J Grid Computing (2006) 4: 247–263
DOI 10.1007/s10723-006-9041-9

Trust Brokering and Its Use for Resource Matchmaking in Public-Resource Grids

Farag Azzedin · Muthucumaru Maheswaran · Arindam Mitra

Received: 1 September 2005 / Accepted: 14 April 2006 / Published online: 18 July 2006

Abstract This paper presents a trust brokering system that operates in a peer-to-peer manner. The network of trust brokers operate by providing peer reviews in the form of recommendations regarding potential resource targets. One of the distinguishing features of our work is that it separately models the accuracy and honesty concepts. By separately modeling these concepts, our model is able to significantly improve the performance. We apply the trust brokering system to a resource manager to illustrate its utility in a public-resource Grid environment. The simulations performed to evaluate the trust-aware resource matchmaking strategies indicate that high levels of 'robustness' can be attained by considering trust while matchmaking and allocating resources.

Key words trust brokering · Grid computing · trust aware matchmaking · resource allocation

Dr Azzedin contributed towards the work in this paper while he was at University of Manitoba, Winnipeg, Canada.

F. Azzedin (✉)
Department of Information and Computer Science,
King Fahd Univ. of Petroleum and Minerals,
Dhahran, Saudi Arabia
e-mail: fazzedin@ccse.kfupm.edu.sa

M. Maheswaran
School of Computer Science, McGill University,
Montreal, Quebec, Canada
e-mail: maheswar@cs.mcgill.ca

A. Mitra
Department of Computer Science,
University of Manitoba,
Winnipeg, Canada
e-mail: arindam@cs.umanitoba.ca

1. Introduction

The Grid computing system [7, 9] is a highly scalable network computing system that is emerging as a popular mechanism for harnessing resources that are distributed on a wide-area network. Conceptually, a Grid computing system allows resources from different administrative domains to participate in it and ensures the autonomy of the different sites (referred hereafter as domains). However, in current practice, Grid computing systems are built from resources that are contributed by institutions that agree to work together due to off-line trust relationships that exist or are forged among them. To scale a Grid beyond hundreds of nodes, it is necessary to accommodate 'public' resources, where *a priori* trust relationships do not exist among the resources [8].

The term *public-resource* Grids refers to a class of Grids, where the resources do not have a priori trust relationships. A variety of different

approaches can be used to construct Grid systems that fit into this class including: (a) combining Grids and P2P systems and (b) generalizing P2P systems with resource matching and resource management systems. One of the most desirable features of the public-resource Grids is that it opens up the membership of the Grid very much like P2P file sharing systems. This provides an opportunity for the Grid to scale in terms of participants and increases the eligible Grid participants. Currently, only best-effort application specific Grids such as SETI@home bring public resources under a single virtual entity. One way of increasing the applicability of public-resource Grids is to make them QoS aware.

To provide services with QoS, the resources should be managed. Because a public-resource Grid is made of resources with heterogeneous trust relationships [6, 8, 19], the resource manager needs to consider these relationships while managing the resources.

This paper contributes to this important problem in the following ways: (a) presents a trust brokering system that manages and models the trust relationships among the different parts of the network computing system, (b) devises new mechanisms for efficiently maintaining recommendation based systems, and (c) introduces a new methodology for incorporating trust into resource matchmaking that is based on risk minimization.

Our trust broker system is based on a trust model that separates the concepts of accuracy and honesty. The concept of accuracy enables peer review-based mechanisms to function with imprecise trust metrics, where different peers can evaluate the same situation differently. By introducing the concept of honesty, we handle the situation where peers intentionally lie about others for their own benefit. Using simulations, we demonstrate that these two conditions can be handled by our mechanisms and the importance of properly handling these conditions.

Section 2 defines the notions of trust and reputation and outlines mechanisms for computing them. In Section 3, we describe the architecture of the trust brokering system. Section 4 discusses the results from a simulation study performed to evaluate the proposed trust model. Section 5 presents a case study designed to investigate the utility of the proposed trust model. Related work is briefly discussed in Section 6.

2. Trust Model

This section describes the notions of trust and outlines the assumptions for sustaining the trust modeling framework.

2.1. Fundamental Trust Concepts

In this paper, *behavior trust* is quantified by the dynamic parameter *trust level* (TL) that ranges from *very untrustworthy* (TL = 1) to *very trustworthy* (TL = 5). The TL is computed based on past experiences for a specific context. Ideally, *reputation* of an entity y is the behavior trust value reached by global consensus. In practice, it is estimated by polling sufficiently large number of recommenders regarding the behavior trust of entity y. Because recommenders can be dishonest and distort the reputation estimates, we introduce an *honesty* concept that tracks the truthfulness of a recommender. A recommender is considered to be truthful if it says what it actually knows. *Accuracy* is another important concept that tracks how correctly a recommender estimates the underlying trust values.

2.2. Assumptions and Trust Model Elements

Our trust model assumes that each entity x maintains a *set of recommenders* (R) and a *set of trusted allies* (T). Entity x completely trusts its trusted allies that are chosen based on off-line trust relationships such as the ones maintained by a social network [12]. Trusted allies are used by an entity to determine the honesty of its recommenders. In general, trusted allies of an entity x do not have sufficient knowledge to provide recommendations themselves. The recommenders of x are maintained in a *recommender trust table* (RTT), where a two-tuple (honesty, accuracy) is associated with each entry. The initial membership of RTT is randomly chosen and it evolves as described below. Similarly, x maintains another table called *direct trust table* (DTT) for tracking transactions that x had with other entities. In addition to the above,

Table 1 List of notations used.

Notation	Description	Values
TL	Indicates a peer's *trust level*; quantify behavior trust. Higher values mean more trustworthy.	[1,5]
$TTL_x(y, t, c)$	*True trust level* as observed by entity x for entity y during a transaction at time t for context c.	Computed
RTT	*Recommender trust table* used by entity x to maintain a list of its recommenders	2-tuple $< H, A >$
$H(x, y)$	*Honesty* measurement for entity y as seen by x	Conditional; 0 or 1
$A(x, y, t, c)$	*Accuracy* measurement for entity y as observed by x during time t for context c	Computed
$RE_k(x, y, t, c)$	*Recommendation* given by entity k to x about y during time t for context c	Computed
S(A, RE)	*Shift function* used for adjusting a recommendation RE using the accuracy A of the recommender.	Computed; conditional
$DTT(x, y, t, c)$	An entry in the *direct trust table* (DTT) maintained by entity x used for tracking transactions that it had with y during the time t for context c	Computed

we make the following assumptions as well: (a) behavior trust is a slowly varying parameter, (b) transactions between entities are secure and the source and destination are properly authenticated, and (c) trustworthiness and honesty are independent notions. Table 1 shows the list of notations used in rest of the paper for defining and computing the trust measurements.

Although in the performance evaluations we assumed dense (fully populated) DTT and RTT matrices, the trust framework is applicable to sparse matrices as well. That is, not all brokers need to engage in transactions with all other brokers.

