

# Enforce Cooperative Resource Sharing in Untrusted Peer-to-Peer Computing Environment

Technical Report: MIST-TR-2004-014

Zhengqiang Liang and Weisong Shi

Mobile and Internet Systems Group  
Department of Computer Science  
Wayne State University  
{sean,weisong}@wayne.edu

## Abstract

Peer-to-Peer (P2P) computing is widely recognized as a promising paradigm for building next generation distributed applications, ranging from large scale scientific applications to mobile ad hoc information sharing, by federating dispersed pools of geographically distributed resources under loosely coordinated control. However, the autonomous, heterogeneous, and decentralized nature of participating peers across multiple administrative domain introduces the challenge for resource sharing in such an environment: how to make the peers profitable in the decentralized resource sharing under the untrusted P2P environment. To address the problem, in this paper 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 an untrusted P2P computing environment. *PET* is a flexible trust model that can adapt to different requirements, and provides the solid support for the currency management in *M-CUBE*. With the help of the trust management and the merits of the economics, *M-CUBE* provides a novel self-policing and quality-aware framework for the sharing of multiple resources, including homogenous 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 the system with large scale.

**Keywords:** Cooperative, Heterogeneous, Resource Sharing, Untrusted Environment, Peer-to-Peer, Economic model, Trust Model.

## 1 Introduction

Peer-to-Peer (P2P) computing — federated sharing of dispersed pools of geographically distributed computing re-

sources 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 from participating peers, and coordination 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 [2, 40], 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 are 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 [14].

In this paper, we propose our approach combining 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 seamlessly 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-level 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. In the following section, we provide an overview of our approach first. 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 Section 6 and Section 7 respectively.

## 2 Overview

We first give a brief description about the service-oriented architecture, which provides a foundation for our system. After that, five problems to be addressed in our system are listed, followed by a general overview of our design.

### 2.1 Service-oriented View

Traditionally, the term “resource” in resource sharing has been interpreted narrowly as denoting a physical entity, such as computer, network, or disk storage systems. In contrast, we envision that the notion of resource should be in a more generic sense to denote any capability that may be shared and exploited in a networked environment, i.e., *service*. In the rest of this paper, we shall use “service” and “resource” interchangeably. Figure 1 shows a general scenario

of P2P resource sharing over the Internet, where multiple peers are distributed across multiple administrative domains (also known as autonomous systems (ASes)). The scenario can be used in the Pure P2P environment, where all the nodes are peers; at the same time it can also be applied in the Hybrid P2P environment, where the nodes provide service (we call such a node the peer server *PS*) form the P2P community, while the nodes asking for the services (we call such a node the client) locate outside the P2P community. For the Hybrid architecture, we use dotted line to connect clients to peers in the figure. Note that some ASes may have multiple peers, e.g.,  $A_3$  has two peers  $P_3$  and  $P_4$ . This will demand the system to provide flexible security and trustworthiness control to treat intra-AS and inter-AS in different ways, which is within the capacity of our approach: M-CUBE and PET.

### 2.2 Problems of Resource Sharing

Here, we list the problems related to resource sharing in an open P2P environment.

- **Heterogeneity** The heterogeneity of resources makes the multiple-resource sharing difficult, because of lacking of a formal metric for the trading among different resources.
- **Untrustedness** Enforcing a cooperative, adaptive, and anti-maliciousness P2P sharing environment on top of an untrusted and private P2P community is really a challenge.
- **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.
- **Autonomy and Cooperation** Peers usually belong to different administrative domains which may have different local policies. How to effectively and efficiently integrate these local policies and general resource sharing is a challenge.
- **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.

### 2.3 Overview of Our Approach

We conjecture that the fundamental problem of P2P resource sharing is a trustworthiness mechanism for resource trading, as the dependable trading to world economics. Based on

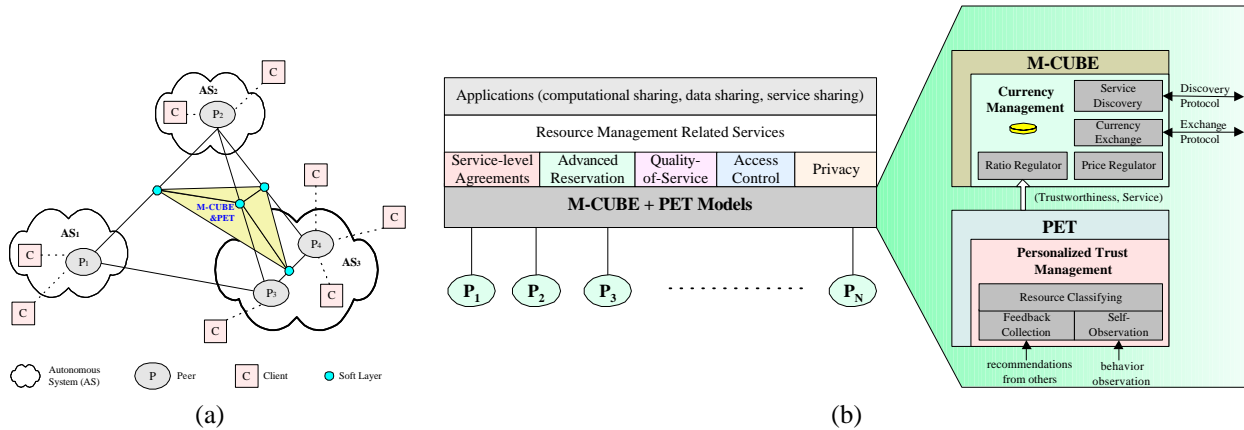


Figure 1: (a) A general example of peer-to-peer resource sharing; (b) Overview of the system.

these, a lot of high level resource management related services, such as service level agreements, access cost negotiation, resource coallocation, advanced reservation, quality-of-service scheduling, dynamic adaption and reconfiguration, and even access control (partially) 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(b), 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. The weights of the reputation and risk are adjustable according to different environments and requirements. Integrating with the risk evaluation distinguishes PET from the previous work [10, 20, 29, 46].

### 2.3.1 Assumptions

Before describing the PET and M-CUBE model, we give the basic assumptions first:

- Each peer has a relative unique and stable ID. This will make reputation and trustworthiness make sense.

- Coordinated access to diverse and geographically distributed resources is valuable for participating peers.
- Each peer has an associated public-private key pair to make its currency unforgeable.
- Most peers in the system need the cooperation so as to gain profitable through sharing the resources.
- Each peer is selfish.
- Each peer has a pair of public/private keys for peer authentication and building secure communication channel. Peers use PKI for public key distribution.

## 3 PET Design

PET is the underpinning of our system, which provides the trustworthiness to trigger M-CUBE to evolve. Before depicting the PET model, we list four principles for the design first:

- P.1** Peer will always trust itself.
- P.2** Bad behavior makes the trustworthiness value drop faster and good behavior increases the value slower, which make the new joiner not to be preferred.
- P.3** If a peer continually behaves badly, it will be bad peer prone.
- P.4** The recommendations from others will not dominate the calculation of the trustworthiness value, but it will gain more weight when no direct interactions happen before.

### 3.1 Trustworthiness

PET [24] underpins M-CUBE through providing accurate trustworthiness value calculation. Trustworthiness is service-specific. One peer can have different trustworthiness values corresponding to different services in eyes of other peers, so PET actually provides the *trustworthiness, service* pair for M-CUBE. In PET the trustworthiness  $T$  is directly derived from two parts: reputation  $R_e$  and risk  $R_i$ , as shown

in the upper part of Figure 2.  $W_{Re}$  and  $W_{Ri}$  are the weights of  $R_e$  and  $R_i$  respectively, which are adjustable according to different environments and requirements. Reputation 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 ( $I_r$ ).  $W_{Er}$  and  $W_{Ir}$  are their corresponding weights. Recommendation is the referred opinions from other peers, which is collected by the *Feedback Collection* component in PET. Interaction-derived information is the self-opinion resulting from the direct interaction. This kind of information is from feedbacks from peer’s own agent [37] (also collected by the feedback collecting component) and behavior pattern information collected by the *Self-Observation* module. 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 of the short-term behavior. Integrating the risk evaluation into the trustworthiness calculation is one distinguished characteristic of the PET model. All values of  $T$ ,  $R_e$ ,  $R_i$ ,  $E_r$ , and  $I_r$  are values from interval  $[0, 1]$ .

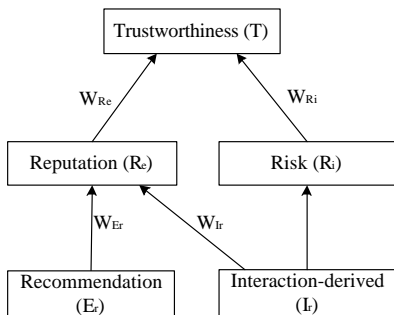


Figure 2: Derivation of the trustworthiness.

