Inf Syst Front (2006) 8:321–333
DOI 10.1007/s10796-006-9005-4

# Parameter learning of personalized trust models in broker-based distributed trust management

**Jane Yung-jen Hsu · Kwei-Jay Lin · Tsung-Hsiang Chang · Chien-ju Ho · Han-Shen Huang · Wan-rong Jih**

Published online: 5 December 2006


**Abstract** Distributed trust management addresses the challenges of eliciting, evaluating and propagating trust for service providers on the distributed network. By delegating trust management to brokers, individual users can share their feedbacks for services without the overhead of maintaining their own ratings. This research proposes a two-tier trust hierarchy, in which a user relies on her broker to provide reputation rating about any service provider, while brokers leverage their connected partners in aggregating the reputation of unfamiliar service providers. Each broker collects feedbacks from its users on past transactions. To accommodate individual differences, *personalized trust* is modeled with a Bayesian network. Training strategies such as the *expectation maximization* (EM) algorithm can be deployed to estimate both *server reputation* and *user bias*. This paper presents the design and implementation of a *distributed trust simulator*, which supports experiments under different configurations. In addition, we have conducted experiments to show the following. 1) Personal rating error converges to below 5% consistently within 10,000 transactions regardless of the training strategy or bias distribution. 2) The choice of trust model has a significant impact on the performance of reputation prediction. 3) The two-tier trust framework scales well to distributed environments. In summary, parameter learning of trust models in the broker-based framework enables both aggregation of feedbacks and personalized reputation prediction.

**Keywords** Distributed trust management · Reputation mechanism · Probabilistic trust model · Personalized feedback rating · Parameter learning · Expectation maximization

## 1 Introduction

Trust is an important relationship between individual entities engaging in any transactions. Each individual has a belief on certain attributes about the other. In addition to *identification*, who the subject entity is, and *qualification*, whether the subject entity is capable of performing the requested service, the trust relationship gauges *consistency*, that is, how well/certain the subject entity is able to deliver a service or a result (Lin, Lu, Yu, & Tai, 2005).

In the online world, the ability to identify the *trustworthiness* of a target partner/server has become critically important. For example, how does an eBay buyer decide which seller will deliver the requested item as promised? Similarly, how does an enterprise application select the web services to help achieve its goals? While it is relatively easy to be deceitful online due to

This research is supported in part by the National Science Council of Taiwan #NSC-94-2218-E-002-057, Institute for Information Industry #94-CS-0457, UC MICRO project #04-0511 and GeoSpatial Technologies, Inc.

J. Y.-j. Hsu (✉) · T.-H. Chang · C.-j. Ho · W.-r. Jih
Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan
e-mail: yjhsu@ntu.edu.tw

K.-J. Lin
Electrical Engineering and Computer Science, University of California, Irvine, CA 92697, USA
e-mail: klin@uci.edu

H.-S. Huang
Institute of Information Science, Academia Sinica, Taipei 115, Taiwan

the lack of physical contact, deceptive behaviors will be discouraged in repeated interactions if experiences may be captured. The rapid growth of online transactions and e-business activities suggests that traditional encryption and authentication mechanisms are no longer sufficient to adequately address the trust issue. It is imperative to provide *trust management* through a reputation mechanism based on user feedbacks from past transactions as in Zacharia and Maes (2000).

Amazon and eBay are successful examples of *centralized* reputation systems, which help foster trust for vendors. With a single trust authority controlling all reputation information, such systems may be vulnerable, inflexible, and difficult to scale up. When the centralized reputation system is *owned* by a single business entity, one may also raise issues about *subjectivity*. In contrast, a software agent working on behalf of its users may choose to maintain a reputation rating for every service provider. Building up a distributed trust relationship can facilitate collaboration in a multi-agent system. However, eliciting reputation information for each agent individually can be challenging. Moreover, when the number of feedbacks collected is small, the ratings can be easily skewed by potentially biased agents.

This research proposes a two-tier trust hierarchy, in which a user relies on its trust broker to provide reputation information about any service provider, while brokers leverage their connected partners in aggregating the reputation of unfamiliar services. The software brokers act as trusted domain experts that manage the trust relationship for general web users. Trust brokers are independently maintained and operated; users are free to choose among many brokers available, much like people can choose their own CPAs and lawyers. Each broker is in charge of collecting and aggregating feedbacks from its users. The broker-based trust framework avoids the pitfalls of the centralized approach, while ensuring meaningful trust ratings in standard operations.

Even though all users belonging to the same broker are assumed to share certain characteristics, e.g. membership, locations, or common interests, they are not without personal differences. To accommodate individual differences, this paper presents a probabilistic approach to modeling personalized feedback with a Bayesian network. To tease apart the *subjective* user bias from the *objective* server performance, a broker may learn the trust model given the only observable data of *user feedback*. In our design, model fitting training strategies such as the *expectation maximization* (EM) algorithm (Dempster, Laird, & Rubin, 1977) are used to approximate both the server performance and user bias by searching for the local maximum of the likelihood function based on a probabilistic trust model and observed user feedbacks.