2.3. Computing Honesty and Accuracy

To determine the honesty of recommender z, entity x instructs the entities in its T to request recommendations from z regarding entity y for a specific context. These requests are launched such that they arrive at z as closely spaced in time as possible. Because behavior trust is a slow varying parameter, if z is honest, it cannot give away largely different answers. Let the honesty of recommender z as observed by entity x be denoted as $H(x, z)$ and let $RE_k(z, y, t, c)$ denote the recommendation for entity y given by z to entity k for context c and time t, where $k \in T$. Let $TL_{min} = \min_{k \in T}\{RE_k(z, y, t, c)\}$ and $TL_{max} = \max_{k \in T}\{RE_k(z, y, t, c)\}$. Let Δ_{RE} denote the difference and be given by $\Delta_{RE} = TL_{max} - TL_{min}$. The value of Δ_{RE} will be less than a small value ϵ_{RE} if recommender z is honest. Consequently, $H(x, z)$ is computed as follows:

$$H(x, z) = \begin{cases} 0 \text{ if } \Delta_{RE} > \epsilon_{RE} \\ 1 \text{ otherwise} \end{cases}$$

Let the accuracy of recommender z as observed by entity x for a specific context c at a given time t be denoted as $A(x, z, t, c)$, where $0 \leq A(x, z, t, c) \leq 1$. Let $TTL_x(y, t, c)$ denote the *true trust level* (TTL) of y obtained by x as a result of monitoring its transaction with y for context c at time t. Let $\Psi_{RE} = RE_x(z, y, t, c) - TTL_x(y, t, c)$. The value of $|\Psi_{RE}|$ is an integer value ranging from 0 to 4 because $RE_x(z, y, t, c)$ and $TTL_x(y, t, c)$ are in [1..5]. Then, $A(x, z, t, c)$ can be computed as:

$$A(x, z, t, c) = -\frac{1}{4}\ |\Psi_{RE}| + 1 \tag{1}$$

Monitoring each transaction is an onerous task. Therefore, Equation (1) will be used to update accuracy every nth transaction and a weighted averaging process as shown in Section 4.3.1 will be used to maintain this parameter in between

the updates. Before x can use $RE_x(z, y, t, c)$ to calculate the reputation of y, $RE_x(z, y, t, c)$ must be adjusted to reflect recommender z's accuracy. Therefore, a *shift function* (S) that uses the accuracy $A(x, z, t, c)$ to correct $RE_x(z, y, t, c)$ is formulated as follows:

$$S(A(x, z, t, c), RE_x(z, y, t, c)) = \begin{cases} RE_x(z, y, t, c) + 4(1 - A(x, z, t, c)) \text{ if } \Psi^*_{RE} < 0 \\ RE_y(z, y, t, c) - 4(1 - A(x, z, t, c)) \text{ if } \Psi^*_{RE} \geq 0 \end{cases} \quad (2)$$

Because monitoring is done every nth transaction, Ψ^*_{RE} is equal to the Ψ_{RE} that was obtained at the last monitoring event.

2.4. Computing Trust and Reputation

The trust level that quantifies behavior trust between two entities is assumed to be made up of direct trust and reputation. Let the behavior trust for a given context c and time t between two entities x and y be $\Gamma(x, y, t, c)$, direct trust between the entities for the same context and time be $\Theta(x, y, t, c)$, and the reputation of y for the same context and time be $\Omega(y, t, c)$. Let the weights given to direct and reputation trusts be α and β, respectively such that $\alpha + \beta = 1$ and $\alpha, \beta \geq 0$. If the trustworthiness of y (as far as x is concerned) is based more on direct relationship with x than the reputation of y then α should be larger than β.

The reputation of y is computed as the average of the product of the trust level in the DTT shifted by the *shift function* S, for all recommenders $z \in R \neq x$. In practice, DTT will be used to give recommendations and obtain direct trust levels.

$$\Gamma(x, y, t, c) = \alpha\ \Theta(x, y, t, c) + \beta\ \Omega(y, t, c)$$
$$\Theta(x, y, t, c) = DTT(x, y, t, c) \quad (3)$$

One way of estimating the reputation $\Omega(y, t, c)$ is to use recommendations. Let $\Omega_x(y, t, c)$ be the estimate of $\Omega(y, t, c)$ computed by x based on the recommendations it received from its recommenders. In general, this estimate will not be the same as the actual value $\Omega(y, t, c)$. However, as a simplification measure, we assume the estimate to be sufficiently accurate.

$$\Omega_x(y, t, c) = \frac{\sum_{z \in R} S(A(x, z, t, c), RE_x(z, y, t, c))}{|R|}$$

where $z \neq y$

3. The Trust Brokering System

Here, we present the trust brokering architecture. The different steps for the trust brokering mechanism involving the trust notions defined in the earlier sections are described in detailed.

3.1. Trust Brokering Model

The core of the trust brokering model is a peer-to-peer network of trust brokers (hereafter referred to as brokers). A broker is responsible for managing the trust of the resources and clients that are within its domain. The resources within a single domain are grouped into different classes by their expected reputation by the broker. A broker's reputation will depend on how accurate and honest it is in representing the reputations of its resources. For instance, when a resource misbehaves despite it being presented as a highly reputed resource by the broker, the reputation of the broker will be reduced. Conversely, if a broker conservatively estimates the reputation of its resources, then the resources within its purview can be unnecessarily shunned by other resources. Therefore, a broker is compelled to place a resource in the most appropriate class by weighing these conflicting requirements. When a resource joins a domain, it negotiates with the broker the trust level (reputation) that will be placed on it. The resource could use recommendations from prior broker associations to lay its claim for higher trust levels.

Often a broker may need to know about resources that are under the purview of brokers with whom it does not have any relationships. In this case, a broker will request recommendations regarding the target broker from its peering brokers. The predicted reputation of the target resource will depend on the reputation of the broker that manages it and the reputation bestowed upon the

resource by the broker. The post mortem analysis of the transactions will determine the validity of the predictions. The reputations and other trust levels of the brokers are adjusted based on the match between the predicted values and post-mortem detected values.

3.2. Trust Representation and Usage

The trust that exists among the brokers is represented by a DTT, where a specific row of the DTT shows how a particular source broker (represented as B_s) trusts other target brokers (represented as B_ts). For a specific context c_i, B_s trusts B_t at trust level $TL_{st}^{c_i}$ and this trust level is based on direct experience with B_t. If B_t is *unknown* to B_s, $TL_{st}^{c_i} = -1$. The trust levels in the DTT are time stamped to indicate the time of last update and are maintained distributively such that B_i maintains row i of the DTT.

Suppose B_j is highly trustworthy in the 'global' sense, then the DTT should have very high trust levels along column j. A DTT is considered *consistent* if the variation along *any* given column of the DTT is below a given threshold. Otherwise, the trust model is considered to be *inconsistent*. With a consistent DTT, when a broker is considered trustworthy by another broker, there is a high probability that other brokers will also find it trustworthy. This property is essential for recommendations to be useful.