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 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 service, low-grade service, no response, and Byzantine behavior, in P2P system as listed in Table 1, so that PET can be applied for most resource sharing cases by just modifying its feedback collecting and resource classifying components, which we argue is application-specific. To be worth noting 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. Though this strategy puts more strict limitation on the peer’s behavior, it is good for the model to find good peers considering both user’s intention

and the physical condition. Finally the dynamic behavior in the table is introduced to simulate the dynamics of the P2P system, and is used in the simulation design.

According to different requirements, we can assign different weights to the reputation and risk components, through which PET can meet most demands, no matter preferring to the long-term assessment or caring for the short-term assessment. Equation 1 describes the derivation of the trustworthiness  $T$ :

$$T = \begin{cases} \alpha \times R_e + (1 - \alpha) \times (1 - R_i), 0 \leq \alpha \leq 1 \\ R_e, \text{ if } R_i \text{ is NULL} \\ R_i, \text{ if } R_e \text{ is NULL} \\ 0.4, \text{ both } R_i \text{ and } R_e \text{ are NULL} \end{cases} \quad (1)$$

where  $R_e$  is the reputation value, and  $R_i$  is the risk value. The values of  $T$ ,  $R_e$ , and  $R_i$  are all from 0 to 1. Here  $W_{Re} = \alpha$  and  $W_{Ri} = 1 - \alpha$ . If we set  $\alpha = 1$ , that means the weight of the risk is 0, then PET will degenerate to the traditional reputation system. However our simulation results show that risk evaluation is a very helpful component to build the trust model. Normally when a 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), that is, set a lower value to  $\alpha$  (e.g., 0.3), 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 also the safe recommended value. For a complete new peer who does not have any related  $R_e$  and  $R_i$  information, its trustworthiness is set to 0.4. Normally, this value is used only when the peer is newcoming and needs to find some neighbors (partners), such as at the very beginning of the system. It is worth noting that, the  $T$  defined in Equation 1 is associated with only one service type. Peers who provide multiple services can have multiple trustworthiness values.

### 3.2 Reputation Model

In the following model, we call the peer to evaluate other peers a *valuer*, the peer to be evaluated a *valuee*, and the peer that sends the trustworthiness value of the known peers to others the *recommender*. For example, when peer  $A$  tells peer  $C$  the trustworthiness value of peer  $B$ ,  $A$  will be the *valuee* of  $B$ , and the recommender of  $C$ .

We classify services provided by the peers into the following four categories, as shown in Table 1. We formalize the quality set as  $Q = \{G, L, N, B, D\}$ . Correspondingly, the peers providing  $G$  service is called **G-peer**. Then we have **L-peer**, **N-peer**, **B-peer**, and **D-peer** similarly. This classification is flexible enough to apply to any resource sharing, but also with coarse grain. More subclasses can be introduced if necessary. All three  $L$ ,  $N$ , and  $B$  services are bad services, and will cause the *valuer* to decrease the *valuee*’s score. Considering the dynamic behaviors of the peers in the real P2P community, the Dynamic quality ( $D$ )

Peer Behaviors	Definition
Good (G)	Provide services as good as expected.
Low Grade (L)	Provide correct services, but with some degradation, e.g., delay for service.
No Response (N)	Reject any incoming service requests.
Byzantine Behavior (B)	Give the wrong or even malicious response for the incoming requests.
Dynamic Behavior (D)	Change the behavior among G,L,N,B one after another and repeatedly.

Table 1: Five different peer behaviors.

is also introduced in the simulation in addition to the four qualities (G, L, N, B), and the corresponding peers are called **D-peers**. For the D-peer, it will change its behavior among G, L, N, and B repeatedly and uniformly, so 75% of the behavior of D-peer is bad, and its score actually also will decrease gradually. For the *valuer* there is a map  $h$  from Q to a score for one cooperation:

$$h(x) = \begin{cases} S_1 & , x = G, S_1 > 0 \\ S_2 & , x = L, S_2 < 0 \text{ and } |S_2| > S_1 \\ S_3 & , x = N, S_3 < S_2 \\ S_4 & , x = B, S_4 < S_3 \end{cases} \quad (2)$$

For example, let  $h(B) = -6$ , which means when the *valuee*'s service category is known as Byzantine behavior, *i*'s total score  $S$  will be dropped by six. As shown in Equation (2), the bad behaviors (L, N, B) will lead to more extent of decrease of the score than the extend of increase for the good behavior, and the B is the most harmful action which cause the most severe decrease. This alters the peer not to act badly. The score is used to calculate the reputation, as seen in Equation 3. Simply we can choose a constant value for  $S_1$  to  $S_4$  (Note,  $S_2, S_3,$  and  $S_4$  are negative), but also we can adjust these values with a complex adaptive mechanism.

Reputation value is the historical accumulation for *valuee*'s past behavior from the *valuer*'s viewpoint. It will reflect the overall quality of the peer for a long time. 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  $\alpha$  ( $\alpha > 0.5$  for example) to make the trustworthiness preferring to the reputation value. Reputation is derived from  $E_r$  and  $I_r$ , as shown in Equation 3:

$$R_e = \beta \times E_r + (1 - \beta) \times I_r, 0 \leq \beta \leq 1 \quad (3)$$

where

$$E_r = \frac{\sum T_e}{N_e}, \quad I_r = \begin{cases} 1, & S \geq T_{good} \\ \frac{S}{T_{good}}, & 0 < S < T_{good} \\ 0, & S \leq 0 \end{cases}$$

Here  $W_{E_r} = \beta$  and  $W_{I_r} = 1 - \beta$ .  $\sum(T_e)$  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.  $T_{good}$  is the score threshold to normalize the score.  $S$

is the total score. For example, when  $T_{good}$  is set to 100 and the valuee's score is higher or equal to 100, its  $I_r$  will be set to one.

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 may have different views on the same resource provider because different peers may have different situation-specific criteria and requirements for the sharing.
2. Peer's behavior can change dynamically, which implies that we can not compute the trustworthiness relying much on the recommendations from others.
3. Fraudulent recommendation, especially the collusion on the recommendation is very difficult to handle if the trustworthiness calculation relies much on the recommendation.

However, as mentioned before, it is not a good answer to ignore the recommendation. Assigning it a lower weight  $\beta$  is good for the solution, which is supported by the simulation results in Section 5.

### 3.3 Risk Model

Reputation is the accumulative value for the past behavior and reflects the overall evaluation for the *valuee*. However, it 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.

The risk  $R_i$  is defined as the ratio between bad services and total services, as shown in Equation 4, where B, N and L are the service qualities defined in Table 1 and  $h(i)$  is the score for the cooperation with service quality  $i$  define in Equation 2.  $S_w$  is the size of the window, and  $N$  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. From Equation 2 we can know that the value of  $|h(B)|$  is largest unit to change the score, so with it we can normalize  $R_i$  to be a value within the range  $[0,1]$ . 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 the Sybil attack [13], but actually not. Once a

peer behaves badly, the risk value will increase significantly, so that the door will close very fast for the bad peers. With the same reason, even the complete new stranger hold a relatively high trustworthiness value 0.5, the risk evaluation can reduce the threats if the risk component has a high weight, which will be validated in in the Section 5.3.5.

$$R_i = \begin{cases} \frac{\sum_{i=B,N,L}(N_i * h(i))}{h(B) * \max(S_w, N)} & \max(S_w, N) \neq 0 \\ 0 & \max(S_w, N) = 0 \end{cases} \quad (4)$$

For the implementation, a risk window is employed to limit the assessing range. The smaller the window size is, the more the shorter-term assessment is favorite by the trustworthiness calculation. In the simulation, we will see the results about the effects of changing the window size for the trustworthiness value.

With the window shifting forward, the risk value reflects the fresh statistics of the *valuee*'s recent behaviors. 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, decay function is used in [4]. However, it is difficult to choose a unique decay function for all peers, because different peers have different behavior patterns. In PET, risk window is a more graceful alternate. The risk contribute to the trustworthiness with the *valuee*'s most recent behaviors, which integrates the temporal factors into the trustworthiness value. Every *valuee* has the individual risk value, which makes the temporal factor more precise than the decay function. To reduce the risk from the cooperation, users can focus more on the risk value  $R_i$  by assigning it a high weight. Yet this will decrease the availability of the resources, because the lower the risk is requested, the less resources are qualified to be used. The user can make a tradeoff between the risk and the resource availability by adjust the weight of the risk.