To evaluate the performance of the proposed trust framework, we have implemented a distributed trust simulator, and conducted extensive simulations. In particular, we performed experiments to compare performance of two training strategies under shifted bias distributions; to examine how different trust models affect the reputation prediction accuracy; and to illustrate scalability of *personalized reputation rating* using broker-based trust management in a distributed environment.

This paper is organized as follows. Section 2 provides an overview of related research on trust management. The broker-based distributed trust framework is introduced in Section 3. In Section 4, we define the probabilistic trust model, and describe the procedure for training such a model with EM. The design of a distributed trust simulator and the simulation process are detailed in Section 5, followed by experimental results in Section 6 and the conclusion in Section 7.

## 2 Related work

In recent years, the design of *trust management framework* has gained much attention in e-commerce, online auctions, peer-to-peer systems, web services and multi-agent systems. Trust management generally relies on a reputation mechanism based on user feedbacks from past transactions. In a comprehensive survey by Dellarocas and Resnick (2003), approaches to online reputation mechanisms are classified into centralized (Zacharia and Maes, 2000) and distributed (Kamvar, Schlosser, & Garcia-Molina, 2003; Yu & Singh, 2000). For example, Amazon computes the average of product ratings according to customer reviews in a centralized fashion. Similarly, eBay utilizes a centralized server to keep track of trust scores based on simple accumulation of user feedbacks of positive, negative, or neutral. Buyers and sellers have the opportunity to rate each other after each transaction, and ratings (with specific comments) over the last six months are maintained.

For peer-to-peer networked environments, trust ratings are usually collected locally. The Eigentrust algorithm proposed by Kamvar et al. (2003) is based on the notion of *transitive trust*, in which all peers in the file-sharing network cooperate to compute and store the global trust vector using power iteration. The approach, similar to the idea of *PageRank* in Google search, was shown to be resistant to various attacks. In

addition to combining feedbacks using simple weighted average in Zacharia and Maes (2000), a Dempster–Shafer *evidential* model based on the word-of-mouth topology is proposed by Yu and Singh (2002). The model distinguishes between uncertainty and negative feedbacks to provide more accurate ratings. A Bayesian network model is proposed by Wang and Vassileva (2003) to combine ratings on different aspects of a server .

A broker framework for web applications was introduced by Lin et al. (2005), where service brokers manage trust information for their respective users. The framework combines three levels of trust and utilizes security broker, trust network, and reputation authority at each level respectively. By delegating trust management to brokers, individual users only need to ask their brokers about the reputation of a service before engaging in any transaction. Each user only needs to share her feedback with her broker. Experiments were conducted to evaluate the performance of the proposed broker framework. While the proposed broker framework performed effectively with low computational overhead, there is no guarantee in error convergence, which motivated the current research.

One important challenge in any reputation mechanism is the difficulty in soliciting feedbacks. In addition to the general lack of incentives for the users, people are reluctant to share information for fear that it will give competitive advantage to others. Rewards are provided in Fernandes, Kotsovinos, Ostring, and Dragovic (2004) as an incentives for honest participation. In Jurca and Faltings (2003), an incentive-compatible protocol is proposed based on the upper bound of deception probability from game theory. Pavlov, Rosenschein, and Topol (2004) proposed supporting privacy as an incentive for truthful feedbacks, while Jurca and Faltings (2004) designed a broker-based protocol to elicit truthful feedbacks.

Instead of deception, this research focuses on the problem of potentially *biased* feedbacks due to individual differences. Some people tend to give negative feedbacks, while others are more positive. Some people tend to have extreme opinions, while others are more moderate. In fact, the general distributed trust framework that will be introduced below can be extended to model other factors affecting user feedbacks.

## 3 Distributed trust framework

This research proposes a two-tier broker-based distributed trust management framework for online service transaction systems. Figure 1 shows the overall structure of the proposed framework consisting of two types of agents, the *brokers* and the *users*. A broker typically works for multiple users who share (localized) common features, and are willing to share information among the group.

Each user may function as either a service provider (e.g. server) or a service requester (e.g. client) in a transaction. In e-commerce scenarios, a client user is often called a *buyer*, and a server user is called a *seller*. When a user acts as a seller, its server performance is dictated by a *consistency factor* (CF) or *server reputation* that controls the probability of a successful service delivery. Let's assume CF is an inherent and consistent property of the server (similar to the reliability of a hardware) with a real value ranging from 0 to 1.

