Mobile Networks and Applications 10, 971–983, 2005
© 2005 Springer Science + Business Media, Inc. Manufactured in The Netherlands.
DOI: 10.1007/s11036-005-4453-5

Enforcing Cooperative Resource Sharing in Untrusted P2P Computing Environments

ZHENGQIANG LIANG and WEISONG SHI
Department of Computer Science, Wayne State University, Detroit, MI 48202, USA

Published online: 24 October 2005

Abstract. Peer-to-Peer (P2P) computing is widely recognized as a promising paradigm for building next generation distributed applications. However, the autonomous, heterogeneous, and decentralized nature of participating peers introduces the following challenge for resource sharing: *how to make peers profitable in the untrusted P2P environment?* To address the problem, we present a self-policing and distributed approach by combining two models: *PET*, a personalized trust model, and *M-CUBE*, a multiple-currency based economic model, to lay a foundation for resource sharing in untrusted P2P computing environments. PET is a flexible trust model that can adapt to different requirements, and provides the solid support for the currency management in M-CUBE. M-CUBE provides a novel self-policing and quality-aware framework for the sharing of multiple resources, including both homogeneous and heterogeneous resources. We evaluate the efficacy and performance of this approach in the context of a real application, a peer-to-peer Web server sharing. Our results show that our approach is flexible enough to adapt to different situations and effective to make the system profitable, especially for systems with large scale.

Keywords: cooperative, heterogeneous, resource sharing, untrusted environment, P2P, economic model, trust model

1. Introduction

Peer-to-Peer (P2P) computing—federated sharing of dispersed pools of geographically distributed computing resources under coordinated control—has been considered as a promising platform for solving large-scale problems in science and engineering. However, resource management in these environments is a complex undertaking. These systems need effective mechanism for fair sharing of community resources, adaptability to dynamic changing conditions, prevention of denial-of-service (DoS) attacks, and coordinaton of the diverse policies, cost models, and varying loads different peers. As one motivating example, a classical "tragedy of the commons" for peer-to-peer file sharing is 50 to 70% of peers are free riders [1], which results in a great load imbalance of the systems. Resource trading can enforce a cooperative approach for the resource sharing and is promising to address the above problems.

The autonomous, heterogeneous, and decentralized nature of participating peers across multiple administrative domain introduces two challenging issues related to resource trading: *decentralized trading scheme*, which means the decision of resource exchange and negotiation is determined by each peer based on its personalized view of the partner and its own policy; *self-policing personalized trustworthiness management*, which means different peers may have different opinions on the trustworthiness of the same peer, instead of unique global trustworthiness value like eBay.

In this paper, we propose an approach that consists of two models: M-CUBE, a Multiple CUrrency Based Economic model, as the decentralized trading scheme, and PET, a PErsonalized Trust Model, to provide the trustworthiness of the peer to support M-CUBE. The M-CUBE model provides a general and flexible substrate to support most of high level resource management services required by the P2P computing, such as resource coallocation, quality of service (QoS) control, advance reservation and scheduling algorithms. PET derives the trustworthiness from the reputation evaluation and risk evaluation. The trustworthiness value provided by PET will be treated as the view of the peer by M-CUBE. The unique feature of our approach is seamless integrating the trustworthiness and dependability of peers into the resource trading.

The major contributions of this paper include: (1) We propose a formal trust model, including the reputation evaluation and risk evaluation, for calculating the trustworthiness of other peers in a self-policing way; (2) We propose a multiple-currency based economic model, which seamless integrates the trustworthiness to provide a self-policing method to enforce the cooperative sharing of heterogeneous P2P resources. Regarding to the pricing problem, we price resources according to their prices in the real economic market. By this mean, prices of heterogeneous resources are comparable, so that heterogeneous resource sharing is feasible in M-CUBE; (3) We design an efficient resource trading protocol which has the capability to prevent multiple rebellious problems related to resource sharing, such as *free rider*, *boaster*, and *application DoS attack*; (4) We evaluate the efficacy and performance of this approach in the context of a real application, peer-to-peer Web server sharing. Our results show that our approach is flexible enough to adapt to different situations and effective to make the system profitable, especially for the system with large scale.

The rest of this paper is organized as follows. We provide an overview of our approach in Section 2. The details of the PET model is provided in Section 3. Section 4 depicts the design of the M-CUBE model. In Section 5 we describe an application scenario and give the detailed analysis with simulation based on our approach. Finally, related work and concluding remarks are listed in Sections 6 and 7 respectively.

2. Overview

There are five problems related to resource sharing in an open P2P environment.

(1) *Heterogeneity*. The heterogeneity of resources makes the multiple-resource sharing difficult, because of lacking a formal metric for the trading among different resources;

(2) *Untrustedness*. Enforcing a cooperative, adaptive, and anti-maliciousness P2P sharing environment on top of an untrusted and private P2P community is really a challenge;

(3) *Selfishness*. The possible threats launched by selfish peers, such as cheating and boasting, can destroy the cooperative resource sharing. Enforcing a fair resource sharing framework to limit the negative effect of the selfish peers is one of the goals for our approach;

(4) *Autonomy and cooperation*. Peers usually belong to different administrative domains which may have different local policies. Effective and efficient integration of these local policies and general resource sharing is a challenge;

(5) *Incentives*. Free riders are the considerable population in the P2P community. To attract the peer to contribute to the community is an old but still ongoing problem. In this paper, we intend to address all these problems.

We conjecture that the fundamental solution for these problems is a trustworthiness mechanism for resource trading, as the dependable trading to world economics. Based on this mechanism, a lot of high level resource management related services, such as service level agreements, access cost negotiation, and advanced reservation can be easily built. Therefore, the major objective of this paper is to build a trusted dependable trading approach, which include two components: the M-CUBE model and the PET model.

As shown in figure 1, the M-CUBE model is a flexible *universal* infrastructure for building high-level resource management related services, and provides a comprehensive solution to all challenges listed above. There are four major modules in M-CUBE: the *Price Regulator* decides the price of the resources; the *Ratio Regulator* determines the exchange ratio of the currency based on the trustworthiness value provided by the PET model; the *Service Discovery* module is in charge of discovering the available resources provided by remote peers; finally the *Currency Exchange* module enables peers to bargain until the agreement of the currency exchange is reached, and then makes the exchange. PET underpins M-CUBE through providing the accurate trustworthiness value. Trustworthiness is service-specific. One peer can have different trustworthiness value corresponding to different services in the eyes of other peers, so PET actually provides the (trustworthiness, service) pair for M-CUBE. PET models the reputation, and treats the risk as the opinion of the short-term behavior and makes it be quantified.

In the design, we assume the use of Public Key Infrastructure (PKI) for naming and authentication in both M-CUBE and PET. Before describing the PET and M-CUBE model, we give the basic assumptions first:

(1) Each peer has a relative unique and stable ID. This will make reputation and trustworthiness make sense;

(2) Coordinated access to diverse and geographically distributed resources is valuable for participating peers;

(3) Each peer has an associated public-private key pair to make its currency unforgeable;

(4) Most peers in the system need the cooperation so as to gain profitable through sharing the resources;

(5) Each peer is selfish;

(6) Each peer has a pair of public/private keys.

3. PET design

PET is the underpinning of our system, which provides the trustworthiness to trigger M-CUBE to evolve. In PET the trustworthiness T is directly derived from two parts: reputation R_e and risk R_i, as shown in the upper right corner of figure 2. W_{Re} and W_{Ri} are the weights of R_e and R_i respectively. For a complete new peer without any related R_e and R_i information, its trustworthiness value is set as 0.4, a relatively lower value. Actually T is related to one type of service. Without specially pointing out, all variables in one formula is related to the same service. Reputation R_e is the accumulative opinion, which reflects the quality of target peer within a long term. PET models the reputation through combining the recommendation (E_r, also called referral or second hand information) and interaction-derived information (I_r, also called first-hand information). W_{Er} and W_{Ir} are their corresponding weights. Recommendation is the opinions from other peers, which is collected by the *Feedback Collection* component in PET. Interaction-derived information is the self-opinion from the direct interaction. Basically I_r is the self-knowledge, so it is reliable and self-determined. The interaction-derived information is also the base of the risk calculation. Risk is treated as the opinion on the short-term behavior. All values of T, R_e, R_i, E_r, and I_r are values from interval [0, 1].