Using risk evaluation, the risk sensitive users can find the bad peers much earlier than just using the reputation value, which is illustrated in Figure 3. In this figure, x-axis represents the time, and y-axis shows the behavior of the peer. The line reflects the variation of the peer's behavior. At time  $t_0$ , the peer starts to behave badly. The bad peer will not be discovered until time  $t_2$  when only the reputation value is taken into consideration. If the risk evaluation is federated and paid enough attention, the time will be efficiently shortened to time  $t_1$  due to the high risk value  $R_i$ .

## 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.

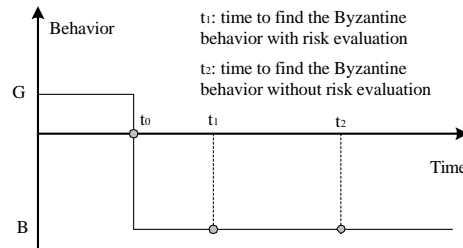


Figure 3: Risk evaluation helps to find the peer with dramatic spoiling earlier.

### 4.1 World Economic Model

The ability of trading and price mechanisms combine local decisions by diverse entities into globally effective characteristics, and imply their value for organizing computations in large systems such as P2P system. Based on the success of economic institutions in the real world as a sustainable model for exchanging and regulating resources, goods, and services, we propose the M-CUBE model, a computational economics framework. Several features of world economic model motivate the design of our own economic model for trusted resource sharing in P2P environment:

- (1) Each country has their own currency;
- (2) The consumer (buyer) needs provider (vendor)'s currency for trading, rather than its own currency;
- (3) Currency exchange ratio is floating dynamically to stabilize the world's economics;
- (4) Countries must have incentives for exchange. That means the two trading sides must be interested in counterpart's goods or services;
- (5) Some countries may have some restrictions on trading with some specific countries based on previous reputation;
- (6) After finishing the trading, both side can get the information of the cooperator's quality and adjust the reputation of its cooperator. The trust and reputation information will be used for the next possible trading.

### 4.2 Currency Model

Inspired by the above features, 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.

#### 4.2.1 Currency Model

M-CUBE is built upon currency-based mechanism, where the uniqueness of M-CUBE is each peer has its own cur-

rency. Unlike many of previous work, in addition to associating the currency with physical resources directly, such as CPU and disk, M-CUBE also associates currency with application-level services directly. For example, in the P2P Web server sharing application [38], the currency is related to the service for the HTTP requests.

**Pricing and Ratio** In M-CUBE, pricing is the first problem that needs 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 of pricing and trading. Peer A wants to share its 100G harddisk, and peer B wants to share its 2GHz CPU. Assume in the real market A's harddisk 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)<sup>1</sup>. 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 1G 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

<sup>1</sup>25MHz CPU is  $\frac{1}{80}$  of 2GHz CPU. From the application view, it represents  $\frac{1}{80}$  CPU usage.

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 [16, 17, 19, 25, 30, 51], which are the good complementary work for our extensions to decide the price independent on the real economic market.

**Currency Format and Management** Figure 4(a) shows the logical relationship among peers, where each peer issues its own currency according to its contributed resources. When two peer exchange their currencies, there is a exchange ratio  $R_c$  that they agree with. Initially, since every currency is corresponding to \$1, so  $R_c$  is equal to one, as shown in Figure 4(a). After sometime,  $R_c$  will be adjusted according to the trustworthiness value provided by PET. It is the function of trustworthiness value  $T$  and old value of  $R_c$ :  $R_c = f(T, R_c)$ . A simple definiton of  $f$  is  $R_c = T$ . That is, just use the trustworthiness as 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. In the principle, 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  $P2$ . When  $P1$  issues its own currencies, it promises to share its resources related to those currency at the same time. 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 normal days. 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. More details about the boaster will be depicted in the subsection 4.2.2. 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 4(b). *Type-Vec* 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

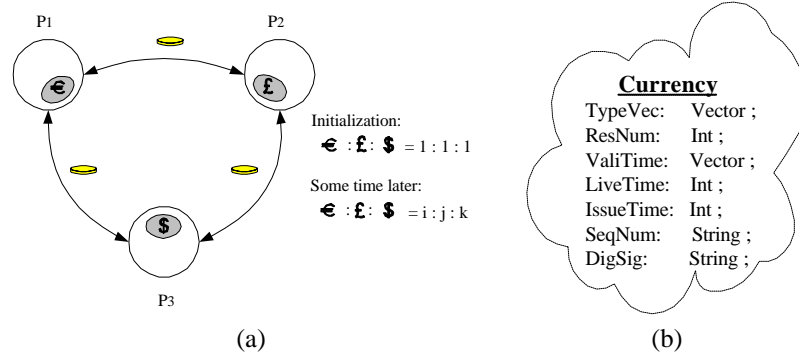


Figure 4: An overview of the currency model: (a) Evolution of the M-CUBE, (b) the format of the currency.

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 so 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 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.

**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, and (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, as shown in Figure 5(a). The requester sends out a request ( $C_{num}, C_{type}, R, V$ ) to its co-

operators (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 small world phenomenon, it is expected that after several hops the wanted currency can be discovered. In M-CUBE the maximum number of hops is set to six [31]. 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, and (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 want 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 request 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 provider). Then one peer from the candidate list is chosen to proceed the currency exchange, whose details are described in the following paragraph.

If the currency exchange can not be fulfilled (for example, the exchange ratio is very high for the requester peer, so that it doesn't want to continue exchanging), another peer from the list is chosen to continue the exchange. When the requester peer doesn't have the right currency as the respon-

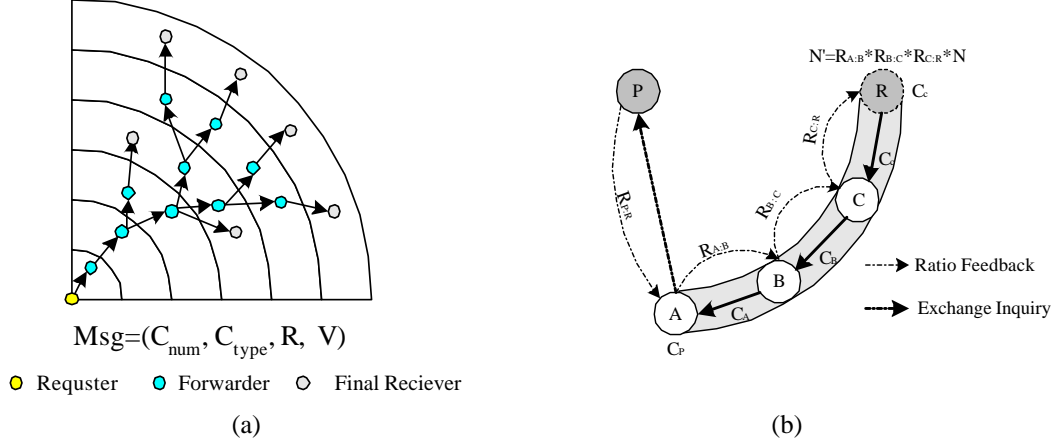


Figure 5: The currency discovery protocol: (a) multicast protocol, and (b) an example of exchange chain.

der peer needs, it must try to get the currency the responder peer wants first. In a worse case, the requester may have to get several intermediate currencies to finally get the wanted currency which can be used for exchange. In this case, the exchange chain shapes up.

Figure 5(b) is an example of the exchange chain combining the delegation, where peer  $P$  possesses the resource the requester  $R$  wants. 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 tell  $R$  it just accepts  $C$ 's currency. Fortunately,  $C$  accepts  $R$ 's currency (another multicast to find who has  $C$ 's currency). Now  $R$  can build a 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 peer agree all the ratio from the peers in the chain. So during the formation of the chain,  $R$  just keep the ratio reported from  $A$ ,  $B$ , and  $C$ . 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 gets  $P$ 's currency from  $A$ . One detail is, when  $A$  gets  $R$ 's exchange request finally,  $A$  will inquiry  $P$  the ratio  $R_{P:R}$  first, to defense the *exchange shortcut attack* which is depicted in Section 4.3. If  $R$  is qualified,  $A$  will give  $R$   $P$ 's currency. Actually finally  $N$  units of  $R$ 's currencies can exchange for  $R_{A:B} * R_{B:C} * R_{C:R} * N$  units of  $P$ 's currencies. For resource discovery in a large-scale network, we can resort to distributed hash table techniques [36, 43] to uniformly distribute the overhead of resource advertisement and lookup, which is beyond the scope of this paper.