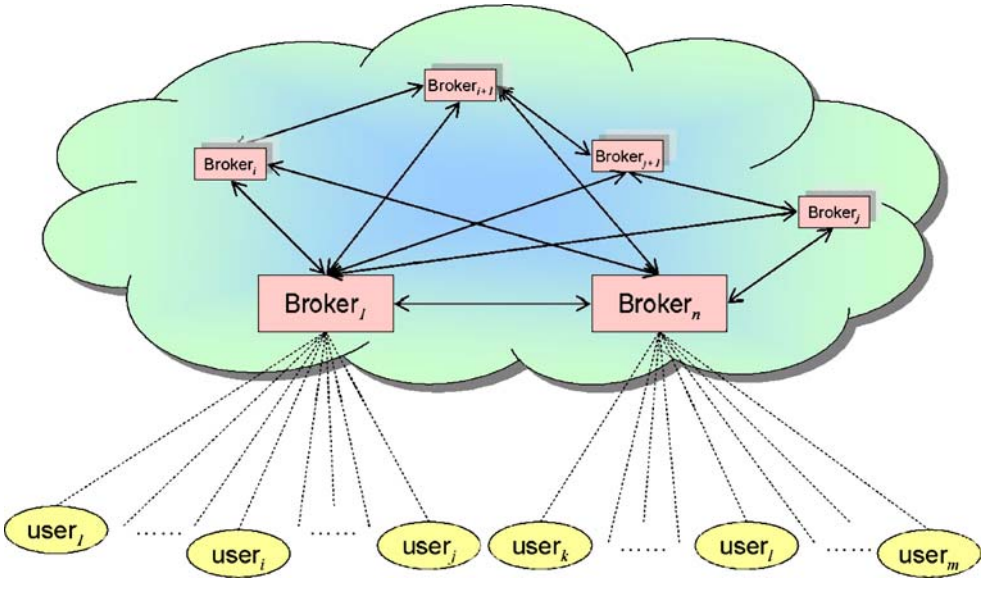



**Fig. 1** Users and brokers in a two-tier distributed trust framework

For example, a CF value of 0.8 means that the server has an 80% chance to deliver a satisfactory service as requested.

In this framework, a client user relies on its respective broker to maintain server reputation ratings for all service providers that have engaged in transactions with any user managed by the same broker. For each transaction, the client user $u_j$ requests its broker $Broker_i$ to provide a reputation rating about a specific server user $u_k$, and initiates the transaction provided that the rating is above a configurable threshold representing $u_j$'s risk-taking attitude. When the transaction is over, $Broker_i$ collects the feedback rating $f$ on the server user $u_k$ from the client user $u_j$ based on the success or failure of the current transaction.

Figure 2 shows the components of a broker, in which the *Reputation Manager* collects all feedback ratings generated by all its users, and the *Trust Manager* exchanges reputation ratings with other connected brokers when necessary. By aggregating feedbacks from all its users, a broker has the opportunity to accumulate enough rating information about any server. In the case when a broker does find its local trust database to be inadequate for making a confident recommendation, it will request additional reputation information from neighboring brokers.

In the proposed framework, we assume that the brokers are connected according to some pre-defined properties, e.g. physical proximity, social connections or business relationships. Brokers communicate through standard protocols (e.g. SOAP) in a peer-to-peer fashion. Based on its past performance, each broker is assigned a *trust* rating, which will be used in aggregating the server reputation ratings from multiple brokers. A simple weighted sum of all ratings from neighboring brokers will be returned to the requesting client user, who then decides whether to carry out the transaction with the specific server. Given that trust brokers are independently operated, they may not always be *cooperative* and *truthful* in providing reputation information and feedback when requested by other brokers. The EigenTrust mechanism is deployed to reward good behaviors and minimize malicious attacks by some brokers (Kamvar et al., 2003).

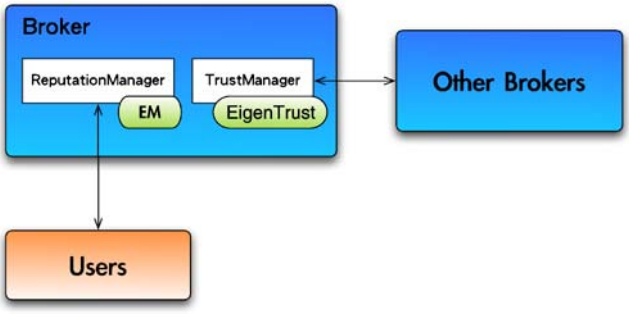



**Fig. 2** Trust broker architecture

The reputation and trust management broker framework introduced in Lin et al. (2005) assumes users and brokers to be diligent in providing *honest* feedbacks. The ad hoc aggregation methods can compute trust ratings efficiently, but there is no guarantee of convergence. On the other hand, the computation defined by EigenTrust converges nicely to a global trust vector, and it meets the demand at the broker level satisfactorily. However, a global trust value may not be the right choice at the individual user level. For example, a small retailer may provide speedy delivery and great service in its local geographical area, but it may be limited in logistics and does not perform satisfactorily globally. In addition to variation in ratings due to locality or other factors, a client user may have her own personal *bias*, either positively or negatively. As a result, a broker needs to *learn* both the server performance and the user bias from feedback data collected over time. In the following section, we will present a Bayesian network trust model for biased user feedback, and explain how the EM algorithm can be adopted to train the probabilistic trust model.

## 4 Probabilistic trust model

One of the most important and challenging issues in trust management is the problem of *trust rating prediction*. A broker needs to provide accurate predictions to keep its users informed, even with a small rating database at the beginning of the broker's operation (e.g. during startup time) or when a new service has become available recently. What makes trust rating prediction especially hard is that the rating database may consist of subjective ratings from various users. Consequently, the predictive trust rating must be personalized to fit the subjective views of various users while maximizing the satisfaction from each individual.