Suppose a resource under B_s is interested in engaging in a transaction with a resource under B_t. To determine the suitability of target the resource, B_s consults its DTT to obtain the direct trust level and its recommenders in RTT to obtain the reputation. The requests sent to the recommenders can trigger recursive queries yielding a *recommender tree*. A recommendation tree has DTT lookups at its leaf nodes and RTT lookups at the intermediate nodes. To avoid cycles in the recommender tree, a recommendation request carries the list of visited brokers.

3.3. Trust Evolution

After a resource decides to pursue a transaction based on the estimated trust levels, that transaction is either monitored or logged for subsequent analysis by the *transaction monitoring proxies* (TM proxies) of B_s and B_t. Because a TM proxy is controlled by the broker of the associated domain, TM proxies of B_s and B_t might evaluate the same transaction differently. The exact definition of 'breaches' vary between two TM proxies and some examples include: (a) holding the resources for longer periods that initially requested, (b) trying to access protected local data, (c) instantiating illegal tasks on the resources, and (d) reneging on promises to provide resources [6]. Monitoring the transactions in a real-time fashion can cause significant overhead for trust computation. One way to reduce the overhead is to combine online and offline mechanisms in the monitoring process. [11, 14, 18].

The TTLs obtained by the TM proxies are periodically used to evaluate direct and reputation trust sources that are evaluated differently. Let $TTL(B_t, t, c)$ denote B_t's TTL for context c at time t and $DTT(B_s, B_t, t_{st}, c)$ be the current trust level of the DTT entry that corresponds to B_s and B_t in context c and was last updated on t_{st}. Let δ be a real number between 0 and 1.

$$DTT(B_s, B_t, t, c) = (1 - \delta)\ DTT(B_t, B_s, t_{st}, c) + \delta\ TTL(B_t, t, c)$$

If $\delta > 0.5$, preference is given to the TTL determined through the analysis of the last transaction between the two brokers.

To evaluate the set of recommenders, B_s needs to compute the *honesty* as well as the *accuracy* measures. A formula similar to above can be used to update the average accuracy. However, the *honesty* of the recommenders is updated differently. Suppose $H(B_s, B_z)$ is the honesty of recommender B_z based on the recommendation given to B_s and $H_{RTT}(B_s, B_z)$ is its honesty as maintained in the RTT based on all previous recommendations. The following simple formula updates the honesty parameter.

$$H_{RTT}(B_s, B_z) = \min(H_{RTT}(B_s, B_z, c), H(B_s, B_z))$$

The above equation penalizes a broker for lying even once. When the honesty value of a broker reaches 0, it is removed from the recommendation set for a random interval. For the direct trust and accuracy, a weighted moving average

algorithm was used to update the parameters. In Section 4.3.1, we show alternative approach and examine their properties.

4. Performance Evaluation

In this section, we discuss the results from our simulation experiments to analyze the performance of the proposed trust model.

4.1. Overview

We conducted a series of simulation studies to examine various properties of the proposed trust model. One performance measure of the trust model is its ability to correctly predict the trust that exists between the brokers. A prediction is considered successful when: (a) a trustworthy broker is predicted as trustworthy and (b) an untrustworthy broker is predicted as untrustworthy. A broker is considered to be trustworthy if its trust level is in [3, 5] and considered to be untrustworthy if its trust level is in [1, 2]. Let the value of the prediction function, $\Phi(B_k)$, be 1 if it correctly predicts B_k's trustworthiness and 0 otherwise. Hence, the *success rate* (SR) of prediction is computed for n brokers at time t as follows:

$$SR(t) = \frac{\sum_{k=1}^{n} \Phi(B_k)}{n} \times 100$$

4.2. Simulation Model and Setup

Not all trust brokers are trustworthy at the same level. One of the objectives of the simulation is to model the process of uncovering the trustworthiness of the trust brokers by the trust system through the observation of the transactions that take place among them. We model the trust values that underly among the brokers by an *actual direct trust table* (ADTT). For simplicity, we assume that these trust relationships do not change for the duration of the simulation time.

The *computed direct trust table* (CDTT) is another table that is similar to ADTT that is used to keep track of the true trust levels that are revealed by the post mortem processes carried out by the TM proxies. The elicitation of the true trust levels by the post mortem processes are simulated by initially setting CDTT to ADTT plus a random noise generated from [0, 4] and setting those values related to the transactions to a small value that is randomly chosen in the range [0–2]. This causes the CDTT to contain values closer to 'true' values for those relationships for which post mortem analysis have been carried out. In addition to ADTT and CDTT, we maintain a *predicted direct trust table* (PDTT) to track the evolution of the trust relationships among the brokers. The PDTT values are initially set to -1 and are updated using Equation (3) and the latest values of CDTT.

4.3. Results and Discussion

The requests initiating inter-broker transactions are assumed to have a Poisson arrival process. The number of brokers was set to 30, the size of R was fixed at 4 for all brokers, and the size of T was fixed at 3 for all brokers. The source and the target brokers for each transaction were randomly generated from [0, 29]. For the simulations performed here, a consistent DTT was used. The length of the monitoring interval is set at 1, 5, 10, or 20 meaning that the TM proxy is monitoring every, every fifth, every 10th, or every twentieth transaction, respectively. The value of α is varied from 0 to 1 in 0.5 increments. By varying the monitoring interval, we can examine the sensitivity of the trust model on true trust levels. By varying α, we can examine the dependence of the trust model on the different trust components.

4.3.1. *Estimating Trust Levels*

Previously we used an *exponential weighted moving average* (W-AVE) filter for estimating trust levels. One of the drawbacks of this scheme is that it returns high estimates despite periodic occurrences of low values in a sequence of trust values (i.e., a broker can periodically cheat and still maintain a high trust level). The W-AVE filter produces an estimate using $O_t = \omega O_{t-1} + (1 - \omega) O_c$, where O_t is the newly generated estimate,

O_{t-1} is the prior estimate, and O_c is the new observation. If ω is large, the W-AVE filter resists rapid changes in individual observations and said to provide stability. For low ω values, the filter is able to detect changes quickly and said to be agile.

In [13], the stable and agile filters were combined to create a *flipflop* filter. We develop a variation called *modified flipflop* (MFF), where the agile filter is activated as soon as we detect a drop in value of the trust parameter beyond an acceptable threshold from the previously estimated value. The agile filter quickly downgrades the estimate. For the subsequent estimates, we switch back to the stable filter assuming that the trust parameter does not experience any further depreciations.

One of the drawbacks of this filter is that it does not penalize those brokers that continue to periodically cheat. We further modify flipflop to obtain *weighted modified flipflop* (WMFF) to take periodic cheating into considerations by having a history queue that has the past n values of the trust level. When the kth cheating incident is detected, k low trust values are inserted into the history queue. Because the history queue is limited to n entries, only $n - k$ entries from the past remains in the history queue. In computing the trust level, the history queue entries are weighted such that the weight increases linearly from the head to the tail. From the simulation results shown in Figure 1, we can observe that WMFF can detect periodic cheating. Consistent *versus* Inconsistent Trust Models.