**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. When the expected service is time-critical such as CPU service, peers can pre-exchange the wanted currency. Af-

ter 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 needed to be modified is to combine the exchange in the chain together). The requester sends its request of currency exchange to the candidate, and the candidate decides the exchange amount based on the exchange ratio. The pseudocode of the exchange protocol is shown in Figure 6(a). 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 Figure 6(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  $C_{P2}(T_1)$  to  $C_{P1}(T_1)$ . 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  $C_{P1}(T_i)$ , 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.

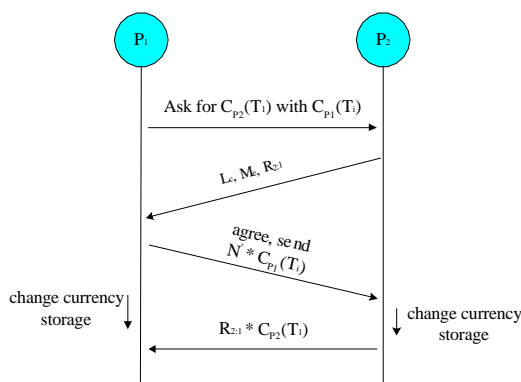
After  $P1$  receives the currencies from  $P2$ , it can ask for

```

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

```

(a)



(b)

Figure 6: The currency exchange protocol: (a) the protocol, and (b) the SUBEXCHANGE process.

$P2$ 's resource service immediately or later as long as within the lifetime of  $P2$ 's currency. If all things are legal,  $P2$  will provide the service for  $P1$ .  $P2$  can't reject  $P1$ 's request in order to maintain a good reputation in the community. Otherwise its bad reputation will lead to the ratio of its currency to  $P1$ 's decrease. In the Subsection 5.3.5, we can see how the ratios change for different peers with different behaviors. 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.

#### 4.2.2 Advantages of the Model

Benefit 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 prospective 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 of the resource let peers known when can find the available resource, which is necessary to make the system profitable. Actually from the view of resource quantity, it is impossible to make the system profitable, for one contributing one unit of resource is supposed to get back another unit of resource, however, when one peer facing the emergency or a complex task, and its resource is not enough, the value of resources from others is more than the normal days. This is the reason why the

resource sharing system can get benefits. The predictability is necessary for well planned and scheduled usage of resources, without which there is no way to cope with the emergency and complexity.

**Eliminating Free-rider** Free-rider [2] 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 it receives its currency, otherwise its credit will be decrease and finally will be kicked out from the community.

**DoS Attack Free** Since every 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 when the peer issues the currency legally. It is worth noting that when we say DoS attack free here, we refer to the possible attack resulting from our currency model, it is located in the application-level. The network-layer attack such as the TCP/IP SYN attack is not our major concern. Taking the Web server sharing case as example, one peer knows it is able to provide 100 *request/sec*, which won't bring itself a overload. When it issues 100 currencies (we assume  $1C/request$ ), the maximum load for the issuer is 100 *request/sec*.

**Anti-Boaster and Inflation** One malicious peer may issue more currencies than its actual resource capacity to try to get more benefits, or even issue the valueless currencies for it won't honor its currency with its service. We called such a peer a *boaster*. For example,  $P1$ 's capacity is 100GB disk and the market price of hard disk is 1G/\$, so normally it can just issue 100 currencies. However it issues 200 currencies to try to get 100 currencies free. In [15], this action is considered as legal and useful, but we doubt it because it will lead to the uncontrollable state of the system. In M-CUBE, this behavior is treated as illegal. In the real economic market, if the value of total currencies is larger than the actual value of total merchandize, it leads to the inflation and disorder the market trading. However, our currency mechanism

has a natural essence against 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 who have the currency of the boaster will ask 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 boosters, our approach is still can make the system profitable.

### 4.3 Threats and Extensions

The basic model works fine in the normal case, but needs some extensions to handle different possible attacks. Next, four possible attacks are discussed sequentially.

**Dare-to-Die attack** When we talk about the mechanism against the boaster, we assume every node want to get service from others, therefore maintaining a good credit in the community is very important. However, if the objective of a peer is disturb and destroy the system, so that punishment does not make any sense. We call this behavior *dare-to-die attack*. Actually we believe it is impossible to eliminate such attack fundamentally. However, we propose to use three strategies to weaken the attack. First, when one receives a new exchange request from others, it will base on the requester’s trustworthiness value to decide the amount of the request can get. When it is a new requester without any credit history, the exchange will be limited to a small amount. After the exchange, if the requester is a dare-to-die attacker, its failure in the service lead to its credit decreasing. Second, the provider will decrease the credit of requester when the provider doesn’t use the requester’s currency after a long time. Next time the same requester can get even less currency through the exchange. The second one is to prevent such case happen: provider doesn’t send request to the attacker, so it won’t know the attacker. Third, after a certain period LiveTime (Figure 4(b)), the currency in the attacker will be expired and be regenerated. Through these, we can limit the system’s damage to a low bound.

**Vampire Attack** In P2P community, several peers attack one peer cooperatively is a normal and severe problem. For M-CUBE, though we limit the exchange quota when exchanging with the stranger, but if the number of the stranger is enough, and consecutively they exchange with peer A, then peer A will still have risk in exhausting its currency, then unable to cooperate with others, that is, unable to provide service for others and ask service from others. we call this Vampire attack. In order to solve this problem, every peer will reserve a part of currency for its own use, which won’t be used for the exchange. So even facing with this

attack, peer won’t be shoot dead.

**Unidirectional Service Attack** In some cases, a peer will be able to ask for others’ service continually but without giving back its service. It happens when the victim won’t ask back the service from the profiteer directly or indirectly, because the profiteer will be located in a remote place (or low bandwidth, low process rate), which makes victim unbenefitable when asking back the service. Then the currency from the profiteer will be useless for victim because the victim won’t choose to ask for the profiteer’s service. But the profiteer may ask for the exchange again and again no matter it is malicious or not. Making the resource single-flow is the essence of the case.

Again, the three strategies in the first extension for weakening the dare-to-die attack, such as *limiting the exchange amount each time*, *decreasing the trustworthiness value of the peer whose currency is not used for a long time*, and *making the currency able to expired*, are helpful to address this problem as well.

**Exchange Shortcut Attack** In the basic currency model, exchange chain is allowed. But it enables the possibility of another attack that malicious peers ask other peers help to get the currency of third party, who refuses to exchange currency with the malicious people. We call this behavior *exchange shortcut attack*. Figure 7 shows a snapshot of the system which consists of three peers. Since the credit of P3

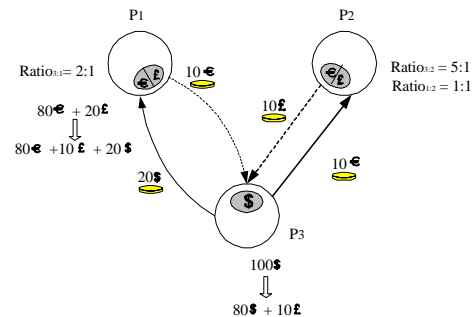


Figure 7: An example scenario for selecting profitable peer to exchange.

is low in P2, but is relative high in P1, so if P3 wants the currency  $C_2$  of P2, exchange with P1 to get the currency  $C_1$  first, then use currency  $C_1$  to exchange currency  $C_2$  will be much profitable then direct exchange currency  $C_2$  with currency  $C_3$ . But this action may be harmful to P1 because the currency of P3 in P1 is low-value for other peers when P3 is known as a bad peer by others.

In order to eliminate the profiteer, additional extension is needed. Every delegation will have to inquiry the currency owner if the delegation will represent the currency owner to exchange the currency to the requester. If the currency owner know the requester is a bad peer, then it will tell the delegation, and the delegation will reject the exchange request from the requester. If the requester is known as

good peer by the currency owner, the exchange will continue. Let's take Figure 5(b) as an example, after delegation A gets requester R's request, it will inquiry currency owner O to ask if R is a bad peer for O. If the answer is yes, A will reject R's exchange. This extension will make the legal chain exchange be able to proceed, but the bad peers unable to gain profit.

## 5 Experimental Analysis

The simulation is conducted in the context of P2P Web server sharing application [37], 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 bound 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 peer to peer network will have peak loads on their web sites simultaneously. PET will be integrated with the proposed M-CUBE model [23] for the application in the simulation. we assess the effects of different components of PET under various environment options first, then we analyze the ability of the approach to attract the good services and resist the bad services, and finally, we study the relationship of different components. In the following subsections, we will state the related concepts at first; then we depict the simulation settings; after that the discussions on the results are presented; finally we will give the summary of the results.