Much previous work uses graphical models to represent existing interactions among users. In their models, each user is expressed as a node. A feedback rating $f$ that client/buyer $u_j$ gives server/seller $u_k$ is recorded as a directed link from node $u_j$ to node $u_k$ with weight $f$. To predict the trust of $u_j$ on another server $u_l$, those models compute the weighted average rating based on all the ratings on all paths from $u_j$ to $u_l$. The common underlying assumption for computing weights is that, if $u_j$ trusts $u_k$ based on the performance of $u_k$, $u_j$ and $u_k$ are likely to give similar ratings to the same servers.

However, those methods do not provide satisfactory solutions to our concerns. First, it is hard to predict the ratings between client and server at startup time or for a new server. The number of links is usually not enough to compose paths between two arbitrary nodes except for some special topologies. Second, although the weighted average mechanism gives personalized prediction, it does not take subjectivity into better consideration. For example, a server that always performs perfectly may give strict ratings on others, using its own performance as the rating standard. In this case, the ratings tend to be underestimated by the strict user.

This research proposes handling the problems by modeling server performance and client subjectivity explicitly. The rating that client $u_j$ would give to server $u_k$ comes from a function that takes the subjectivity of $u_j$ and the performance of $u_k$ as parameters. Such a model considers the subjectivity issue and is beneficial to the startup time issue since ratings can be computed even if there is no direct path from $u_j$ to $u_k$.

A Bayesian network trust model is adopted by the proposed distributed trust management system. Our trust model is distinct from previous work in that it models the relation between a client and server within a transaction. Section 4.1 describes the details of the model. In this work, the subjectivity of *rating bias* is considered, namely, a client's tendency to give strict or generous ratings. Section 4.2 shows how the model computes biased trust. Finally, we present the EM algorithm (Dempster et al., 1977) for training our model.

### 4.1 A sample model

Figure 3 shows the Bayesian network trust model consisting of five random variables, the client user ($C$), server user ($S$), user bias ($B$), server reputation ($R$) and the feedback rating ($F$) for a given transaction. The links in Fig. 3 represent causal dependencies among the random variables. Given the natural variations in server reputation and user bias, we introduce hidden nodes $R$ and $B$ as intermediate random variables to determine the rating. For each transaction, the server reputation is decided by the objective server performance (link from $S$ to $R$), while the user bias is decided by the subjective view of the client (link from $C$ to $B$). For any transaction, the feedback rating a seller receives is based on both server performance and client bias (links from $B$ and $R$ to $F$). Throughout this paper, we sometimes use *buyers* to refer to client users, and *sellers* to refer to servers.

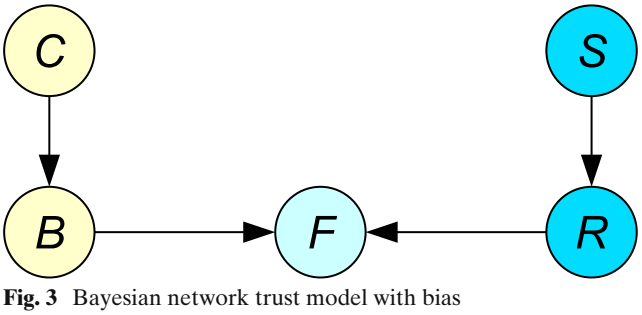


**Fig. 3** Bayesian network trust model with bias

Let us examine the variables and distributions in more detail. The values of $C$ and $S$ are the unique user IDs. Multinomial distributions $P(C)$ and $P(S)$ are used to represent their transaction frequency. Each buyer has its own bias distribution $P(B|C)$, and each seller has its own performance distribution $P(R|S)$. While buyer, seller, and rating for each transaction are observable, the actual server performance and user bias are unknown to the trust brokers. That is, $R$ and $B$ are latent variables. Intuitively, performance and bias can be expressed as real values. To reduce the complexity in computation, our implementation discretizes performance and bias into $n_R$ and $n_B$ bins, respectively. The degree of user bias or server performance is approximated by its expected value. The value of $B$ falls within a lower bound $b_l$ and an upper bound $b_u$. The value of $R$ is within 0.0 and 1.0. It follows that $P(B|C)$ and $P(R|S)$ are also multinomial distributions.

Assume $B$ and $R$ are instantiated as $b$ and $r$ in a specific transaction involving client $c$ and server $s$, we can define the feedback rating to be $f = b + r$. However, the real-numbered rating cannot be obtained by summing up discretized performance and bias directly. In order to handle both discrete-number and real-number ratings with a single model, we define $P(F|B, R)$ as a normal distribution with $\mu = B + R$:

$$P(F|B, R) \propto \exp\left(-\frac{(F-(B+R))^2}{2\sigma^2}\right), \tag{1}$$

where $\sigma$ is a constant. Rating $f = b + r$ has the highest probability to appear in this distribution, which is adaptive to the model training.