4.3.2. *Consistent versus Inconsistent Trust Models*

Here we investigate the dependence of the trust model on the consistence or inconsistence of the DTT. From the simulation results, we found that the success rate is in the range 88.14 to 100.00%, when the DTT consistent. When the DTT is inconsistent, the success rate is around 50%. This shows that with inconsistent DTT a broker cannot learn actual trust because it is getting conflicting reports on other brokers. An interesting observation is that this low success rate is not affected by a variation in the number of the dishonest brokers.

4.3.3. *Agility of the Trust Model*

Tables 2 and 3 show the success rate of the trust model when using the accuracy alone and accuracy and honesty together, respectively. In Table 2, when there are 0 dishonest brokers, it can be observed that combining direct trust and reputation (i.e., when $\alpha = 0.5$), outperforms the others (i.e., when $\alpha = 1.0$ or when $\alpha = 0.0$). Because all recommenders are honest, reputation reinforces direct trust and increases the overall success rate. We can also observe that as the trust monitoring interval is increased, the trust model takes longer

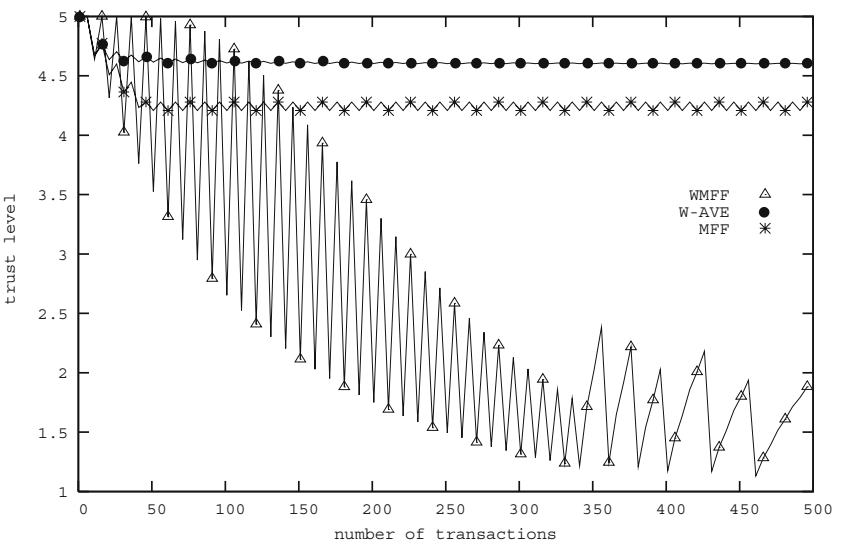

Figure 1 Performance of different filters in detecting malice in updating trust levels.

Table 2 Success rates for a consistent trust model using only the accuracy measure, where B_d is the number of dishonest brokers and F_m the monitoring frequency.

B_d/F_m	α value	Number of transactions per relation		
		5 (%)	50 (%)	150 (%)
0/1	1.0	71.83	99.86	100.00
	0.5	86.55	99.54	99.54
	0.0	79.77	95.29	99.01
0/5	1.0	57.01	86.90	99.31
	0.5	60.57	97.47	99.11
	0.0	60.23	86.32	89.77
0/10	1.0	54.02	77.36	92.18
	0.5	56.44	88.16	98.85
	0.0	54.94	79.66	88.74
0/20	1.0	51.95	64.37	81.84
	0.5	53.68	75.86	95.98
	0.0	51.84	69.54	82.53
15/1	1.0	71.84	99.89	100.00
	0.5	76.90	91.95	91.49
	0.0	55.86	62.99	64.25
15/5	1.0	57.01	86.90	99.31
	0.5	55.17	85.52	89.77
	0.0	50.69	53.91	54.60
15/10	1.0	54.02	77.36	92.18
	0.5	52.99	75.63	88.28
	0.0	49.77	51.72	52.41
15/20	1.0	51.95	64.37	81.84
	0.5	51.03	63.44	81.61
	0.0	50.92	50.11	52.18
20/1	1.0	71.67	99.89	100.00
	0.5	72.01	89.54	89.54
	0.0	49.89	57.13	57.24
20/5	1.0	57.09	86.90	99.31
	0.5	54.60	76.90	83.91
	0.0	44.14	45.75	47.47
20/10	1.0	54.02	77.36	92.18
	0.5	52.07	68.28	79.20
	0.0	46.32	44.60	45.75
20/20	1.0	51.95	64.37	81.84
	0.5	51.90	62.87	73.68
	0.0	49.08	46.09	44.25

to reach a given acceptable success rate. In this paper, we arbitrarily set the acceptable success rate as 85%. Tables 2 and 3 show the acceptable successes as darkened entries.

Because the accuracy measure is the difference between a recommender's opinion and the true trust level obtained by the TM proxy, with a shorter monitoring interval, the accuracy will be higher and the trust model will converge faster.

As the dishonest brokers increase to 15 or 20, any mechanism that relies on the reputation gives poor success rate and the accuracy measure loses its effectiveness. In this case, the solution is to depend on direct trust. However, just using direct trust lowers the convergence rate and also does not exploit the opportunities for cooperation among the brokers, which is a major feature of a network computing system.

To reduce the trust model's sensitivity to dishonest brokers, we use the honesty measure so that dishonest recommenders are filtered out to prevent them from contributing to the recommender

Table 3 Success rates for a consistent trust model using the accuracy and the honesty measures, where B_d is number of dishonest brokers and F_m the monitoring frequency.

B_d/F_m	α value	Number of transactions per relation		
		5 (%)	50 (%)	150 (%)
0/1	1.0	71.22	100.00	100.00
	0.5	87.36	99.54	100.00
	0.0	80.15	95.67	100.00
0/5	1.0	56.78	88.16	100.00
	0.5	62.18	98.16	100.00
	0.0	60.46	87.47	90.03
0/10	1.0	54.28	77.13	92.10
	0.5	56.67	90.34	99.01
	0.0	55.63	80.23	88.14
0/20	1.0	51.49	66.44	82.79
	0.5	52.18	76.32	95.66
	0.0	51.84	70.57	82.70
15/1	1.0	71.20	100.00	100.00
	0.5	87.70	99.67	100.00
	0.0	77.82	97.01	100.00
15/5	1.0	56.78	88.16	100.00
	0.5	63.45	97.82	100.00
	0.0	62.41	86.44	91.78
15/10	1.0	54.25	77.13	92.19
	0.5	55.40	90.11	100.00
	0.0	55.98	78.62	88.91
15/20	1.0	51.49	66.44	82.02
	0.5	53.33	77.47	96.59
	0.0	53.79	68.97	84.14
20/1	1.0	71.12	100.00	100.00
	0.5	89.08	99.89	100.00
	0.0	82.18	98.85	100.00
20/5	1.0	56.78	88.16	100.00
	0.5	63.22	98.16	100.00
	0.0	61.72	89.89	94.01
20/10	1.0	54.25	77.10	92.13
	0.5	55.98	93.33	100.00
	0.0	55.98	80.52	89.47
20/20	1.0	51.49	66.86	82.49
	0.5	52.53	78.31	94.93
	0.0	52.30	70.57	85.01

network. The overall performance of this strategy is shown in Table 3.