There are a wide range of resource categories in P2P resource sharing such as CPU, hard disk, and so on. In the context of heterogeneous resource sharing, another program component *resource classifying* in PET is employed to

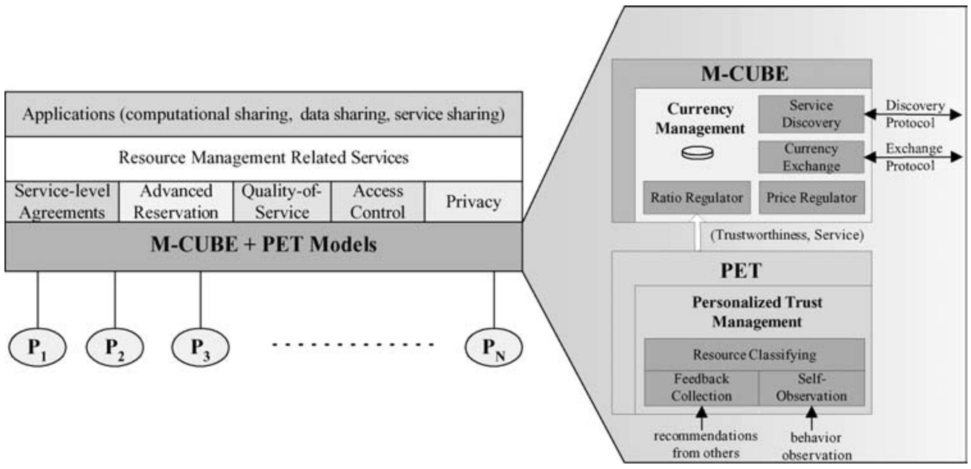


Figure 1. Overview of the proposed approach, including two models: PET and M-CUBE.

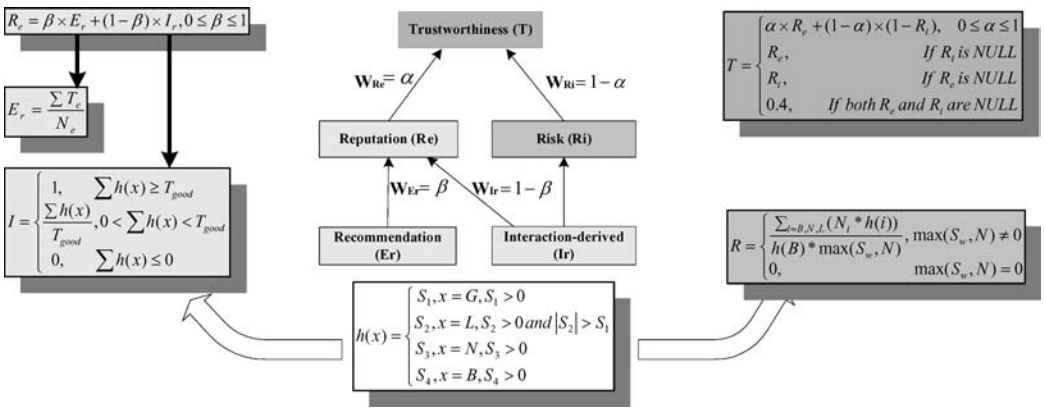

$R_e = \beta \times E_r + (1-\beta) \times I_r, 0 \le \beta \le 1$
$E_r = \frac{\sum T_e}{N_e}$
$I = \begin{cases} 1, & \sum h(x) \ge T_{good} \\ \frac{\sum h(x)}{T_{good}}, & 0 < \sum h(x) < T_{good} \\ 0, & \sum h(x) \le 0 \end{cases}$
Trustworthiness (T)
$W_{Re} = \alpha$
$W_{Ri} = 1-\alpha$
Reputation (Re)
Risk (Ri)
$W_{Er} = \beta$
$W_{Ir} = 1-\beta$
Recommendation (Er)
Interaction-derived (Ir)
$T = \begin{cases} \alpha \times R_e + (1-\alpha) \times (1-R_i), & 0 \le \alpha \le 1 \\ R_e, & \text{If } R_i \text{ is NULL} \\ R_i, & \text{If } R_e \text{ is NULL} \\ 0.4, & \text{If both } R_e \text{ and } R_i \text{ are NULL} \end{cases}$
$R = \begin{cases} \frac{\sum_{i=B,N,L}(N_i * h(i))}{h(B) * \max(S_w, N)}, & \max(S_w, N) \ne 0 \\ 0, & \max(S_w, N) = 0 \end{cases}$
$h(x) = \begin{cases} S_1, x = G, S_1 > 0 \\ S_2, x = L, S_2 > 0 \, and \, |S_2| > S_1 \\ S_3, x = N, S_3 > 0 \\ S_4, x = B, S_4 > 0 \end{cases}$

Figure 2. Derivation of the trustworthiness value.

identify the resource category to which the feedback and self-observation information belong, then this component adopts different strategies to process these information. In addition, we abstract four general behaviors, good, low-grade, no response, and Byzantine behavior in P2P systems. Peers with good behavior (G) provide the service as good as expected; peers with low-grade (L) behavior provides correct services, but with some degradation, e.g., delay HTTP response; no-response (N) behavior is from the view of requesters—the requester can not get any response from the service provider with this kind of behavior; finally peers with Byzantine (B) behavior give the wrong, or even malicious answer to the requester. All three L, N, and B services are bad services, and have increasing extent of harmfulness to the system.[1] To simulate the dynamics in P2P environment, the dynamic (D) quality is introduced as an additional quality. Peers with this quality changes its behaviors among G, L, N, and B uniformly and repeatedly. We formalize the quality set as Q = G, L, N, B, D. Correspondingly, the peers providing G service is called *G-peer*, so do *L-peer*, *N-peer*, *B-peer*, and *D-peer*. This coarse-grain classification is flexible enough to apply to any resource sharing, and more subclasses can be introduced if necessary.

From figure 2 it can been seen that $W_{Re} = \alpha$ and $W_{Ri} = 1 - \alpha$. When $\alpha = 1$, which means the weight of the risk is 0, PET degrades to the traditional reputation model that just considers the peer's history only. However our simulation results show that risk evaluation is a very helpful component to build the trust model. Normally when the system is highly dynamic and most nodes are not good, it is recommended to set the risk with a high weight (e.g. 0.7), which is supported by the simulation results in Section 5. For the blind users (i.e., users who do not know how to tune the parameters of the underlying trust model), $\alpha = 0.3$ is the safe recommended value. Integrating with risk evaluation distinguishes PET from the previous work [7,13,17,19].

Reputation model. In the following, we call the peer to evaluate other peers a *valuer*, the peer to be evaluated a *valuee*, and the peer sending out the recommendation the *recommender*. For example, when peer A tells peer C the trustworthiness value of peer B, A will be the *valuer* of B, and the *recommender* of C about B. Correspondingly, B is the *valuee* of A.

[1]Note that in our classification, "no response from the service provider" is a special behavior. We treat this kind of behavior as a bad behavior, no matter it is because of subjective factor, e.g., rejecting the service request intentionally, or objective factor, e.g., the physical link gets broken.

Reputation value is the historical accumulation for *valuee*'s past behavior from the *valuer*'s viewpoint. Sometimes some good peers will misbehave for nonsubjective factors, for example, the good peer rejects the network service requests for the breakdown of the physical link, but after recovery, it will provide good service continually. If wanting to forgive the occasional nonsubjective misbehavior, we can set a high value to α (e.g., $\alpha > 0.5$ for example) to make the trustworthiness preferring to the reputation value. Reputation is derived from E_r and I_r. Here $W_{Er} = \beta$ and $W_{Ir} = 1 - \beta$. $\sum(T_e)$ stands for the sum of the recommendations, and N_e is the amount of the recommendations. So E_r is the average value of recommendations. I_r is derived from the peer's accumulative score. After each transaction, the requester evaluates the quality of the cooperation. The peers providing `G` service will lead to their scores (`S`) increase, while peers providing `L`, `N`, or `B` service will cause their scores to decease. For the D-peer, it changes its behaviors among `G`, `L`, `N`, and `B` repeatedly and uniformly, so 75% of the behavior of `D`-peer is bad, and its score actually also decrease gradually. For the *valuer* there is a score function h mapping from Q to a score for one cooperation. As shown in function h in figure 2, the bad behaviors (`L`, `N`, `B`) will lead to more decrease of the score than the increase for the good behavior, and the `B` is the most harmful action which cause the most severe decrease. In the simulation, h is defined as: $h(\texttt{G}) = 1$, $h(\texttt{L}) = -2$, $h(\texttt{N}) = -3$, and $h(\texttt{B}) = -4$. The sum of score is normalized by a threshold value T_{good} to derive I_r.

