation evaluation attribute evaluation matrix $R_d(r_{ij})_{n\times M}$; and the weight allocation of each reputation evaluation attribute $W = (w_1, w_2, \ldots w_n)$.

Service reputation evaluation attribute set E contains all attributes that constitute the trust of the entity. The evaluation set D is composed of different grades of evaluation for the specific subject attribute. The evaluation grade number is M. r_{ij} indicates the possibility for making d_j evaluations of the reputation evaluation attribute re_i. According to the membership function that calculates the reputation evaluation attributes for the evaluation of the degree of membership, in order to obtain the membership matrix R_d,

$$R_d = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ r_{21} & r_{22} & \cdots & r_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nm} \end{bmatrix}$$

W is a weight distribution vector that represents the different preferences for each reputation evaluation attribute of the user. Use the vector $W = [w_1, w_2, \ldots w_n]$ to denote the weight distribution vector, $w_i \in [0,1]$, and $\sum_{i=1}^{n} w_i = 1$.

We use the weigh value calculated from the final membership degree vector

$$F = w \circ R_d = [f_1, f_2, \cdots, f_m]$$

F is just a fuzzy vector, while the calculated value for the evaluation of service entities is an exact number. Use the following method to calculate the user i for the service entity:

$$T_{uNew}^{s}(i,j) = \frac{\sum_{k=1}^{m} w_k f_{sk}}{\sum_{k=1}^{m} f_{sk}} \tag{10}$$

Next, we take incentive measures to update the user i's trust value to the service entity j. Either the user or the service entity, its behavior trust with characteristics of "rise slow" and "fast down," namely to achieve high trust values requires a long process. However, fraud will lead to quickly reduced trust. Thus, we introduce an incentive factor. At the same time, according to the user before trust for the service value of the entity, to determine whether the $T_{uNew}^{s}(i,j) > \tau$. If the inequality holds, it express

that the trust value is not decreased. We should take incentives. The user *i* to the service entity *j* trust updating formula as follows:

$$T''^{s}_{ud}(i,j) = T'^{s}_{ud}(i,j) + \mu \cdot (T^{s}_{uNew}(i,j) - \tau) \cdot e^{-(T^{s}_{uNew}(i,j)-\tau)} \tag{11}$$

$T'^{s}_{ud}(i,j)$ is the trust evaluation value before the transaction by the time decay for user *i* and service entity *j*. $(T^{s}_{uNew}(i,j)-\tau)e^{-(T^{s}_{uNew}(i,j)-\tau)}$ is the incentive factor, which is used to adjust the service entity's trust value after the transaction. *i* is influence factor of the space time complexity and transaction importance, and $T''^{s}_{ud}(i,j)$ is the updated trust value.

According to the evaluation of users for the credence property service entity, the evaluated value of the credit attribute of each service entity should be adjusted dynamically to make it more reasonable and consistent with the actual situation. Taking the service quality as an example, when many users claim that the description of service quality is incompatible with reality, it is necessary to reduce the credibility of its quality of service.

6.2 Trust Evaluation of Users to Recommend Users

User *i* evaluates service entity *j* after using the services, and judges whether $\left|T''^{s}_{ud}(i,j) - T'^{s}_{ud}(r,j)\right| - \theta > 0$. If it is true, the evaluation of two parties on the same service entity information is not close. If the evaluation information is close to the user *r*'s evaluation information, the trust relationship of user *i* to user *r* is to be set. We can establish the trust relationship and implement the recommendation of service entities between users. The user-*i*-to-user-*r* trust evaluation value update formula is as follows:

$$T''^{u}_{ud}(i,r) = T'^{u}_{ud}(i,r) - \mu \times (\left|T''^{s}_{ud}(i,j) - T'^{s}_{ud}(r,j)\right| - \theta) e^{\left|T''^{s}_{ud}(i,j) - T'^{s}_{ud}(r,j)\right| - \theta} \tag{12}$$

$T'^{u}_{ud}(i,r)$ is the direct trust of user *i* to user *r* after time attenuation, $T''^{s}_{ud}(i,j)$ is the direct trust of user *i* to user *r* after the transaction, $T'^{s}_{ud}(r,j)$ is the direct trust of user *r* to the services entity after time attenuation, and $(\left|T''^{s}_{ud}(i,j) - T'^{s}_{ud}(r,j)\right| - \theta) e^{\left|T''^{s}_{ud}(i,j) - T'^{s}_{ud}(r,j)\right| - \theta}$ is the incentive factor used to adjust the trust of user *i* to user *r* after the transaction. The parameter *i* is the influence factor of the space time complexity and transaction importance.