As the number of dishonest brokers increase, we observe the behavior to significantly differ between Tables 2 and 3. In Table 3, combining both components (i.e., direct trust and reputation) gives a higher success rate than relying only on one of them. For example, when the monitoring interval length = 5, the number of dishonest brokers = 15, and the number of transactions = 50, the success rate reaches 77.13% when $\alpha = 1.0$ and 78.62% when $\alpha = 0.0$ but 90.11% when relying on both. We can conclude that once the dishonest recommenders are filtered out from the recommendation sets, reputation reinforces direct trust and therefore combining these two components yields a higher success rate than relying on only one of them.

The darkened entries in Tables 2 and 3 illustrate the benefit of incorporating honesty into the trust model. The maximum success rates are achieved when the number of dishonest brokers equal zero.

As the number of dishonest brokers increase, the number of darkened entries in Table 2 start to reduce. This shows that the accuracy measure is not effective in limiting and preventing the dishonest brokers from influencing the recommenders set. On the other hand, Table 3 show that the number of darkened entries remain almost the same as the number of dishonest brokers increase. This demonstrates the necessity to model honesty and incorporate into the overall trust model.

5. Case Study: Trust Modeling on Public-Resource Grids

5.1. Overview

In our Grid model, we assume that each domain autonomously finds a matching target resource for the requests emerging from within the domain. The matchmaking process uses the truest information the domain can learn using the trust modeling framework previously discussed. The focus here is to observe how the matchmaking process is able to utilize the trust model for pairing resource requesters to providers.

The overall objective of the trust-based matchmaking process is to bring together resources from domains that have high mutual trust in a particular allocation. As a result, the resource forming a *virtual collection* have higher trust among them and can provide higher levels of assurance on the delivered performance.

5.2. System Model and Assumptions for Public-Resource Grids

In our Grid model, a public-resource Grid is composed of several domains, where each domain is managed by one or more brokers. In the case of multiple brokers within a domain, we assume them to be strongly consistent with each other. A broker provides several services to the resources within its domain, including: (a) serving as a resource manager/negotiator and acting as the local domain representative, (b) serving as a matchmaker to find resource matches for local requests among local and remote resources, and (c) computing and maintaining the DTT for remote domains. We make the following assumptions regarding within the overall system:

A1: Privacy and secure communications within the system is ensured by following:

(1) brokers and resources are cryptographically bound to public keys via digitally signed certificates. There is no single root CA (certification authority) but there may be a fixed set of predetermined root CAs.
(2) brokers and resources hold only one set of cryptographic keys. This can be guaranteed by fingerprinting the principals (e.g., using software and hardware fingerprints).
(3) each resource establishes out-of-band trust relationship with the local broker in its domain. This includes exchange and authentication of public keys using arbitrary exchange mechanisms.
(4) peers (brokers and resources) will validate each other before engaging in any transaction.

A2: A resource can be attacked by other resources in the system but never attacked/threatened by its local broker.

A3: The broker (also the resource manager and matchmaker) is non-biased. Meaning, it will not favor some resources over others within a domain.

A4: For trust-aware resource matchmaking purposes, we consider the following:

(1) a resource has the following attributes: *type of contexts* (ToCs) it supports and a trust level for each ToC. Each task allocated on a resource related to a particular *type of context* (ToC) is time limited. Associating a trust level with each ToC provides the flexibility of selectively opening services to clients. Client domains also have their own trust attributes including: ToCs sought and trust levels associated with the ToCs.
(2) resources will establish trust for each other via their brokers before engaging in transactions. The broker and its

resources negotiate on the ToCs and the corresponding trust levels.

(3) broker will have some external means to verify correctness of the ToCs provided by its local resources resources.

A5: Brokers will collect recommendations from multiple sources to establish trust levels for local and remote resources, domains, and brokers.

5.3. A Trust Model for Public-Resource Grids

From the DTT. $TL_{ij}^{c_k}$, which is the *offered trust level* (OTL) that B_i offers to B_j to engage in activity within context c_k, can be determined. Suppose we have client X from B_i wanting to engage in activities within contexts c_p, c_q, and c_r on resource Y in B_j. Because resource and client domains inherit the trust levels from the broker they are associated with, we can compute the OTL for the composite activity between X and Y, i.e., OTL $= \min(TL \text{ for } c_p, TL \text{ for } c_q, TL \text{ for } c_r)$. There are two *required trust levels* (RTLs), one from the client side and the other from the resource side. If the OTL is greater than or equal to the maximum of client and resource RTLs, then the activity can proceed with no additional concern. Otherwise, there will be a risk involved in carrying out the activity. The *expected risk factor* (ERF) for given RTL and OTL values are provided by RTL − OTL and is 0, when RTL − OTL < 0.

5.4. A Model for Trust Aware Resource Matchmaking

The resource matchmaking model presented here is based on the following additional assumptions: (a) tasks are mapped non-preemptively, (b) tasks are indivisible (i.e., a task cannot be distributed over multiple machines), and (c) tasks are independent (i.e., there is no data dependency among the different tasks).

For request r_j, let $t(r_j)$ and $c(r_j)$ denote the task and originating client, respectively. Suppose that we have a set of tasks $t(r_0) \ldots t(r_{n-1})$ and a set of machines $m_0 \ldots m_{k-1}$. We can match the tasks onto the machines in $n \times k$ different ways. With each match, we can associate a completion time ($\alpha(t(r_j), m_i)$) and an ERF ($\beta(t(r_j), m_i)$). The completion time indicates when m_i is going to be available after completing task $t(r_j)$, whereas $\beta(t(r_j), m_i)$ indicates the expected risk, when assigning $t(r_j)$ to m_i. Further, let γ_i be the available time of machine m_i after executing all prior requests assigned to it. Table 4 describes the notations used in the rest of this section.

5.5. Matchmaking Algorithms

In this section, we show how a simple resource matching heuristic can be modified to perform trust aware resource matchmaking. The objective of this exercise is to show the utility of the trust model and not to solve the resource matching

Table 4 List of notations used.

Notation	Description
OTL	*Offered trust level* used by a broker to advertise its resources to another domain.
RTL	*Required trust level* is the trust level that a domain X sets for a domain Y. Domain X uses it to compute the expected risk factor (ERF) involved in engaging in transaction with Y.
ERF	*Expected risk factor* measured as: $ERF = RTL - OTL$. $ERF \leq 0 \Rightarrow$ no risk; otherwise, there is a risk involved.
$\alpha(t(r_j), m_i)$	Completion time for task $t(r_j)$ on machine m_i.
$\beta(t(r_j), m_i)$	ERF involved in executing task $t(r_j)$ on machine m_i.
γ_i	Available time on machine m_i after executing all prior requests assigned to it.
EEC$(t(r_j), m_i)$	*Expected execution cost* for task $t(r_j)$ on machine m_i
ECC$(t(r_j), m_i)$	*Expected completion cost* for task $t(r_j)$ on machine m_i, measured as $EEC(t(r_j), m_i) + \gamma_i$

problem optimally. In this study, we choose the *min-min* heuristic [15] as the base algorithm.