Since PET aims to deploy in the P2P community, in which there are malicious recommenders providing the misleading recommendations, it is good to lower the role of the recommendation. The reasons are:

(1) Different peers are hardly consistent on the same service provider because of their different criteria and experiences.

(2) Peer's behavior can change dynamically, while recommenders need time to perceive this change, and there are delays for the recommendation to spread, so recommendations somehow inevitably departure from the truth.

(3) Fraudulent recommendation, especially the collusion on the recommendation is very difficult to handle if the trustworthiness calculation relies much on the recommendation.

However, it is not a good answer to ignore the recommendation. As the knowledge from other peers, the recommendation is helpful to know more about the system without direct interaction. Assigning it a lower weight β is a good solution, which is supported by the simulation results in Section 5. More detailed analysis of the effect of recommendations can be found in [15].

Risk model. Reputation is not sensitive enough to perceive the suddenly spoiling peer because it needs time to decrease the accumulative score. Risk evaluation can help to solve this problem. Trustworthiness is a temporal value, because the behavior of the peer will change dynamically. The old trustworthiness value may totally misrate one peer after some time passes. To solve this problem, a risk window is employed to limit the assessing range. While the window is shifting forward, the risk value reflects the fresh statistics of the *valuee*'s recent behaviors, which integrates the temporal factors into the trustworthiness value. To calculate the risk R_i, we first get the total score of the bad service within the window, then make it normalized with the product of $h(B)$ and the window size S_w or N. N is the number of total history events in the queue. Normally S_w and N are equal, except that at the beginning, the queue is not full. When there is no interaction at the beginning, the risk value is set to be zero, that is, no risk for the new stranger. It seems that this strategy opens a door for the new stranger and brings corresponding threats such as Sybil attack [8], but actually not. Once a peer behaves badly, the risk will increase significantly, so that the door will close very fast for the bad peers, which will be validated in the Section 5.3. To reduce the risk of the cooperation, users can focus more on the risk by assigning $1 - R_i$ a high weight by trading off the resource availability.

4. M-CUBE model design

M-CUBE makes use of the (trustworthiness, service) pair provided by PET to change the view on the quality of others, then adjusts the corresponding policies to control the currency. In this section, we first give a brief introduction about the world economic model, which inspires the design of M-CUBE, then present the details of the M-CUBE model. Finally, the advantages of this model are discussed.

4.1. Currency model

Inspired by the features of the world economic model, we propose the M-CUBE model, which is a multiple-currency based, self-policing, dependable and unified method for heterogeneous resource sharing; however, our approach differs from the real economic market in the grain of economic entity. That is in our currency model, every peer, namely one machine or one organization, issues and regulates its own currency, while in the real world economic market, each country (not a single person) is the smallest entity to control the its currency issuing and exchange. M-CUBE is built upon currency-based mechanisms, where the uniqueness of M-CUBE is each peer has its own currency. Unlike many of previous work, in addition to associating the currency with physical resources directly, such as CPU and disk, M-CUBE also can associate currency with application-level services directly, e.g., each HTTP request.

Pricing and ratio. In M-CUBE, pricing is the first problem to be addressed before building the currency model. Since most computer users live in the market-economy society, it is reasonable and acceptable to price our resources in the virtual community referring to the real price of the physical devices. On the other hand, the shared resources have their own period of validity. So from the view of trading, the currency in the

M-CUBE is mainly expressed as a 3-tuple: *(t, p, v)*, where t is the type of the resource, p is the number of this type resource which \$1 can buy in the real economical society, and v is the validity period of the resource. In the following subsection, more details about the format of the currency are described. However, regarding to the application-level service not only related to a single device, for example, one calculation request in SETI@Home, it needs to consider multiple devices related to this service to fix p. Normally the pricing normally is self-decided, but it also uses \$1 as the basic unit to define p. For example, if \$1 is decided to be able to buy 10 requests, then the value of p is equal to 10.

Based on the value of p, heterogeneous resource trading is feasible. Here is an example. Peer A wants to share its 100 G hard drive, and peer B wants to share its 2 GHz CPU. Assume in the real market A's hard drive cost \$100, and B's CPU cost \$80. Then the value of p in A's currency is 1 (unit is GB), and B's is 25 (unit is MHz).[2] Ignoring the consideration of the validity period, one unit of currency of A (C_A) is expected to get one unit of currency of B (C_B) because one currency is corresponding to \$1. In this case, one C_A can be used to exchange for 25 MHz CPU resource from peer B, or one C_B can be used to exchange for 1 G harddisk from peer A. Two advantages can be expected with this approach:

(1) People are willing to accept this approach because it is similar to their dairy life;
(2) Base on this approach, heterogenous resources can be easily exchanged, because all currencies are introduced based on \$1 value in the real economic market.

In order to simplify the system design, a pricing bootstrap peer will be introduced in our system. The bootstrap peer is taking care of one additional task to update the device price. Other peers in the system contact the bootstrap peer to get the reference price. However the reference price is not mandatory; other peers can price their resources based on their own experience, disregard for the price from the bootstrap peer. So the bootstrap peer is not necessary in the system. The reference price also can be referred to see whether the price from the counterpart peer between the resource exchange is reasonable or not. The module *Price Regulator* is employed to manage the price, whose job is either to contact the price bootstrap peer periodically or self-decide the price according to some pricing mechanism. A lot of previous work has been done on the pricing [10–12], which complements to our work and can be used for pricing in M-CUBE.

Currency format and management. When two peer exchange their currencies, there is an exchange ratio R_c that they agree with. Initially, since every currency is corresponding to \$1, so R_c is equal to one. After sometime, R_c will be self-adjusted according to the trustworthiness value provided by PET, which is defined by a function f: $R_c = f(T, R_c)$. A simple definiton of f, $R_c = T$, is adopted in our following sections. That is, just use the trustworthiness as the exchange ratio. For example, if for peer B, peer A's trustworthiness value is 0.5, then one unit of A's currency can just exchange for 0.5 unit of B's currency when A asks for B's currency. R_c is regulated by the module *Ratio Regulator* in figure 1. When *P1* wants *P2*'s service, it must use *P2*'s currency. But *P1* must have its own currency first which it can use to exchange with *P2*. Once *P1* issues its own currencies, it promises to share the corresponding resources those currencies standing for. So if one doesn't contribute any resource to the community, it has no currency for exchange so that it won't get any services from others. The incentive brought by M-CUBE to the resource sharing is: the more the contribution a peer provides, the more services it can get from others; the peer will get more benefits than its actual contribution because sometimes resources from others can help the peer to pull through the difficult period such as overloaded time, which are definitely more valuable than the sharing in the common situation. However, issuing more currencies than one's capacity blindly is not a good way either. This is what we call the *boaster*. The total number of currencies stands for the peer's outward service capacity. The boaster may incur trustworthiness loss because it will be unable to serve the legal requests when most of its currency holders ask for the services at the same time. So, the peers must issue the currency according to its actual service capacity. One peer will provide its service only when it receives its own currency and it must do so in order to maintain its reputation in the community. The format of the currency is shown in figure 3(a). *TypeVec* stands for type vector, to specify which resource this currency related with. It should be made clear that, though *P1*'s currencies can relate to different resources, they are all *P1*'s currencies. When we talk about multiple-currency, we are from the view of different peers, not different resources. When a peer generates a new currency, it will fill in this field to make the currency to be used only for the specified resource. Because the resources are limited, the number of currencies corresponding to one resource is also limited. In M-CUBE, every issuer must take care of the currency issuing itself to avoid to be a boaster. *ResNum* is the number of corresponding resources which this currency can buy. *ValiTime* stands for the validation time (Time-to-Live) of the resource the contributor guarantees. Based on *ValiTime*, the receiver of the currency will know when the service is available on the issuer side. The validity period p mentioned in above paragraph can be achieved by subtracting current time from the *ValiTime*. *LiveTime* specifies the validate time interval of currency. If after the exchange the currency is not used within *LiveTime*, the currency will expired, and the issuer will re-issue the currency with another *LiveTime*. Different with *ValiTime*, *LiveTime* is from the angle of currency, not the resource. Normally *LiveTime* is less than *ValiTime*. *IssuedTime* is the time stamp indicating issue time of the currency. When an issuer receives its currency, it will check if the currency is valid by comparing the *IssuedTime* + *LiveTime* to the its current time, but this limitation is not strict because the time is not strictly synchronized in the distributed system. Through this way, the service consumer signs a