6.3 Trust Evaluation of Cloud to Service Entity

In this study, the user evaluates the trust about the service in the cloud service entity, and the cloud services agency evaluates the user's behavior.

When the user's comprehensive trust value is lower than a certain threshold of trust, this domain service entity is no longer providing services to the user.

With regard to the evaluation of trust, many factors may need to be considered, including the user's reputation evaluation attribute set UE = {ue_1, ue_2, … ue_p}, for example, UE = {garbage data, timeout, additional request resources}; the service entity evaluation set D = {d_1, d_2, … d_m}, for example, D = {not credible, critical credible, generally credible, very credible, absolutely credible}; reputation evaluation

attribute evaluation matrix $R = (r_{ij})_{p\times Q}$; and the weight allocation of each reputation evaluation attribute $W = (w_1, w_2, \ldots w_p)$.

W is a weight distribution vector that represents the different preferences for each reputation evaluation attribute for the user. Use vector $W = [w_1, w_2, \ldots w_p]$ to denote the weight distribution vector, $w_i \in [0,1]$, and $\sum_{i=1}^{p} w_i = 1$.

Use the weigh value calculated from the final membership degree vector

$$F_u = w \circ R_d = [f_1, f_2, \cdots, f_m]$$

F_u is just a fuzzy vector, while the calculated value for the evaluation of the user is an exact number. Use the following method to calculate user j to user i:

$$T_{sNew}^{u}(j,i) = \frac{\sum_{k=1}^{q} w_k f_k}{\sum_{k=1}^{q} f_k} \tag{13}$$

After evaluation, the trust value is $T_{sNew}^{u}(j,i)$, according to trust with the characteristics of "rise slow" and "fast down." $T_{sNew}^{u}(j,i) > \sigma$. If the inequality is tenable, it denotes that the trust value should not decrease. We use incentives. The trust update formula of service entity j to user i is as follows:

$$T^{''}{}_{s}^{u}(j,i) = T^{'}{}_{s}^{u}(j,i) + \mu \times (T_{sNew}^{u}(j,i) - \sigma) e^{-(T_{sNew}^{u}(j,i)-\sigma)} \tag{14}$$

$T^{'}{}_{s}^{u}(j,i)$ is the direct trust of user i to services entity j after time attenuation, $(T_{sNew}^{u}(j,i) - \sigma) e^{-(T_{sNew}^{u}(j,i)-\sigma)}$ is the incentive factor, and $\dot{\imath}$ is influence factor of the space time complexity and transaction importance. $T^{''}{}_{s}^{u}(j,i)$ is the newest trust value of services entity j to user i after this transaction.

After punishing the user, the trust value is reduced. When the user allocates a resource in the future, the user's position in the waiting queue relative to other users with the same conditions should be in the rear. That is, high-reputation users prioritize chosen resources. We set a sub-trust threshold. If the user's trust is less than a certain trust threshold Ö, the system refuses to provide reserves resources for that user. If the user's trust is higher than trust threshold Ö but less than a certain reputation threshold ø, the reserve request position in the waiting queue should be in the rear.

7. Simulation and Result Analysis

The simulation generates a set of cloud resources. Each resource has a reserved queue and buffer allocated to its reservation tasks. The reservation time and reservation window of each reservation request are distributed respectively and uniformly at 1–20 and 5–50. The capability of resources and reservation requests of random resource demands are divided into levels 1–5. The resource ability of a reservation request candidate resource must be not less than its resource demand levels. For each resource and the reservation request, the user randomly sets the reputation value.