The min-min is a trust unaware algorithm and it works as follows. Let $\mathrm{EEC}(t(r_j), m_i)$ be the *expected execution cost* of $t(r_j)$ on machine m_i. In addition, let $\mathrm{ECC}(t(r_j), m_i)$ denote the *expected completion cost* of $t(r_j)$ on machine m_i, which is computed by adding the EEC of $t(r_j)$ on machine m_i and the available time of m_i. The *min-min* heuristic has two phases. In the first phase: For each task $t(r_j)$, the machine that gives the earliest expected completion cost is determined using the EEC matrix and the machine available times. In the second phase, task $t(r_k)$ with the minimum earliest completion cost is found and is assigned to the corresponding machine. The goal of the *min-min* heuristic is to assign a set of requests $\{r_0 \ldots r_{n-1}\}$ such that $\{\max_i\{\gamma_i\}\}$ is minimized for $0 \leq i < m$, where n is the number of requests and m is the number of machines.

5.5.1. *Trade-off Algorithm*

With trust aware resource matchmaking, we need to perform matchmaking such that the overall completion times are minimized while the risk associated with the matchmaking are simultaneously reduced. This 'biobjective' matchmaking is harder to achieve. This algorithm trades off one objective for another by weighing them differently.

The trade-off algorithm works as follows. For each $t(r_j)$, it finds the maximum $\alpha(t(r_j), m_i)$ and the maximum $\beta(t(r_j), m_i)$ and normalizes the entries in matrices ECC and ERF by these two numbers, respectively. This results in normalized ECC and ERF matrices. Because the completion time (i.e., ECC matrix) changes with each assignment, the ECC matrix should be recomputed and renormalized with every assignment.

Let w_c and w_r represent the weights associated with the completion time and risk components, respectively. The trade-off process can be carried out by changing these weights. It should be noted that the relative value of these weights do not imply that some risk value is equivalent to some value of completion time. When the trade-off is applied, we are dealing with normalized completion times and risks. Once the biobjective trust-aware resource matchmaking problem is transformed into a uniobjective formulation through this trade-off process, a single parameter minimization heuristic such as min-min can be directly applied.

5.5.2. *Maximum Risk Algorithm*

Suppose R_{max} is the maximum risk that the matchmaker is willing to take for any individual matchmaking. A penalty factor $\mathrm{diff}(\beta(\mathrm{t}(\mathrm{r_j}), \mathrm{m_i}) - \mathrm{R_{max}})\mathrm{Q}$ is added to $\alpha(t(r_j), m_i)$, where Q is a large penalty factor and $\mathrm{diff}(\mathrm{x}) = 0$ if $x < 0$ and 1 otherwise. Once the penalty factor is added to the completion times, the min-min algorithm is used to select the task-to-machine matchmakings. By adding the penalty factor, we are able to avoid high risk matchmakings.

5.6. Performance of Trust Aware Resource Matchmaking

To highlight the benefits of trust aware resource matchmaking, we investigate two factors that impact performance: (a) makespan for the complete schedule (b) makespan variability. The makespan is defined as $\max_{\mathrm{i}\in\mathrm{K}}(\alpha(\mathrm{t}(\mathrm{r_j}), \mathrm{m_i}))$ and is a measure of the throughput of the whole resource matchmaking process. The makespan variability is the defined as the variation in makespan as the risk penalty value Q changes.

To simulate the impact of risk, we compute the *expected* completion time of a request r_j mapped onto machine m_i without considering the risk penalty and then compute the *actual* completion time for the same mapping by adding risk penalty to the above completion time. The risk penalty determines the cost due to misbehaving resources. The ERF (i.e., $\beta(t(r_j), m_i)$) indicates how likely it is to incur the risk penalty. For example, ERF of 3 indicates that there is a $(3/5)100 = 20\%$ chance of incurring the risk penalty. The actual risk penalty itself is computed as a fraction x of the corresponding EEC value.

The resource matchmaking process was simulated using a discrete event simulator with Poisson request arrivals. We consider 20 resource domains. The source and target domains were

Table 5 Comparison of makespan and robustness of various resource matchmaking algorithms using a consistent trust model with $w_r = 1 - w_c$ The makespan values are given in milliseconds.

x value	RMS algo.	w_c value	Expected makespan	Actual makespan
0.01	Min–min	NA	16, 011.28	16, 094.54
	Trade-off	0.0	178, 672.41	178, 686.17
		0.2	40, 323.55	40, 325.78
		0.4	39, 474.82	39, 477.04
		0.6	40, 244.76	40, 244.76
		0.8	35, 273.99	35, 274.98
		1.0	15, 985.72	16, 055.42
	Max. risk	NA	30, 195.42	30, 211.99
1.0	Min–min	NA	16, 011.28	24, 336.83
	Trade-off	0.0	178, 672.41	180, 048.14
		0.2	40, 323.55	40, 546.34
		0.4	39, 474.82	39, 697.60
		0.6	40, 244.76	40, 244.76
		0.8	35, 273.99	35, 372.55
		1.0	15, 985.72	25, 297.04
	Max. risk	NA	30, 195.42	31, 853.08
10.0	Min–min	NA	16, 011.28	101, 871.41
	Trade-off	0.0	178, 672.41	192, 429.71
		0.2	40, 323.55	42, 899.47
		0.4	39, 474.82	42, 451.45
		0.6	40, 244.76	42, 798.63
		0.8	35, 273.99	41, 317.49
		1.0	15, 985.72	110, 686.16
	Max. risk	NA	30, 195.42	49, 040.93

randomly picked from the range [1–20]. The ToCs required for each request were randomly generated from the range [1–4] meaning that each $t(r_i)$ involves at least one ToC but no more than four ToCs. The two RTLs were randomly generated from the range [1–6] and the OTLs were randomly generated from the range [1–5]. The simulations results for 10,000 tasks and 20 machines are shown in Table 5.

As expected, the min-min algorithm that only considers completion times perform best in terms of the expected makespan. However, the actual makespan varies significantly from the expected makespan due to the addition of the risk penalty. The mapping computed by min-min gives the highest variability on the makespan (i.e., considering risk while mapping results in mappings that are resistant to risk related variations at run time. This 'robustness' of the resource matchmaking is highly desirable because if the expected makespan of a matchmaking process is not representative of the actual makespan, then the resource manager cannot do meaningful capacity planning.

5.7. Threat Analysis of Public-Resource Grids

This section presents a qualitative analysis of the attack scenarios that can impede the operation of the public-resource Grids and what counter-measures are required to provide resiliency against such attacks. First, we model the threats that can be used to attack a public-resource Grid system and then enumerate the broad strategies used to combat the threats. Next, we examine the threats originating from the actors in the system and describe how the strategies can be used to prevent/dilute the attacks.