### 4.2 Biased trust

The probabilistic trust model can be used to make prediction about a buyer's biased trust, or the *subjective* trust, in a given seller. We compute the estimated reputation (i.e. consistency factor or expected performance) of seller $s$ as follows.

$$\hat{R}_s = \sum_r P(R = r|S = s) \cdot r. \tag{2}$$

Then, the estimated bias $\hat{B}_c$ of client/buyer $c$ can be calculated in a similar way.

$$\hat{B}_c = \sum_b P(B = b\,|C = c) \cdot b\,. \tag{3}$$

Finally, $\hat{B}_c + \hat{R}_s$ is returned as the personalized reputation rating back to client $c$.

### 4.3 Model training

Given a set of $T$ transactions $\mathcal{D} = \{d_t \mid 1 \le t \le T\}$, the parameters of a trust model can be trained using a model fitting method. The $t$-th transaction $d_t$ consists of a client $c_t$, a server $s_t$, and the feedback rating $f_t$ for the specific transaction.

For any transaction, the participants, client $C$ and server $S$, are observable. As a result, the probability distributions $P(C)$ and $P(S)$ can be estimated easily by normalizing their frequency of occurrences. Let $N_c$ and $N_s$ be the numbers of transactions with client $c$ and server $s$, respectively. We have $P(C = c) = \frac{N_c}{T}$ and $P(S = s) = \frac{N_s}{T}$. The conditional distribution $P(F|B, R)$ is defined for all circumstances in Eq. 1, and no update is needed. The distributions of latent variables $B$ and $R$ are estimated by employing the EM algorithm.

The EM algorithm is a general algorithmic framework that searches for local maxima of data likelihood function in the parameter space of probabilistic models. It consists of repeated applications of the E-step and the M-step. The E-step estimates the posterior distributions of the latent variables using the current model. The M-step updates the current model with the results from the E-step. The EM algorithm terminates at a local maximum of the likelihood function or the maximum number of iterations.

In our work, the E-step computes $P(B, R|d_t)$, the joint probability distribution of the latent variables given each transaction:

$$P(B = b\,,\, R = r|d_t) = \frac{P(b\,, r, c_t, s_t,\, f_t)}{\sum_{b',r'} P(b', r', c_t, s_t,\, f_t)}.$$

Let $p_{b,t}$ denote $\sum_r P(b\,, r|d_t)$, and $p_{r,t}$ denote $\sum_b P(b\,, r|d_t)$. In the M-step, $P(B|C)$ and $P(R|S)$ are updated by calculating the expected number of occurrences as follows.

$$P(B = b\,|C = c) = \frac{\sum_{\{t|c_t=c\}} p_{b,t}}{\sum_{b',\{t|c_t=c\}} p_{b',t}},$$

$$P(R = r|S = s) = \frac{\sum_{\{t|s_t=s\}} p_{r,t}}{\sum_{r',\{t|s_t=s\}} p_{r',t}}.$$

It should be noted that there are a number of methods for searching in the parameter space to maximize data likelihood function. For example, simple average or *gradient ascent* may work fine in finding a solution, even in the presence of latent variables. Different methods may stop at different local maxima, so will the same search method with different initial points. In general, good model design may have a stronger impact on the overall search result than the choice of search methods. The most important advantage of adopting EM is in its ease of implementation (or flexibility) to deal with new models, and its ability to learn from a relatively small set of data.

## 5 Implementation

We have designed and implemented a working prototype of the *trust broker*, which can run in either simulation mode or deployment mode. For performance evaluation of the proposed broker-based trust framework, a configurable *trust simulator* has been implemented. This section presents the details of both designs, with a detailed description of the simulation process. The updated and robust implementation in Python is produced to replace the previous Java implementation that was used in Lin et al. (2005). This implementation adopts a completely distributed design, in which all brokers and users can be independently running on any networked machines. Each broker or user is assigned a unique *Universal Resource Identifier* (URI), and all communications use standard SOAP APIs for enhanced interoperability and flexibility. A series of experiments using the trust brokers and simulator have been conducted and the results are reported in Section 6.

### 5.1 Trust broker

Trust brokers form the core of the proposed two-tier trust management framework. In particular, a trust broker performs the following functions:

1. Collecting feedback from its user after each transaction.

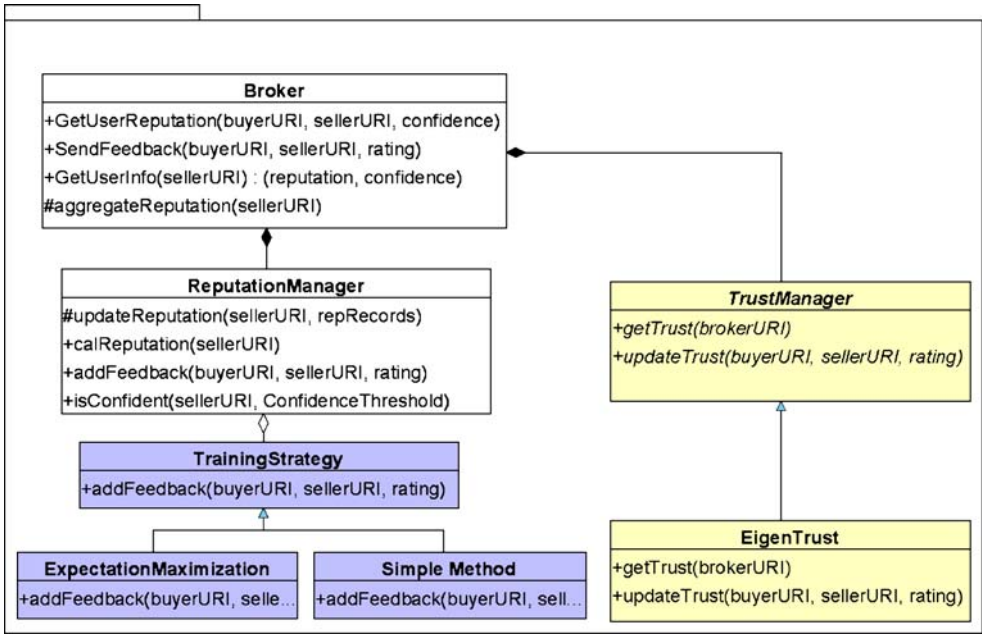



**Fig. 4** Trust broker UML diagram

2. Maintaining ratings of all servers engaged in transactions with its users.
3. Aggregating seller reputation ratings from connected brokers.
4. Managing personal bias of its users.
5. Providing *personalized estimation of server performance* with confidence to its client user.

As was shown in Fig. 2, there are two main components in a trust broker: the reputation manager and the trust manager. Fig. 4 depicts a more detailed design of the trust broker processes and APIs.

The broker module provides two interfaces to the users: `GetUserReputation` and `SendFeedback`. A client user checks the target server's reputation from its broker using *GetUserReputation*. A transaction is initiated only if the client is satisfied with the server rating. After each transaction, the client user is obligated to provide a feedback on the server performance to its broker via *SendFeedback*. Our implementation handles two types of feedback rating: *continuous*, which is a real value between 0.0 and 1.0.; and *binary*, which is either 0 or 1.

A broker utilizes *ReputationManager* to manage user information based on feedbacks collected from past transactions. Given that server reputation and client bias are not directly observable to the broker, the *TrainingStrategy* module attempts to find the optimal fitting of the data collected to the probabilistic user model. In particular, we have implemented two model fitting strategies, *ExpectationMaximization* and *Simple Method* for our experiments. The EM algorithm has been detailed in Section 4.3. The Simple Method estimates the reputation (or CF) of a specific seller by computing the average of all feedbacks on the seller

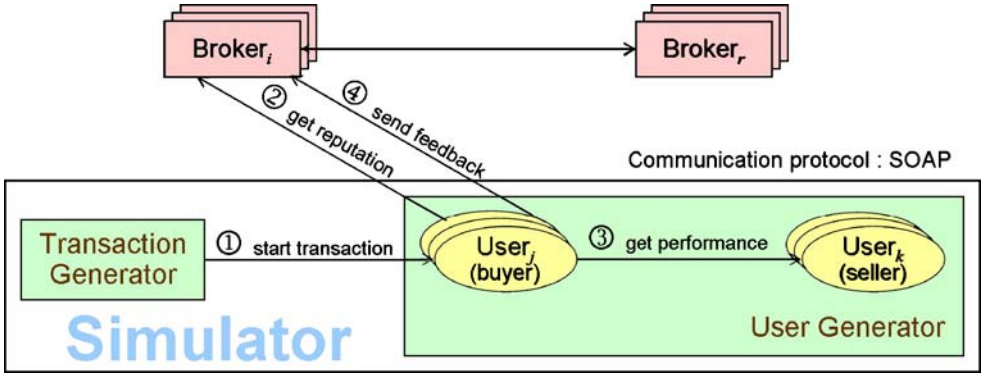



**Fig. 5** Trust simulator

collected by a broker, and it estimates the bias of a user by computing the average over the difference of any feedback with the seller's estimated CF.

Additionally, the *TrustManager* module maintains the trust values of all brokers. The *EigenTrust* mechanism (Kamvar et al., 2003) has been adopted to perform peer-to-peer trust management among brokers. A broker aggregates reputation ratings from connected brokers by summing the ratings weighted by their trust values. EigenTrust offers the advantages of global convergence as well as resistance from attacks by malicious, incompetent, or disagreeing neighbors.

### 5.2 Trust simulator

In order to validate the proposed probabilistic user model and to evaluate the performance of specific parameter learning strategies, e.g. EM, a trust simulator is designed to conduct experiments under various system and environmental configurations. The trust simulator is in charge of starting the broker processes, initializing each simulated user with an inherent CF and bias, generating the set of simulated transactions, configuring the training strategy and schedule, sampling and calculating the prediction errors in reputation and bias. Figure 5 shows the overall structure and the sequence of operations performed by the trust simulator.

The simulator synthesizes each transaction by randomly selecting a buyer and a seller, and the transaction proceeds just like a real transaction. All communications among buyers, sellers, brokers, and the simulator are conducted with SOAP APIs. The seller performance (or client bias) is sampled based on a Gaussian distribution of the user's inherent CF (or bias). For example, given a seller CF of 0.8 and a buyer bias of +0.05, the probability of a successful transaction is 0.8 with an expected value of 0.85 for the feedback rating. Continuous feedbacks range from 0.0 to 1.0, and binary feedbacks can be generated with the continuous rating as the probability of positive feedbacks.