### 5.1 Concepts

Before presenting the experimental results, it is necessary to make some concepts clear.

**Cooperation:** When peer A uses B's currency to ask for B's service, and B satisfies A's request, we say A has one cooperation with B, or A cooperates with B.

**Active Cooperator:** We call the cooperators which are ready for the cooperation the active cooperators. When the cooperation is needed, active cooperators will be considered first.

**Inactive Cooperator:** The cooperators which are not ready for the cooperation are called inactive cooperators. They may be the peers exchanged the currency before, or the peers heard from others through the recommendations. They can also be the active cooperators before, but now are purged because of their bad trustworthiness values.

**Cooperated Cooperator:** When A has cooperated with B, B will be A's cooperated cooperator. A cooperated cooperator can be either an active cooperator or an inactive cooperator.

**Good-Known-Cooperator:** When the trustworthiness value of one cooperator is over a certain threshold (the value is set to 0.7 in our simulation.), the cooperator will be called the good-known-cooperator.

**Active Cooperator Table:** The information of the active cooperator will be stored in this table. The main contents include the cooperator ID and its corresponding number of the currency.

**History Table:** The information of all cooperators, including the active cooperators and the inactive cooperators, will be stored in this table. The main contents include the cooperator ID, trustworthiness, type, the number of the currency, etc. When one peer receives the recommendation from other peers, it will store the recommendation information into this table. When an active cooperator is purged, its information will be kept inside this table also, and a reselection of active cooperator will be based on this table in priority.

### 5.2 Experiment Design and Settings

The simulation is thread-based and written in Perl language. Table 2 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 [5]. 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 2. 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 2.

### 5.3 Results and Analysis

In the following subsections, we will present the simulation results with different experiment options. Before analyzing the results it is important to understand four metrics used to evaluate our model.

**Sensitiveness:** This metric is implied by the total number of cooperated cooperators in the history table. It can be expected that the more sensitive the model is, the more peers will be checked, because bad cooperators will be purged and

	Settings		Illustrations
Client Number	<b>C1</b>	4700 Clients	Small-size population.
	<b>C2</b>	9400 Clients	Large-scale population.
The Proportion of Peer with Different Quality (G:L:N:B:D)	<b>P1</b>	20% : 10% : 10% : 30% : 30%	To simulate the community with less good peers and all kinds of peers coexist.
	<b>P2</b>	20% : 0% : 0% : 0% : 80%	To simulate high dynamic community with many dynamic peers.
	<b>P3</b>	20% : 20% : 20% : 40% : 0%	To simulate the stable community without dynamic peers.
	<b>P4</b>	50% : 10% : 20% : 10% : 10%	To simulate a half-good community.
	<b>P5</b>	80% : 5% : 5% : 5% : 5%	To simulate a terrific community.
Malicious Recommendation	<b>M1</b>	Malicious recommendation	Spreading the distorted facts.
	<b>M2</b>	Correct recommendation	Spreading the true facts.
Weight of Different Components	<b>W1</b>	$\alpha = 0.3, \beta = 0.2$	Emphasizing $R_i$ and $I_r$ .
	<b>W2</b>	$\alpha = 0.3, \beta = 0.5$	Emphasizing $R_i$ and relying more on $E_r$ .
	<b>W3</b>	$\alpha = 0.7, \beta = 0$	Emphasizing $R_e$ and ignoring $E_r$ .
	<b>W4</b>	$\alpha = 0.7, \beta = 0.2$	Emphasizing $R_e$ and $I_r$ .
	<b>W5</b>	$\alpha = 0.7, \beta = 0.5$	Emphasizing $R_e$ and relying more on $E_r$ .
	<b>W6</b>	$\alpha = 1, \beta = 0.2$	Ignoring $R_i$ and Emphasizing $I_r$ .
Size of Risk Window	<b>S1</b>	4	Small window size. Based on last four services.
	<b>S2</b>	32	Large window size. Based on last 32 services.
Boaster	<b>B1</b>	No boaster	The peers issue their currency limitedly.
	<b>B2</b>	With boaster	The peers issue their currency unlimitedly.

Table 2: Simulation settings and their illustrations.

new cooperators will be chosen until all active cooperators are good. Generally speaking, high sensitiveness is favorite for the model, because it shows that the model is active. The sensitiveness will be studied in all the sub-figures (a) from Figure 8 to Figure 11, in which the x-axis is the number of the cooperated cooperators, and the y-axis is the the cumulative distribution of the percentage of the peers which have corresponding amount of cooperated cooperators in the x-axis.

**Hit Ratio:** The hit ratio metric is reflected by the number of the good-known-cooperators. Note, even some peers are the peers always providing the service with good quality, we can not say they are good-known-cooperators until they are discovered to be good (the trustworthiness value is over the threshold). Even without any cooperation, the good peers still can be perceived through the recommendation from others. In order to prevent malicious recommendations, the threshold to be a good cooperator is high so that it is very difficult for one malicious peer to fool other peers to take the bad peers as the good ones. The more the good-known-cooperators, the higher the hit ratio is. This is important because picking up the good cooperators from the community is the goal of our trust model. The hit ratio will be studied in all the sub-figures (b) from Figure 8 to Figure 11, in which the x-axis is the number of the good-known-cooperators, and the y-axis is the the cumulative distribution of the percentage of the peers which have corresponding amount of good-known-cooperators in the x-axis.

**Efficiency:** Efficiency is the average number of cooperations required to discover certain number of the good-known-cooperators. The lower number of this value, the

more efficient the model is. All the sub-figures (c) from Figure 8 to Figure 10 show the efficiency, in which the x-axis is the number of good-known-cooperators, and the y-axis is the average number of cooperations to find the corresponding number of good-known-cooperators of the x-axis.

**Applicability:** Applicability is reflected by the percentage of good services for clients' requests. The more good services the clients get, the more applicability the model has. This metric shows the effect of our approach most directly. The applicability will be studied in the sub-figure (d) of every figure from Figure 8 to Figure 10, Figure 11(c), and Figure 13 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. For each evaluation sub-scenarios (Subsection 5.3.1–5.3.4), we group the results of these four metrics into four graphs, and three in Subsection 5.3.2. In Subsection 5.3.5 we will analyze the relationship among different components, and in Subsection 5.3.6 we will focus on the applicability to study the effect of our whole approach.

### 5.3.1 Effect of Risk Evaluation

There are two ways to change the effect the risk value: changing the weight of the risk  $W_{Ri}$  and adjusting the risk window size  $S_w$ . This group of experiments explore how the change of  $W_{Ri}$  influences the model while remaining  $S_w$  the same.

**Setup:** In order to compare the result more precisely, 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

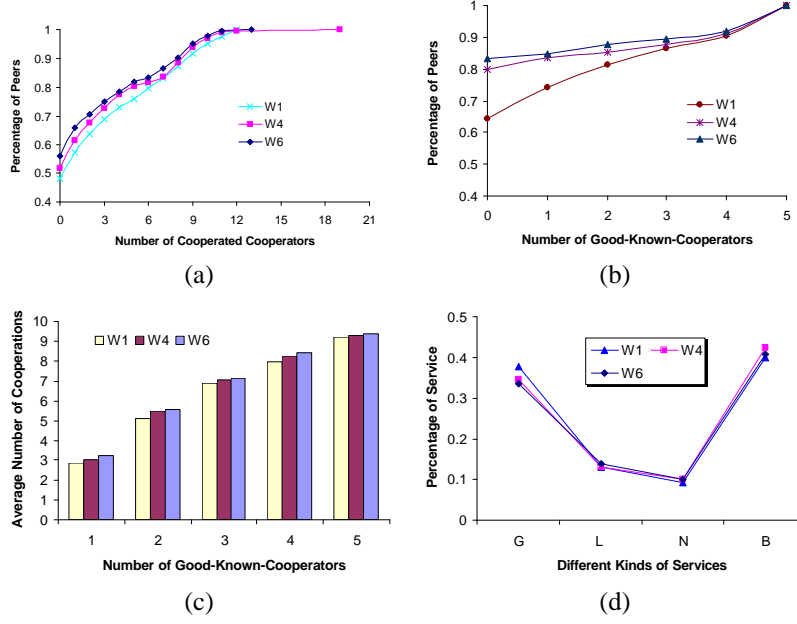


Figure 8: Risk evaluation with different weights. Other setting options of this experiment group are: B1, C1, M2, P1, and S2. (a) CDF of the total number of cooperated cooperators in the history table, (b) CDF of the number of good-known-cooperators, (c) Number of good-known-cooperators versus related average number of cooperations, and (d) Distribution of served services.

options are: C1, M2, P1 and S2.