[2] 25 MHz CPU is $\frac{1}{80}$ of 2 GHz CPU. From the application view, it represents $\frac{1}{80}$ CPU usage.

usage contract with the service provider and takes on the duty for the expiration of the provider's currencies. *SeqNum* is the sequence number of the currency, which is used for deciding the authenticity of the currency by the issuer. Finally, *DigSig* is the digital signature signed by the issuer. The issuer uses the node's private key K^- to encrypt the *TypeVec, ResNum, ValiTime, LiveTime, IssuedTime, SeqNum* to generate digital signature *DigSig*. The currency is totally self-determined and self-policing. It meets the demand of high independence of the P2P community. In order to decrease the space consuming for the currency storage, every issuer keeps a table to track where its currencies in terms of (*TypeVec*, *Resnum*, *SeqNum*, Receiver), so that currency receiver can merge the currency with the same *TypeVec* and issuer, just by requesting a new large-amount currency from the issuer. The issuer just needs to delete the corresponding small-amount currencies, and add a new tracking item with new *Resnum* and *SeqNum*. The similar process happens in the process of currency consuming, but the direction is change the large-amount currency to small-amount currency. All the communication channels are secure and authenticated with the help of PKI.

Service discovery protocol. Before one peer exchanges currency with another, it must know who has the currency related to the resource it wants, which is taken care by *Service Discovery* module in figure 1. Two functionalities the module performs:

(1) Locating the wanted currency,
(2) Making sure the validity period of the wanted currency is long enough.

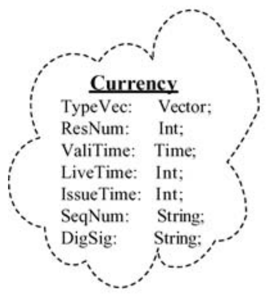
In M-CUBE, limited-hop multicast is used here for the service discovery. The requester sends out a request (C_{num}, C_{type}, R, V) to its cooperators (cooperators refer to the peers having the history of currency exchange before), where C_{num} is the currency number it wants, C_{type} is the resource type to which the wanted currency related, R is the identifier of the requester, and V is the minimum validity period for the request resource. The cooperators will forward the request to their cooperators again. According to the social network phenomenon, it is expected that after several hops the wanted currency can be discovered. In M-CUBE the maximum number of hops is set as six. All the receivers piggyback the response to the requester peer. If all the following two conditions are met, that is,

(1) The currency associated with the requested resource is available,
(2) The validity period is long enough,
the receiver will confirm the requester peer, and tell the requester peer what kind of currency the receiver needs if the requester peer wants to proceed the exchange.

One important thing is, delegation peer is allowed in M-CUBE to improve the efficiency of the resource sharing. Peers are not limited to just exchange with the issuer directly. In other words, if peer A has peer B's currency, and peer C wants peer B's currency, peer C can exchange peer B's currency with peer A. We call A an *exchange delegation*. This will improve the resource availability and efficiency. Considering the case that when peer B and peer C does not know each other, so peer B and peer C can not build the cooperation relationship. But peer C needs peer B's service. Without the exchange delegation, C won't get B's help, and B's resources can not be known by C.The requester picks up some peers and builds the candidate list according to the trustworthiness of the peer who issues the wanted currency (not the peer performing the exchange. Remember there are delegations here, and the trustworthiness just cares about the service provider, not the currency exchanger). Then one peer from the candidate list is chosen to proceed the currency exchange.

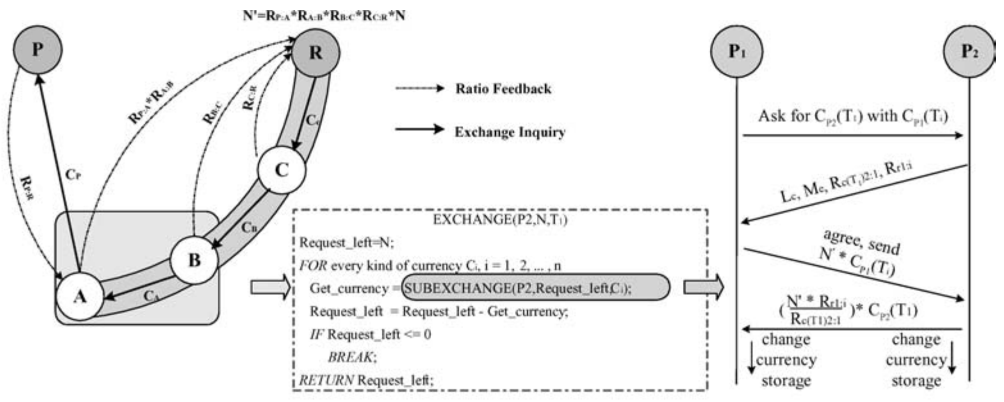
If the currency exchange can not be fulfilled, e.g., the exchange ratio is very high for the requester, so that it doesn't want to continue exchanging, another peer from the list is chosen to continue the exchange. When the requester doesn't have the right currency as the responder needs, it must try to get the currency the responder wants first. In a worse case, the requester may have to get several intermediate currencies to finally get the wanted currency. In this case, the exchange chain shapes up, as shown in the left part of figure 3(b). The intermediate nodes in the exchange chain are the delegations A, B, and C. Two end points of the chain are issuer P and the requester R. Now R wants P's currency. After multicast, R knows that peer A has P's currency. In this scenario, A is the delegation of P. But A just accepts the currency of peer B, which R doesn't have. Then R uses another multicast to find out who has B's currency, and B responses and tells R it just accepts C's currency. Fortunately, C accepts R's currency (through another multicast to find who has C's currency). Now R can build an exchange channel to A through the chain $R \Rightarrow C \Rightarrow B \Rightarrow A$. To improve the performance, the exchange activity won't happen until the requester agrees with all ratios from the delegations in the chain. So R records the ratio reported from A, B, and C first. If R agrees with all the ratios, the exchange chain forms: R will trigger the chain exchange by exchanging with C first, and then uses C's currencies to exchange with B, and the process goes on until it gets P's currency from A. One detail is, when A gets R's exchange request finally, A will inquiry P the ratios $R_{P:R}$ (The currency ratio for R in P) first. A rejects the exchange in case $R_{P:R}$ is low, that is R's trustworthiness in P is low (remember $R = T$). If R has a good reputation, A gives R P's currency with ratio $R_{P:A} * R_{A:B}$. Actually finally N units of R's currencies can exchange for $R_{P:A} * R_{A:B} * R_{B:C} * R_{C:R} * N$ units of P's currencies.