7.1 Change Rules of Acceptance Rate

The number of user reservation tasks is 200. With an increase in the number of system resources, the system accepts a user reservation task ratio that will be different. Choose a user whose reputation value is higher than a certain reputation threshold and test the user according to proportions of 100%, 70%, 40%, and 10%. The acceptance rate of a no-trust strategy (NTM) and trust strategy (TM) are compared. The acceptance rate of a reservation task that satisfies a certain reputation threshold and different reputation thresholds of user reservation requests are compared. Randomly generated user requests are allowed in the time range. So, the acceptance rate only needs to be considered in relation to the reputation value.

The system acceptance rate is shown in Fig. 4.

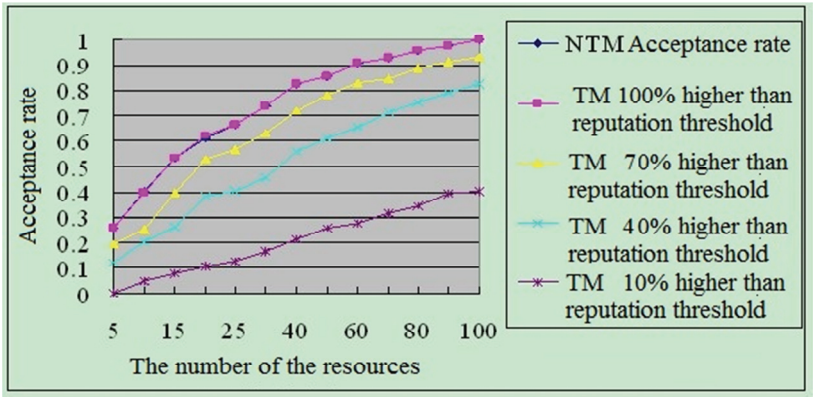


Fig. 4. Variation of accepted rate as number of resources increases.

When the reputation values of all reservation requests are larger than the reputation threshold specified, the acceptance rates of the TM and NTM have no obvious difference. However, when the reservation request reputation is lower than the specified reputation threshold, the acceptance rate decreases as the reservation request's reputation value decreases. This is because the system will reject the reservation request when the user's reputation is less than the specified threshold, which causes a decrease in the acceptance rate.

7.2 Change Rules of Success Rate of Transaction

This experiment tests the existence of malicious cloud resources. The number of transactions is increased, and the change in the transaction success rate under a TM and NTM is examined.

We set 50 available resources in the cloud management domain, which includes absolutely trusted resources, generally trusted resources, and critical trusted resource random distribution (there are no entrusted resources because entrusted resources do not provide services). Select 10%, 40%, and 80% of the resources as malicious resources.

When there are malicious resources, the change in the transaction success rates under a TM and NTM are as shown in Fig. 5.

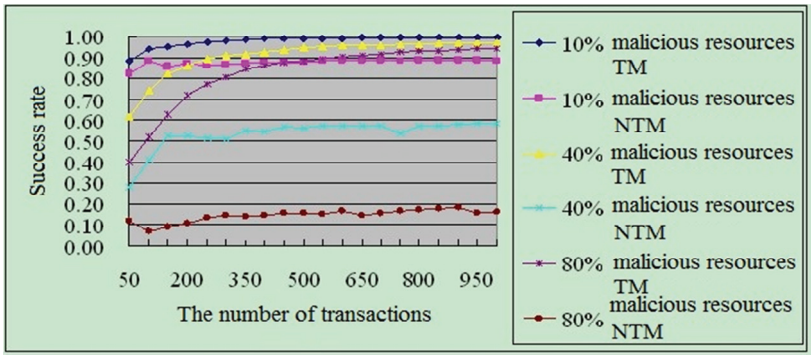


Fig. 5. Variation in transaction success rate as number of transactions increases.

As can be seen from Fig. 5, with an increase in the number of transactions, the success rate under the TM continues to rise. This is because the reputation of those that provide malicious service resources rapidly decreases under the TM. The service entity can no longer provide services when the reputation is reduced to a certain threshold, so the success rate is on the rise. Under a NTM, when selecting resources without reference to the credibility values, the transaction fails after malicious resources provide malicious service. The resources that provide malicious service are not punished. They are likely to provide malicious services again; therefore, the success rate decreases with an increase in the number of malicious resources.