**Discussion:** Intuitively, when there are bad peers (L, N, B, D), high value of  $W_{Ri}$  will help to get more satisfactory results compared with the model with lower value of  $W_{Ri}$ . In Figure 8(a) and Figure 8(b), the model with W1, the highest value of  $W_{Ri}$  in the three experiments, shows more sensitive and has more hit ratio than other two, which meets our expectation; we can also see that high value of  $W_{Ri}$  will make the model more effective from Figure 8(c). In Figure 8(d), the improvement for the applicability by the model with W1 is also the highest. All the results show that, in the P2P community where most peers are not good, the high value of  $W_{Ri}$  is helpful to improve the model.

### 5.3.2 Selecting the Weight of Recommendation

To select a suitable weight of recommendation is important for PET, we conduct this group experiments. Through the results, we will have the idea how to select the weight of the recommendation  $W_{Er}$  (defined in Equation 3) to decrease the negative effects of the malicious recommendation.

**Setup:** 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 options include C1 and S1.

**Discussion:** Let's focus on the lines and bars with M1 option in the following discussion. In Figure 9(a), when  $W_{Er}$  is set to W3 or W4, more than 25% peers will have three

cooperated cooperators. However, when  $W_{Er}$  is set to W5, the percentage will drop to about 15%. So the model will gain more sensitiveness when  $W_{Er}$  is set to be low. The effect is even much close to the model with correct recommendation. In Figure 9(b), it shows that setting  $W_{Er}$  higher will make the model's hit ratio increased: the number of peers having more than two good-known-cooperators increases from 10% to 15%. Figure 9(c) shows that with the lower value of  $W_{Er}$  (W4), our approach can gain the better efficiency than the case ignoring the recommendation (W1) and the case with higher  $W_{Er}$  (W5). Finally in Figure 9(d), 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 than both the case ignoring the recommendation (W1) and the case relying more on the recommendation (W5) to resist the malicious recommendations and discover the G-peers. From the above results, we can conclude that in a community with malicious recommenders, just ignoring others' recommendations is not a good way, even it improves the hit ratio. 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.

### 5.3.3 Risk against Malicious Recommendations

Risk model is very important in the PET model. We can see the ability of the risk model against the malicious recom-

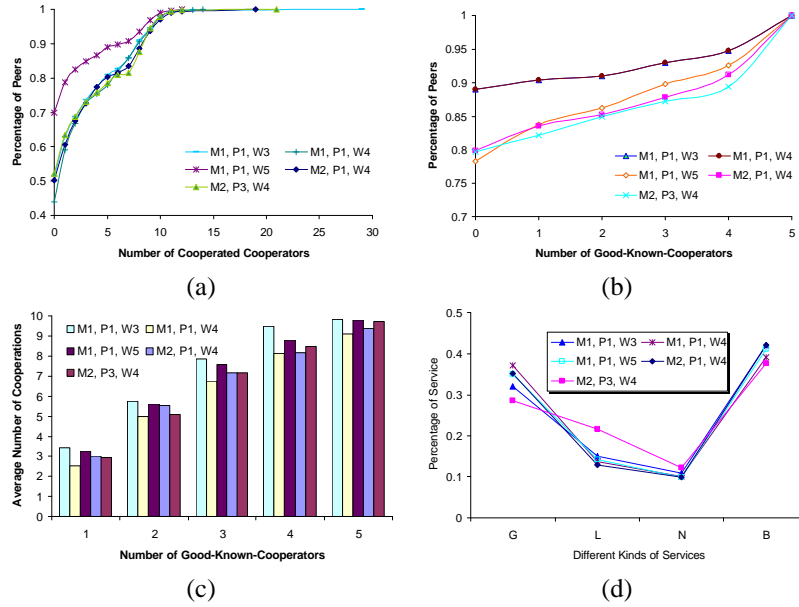


Figure 9: Effect of recommendations. Other setting options of this experiment group are: B1, C1 and S1. (a) CDF of the total number of cooperated cooperators in the history table, (b) CDF of the number of good-known-cooperators, (c) Number of good-known-cooperators versus related average number of cooperations, and (d) Distribution of served services.

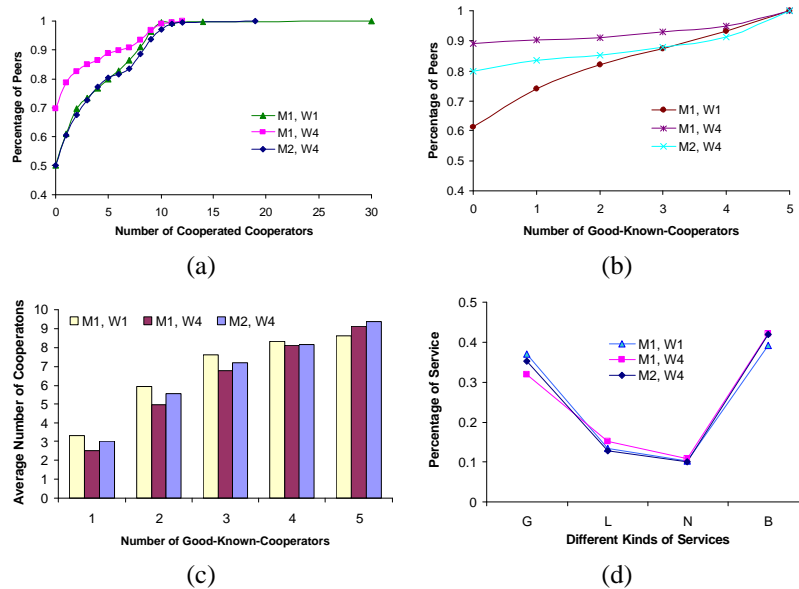


Figure 10: Risk evaluation with the effect of malicious recommendations. Other setting options of this experiment group: B1, C1, P1, and S2. (a) CDF of the total number of cooperated cooperators in the history table, (b) CDF of the number of good-known-cooperators, (c) Number of good-known-cooperators versus related average number of cooperations, and (d) Distribution of served services.

mentation from this group of experiments.

**Setup:** 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 include: C1, P1 and S2.

**Discussion:** From Figure 10(a), Figure 10(b) and Figure 10(d) just with the option M1, it can be seen that *if we increase the value of  $W_{Ri}$ , PET has high resistance to the malicious recommendations for the sensitiveness, hit ratio, and the applicability*: Considering the sensitiveness, about 25% peers have more than four cooperated cooperators with options (M1, W1), while with options (M1, W4) this number drops to less than 15%; for the hit ratio, about 18% peers pick up more than two good-known-cooperators with options (M1, W1), while with option (M1, W4), the number drops to 9%, and the result is even better than the case without malicious recommendations with options (M2, W4); for the applicability, 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. The merit also appears in the anaphase considering the efficiency from the Figure 10(c). In summary, *when the malicious recommendations exist, setting  $W_{Ri}$  with a high value is great helpful to resist the malicious recommendation*.

### 5.3.4 Long Range Effect

Now, we are in a position to investigate the long range effect of our model. In other words, what will happen if the scale of the system increases.

**Setup:** Here we will combine two groups of options, (C1, C2) and (P1, P4), to proceed the simulation. Other options are the same: M2, S1, and W4.

**Discussion:** All the sub-figures show that the more clients, which means the longer the execution time and the larger the scale, the more effective, efficient and applicable the our approach will be. In Figure 11(b), the number of good-known-cooperators with C2 option is seven, 40% more than the one with C1 option. From the Figure 11(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 expected that in the large scale P2P community, our approach will be great promising.

### 5.3.5 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 12 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 12(a) is increasing overall. It can be 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 12(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 fast, which is helpful to recognize the B-peer. For the D-peer (Figure 12(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).