Currency exchange protocol. At the beginning, one peer only has its own currency. It needs to exchange the currency from other peers when it needs the services from others. The module *Currency Exchange* in figure 1 takes care of this job. After the service discovery stage mentioned in previous paragraph, the requester already has the candidate list, and one candidate peer has been chosen. In the following, we will just state a basic exchange protocol without considering the chain exchange (actually the chain exchange is just a little bit different. What to be modified is to combine the exchange in the chain together). The requester sends its exchange request



```
Currency
TypeVec:   Vector;
ResNum:    Int;
ValiTime:  Time;
LiveTime:  Int;
IssueTime: Int;
SeqNum:    String;
DigSig:    String;
```

(a)



N'=RP:A*RA:B*RB:C*RC:R*N
P
R
RP:A*RA:B
RB:C
RC:R
CR
CP
RP:R
C
CC
CB
B
A
CA
Ratio Feedback
Exchange Inquiry
P₁
P₂
Ask for $C_{P2}(T_1)$ with $C_{P1}(T_i)$
L_c, M_e, $R_{c(T_1)2:1}$, $R_{r1:i}$
agree, send $N' * C_{P1}(T_i)$
$(\frac{N' * R_{r1:i}}{R_{c(T1)2:1}})* C_{P2}(T_1)$
change currency storage
change currency storage

```
EXCHANGE(P2,N,T1)
Request_left=N;
FOR every kind of currency Ci, i = 1, 2, ... , n
  Get_currency = SUBEXCHANGE(P2,Request_left,Ci);
  Request_left = Request_left - Get_currency;
  IF Request_left <= 0
    BREAK;
RETURN Request_left;
```

(b)

Figure 3. Currency format and currency exchange protocol. (a) Currency format, (b) Service discovery and currency exchange protocol.

to the candidate, and the candidate decides the exchange amount based on the exchange ratio. The pseudocode of the exchange protocol is shown in the middle of the figure 3(b). We assume that *P1* wants to get $N\, C_{P2}(T_1)$ (the currency with service type T_1 from *P2*), and the currencies of *P1* are related to total *n* kinds of resources. Here we also don't consider the case of delegation, and assume that all currencies of *P1* used to exchange are issued by *P1* itself. At first, P1 will use the currency $C_{P1}(T_1)$ to ask for exchange. If this kind of currency is not enough, it can use other kinds of currencies to continue the exchange until its request is satisfied or no more kinds of currencies can be used for the exchange. *SUBEXCHANGE* function operates as shown in the right part of the figure 3(b), which illustrates the exchange process for just one kind of currency. *P1* inquiries *P2* if possible to exchange the currency with type T_1 from *P2* using the *P1*'s currency with type T_i. If *P2* can conduct the exchange, it sends back the exchange-related information to *P1*, which includes L_c, M_e, and $R_{2:1}$. L_c means *P1*'s credit limit in *P2*, which is the total maximum number of currencies *P1* can ask from *P2*. M_e is the maximum number of the currency *P1* can ask from *P2* in this exchange. $R_{2:1}$ means the currency exchange ratio for C_{P2} to C_{P1}. The protocol is totally self-policing and negotiable. *P1* can reject the exchange for the low exchange ratio specified by *P2*. If *P1* agrees the ratio, *P1* will send *P2* its N$'$ currencies $C_{P1}(T_1)$, and *P2* will send back N$'$∗R$_2$:1 currencies $C_{P2}(T_1)$ to *P1*. Here N$'$ may be not equal to *N* after negotiation. When the exchange procedure completes, both *P1* and *P2* will change their currency storage. Since the currency exchange tends to attract more attacks, some special security protocols, e.g. Diffie-Hellman key exchange protocol, can be used here to protect the exchange.

4.2. Advantages of the model

Benefiting from the nature of the currency mechanism, we can make the resource sharing controllable, eliminate the free-rider and boaster, and make the system anti-DoS with our M-CUBE model.

Making resource sharing controllable. In M-CUBE, the resource sharing and trading are under control from the prospectives of the number and time, which is important for the open P2P community. Through the usage of the currency, every peer is coupled with the system not only using the system, but also managing the system by discovering and propagating the bad peers through the ratio adjustment. The controllability also provides more benefits of the resource sharing with more *reliability* and *predictability*, which is a potential way to put P2P resource sharing to a good direction against the law violation. Every peer has incentives to keep good reputation, so that they also take the responsibility to maintain the reliability of their resources claimed in their currencies. The predictability let peers know when they can find the available resource so that they can deal with emergency or a complex task with the additional resource from others.

Eliminating free-rider. Free-rider [1] is a severe problem in P2P community. In the M-CUBE model, no free-rider can exist because none of them can get other's currency without exchange its currency with other. When one's currencies are hold by other peers, it will have to provide the service whenever other peers redeem the currency, otherwise its credit will be decreased and finally it will be kicked out from the community.

DoS attack free. Since every law-abiding peer generates its currency based on its service capacity, so even facing the burst of the service requests from others, the peer is still can satisfy the requests at the same time. That is, our currency model can avoid the DoS attack for the law-abiding peer. However for the boaster, it is out of the protection of our model. It is worth noting that when we say DoS attack free here, we refer to the possible attack resulting from our currency model and locating in the application-level. The network-layer attack such as the TCP/IP SYN attack is not our major concern.

Anti-boaster and inflation. In [9], boasters are considered as legal and useful. However in M-CUBE, boaster are illegal because they will lead to the uncontrollable state of the system. In the real economic market, if the value of total currencies is larger than the actual value of the total merchandize, it will lead to the inflation and disorder the market trading. However, our currency mechanism has a natural essence to suppress the inflation. The boaster may be able to exchange for the currencies of other peers at first with its devaluated currencies, but the stealing action will be punished by the community in the long run. Other peers holding the boaster's currency will redeem the currency to ask back for the service later. When the boaster can not provide service with good quality facing the burst of the service requests, its trustworthiness value will be decreased. Finally, when the trustworthiness drops to below a threshold, the boaster will be eliminated from the community. Simulation results in Section 5 show that even most of bad peers in the system are boasters, our approach is still can make the system profitable.

5. Experimental analysis

The simulation is conducted in the context of P2P Web server sharing application [18], which is a new content delivery mechanism for both static and dynamic Web content by federating participating Web servers together in a P2P fashion. It empowers the individual peer which is autonomous with respect to managing the resources and replica placement. Each Web server is a peer and serves a bunch of clients. The peers pool their resources to help each other during individual peer's peak loads and/or system failures. The main concept behind the workability of this arrangement is an understanding that not all companies which form the P2P network will have peak loads on their web sites simultaneously. PET will be integrated with the proposed M-CUBE model for the application in the simulation.

The simulation is thread-based and written in Perl language. Table 1 gives the details of the settings of the simulation. There are 500 peer servers to be simulated. To show the scalability of the model, two sizes of clients are used: 4,700 clients (C1) and 9,400 clients (C2). The peer servers need to cooperate with each other to make full use of the spare (computing) resource to serve the clients. HTTP requests from clients are generated using SURGE [3]. The total number of requests in the simulation is about 300,000 when using 4,700 clients, and 600,000 when using 9,400 clients. Five configurations (from P1-P5) are used to simulate different P2P communities as listed in Table 1. In order to simulate the malicious recommenders, peers will also have a secondary role: sending out the correct recommendation (M1) or malicious recommendation (M2). In our simulation, the malicious recommendation will rate the good peers as bad, and bad peers as good. The B-peers will send out the malicious recommendations when option M2 is chosen. Finally, in order to simulate the worse untrusted environment, we also introduce the role *Boaster* into the simulation. Changing the weights of different model components can adjust the model to different environments. Finding some good weight settings through the simulation is one of our goals as well. To achieve this goal, six weight combinations (from W1-W6) are used as shown in Table 1.

In the following analysis, we will mainly compare the percentage of the good service brought by the system to analyze the effect of different components and the improvement or our approach in different kinds of environments. For all figures except figure 6, the x-axis is the four service categories (G, L, N, B), and the y-axis is the percentage of the requests receiving the corresponding service category. In figure 6 the relationship between PET and M-CUBE are explored. Due to space limit,

Table 1
Simulation settings and their illustrations.