The simulation process is summarized in Algorithms 1 and 2. For each simulation run, a global configuration object *Config* is specified to define the number of brokers $n$, the number of users $m$, the total number of transactions $T$, distribution (uniform or normal) and parameters for server reputation and user bias, confidence threshold $\theta_c$, reputation threshold $\theta_r$, training size, and sampling points $\Sigma$ for prediction error calculation. Each user $u_i$ is initialized with a reputation $r_i$ and a bias $b_i$, which are uniformly sampled between 0.0 and 1.0 for the former, and between a lower bound $b_l$ and an upper bound $b_u$ for the latter. Such user parameters remain constant throughout a simulation run.

**Algorithm 1** Trust Simulation

**Require:** *Config*: a global configuration object
1: Initialize a set of brokers $O = \{o_1, \ldots, o_n\}$, each with a feedback repository $\mathcal{F}_i \leftarrow \emptyset$;
2: Initialize a set of users $U = \{u_1, \ldots, u_m\}$, each with an inherent reputation $r_i$ and bias $b_i$;
3: **for** $t = 1$ **to** $T$ **do**
4: $c_t \leftarrow u_j$, where $j \leftarrow Random(m)$; {randomly select a client $u_j$}
5: $s_t \leftarrow u_k$, where $k \leftarrow Random(m)$; {randomly select a server $u_k$}
6: Simulate the transaction by $Broker(i, c_t, s_t)$; {$c_t = u_j$ is managed by $Broker_i$}
7: **if** $t \in \Sigma$ **then**
8: Record the current $\hat{R}_u$ and $\hat{B}_u$ for all $u \in U$;
9: **end if**
10: **end for**
11: Compute and output the average of $\epsilon_R$ and the average of $\epsilon_B$;

**Algorithm 2** Broker

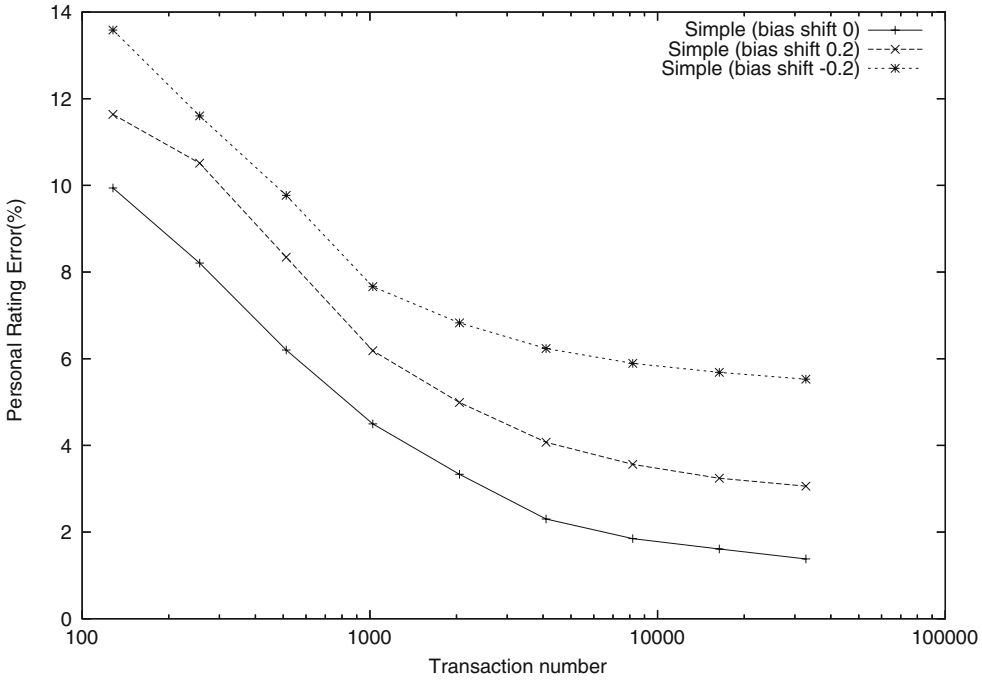
**Require:** a transaction $t$ with client $c_t$ and server $s_t$
1: **if** `GetUserReputation`$(c_t, s_t, \theta_c) \geq \theta_r$ **then**
2: Calculate user feedback rating $r_t$;
3: Update feedback repository $\mathcal{F}_i \leftarrow \mathcal{F} \cup \{r_t\}$;
4: **end if**
5: **if** $|\mathcal{F}_i| == \sigma$ **then**
6: Perform EM training with data in the feedback repository $EM(\mathcal{F}_i)$;
7: Reset the feedback repository $\mathcal{F}_i$;
8: **end if**