The currency exchange ratio is the bridge between the PET model and the M-CUBE model. In our simulation, the function  $f$  (originally defined in Section 4.2.1) is defined as

$$f(T) = \begin{cases} 1 & , T \geq 0.4 \\ T & , T < 0.4 \end{cases} \quad (5)$$

so for the Figure 12(b) and (c), when the trustworthiness value drops below 0.4, the line denoting the trustworthiness will overlap with the line denoting the ratio. We can get more reliable services through setting the threshold higher in

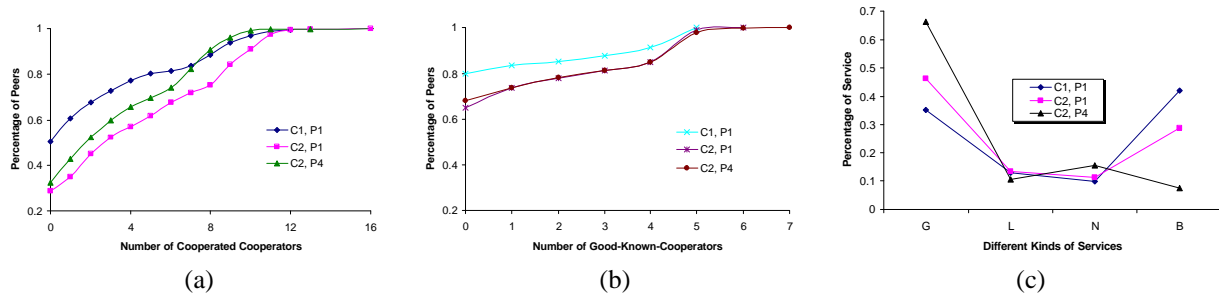


Figure 11: Long-range effect of the Model. Other setting options of this experiment group: B1, M2, S1, and W4. (a)CDF of total number of cooperated cooperators in the history table, (b) CDF of the number of good-known-cooperators, and (c) Distribution of served services.

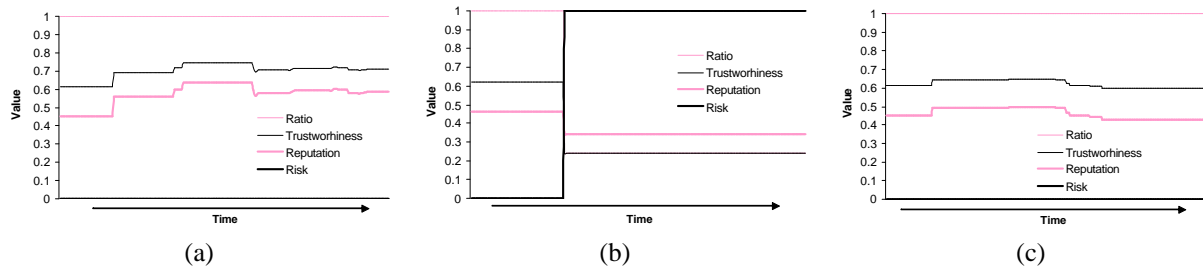


Figure 12: Relationship among the trustworthiness, reputation, risk and ratio. The setting options of this experiments are: B2, M2, P1, S2, and W4. (a) G-peers, (b) B-peers, and (c) D-peers.

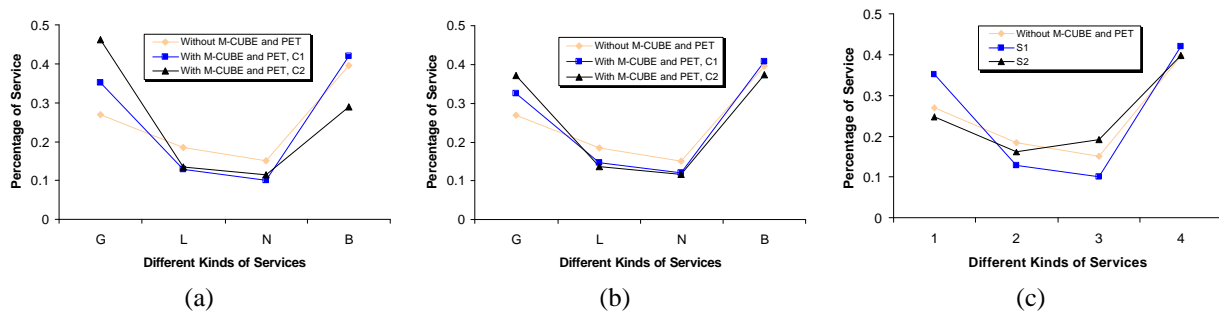


Figure 13: Improvement of our model. (a) Distribution of served services when no malicious recommendations and boasters exist, (b) Distribution of served services with both malicious recommendations and boasters existing, and (c) Distribution of served services with different number of peer servers.

addition to increasing the weight of the risk, which reassure the reliable service.

### 5.3.6 Improvement of Our Approach

In the following, we study the improvement of our approach in terms of **efficacy**, **anti-boaster**, and **scalability**.

**Setup:** The service requests of clients are generated by the SURGE [5], 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). The service provided by D-peers is one of the four services  $G, L, N, B$ . 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. Here  $G : L : N : B$  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 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, also two groups experiments are conducted: the first  $E1$  is with different number of clients, and the second  $E2$  is with different number of peer servers.  $E1$  is to study the effect when the system overload changes. In  $E1$  two sizes of clients ( $C1$  and  $C2$ ) are used for the comparison, while the number of peer servers in both is 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 13 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.

**1. Efficacy** First, let's study the efficacy of M-CUBE. In Figure 13(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 13(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:

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

**2. Anti-boaster** In Figure 13(b) the boaster will exist, and the bad peer can act more intelligently. From Figure 13(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 13(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 cost more time for the system to get convergent, but can not prevent the trend to convergency.

**3. Scalability** In Figure 13(c), in addition 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 boosters in this experiment. The settings and the reasons why choose this have been discussed in setup part. From the Figure 13(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.

## 5.4 Summary

Based on the experiments and analysis in this section, we summarize the major conclusions in the following:

1. High weight of the risk is much more helpful to improve the performance of the model, including sensitiveness, effectiveness, the hit ratio and applicability, when the community has more bad peers; it is also very helpful to resist the negative effect of malicious recommendations.
2. Setting the recommendation a low weight is a good tradeoff to improve the performance while keeping the ability of resistance to the malicious recommendations.
3. The larger the scale of the system, the better effect our approach will bring out.
4. The combination of M-CUBE and PET is effective under the highly untrusted computing environment.
5. Our approach is high sensitive to good services.
6. Setting a high trustworthiness threshold for the currency ratio is another way to get high reliable services in addition to adjusting the weight of the risk.

## 6 Related Work and Discussion

Our work is built upon a great deal of previous work in the field of peer to peer networks, content distribution networks, and distributed resource management. Instead of describing all of them, we cluster them into five groups that are specifically related to our work: *trust management*, *reputation-based system*, *economic model based resource management*, and *cooperative Web caching*.

**Trust Management** The notion of “trust management” was first coined by Blaze, Feigenbaum, and Lacy in their

seminal paper on decentralized trust management [6], 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 literature, Marsh (1994) is the first one to introduce a computational model for trust in the distributed artificial intelligence (DAI) community [27]. However, he did not model reputation in his work. Mui [29] gives a detailed computational model of trust and reputation. In Mui’s model, reputation is well modelled, but it doesn’t take the risk into consideration.

**Reputation-based system** Centralized reputation systems is a very hot topic and has been widely deployed in e-commerce [1, 39, 50], such as eBay (an online auction site), slashdot.com (an online tech-guru site). Recently, in the P2P domain decentralized reputation management schemes like P2Prep [10], EigenTrust [20], and NICE project [22] appears. 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. NICE project [22] discusses the trust inference problems, and [33] proposes a model to build trustworthy software agent. However, the objective of these reputation-based systems are different from that of our effort, which focuses on the 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.

**Economic Model Based Distributed Resource Management** Numerous economic models including microeconomics and macronomics principles for resource management have been proposed in the literature [3, 7, 8, 21, 28, 41, 44], 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 [15, 53, 11, 45]. [53] gives 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 [15] is the closest work related to us. But the details of how to use the currency are different. Different from [15], our infrastructure disagrees with the overbooking and delegation,

and we focus on the solution to the random behavior of resource usage. PPay [52] 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. Samasara [11] and [9] focus on the P2P enforcing storage sharing through the construction of storage claims. It is not applicable to renewable resource in general. Especially, the trustworthiness of the peers are either neglected in these system or treated in different way.

**Cooperative Web Caching** Peer-to-Peer Web server sharing is chosen as a case study to evaluate the efficacy and performance of the proposed model. However, this idea is similar to cooperative Web caching, which has been extensively studied in recent years [12, 18, 26, 32, 34, 35, 42, 47, 48, 49]. Different from these previous work, which are from the perspective of client caching (passive mode), P2P Web server sharing is a proactive approach from the perspective of Web servers. More detailed comparison can be found in [37].

## 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 can prevent several possible attacks launched by participating peers 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. These will be our future work.