Settings			Illustrations
Client number	C1	4700 Clients	Small-size population.
	C2	9400 Clients	Large-scale population.
The proportion of peer with	P1	20%:10%:10%:30%:30%	To simulate the community with less good peers and all kinds of peers coexist.
different quality (G:L:N:B:D)	P2	20%:0%:0%:0%:80%	To simulate high dynamic community with many dynamic peers.
	P3	20%:20%:20%:40%:0%	To simulate the stable community without dynamic peers.
	P4	50%:10%:20%:10%:10%	To simulate a half-good community.
	P5	80%:5%:5%:5%:5%	To simulate a terrific community.
Malicious recommendation	M1	Malicious recommendation	Spreading the distorted facts.
	M2	Correct recommendation	Spreading the true facts.
	W1	$\alpha = 0.3, \beta = 0.2$	Emphasizing R_i and I_r.
	W2	$\alpha = 0.3, \beta = 0.5$	Emphasizing R_i and relying more on E_r.
Weight of different components	W3	$\alpha = 0.7, \beta = 0$	Emphasizing R_e and ignoring E_r.
	W4	$\alpha = 0.7, \beta = 0.2$	Emphasizing R_e and I_r.
	W5	$\alpha = 0.7, \beta = 0.5$	Emphasizing R_e and I_r.
	W6	$\alpha = 1, \beta = 0.2$	Ignoring R_i and Emphasizing I_r.

we report the results of only one metric. Interested readers please refer to the technical report version of this paper [14].

5.1. Effect of different components

In this part, two interesting issues will be studied. We first focus on the risk evaluation with different weights, then turn to the effects of recommendations with different weights and peer constructions

(1) *Risk evaluation with different weights.* In figure 4(a), we fix other options and just change W_{Ri}. Three values of W_{Ri} are used here: W1(0.7), W4(0.3), and W6(0). Other fixed options are: C1, M2, and P1, that is a system with only 20% good peers, but peers send out the correct recommendations. Intuitively, *when there are large number of bad peers* (L, N, B, D), *high value of* W_{Ri} *is helpful to find the bad peers*. In figure 4(a), it can be observed that, with the increase of the weight W_{Ri} ($W6(0) \rightarrow W4(0.3) \rightarrow W1(0.7)$), the more good services the system achieves. However the difference is not that much. It is because we choose a small number clients (C1) in this experiments. From the analysis of the last two experiments, we can expect that, the difference will be enlarged if we select larger number of clients (C2). From this analysis, a hint comes out: in the P2P community where most peers are not good, the high value of W_{Ri} is helpful to improve the model. An interesting topic we will study in figure 4(b) is the ability of the risk model against the malicious recommendation. We choose two weights of the risk W_{Ri}, W1(0.7) and W4(0.3), and take the malicious recommendation into the consideration. The other fixed options is the same with figure 4(a). From the figure, we can see that the options (M1, W1) brings 36% good services, much better than 31% with the options(M1, W4), and it is even better than 35% with options (M2, W4), the case without malicious recommendations. From these data, another hint comes out: *when the malicious recommendations exist, setting* W_{Ri} *with a high value is great helpful to resist the malicious recommendation*.

(2) *Selecting the weight of recommendation.* To select a suitable weight of recommendation is important for PET. We conduct this group experiments to try to know how to select the weight of the recommendation W_{Er} to decrease the negative effects of the malicious recommendation while keeping the same performance. We will mix M1 (Malicious recommendation) and M2 (Correct recommendation), P1 (20% G-peers and 30% D-peers) and P2 (20% G-peers and 80% D-peers), and W3 (W_{Er} is 0), W4 (W_{Er} is 0.2) and W5 (W_{Er} is 0.5) to build different experiments. Other fixed option include C1.

Let's focus on the lines with M1 option in the following discussion. In figure 4(c), with the option W4, among all the requests, 36% are served with good service, higher than 32% from the case with the option W3 (ignore the recommendation). The result is even a little bit higher than the case without malicious recommendations, which implies that *a lower value of* W_{Er} *is better to resist the malicious recommendations and discover the G-peers than both ignoring the recommendation (W1) and relying more on the recommendation (W5)*. From the above results, we can conclude that *in a community with malicious recommenders, just ignoring others' recommendations is not a good way. The right solution is assigning it a low weight to make a tradeoff*. From the simulation results, the tradeoff can lead to a good solution.

5.2. Improvement of our approach

In the following, we will study the improvement of our approach through three angles of efficacy, anti-boaster, and scalability.

Setup: The service requests of clients are generated by SURGE, which are stored in one file. There are some other files used to specify the quality of the peer servers. Combining these files, we can get the results of how the requests will be served without our approach (we call it the standard result). Since the D-peer changes its quality repeatedly and uniformly, we amortize its service to other four services when calculate the standard result, so actually no D service exists. In this experiment group, G : L : N : B : D is 20% : 10% : 10%: 30%: 30%. So after the amortization, G : L : N : B will be 27.5%: 17.5% : 17.5% : 37.5% (each add 30%/4 = 7.5%). We will use this as the expected standard result without M-CUBE, and see the improvement and efficacy compare the results with M-CUBE. To see the effects of the boaster, two groups of experiments are conducted: one is without the boaster and the other with the boaster. In addition to the boaster, in the second group of experiments we also let the bad peers act more intelligently, i.e., the bad peers are able to change the cooperators which have recognized their bad quality, and attempt to find new cooperators, through which to gain more benefits from the new cooperators. Our goal is to simulate a highly untrusted environment to test the effect of our approach. In order to study the scalability, two groups of experiments are conducted also: the first *E1* is with different number of clients, and the second *E2* is with different number of peer servers. *E1* studies the effect when the system overload changes. In *E1* two sizes of clients (C1 and C2) are used for the comparison with the same number of peer servers (500). *E2* is to study the effect when the system scale changes. In *E2*, what we try to do is construct two system scale, one is $\frac{1}{10}$ of the other. Two sizes of peer servers (S1 = 500 and S2 = 50) are simulated. The latter size is $\frac{1}{10}$ scale of former one. When using setting S1 = 500, the size of clients is C1 = 4,700, and the total number of requests generated is about 300,000. When using setting S2 = 50, the size of clients is 1,000, and the total number of requests generated is about 36,000, about $\frac{1}{10}$ of C1.

Discussion: Figure 5 shows the related results, in which the x-axis is the four service categories (G, L, N, B), and the y-axis is the percentage of the requests receiving the corresponding service category.

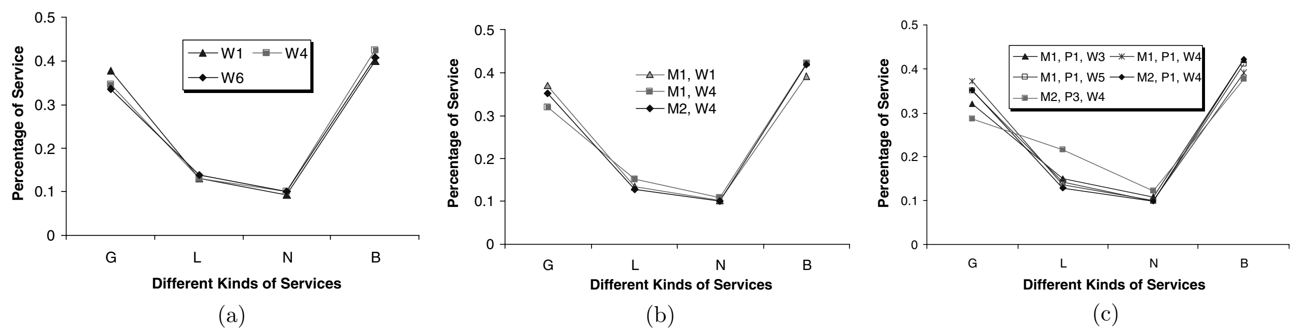


Figure 4. Effect of different components. (a) Risk evaluation with different weights, (b) Risk evaluation with the effect of malicious recommendations, (c) Effect of recommendations.

1. **Efficacy.** First, let's study the efficacy of M-CUBE. In figure 5(a) there are three lines. One is the "Without M-CUBE and PET", which is the standard result we have discussed in the setup part; one is with M-CUBE and PET, and smaller size of clients C1; final one is also with M-CUBE and PET, but with larger size of clients C2. There are no boasters in this group of experiment. What is desired for our approach is to enable more requests to get the good services and depress the Byzantine services, because Byzantine services bring most severe loss among the bad services. From figure 5(a), we can see that once applying our approach, there is great improvement: with small scale ($C1 = 4700$), the percentage of good service is increased from 27.5% (standard result) to 35.5% (relatively 29.1% improvement); when the scale is larger ($C2 = 9400$), the percentage increases significantly to 46.5% (relatively 69.1% improvement). The suppression to Byzantine behavior is not that good. When the size of clients is C1, the service served by Byzantine behavior is even more than the standard result; however when increase the number of clients to C2, the result is much better: from 42.1% with option C1 decrease to 27.3% (relatively 35.2% decrease) with option C2. All these data tell us:

   (1) Our model is very good to bring more good services to the system, and with the increase of the number of clients (service requests), the improvement is even better.