7.3 Change Rules of Failure Rate of Transaction

This experiment tests the existence of malicious users. It examines an increase in the number of transactions, and the change in the transaction failure rate under a TM and NTM.

We set 50 users whose reputation values follow a random distribution but whose initial reputation values are greater than the reputation threshold of the services accepted. We select 10%, 40%, and 80% of the users as malicious users.

When there are malicious users, the change in the transaction failure rates under a TM and NTM are as shown in Fig. 6.

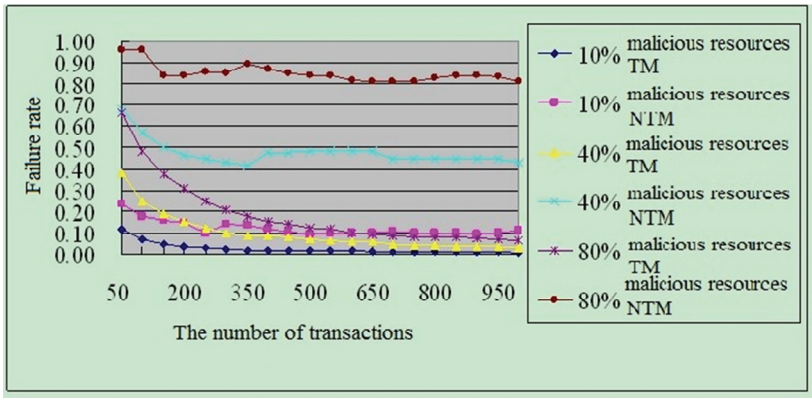


Fig. 6. Variation in transaction failure rate as number of transactions increases.

Under the same principle of malicious resources, a malicious user can also cause transaction failures. As can be seen from Fig. 6, with an increase in the number of transactions, the transaction failure rate continues to decrease under trust strategies. This is because when the user displays malicious behavior, the user's reputation value is reduced quickly. The system no longer provides any service to the user when his reputation is reduced to a certain threshold. The trust policy restrains the malicious behavior of the user. Under a NTM, maliciously behaving users are not punished. The user's malicious behavior may appear again after a transaction fails. Thus, the failure rate of the transaction will continuously increase.

8. Conclusion

This paper introduced a cloud-domain-based management architecture, which is combined with user preferences to provide a dynamic resource reservation service for the user. The service uses a two-way trust evaluation strategy to improve the accuracy of selection of resources and users. The experimental data show that the architecture can provide users with secure, reliable, and flexible reservation services, thus providing resources to satisfy the users' preferences and needs. Simulation results prove that the model can respond to disturbances of malicious users and malicious resources effectively. We improved user satisfaction and the resource access rate.

Acknowledgement

This work was supported by the Handan city science and technology research and development program project (No. 1721202044-3, Design and realization of wisdom logistics public service platform based on cloud computing; and No. 1721211052-1, Study on the realization of atmospheric pressure and large volume uniform discharge and its surface modification) and the Handan College School-level project "Supply Chain Trust and Risk Research Based on Cloud Computing" (No. 1534201095).

References

[1] J. Y. Wu, Q. L. Shen, J. L. Zhang, and Q. Xie, "Cloud computing: cloud security to trusted cloud," *Advanced Materials Research*, vol. 186, pp. 596-600, 2011.

[2] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Proceedings of the Grid Computing Environments Workshop*, Austin, TX, 2008, pp. 1-10.

[3] D. Chen and H. Zhao, "Data security and privacy protection issues in cloud computing," in *Proceedings of the International Conference on Computer Science and Electronics Engineering*, Hangzhou, China, 2012, pp. 647-651.

[4] S. Sun, C. Yan, and Y. Du, "Analysis on the influence of the cloud computing on the safety assessment technique," in *Proceedings of the International Conference on Computer Science and Electronics Engineering*, Hangzhou, China, 2012, pp. 285-288.