Threat Models: One part of the system analysis is to anticipate the threats that can be used to attack the system. Broadly, the threats to the public-resource Grids can be classified as:

M1: *Impersonation*: Peers (resources and brokers) attempt to mask their true behavior by impersonating as reputed entities.

M2: *Service Disruptions*: A peer behavior that deviates from the normal, fair operation of the system with an intent to harm other peers.
M3: *Exploiting vulnerabilities*: Misusing the system vulnerabilities for selfish gains, e.g., longer time to verify access eligibility will encourage free-riding.

Strategies: Securing the trust aware resource matchmaking process largely involves ensuring: execution of the transactions as per their contracts, resource access authorizations, and identifying the malicious resources and brokers. We use the following strategies to achieve these objectives:

S1: Each task binds the resource provider and the resource requester by contracts agreed upon by the participating resources and approved by their respective brokers. Cryptography based authentication are used to identify the resources correctly.
S2: The brokers are implicitly authorized to verify the usage status at the resources in their domain by running some root-privilege daemons on their local resources. This is necessary for the broker to verify any contractual breaches by the resources.
S3: Provider resources monitor the behavior of services run by the requesters during a transaction. This way, providers can detect contractual breaches by the requesters.
S4: A broker maintains an archive of past transactions involving its local resources and other brokers. Periodic auditing of the archive will detect any anomaly in the behaviors of remote brokers and resources.

Threat and Attack Scenarios: Based on the threat models, we explore the threats/attacks that can originate from the actors within the system. Table 6 shows the classification of the threats.

T1: Unauthorized attempts by clients to gain services at the resources.
T2: A resource provider (a) contributes faulty resources for a transaction or (b) rejects the requesters claims for capacity.
T3: A resource requester tries to launch a denial-of-service by (a) asking for resources but leaving them idle when granted the resource by its broker or (b) use the resources granted to launch a DoS on third parties.
T4: Man-in-the-middle modifies the resource requests for its own benefit.
T5: Malicious recommenders (i.e., brokers) giving incorrect recommendations with the intent of harming resources.
T6: Malicious brokers create phantom resources and advertise them to other domains in an attempt to launch a DoS against the other domains.

Counter-measures: Essentially the counter-measures are designed to achieve the objectives described earlier: (a) Authorization – ensuring resource access authorizations, (b) System Integrity – protecting system integrity against improper execution of the contracts for resource accesses, and (c) Non-repudiation – ensuring non-repudiation in order to detect misbehaviors. Table 6 illustrates how these objectives are met to achieve sufficient protection against the attacks. The table is summarized as following.

T1: Resources will only allow clients to gain access when the clients have been verified

Table 6 Counter-measures to prevent threats from peers while ensuring access authorizations, system integrity and non-repudiations.

Threat	Model	Authorization	System integrity	Non-repudiation
T1	M1	A1, A5, S1, S3	–	–
T2(a)	M3	–	A4, S1, S2	A1, S4
T2(b)	M2, M3	A1, A5	S1, S2	S4, –
T3(a, b)	M2	A1, S1	S2, S3, S4	A4, A5, S4
T4	M1, M3	A1, A4, S2	S1, S2, S3	–
T5	M3	A1, A4, S1	A1, A2, A3, A4, S1	A1, A4, A5, S4
T6	M2	–	A5, S2, S3	A5, S2, S3, S4

and considered trustworthy by the brokers. Cryptography based identification schemes followed by the trust evaluation process will preventing any unauthorized access. Fingerprinting is an incentive for peers not to create multiple profiles for malicious reasons. This in turn will make the system resistant to Sybil attacks to the extent that peers can uniquely authenticate their peering partners.

T2: We impose the restriction that managerial entities (i.e., brokers) have to monitor and ensure that the resources are delivering the agreed capacities. This is followed by periodic audits of past transactions to detect anomalies in resource behaviors. The audits are done off-line.

T3: Digital identification schemes followed by periodic audits of transactions also help to prevent any attempts by the clients to launch DoSs. Audits are essential to detect and prove misbehavior attempts.

T4: In order to prevent transaction connection from being hijacked by Man-in-the-middle, proper authentications schemes need to be installed.

T5: In order to dilute the impact from malicious recommendations, the brokers need to collect recommendations from multiple recommenders or use the honest and accuracy metrics for the recommenders to accurately evaluate trust values.

T6: Resources and their respective brokers have to cooperate while monitoring their transactions with remote peers in order to prevent their peering brokers from initiating fraudulent transactions.

6. Related Work

A model for supporting *behavior trust* based on experience and reputation is proposed in [1]. This trust-based model allows entities to decide which other entities are trustworthy and also allows entities to tune their understanding of another entity's recommendations. Each entity keeps two sets: (a) set Q for entities directly trusted and (b) set R for recommenders. One of the drawbacks of this model is its use on an exponentially weighted moving average algorithm for updating Q and R. In this approach, a recommender can give intentionally damaging recommendations about few domains and maintain high overall *accuracy*. Because our model uses the *honesty* concept, such recommenders will be detected and isolated from R. Further, the scalability of the model is not explicitly addressed in this study. In our model, the scalability issue is addressed by the aggregation scheme.

A reputation-based approach for extending Gnutella-like environments is proposed in [5], where an entity uses a polling protocol to query and select target entities Each entity maintains information on its own experience with target entities and shares such experiences when polled by other entities. This approach has no mechanism to filter out dishonest entities from the reputation network. An entity broadcasts its request to all of its neighbors regardless of their honesty. This practice is not just inefficient, but also gives a continued opportunities for dishonest entities to damage and influence the reputation network. Because this approach uses a voting scheme to determine the truth, an entity can be fooled if the majority of recommenders are dishonest. Further, this model is based only on reputation and deals with file sharing or file exchange.

A trust management in a P2P information system is proposed in [2], where the focus is on implementing a generic scalable infrastructure to deploy any trust model. A simple trust model was proposed, where entities file complaints based on bad experiences they had while interacting with other entities. One limitation of this model is that it is based on a binary trust scale (i.e., an entity is either trustworthy or not). Hence, once there is a complaint filed against entity p, p is considered untrustworthy even though it has been trustworthy for all previous transactions. Also, this approach has no mechanism for preventing a malicious entity from inserting arbitrary number of complaints and potentially causing a *denial of service* attack.

A trust model for P2P content distribution networks is presented in [16], where web servers can cooperate to replicate their documents worldwide. This model is based on recommendations and again uses an EWMA algorithm, when updating

the recommender set. Hence, dishonest entities can cheat every n transactions and still be considered trustworthy. A decentralized trust model *Poblano* [4] is implemented in a P2P fashion for the Project JXTA [10]. This model is based on recommendations and provides algorithms to determine the trustworthiness of a peer's data based on its reputation. This approach is used to perform reputation guided searching or to securely distribute signed certificates among peers.