   (2) The effect of suppressing the Byzantine service is not as good as promoting the good service. But the effect will appear when more service requests are served.

   (3) More requests means more time and more information to let the system to get convergent, because more feedback and observation can be received.

   From the above result, we can see that our approach is convergent, for the result gets more improvement when choosing larger number of requests, no matter from the view of increasing good service or the view of depressing the Byzantine service.

   (4) It can expected, when increase the number of requests, the result will get even more competent, because when the system get convergent, most peers know the good peers, and most the service requests will get good service from these good peers.

(2) **Anti-boaster.** In figure 5(b) the boaster will exist, and the bad peer can act more intelligently. From figure 5(b), we can see that the percentage of good service increases from 27.5 to 32.8% (relatively 19.3% improvement) with C1, and to 37.5% (relative 36.4% improvement) with more requests C2. The improvement is quite a bit less than figure 5(a), and the effect on suppressing the Byzantine service is even weaker. However, considering 80% peers are intelligent bad peers, and malicious recommendations and boasters exist (a highly untrusted environment), we still can say our approach is effective and robust for the extremely untrusted computing environment. The fact behind these data is, the highly untrusted environment will

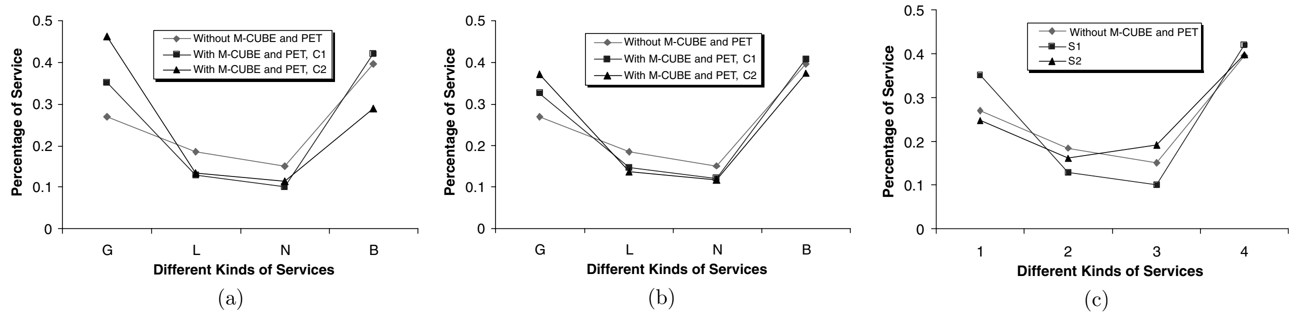


Figure 5. Improvement of our model. (a) No malicious recommendations and boasters, (b) Both malicious recommendations and boasters exist, and (c) Different number of peer servers.

cost more time for the system to get convergent, but can not prevent the trend to convergency.

(3) **Scalability.** In figure 5(c), in additional to the standard results, two experiments are conducted: one is with larger size of peer servers S1, and the other is with the smaller one S2. No boasters exist in this experiment. The settings and the reasons why choose this have been discussed in setup part. From the figure 5(c), we can see that, only about 25.0% good service the system gets with the small scale, much less than the result with larger scale 35.5%, and even less than the standard result 27.5%. It is because when the number of peer server decrease to $\frac{1}{10}$, the number of requests also decrease to $\frac{1}{10}$. Obviously we can see that in the smaller scale, the system is not convergent, so that the performance is not good. But on the other hand, this also tell us that larger scale system can help to improve the performance. So the model is scalable. Of course, there should be one saturate point for the increase of the system scale, which is one of our future work. From figure 5(c), it can be seen that with the increasing of the client number from C1 to C2, the good service percentage increases from 35% to 48%. All these imply that the experiment results will be better if the scale of experiment increases. Thus we expect that in the large scale P2P community, our approach will be great promising.

5.3. Relationship among trustworthiness, reputation, risk and ratio.

In PET, the trustworthiness is derived from the reputation and risk. With the support of the trustworthiness evaluation, M-CUBE can change the ratio of the currency dynamically, which makes our currency model effective and accurate. In this subsection, we will analyze how the reputation and risk combine together to make the trustworthiness more accurate; we can also see how the trustworthiness affects the currency ratio (In the following all the ratio is referred to currency ratio).

Setup: We choose three kinds of peers for the consideration: G-peer, B-peer (the representative for the bad peers include L-peer, N-peer, and B-peer), and D-peer. These three peers are picked up from one peer's history table randomly.

Discussion: The x-axis in every sub-figure of the figure 6 represents the time flow, and the y-axis stands for the value of components. Let's focus on the trustworthiness first. For the G-peer, the trend of the trustworthiness in figure 6(a) is increasing overall. It can been seen that there are some fluctuations. This is because of the affect of recommendations. The malicious recommendations will disrupt the trustworthiness; even the correct recommendations will also delay the convergent process, because at the beginning the G-peer's trustworthiness value will be low, so the recommendation value for the G-peer will be also low. However, the fluctuations tend to disappear as time goes on, because when more interaction-based information has been collected, the role of the recommendation becomes weaker (the risk for the G-peer is always zero, which brings no effect for the fluctuations). For the B-peer, the trend of the trustworthiness is decreasing earlier and suddenly in figure 6(b), which is incurred by the risk evaluation. Because the B-peer always provides Byzantine services, its risk will always be one for its cooperators once the their cooperation begins. The risk will make the trustworthiness drop suddenly and quickly, which is helpful to recognize the B-peer. For the D-peer (figure 6(c)), the trend is fluctuating first, then decreasing later. Different from the G-peer, the overall tendency of the fluctuation is decreasing, while G-peer's is increasing. This is because in addition to the effect of the recommendations, the fluctuation of the D-peer is also because of its dynamic change of the behaviors, which incurs its reputation to decrease. When the cooperation with D-peer begins, the risk is starts having effect and makes the trustworthiness drop suddenly. Different from B-peer, the drop of the trustworthiness is less sharp, but enough to reveal the D-peers. From the above analysis, we can see that *the risk evaluation is great helpful to recognize the bad and dynamic peers, but no effect for the good peers* (For the good peers, our PET model is the same as the reputation model, because the risk is always zero.)

6. Related work

The notion of "trust management" was first coined by Blaze et al. in their seminal paper on decentralized trust management [4], which addresses the authentication of each client request from the perspective of servers (service provider) in terms of security policies, credentials, and trust relationship. This is different from what we proposed, where the trustworthiness of both sides are considered in general, rather than on each individual service request. In the computer science

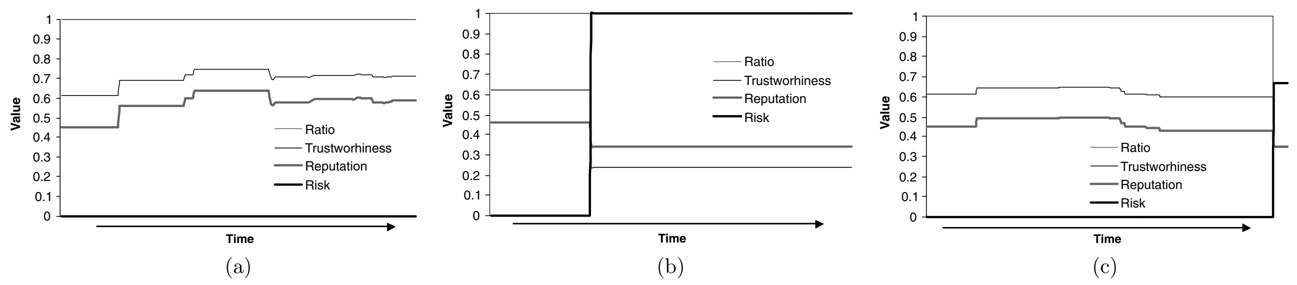


Figure 6. Relationship among the trustworthiness, reputation, risk and ratio The setting options of this experiments are: B2, M2, P1, S2, and W4. (a) G-peers, (b) B-peers, and (c) D-peers.