## References

- [1] K. Aberer and Z. Despotovic. Managing trust in a peer-to-peer information systems. *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM'01)*, 2001.
- [2] E. Adar and B. Huberman. Free riding on gnutella. *First Monday* 5(10), Oct. 2000.
- [3] Y. Amir, B. Awerbuch, and R. Borgstrom. A cost-benefit framework for online management of a metacomputing systems. *Proc. of the first International Conference on Information and Computational Economy*, Oct. 1998.
- [4] F. Azzedin and M. Maheswaran. Evolving and managing trust in grid computing systems. *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE '02)*, May 2002.
- [5] P. Barford and M. E. Crovella. Generating representative web workloads for network and server performance evaluation. *Proceedings of Performance '98/ACM SIGMETRICS '98*, July 1998.
- [6] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. *IEEE Symposium on Security and Privacy*, May 1996.
- [7] R. Buyya, D. Abramson, and J. Giddy. A case for economy grid architecture for service-oriented grid computing. *Proceedings of the 10th IEEE International Heterogeneous Computing Workshop*, Apr. 2001.
- [8] B. Chun. *Market-based Cluster Resource Management*. Ph.D. thesis, Department of Electrical Engineering and Computer Science, UC Berkeley, Oct. 2001.
- [9] B. Cooper and H. Garcia-Molina. Peer-to-peer resource trading in a reliable distributed systems. *Proc. of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, Feb. 2002.
- [10] F. Cornelli, E. Damiani, S. D. C. Vimercati, S. Paraboschi, and P. Samarati. Choosing reputable servants in a p2p network. *Proc. of the 11th International World Wide Web Conference (2002)*, May 2002.
- [11] L. Cox and B. Noble. Samsara: Honor among thieves in peer-to-peer storage. *Proc. of the 19th ACM Symp. on Operating Systems Principles (SOSP-19)*, Oct. 2003.
- [12] M. Dahlin, R. Wang, T. Anderson, and D. Patterson. Cooperative caching: Using remote client memory to improve file system performance. *Proc. of the First USENIX Symposium on Operating Systems Design and Implementation*, Nov. 1994.
- [13] J. Douceur. The sybil attack. *Proc. of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, Feb. 2002.
- [14] eBay, <http://www.ebay.com>.
- [15] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat. Sharp: An architecture for secure resource peering. *Proc. of the 19th ACM Symp. on Operating Systems Principles (SOSP-19)*, Oct. 2003.
- [16] R. Gupta and A. K. Somani. Compup2p: An architecture for sharing of compute power in peer-to-peer networks with selfish nodes. *Second Workshop on the Economics of Peer-to-Peer Systems*, June 2004.
- [17] 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. *Third International Conference on Peer-to-Peer Computing (P2P'03)*, Sept. 2003.
- [18] S. Iyer, A. Rowstron, and P. Druschel. SQUIRREL: A decentralized, peer-to-peer web cache. *Proceedings of the 12th ACM Symposium on Principles of Distributed Computing (PODC 2002)*, July 2002.
- [19] H. Junseok. The economics of dynamic bandwidth transaction service: Toward pricing modeling. *M3I Workshop on Modelling*, pp. 18-19, June 2001.
- [20] S. Kamvar, M. T. Schlosser, and H. Gacia-Molina. The eigen-trust algorithm for reputation management in p2p networks. *Proc. of the 12th International World Wide Web Conference (2003)*, May 2003.
- [21] A. Lazar and N. Semret. Auctions for network resource sharing. Tech. Rep. TR 467-97-02, Computer Science Department, Columbia University, Feb. 1997.
- [22] S. Lee, R. Sherwood, and B. Bhattacharjee. Cooperative peer groups in nice. *Proc. of IEEE Conference on Computer Communications (INFOCOM'03)*, Mar. 2003.
- [23] Z. Liang and W. Shi. M-CUBE: A novel economic model for trusted P2P resource sharing. Tech. Rep. MIST-TR-2004-005, Department of Computer Science, Wayne State University, Feb. 2004.
- [24] Z. Liang and W. Shi. PET: A Personalized Trust model with reputation and risk evaluation for P2P resource sharing. *HICSS-38*, Jan. 2005.

- [25] J. MacKie-Mason, L. Murphy, and J. Murphy. *Responsive Pricing in the Internet*. MIT Press, May 1997, pp. 279-303.
- [26] Y. Mao, Z. Zhu, and W. Shi. Peer-to-peer web caching: Hype or reality? Tech. Rep. MIST-TR-2003-007, Department of Computer Science, Wayne State University, Aug. 2003.
- [27] S. Marsh. *Formalising Trust as a Computational Concept*. Ph.D. thesis, University of Stirling, 1994.
- [28] M. Miller and K. Drexler. *Markets and Computation: Agoric Open Systems*. The Ecology of Computation, B. Huberman (editor), Elsevier Science Publishers, 1998.
- [29] L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation. *Proceedings of the 35th Hawaii International Conference on System Sciences*, 2002.
- [30] L. Murphy and J. Murphy. Feedback and pricing in atm networks. *Proc. of 3rd Workshop on Performance Modelling and Evaluation of ATM Networks*, July 1995.
- [31] A. Oram. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, Mar. 2001.
- [32] V. Padmanabhan and K. Srinidulchai. The case for cooperative networking. *Proc. of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, Feb. 2002.
- [33] A. S. Patrick. Building trustworthy software agents. *IEEE INTERNET COMPUTING* pp. 46-53, Nov. 2002.
- [34] G. Pierre and M. van Steen. Globule: A platform for self-replicating web documents. *Proceedings of the 6th International Conference on Protocols for Multimedia Systems*, pp. 1-11, Oct. 2001.
- [35] M. Rabinovich, I. Rabinovich, R. Rajaraman, and a. Aggarwal. A dynamic object replication and migration protocol for an internet hosting service. *Proceedings of the 19 International Conference on Distributed Computing Systems (ICDCS'99)*, May 1999.
- [36] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content addressable network. *Proc. of ACM SIGCOMM'01*, 2001.
- [37] J. Ravi, Z. Liang, and W. Shi. A case for peer-to-peer web server sharing. Tech. Rep. MIST-TR-2003-010, Department of Computer Science, Wayne State University, Nov. 2003.
- [38] J. Ravi, W. Shi, and C. Xu. Pace: Prefetching and filtering of personalized emails at the network edges. Tech. Rep. MIST-TR-2003-005, Department of Computer Science, Wayne State University, May 2003.
- [39] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara. Reputation systems. *Communications of the ACM* 43(12):45-48, 2001.
- [40] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An analysis of internet content delivery systems. *Proc. of the Fifth USENIX Symposium on Operating Systems Design and Implementation*, Dec. 2002.
- [41] R. Smith and R. Davis. The contract net protocol: High level communication and control in a distributed problem solver. *IEEE Transactions on Computers* C-29(12):1104-1113, 1980.
- [42] T. Stading, P. Maniatis, and M. Baker. Peer-to-peer caching schemes to address flash crowds. *Proc. of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, Feb. 2002.
- [43] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM'2001*, 2001.
- [44] C. Waldspurger, T. Hogg, B. Huberman, J. Kephart, and W. Stornetta. Spawn: A distributed computation economy. *IEEE Transactions on Software Engineering* 18(2):103-117, 1992.
- [45] C. A. Waldspurger and W. E. Weihl. Lottery scheduling-flexible proportional-share resource management. *Proceedings of the First Symposium on Operating Systems Design and Implementation, Usenix Association*, Nov. 1994.
- [46] Y. Wang and J. Vassileva. Trust and reputation model in peer-to-peer networks. *Third International Conference on Peer-to-Peer Computing (P2P'03)*, Sept. 2003.
- [47] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, M. Brown, T. Landray, D. Pinnel, A. Karlin, and H. M. Levy. Organization-based analysis of web-object sharing and caching. *Proc. of the 2nd USENIX Symposium on Internet Technologies and Systems (USITS'99)*, 1999, <http://www.cs.washington.edu/research/networking/websys/pubs/usits99.ps>.
- [48] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy. On the scale and performance of cooperative web proxy caching. *Proc. of 17th ACM Symposium on Operating Systems Principles (SOSP)*, pp. 16-31, Dec. 1999.
- [49] L. Xiao, X. Zhang, and Z. Xu. On reliable and scalable peer-to-peer web document sharing. *Proceedings of 2002 International Parallel and Distributed Processing Symposium*, Apr. 2002.
- [50] L. Xiong and L. Liu. A reputation-based trust model for peer-to-peer ecommerce communities. *Proceedings of the IEEE Conference on E-Commerce*, June 2003.
- [51] Yamamoto, Yasunori, and H. Junseok. Market-based network formation for an ad hoc, p2p wireless data network. *PWC 2002*, Oct. 2002.
- [52] B. Yang and H. Carcia-Molina. Ppay: Micropayments for peer-to-peer systems. *Proceedings of ACM CCS'03*, Oct. 2003.
- [53] T. Zhao and V. Karamcheti. Enforcing resource sharing agreements among distributed server clusters. *Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS)*, Apr. 2002.