Assuming that less than 50% of a population of entities are malicious, a simple reputation polling mechanism is presented in [17]. In this recommendation scheme, an entity's trustworthiness is determined through majority voting.

A reputation management model for a multi-agent system is proposed in [20]. An entity determines the trustworthiness of a correspondent by combining its local experience with the testimonies of recommenders regarding the same correspondent. Again, this approach does not prevent dishonest entities from generating spurious ratings and assumes that the majority of entities offer honest ratings to cancel the effect of dishonest entities.

In summary, one of the major differences between our trust model and the ones examined above is the separation of honesty and trustworthiness in our model. In addition, to the best of our knowledge, no existing literature directly addresses the problem of integrating trust into resource management schemes.

7. Conclusions and Future Work

In this paper, we presented a trust model for public-resource Grid systems. The public-resource Grid systems enable resources without a priori trust relationships to participate as providers in a Grid system. In such conditions, it becomes essential for the resource manager to be trust cognizant to avoid bringing together untrusting parties under a single virtual cluster to solve a given problem.

In this paper, we present a trust brokering system that can be used in a public-resource Grid system. Extensive simulation studies were conducted to evaluate the model under various conditions.

Our trust model uses an *accuracy* concept to enable peer review-based mechanisms to function with imprecise trust metrics, the imprecision is introduced by peers evaluating the same situation differently. Simulation results show that the reputation-based trust model reaches an acceptable level of capability after a certain number of transactions. However, as the number of dishonest domains increase, the model becomes slow in reaching the acceptable level of capability.

To reduce the trust model's sensitivity to dishonest domains, we introduced an *honesty* concept to handle the situation where domains intentionally lie about other domains for their own benefit. Simulation results indicate that incorporating the *honesty* concept into the trust model, limits the effect of dishonest domains by preventing them from providing recommendations.

Another feature of our model is the flexibility to weigh direct trust and reputation differently. Simulation results show that it is better to rely on direct trust when honesty is *not* used. This can be explained by observing that due to malicious recommenders the reputation is tainted and using it can only lead to incorrect decisions. When the honesty is used to isolate the malicious recommenders, we assured of an honest of set of recommenders. In this situation, simulation results indicate that significant benefits can be obtained by using reputations.

Another significant advantage of our scheme is that our scheme does not depend on a majority opinion as previous schemes did. Therefore, our scheme can work even when majority of the recommenders are malicious. Actually as the malicious number of recommenders increase, the recommenders providing recommendations to a query reduces. The number of recommenders also provides another measure of trust on the overall system because all the recommenders are considered honest.

As an application of our trust model, we incorporate trust-awareness into the resource matchmaking process such that the matchmaking decisions are trust cognizant. The simulations performed to evaluate the effectiveness of the modifications indicate that due to trust awareness, the overall performance of the resource matchmaking system improves in different ways.

In summary, our trust model provides two levels of incentives for domains. First, by modeling honesty, the trust model gives incentive to recommenders to truthfully give recommendations and cooperate. If a recommender is dishonest, it will be isolated from the rest of the environment. Second, by modeling trust, the model provides incentives for the domains to be trustworthy and behave as expected. Trust-aware resource matchmaking system is a concrete example, where trust levels maintained by the trust model are used in determining the privileges of a domain.

Acknowledgements This research is supported by a TR*Labs* Scholarship, University of Manitoba Graduate Fellowship, and by Natural Sciences and Engineering Research Council of Canada Discovery Grant RGP220278. The equipments used were supported by a Canada Foundation for Innovation Grant. Some portions of this paper appeared as a conference publication in 2004 International Parallel and Distributed Processing Symposium.

References

1. Abdul-Rahman, A., Hailes, S.: Supporting trust in virtual communities. In: Hawaii International Conference System Sciences, 2000
2. Aberer, K., Despotovic, Z.: Managing trust in a peer-2-peer information system. In: 10th International Conference on Information and Knowledge Management (CIKM'01), pp. 310–317, 2001
3. Azzedin, F., Maheswaran, M.: Integrating trust into Grid resource management systems. In: 2002 International Conference Parallel Processing (ICPP '02), pp. 47–54, 2002
4. Chen, R., Yeager, W.: Poblano: A Distributed Trust Model for Peer-to-Peer Networks. http://security.jxta.org (2001)
5. Damiani, E., di Vimercati, S.D.C., Paraboschi, S., Samarati, P., Violante, F.: A Reputation-based Approach for Choosing Reliable Resources in Peer-to-Peer Networks. In: 9th ACM Conference on Computer and Communications Security, pp. 207–216, 2002
6. Dingledine, R., Freedman, M.J., Molnar, D.: Accountability. In: Oram, A. (ed.) Peer-to-Peer: Harnessing the Power of Disruptive Technologies, pp. 271–340. O'Reilly and Associates, Sebastopol, California (2001)
7. Foster, I., Kesselman, C. (eds.): The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco, California (1999)
8. Foster, I., Iamnitchi, A.: On death, taxes, and the convergence of peer-to-peer and Grid computing. In: 2nd International Workshop on Peer-to-Peer Systems (IPTPS03), 2003
9. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the Grid: Enabling scalable virtual organizations. Int. J. Supercomput. Appl. (2001)
10. Gong, L.: JXTA: A network programming environment. IEEE Internet Computing **5**(3), 88–95 (2001)
11. Habra, N., Chalier, B.L., Mounji, A., Mathieu, I.: ASAX: Software architecture and rule-based language for universal audit trail analysis. In: European Symposium on Research in Computer Security (ESORIC'92), pp. 435–450, 1992
12. Katz, Y., Golbeck, J.: Social network-based trust in prioritized default logic. In: 21st National Conference on Artificial Intelligence (AAAI-06), 2006
13. Kim, M., Noble, B.: Mobile Networks Estimation. In: 7th Annual Conference on Mobile Computing and Networking, 2001
14. Lunt, T.F.: Detecting intruders in computer systems. In: Conference on Auditing and Computer Technology, 1993
15. Maheswaran, M., Ali, S., Siegel, H.J., Hensgen, D., Freund, R.F.: Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. J. Parallel Distrib. Comput. **59**(2), 107–131 (1999)
16. Pierre, G., van Steen, M.: A trust model for peer-to-peer content distribution networks (2001)
17. Sen, S., Sajja, N.: Robustness of reputation-based trust: Boolean case. In: 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02), pp. 288–293, 2002
18. Smaha, S.E., Winslow, J.: Misuse detection tools. J. Comput. Secur. **10**(1), 39–49 (1994)
19. Waldman, M., Cranor, L.F., Rubin, A.: Trust. In: Oram, A. (ed.) Peer-to-Peer: Harnessing the Power of Disruptive Technologies, pp. 271–340. O'Reilly and Associates, Sebastopol, California (2001)
20. Yu, B., Singh, M.P.: An evidential model for distributed reputation management. In: 1st Int'l Joint Conf. Autonomous Agents and Multi-Agent Systems (AAMAS-02), pp. 294–301, 2002