literature, Marsh (1994) is the first one to introduce a computational model for trust in the distributed artificial intelligence (DAI) community [16]. However, he did not model reputation in his work. Mui [17] gives a detailed computational model of trust and reputation. In Mui's model, reputation is well modeled, but it doesn't take the risk into consideration. Recently, in the P2P domain decentralized reputation management schemes like P2Prep [7], EigenTrust [13] appear. P2Prep provides a protocol complementing existing P2P protocols. EigenTrust assumes that trust is transitive and address the weakness of the assumption and the collusion problem by assuming there are pre-trusted nodes in the systems. However, the objective of these reputation-based systems are different from that of our effort enforcing self-policing trustworthiness over other peers, rather than obtaining a global consistent trust value for each other peer. However, we believe that our work will benefit from these reputation-based systems very well. Numerous economic models including microeconomics and macronomics principles for resource management have been proposed in the literature [2,5,6], and various criteria are used for judging effectiveness of an economic model, including social welfare, stability, computation efficiency. However, different from the M-CUBE model proposed here, none of them take the trustworthiness into consideration, also to our knowledge, few of them consider the dependability of the economic model to possible DDoS attacks. Several research systems have explored the use of different economic models for trading resources in different application domains: CPU cycles, storage, database query processing, and computing. Currency and economics based resource management has been extensively studied in the past [9,21]. Zhao and Karamcheti [21] give an approach building on the concepts of tickets and currencies to express resource sharing agreements. Our work is different from these due to our focus on the trust and security. To our knowledge, the SHARP Infrastructure [9] is the closest work related to us. But the details of how to use the currency are different. Different from [9], our infrastructure disagrees with the boaster. PPay [20] is a micropayment-based mechanism for P2P resource sharing and it guarantees that all coin fraud is detectable, traceable and unprofitable. This work complements our work.

7. Summary

In this paper we have presented a novel economic model M-CUBE combining the trust model PET to provide a fundamental mechanism for P2P resource trading in an open environment. The uniqueness of this approach is in its ability to seamlessly integrate the trustworthiness and dependability of peers into currency ratio floating for resource trading. Our analysis show that this model is effective and robust under the untrusted computing environment. To this end, we believe that the proposed model provides a general and flexible infrastructure to build most of high level resource management required by P2P computing, such as resource coallocation and quality of service (QoS) control.

References

[1] E. Adar and B. Huberman, Free riding on gnutella, First Monday **5**(10) (2000).

[2] Y. Amir, B. Awerbuch and R. Borgstrom, A cost-benefit framework for online management of a metacomputing systems, in: *Proc. of the first Inernational Conference on Information and Computational Economy* (1998).

[3] P. Barford and M.E. Crovella, Generating representative web workloads for network and server performance evaluation, in: *Proceedings of Performance '98/ACM SIGMETRICS '98* (1998).

[4] M. Blaze, J. Feigenbaum and J. Lacy, Decentralized trust management, in: *IEEE Symposium on Security and Privacy* (1996).

[5] R. Buyya, D. Abramson and J. Giddy, A case for economy grid architecture for service-oriented grid computing, in: *Proceedings of the 10th IEEE International Heterogeneous Computing Workshop* (2001).

[6] B. Chun, Market-based cluster resource management, Ph.D. thesis, Department of Electrical Engineering and Computer Science, UC Berkeley (2001).

[7] F. Cornelli, E. Damiani, S.D.C. Vimercati, S. Paraboschi and P. Samarati, Choosing reputable servents in a P2P network, in: *Proc. of the 11th International World Wide Web Conference* (2002).

[8] J. Douceur, The Sybil Attack, in: *Proc. of the 1st International Workshop on Peer-to-Peer Systems (IPTPS′02)* (2002).

[9] Y. Fu, J. Chase, B. Chun, S. Schwab and A. Vahdat, SHARP: An Architecture for secure resource peering, In: *Proc. of the 19th ACM Symp. on Operating Systems Principles (SOSP-19)* (2003).

[10] R. Gupta and A.K. Somani, CompuP2P: An architecture for sharing of compute power in peer-to-peer networks with selfish nodes, in: *Second Workshop on the Economics of Peer-to-Peer Systems*, Harvard University (2004).

[11] D. Hausheer, N.C. Liebau, A. Mauthe, R. Steinmetz and B. Stiller, Token-based accounting and distributed pricing to introduce market mechanisms in a peer-to-peer file sharing scenario, in: *Third International Conference on Peer-to-Peer Computing (P2P′03)* (2003).

[12] H. Junseok, The economics of dynamic bandwidth transaction service: toward pricing modeling, in: *MSI Workshop on Modelling*, University College London, (2001) pp. 18–19.

[13] S. Kamvar, M.T. Schlosser and H. Gacia-Molina: The eigentrust algorithm for reputation management in P2P networks, in: *Proc. of the 12th International World Wide Web Conference* (2003).

[14] Z. Liang and W. Shi, Enforcing cooperative resource sharing in untrusted peer-to-peer environment, Technical report MIST-TR-2004-014, Department of Computer Science, Wayne State University (2004).

[15] Z. Liang and W. Shi, Analysis of recommendations on trust inference in the open environment, Technical report MIST-TR.-2005-002, Department of Computer Science, Wayne State University (2005).

[16] S. Marsh, Formalising trust as a computational concept, Ph.D. thesis, University of Stirling (1994).

[17] L. Mui, M. Mohtashemi and A. Halberstadt, A Computational model of trust and reputation, in: *Proceedings of the 35th Hawaii International Conference on System Sciences* (2002).

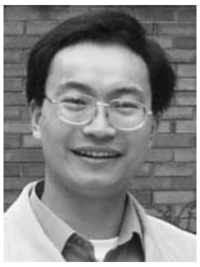
[18] J. Ravi, Z. Liang and W. Shi, A Case for peer-to-peer web server sharing, Technical Report MIST-TR.-2003-010, Department of Computer Science, Wayne State University (2003).

[19] Y. Wang and J. Vassileva, Trust and reputation model in peer-to-peer networks, in: *Third International Conference on Peer-to-Peer Computing (P2P′03)* (2003).

[20] B. Yang and H. Carcia-Molina, PPay: Micropayments for peer-to-peer systems, in: *Proceedings of ACM CCS′03* (2003).

[21] T. Zhao and V. Karamcheti, Enforcing resource sharing agreements among distributed server clusters, in: *Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS)* (2002).

Zhengqiang Liang is a Ph.D. Student in computer science at Wayne State University. His current researches focus on trusted and cooperative resource sharing in the open environment, and computer economics. He received his B.S degree in 1997 and M.S. degree in 2001 from Harbin Institute of Technology (HIT) in China, both in Computer Science and Engineering.
E-mail: sean@wayne.edu



Weisong Shi is an Assistant Professor of Computer Science at Wayne State University. He received his B.S. from Xidian University in 1995, and Ph.D. degree from the Chinese Academy of Sciences in 2000, both in Computer Engineering. His current research focuses on dynamic Web content delivery, trusted resource sharing in peer-to-peer systems, mobile computing, and wireless sensor networks. Dr. Shi has published more than 40 peer-reviewed journal and conference papers in these areas. He is the author of the book "Performance Optimization of Software Distributed Shared Memory Systems" (High Education Press, 2004). He has also served on technical program committees of several international conferences, including the chair of poster track of WWW 2005. He is a recipient of Microsoft Fellowship in 1999, the President outstanding award of the Chinese Academy of Sciences in 2000, one of 100 outstanding Ph.D. dissertations (China) in 2002, "Faculty Research Award" of Wayne State University in 2004, the "Best Paper Award" of ICWE'04 and IPDPS'05. He is a member of ACM, USENIX, and IEEE.
E-mail: weisong@wayne.edu