[5] D. G. Feng, M. Zhang, Y. Zhang, and Z. Xu, "Study on cloud computing security," *Journal of Software*, vol. 22, no. 1, pp. 71-83, 2011.

[6] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy, "A distributed resource management architecture that supports advance reservations and co-allocation," in *Proceedings of the 7th International Workshop on Quality of Service*, London, UK, 1999, pp. 27-36.

[7] C. M. Hu, J. P. Huai, T. Y. Wo, and L. Lei, "Service oriented grid architecture with end to end quality of service," *Ruan Jian Xue Bao (Journal of Software)*, vol. 17, no. 6, pp. 1448-1458, 2006.

[8] Z. A. Wu and J. Z. Luo, "Dynamic multi-resource advance reservation in grid environment," in *Proceedings of the IFIP International Conference on Network and Parallel Computing*, Dalian, China, 2007, pp. 13-22.

[9] K. M. McKeown, "Advance reservation and co-allocation protocol for grid computing," in *Proceedings of the 1st International Conference on e-Science and Grid Computing*, Melbourne, Australia, 2005, pp. 1-8.

[10] Z. Gao and S. W. Luo, "Dynamic grid resource reservation mechanism based on resource-reservation graph," *Ruanjian Xuebao (Journal of Software)*, vol. 22, no. 10, pp. 2497-2508, 2011.

[11] G. F. Wu and Y. He, "Improved domain-based trust model in grid environment," *Computer Engineering*, vol. 37, no. 3, pp. 137-139, 2011.

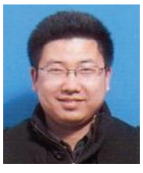
[12] W. Tang and Z. Chen, "Research of subjective trust management model based on the fuzzy set theory," *Ruanjian Xuebao (Journal of Software)*, vol. 14, no. 8, pp. 1401-1408, 2003.

[13] S. W. Cheng and Y. Pan, "Credibility-based dynamic resource distribution strategy under cloud computing environment," *Computer Engineering*, vol. 37, no. 11, pp. 45-48, 2011.

[14] Y. Yao, J. Lin, J. Wang, N. Mi, and B. Sheng, "Admission control in YARN clusters based on dynamic resource reservation," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management*, Ottawa, Canada, 2015, pp. 838-841.

[15] C. Ghribi and D. Zeghlache, "Exact and heuristic graph-coloring for energy efficient advance cloud resource reservation," in *Proceedings of the IEEE 7th International Conference on Cloud Computing*, Anchorage, AK, 2014, pp. 112-119.

[16] Y. Shi, X. Jiang, and K. Ye, "An energy-efficient scheme for cloud resource provisioning based on CloudSim," in *Proceedings of the IEEE International Conference on Cluster Computing*, Austin, TX, 2011, pp. 595-599.



Jiao-Hong Qiang https://orcid.org/0000-0002-9840-6070

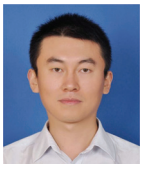
He received his M.S. degree in computer application technology from Hebei University, China in June 2008. He is currently working towards his Ph.D. degree in management science and engineering at Hebei University, China. His current research interest includes network security and trust management.

Ding-Wan Ning https://orcid.org/0000-0002-3993-1916

He received his M.S. degree in College of Information Engineering from Jiangnan University, China in June 2008. He is currently working in Handan College and his research interests include graphic processing, algorithm design and so on.

Tian-Jun Feng https://orcid.org/0000-0003-2243-4355

He was born in 1965 and is Ph.D. Professor (Tutor of doctoral students). He graduated from the Department of Electronics, Hebei University in 1986. In 2004, he graduated from the University of Science and Technology of China. His major in computer science and technology. Director of CCF.



Li-Wei Ping https://orcid.org/0000-0001-7379-9202

He was born in 1988 and received his Ph.D. degree in management science and engineering at Hebei University, China. He is post-doctoral research fellow, School of Economics and Management, Beijing Jiaotong University. His research directions include network security, internet finance, etc.