## 6 Experiments

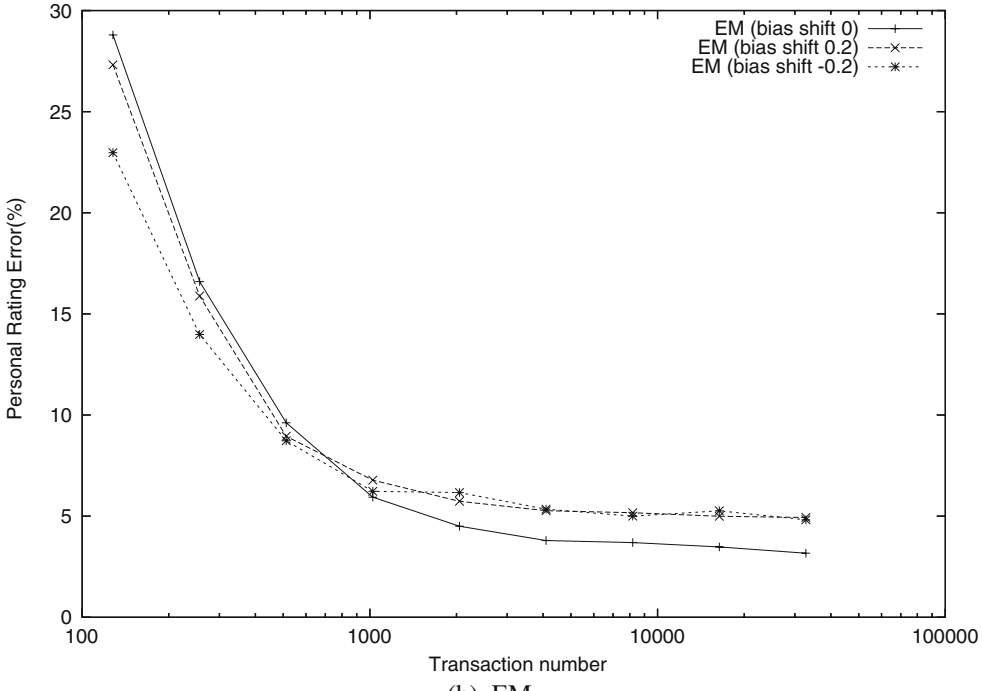
This section presents the results of three sets of simulation experiments that were designed to evaluate the performance of the proposed trust framework. The first set compares performance of two training strategies, *EM* and *Simple*, under shifted bias distributions. The second set examines how different trust models affect the reputation prediction accuracy. The third set illustrates *scalability* of *personalized reputation rating* using broker-based trust management in a distributed environment.

In each simulation, every user is initialized with an inherent CF and bias, which are uniformly sampled

from a given value range. All simulation experiments reported in this paper set the range for CF to be between 0.0 and 1.0. Each simulation is repeated under shifted bias distribution. The range for "Bias shift $\beta$" is defined as $[0.\beta - 0.2, 0.\beta + 0.2]$. For example, the value of bias ranges in $[-0.2, 0.2]$ for "Bias shift 0", $[0.0, 0.4]$ for "Bias shift 2", and $[-0.4, 0.0]$ for "Bias shift $-2$". Both CF and bias remain constant throughout a given simulation run. The confidence threshold, reputation threshold, and training size are constants empirically selected for the experiments. While they have impacts on the speed of convergence, the overall performance trend remains the same regardless of their specific values.




(a) Simple Method




(b) EM

**Fig. 6** Performance with different training strategies ($n = 1, m = 100, T = 35,000$)

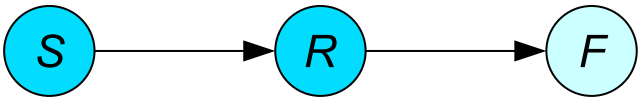



**Fig. 7** Bayesian network trust model without bias

Performance is measured in terms of the *prediction error*, which is defined as the *root mean-squared error* in the estimation. One of the most commonly used measures of success for numeric prediction, the mean-squared error is computed by taking the average of the squared difference between the estimated and correct values. Taking the square root gives the error value the same dimensionality as the actual values. In particular, we use *reputation error* to denote the error in predicting server reputation, and *personal rating error* to denote the error in predicting the summation of server reputation and client bias. In the following experiments, performance is measured at exponentially growing number of transactions. Namely, the sampling points $\Sigma$ is $\{128, 256, \ldots, 32768\}$.

## 6.1 Training strategies

This experiment explores the effectiveness of learning under shifted bias distribution. The simulations compare performance of two training strategies, *Simple* and *EM*. A broker uses the Simple method to estimate server reputation by computing the average rating of all feedbacks collected about the server, and to estimate client bias by computing the average difference between any feedback rating by the user and the corresponding estimated server reputation. Alternatively, a broker may use EM to fit the trust model for predicting server reputation and client bias as described in Section 4.3.

The simulations are set up with a single broker, 100 simulated users, and 35,000 transactions. The results in Figs. 6(a) and 6(b) show that both strategies performed reasonably well, and the personal rating error converges to less than 5% around 10,000 transactions under different bias distributions. With the relatively simple trust model in Fig. 3, we do not observe much performance advantage of EM except that it is less sensitive to bias shift. Given a good trust model, this experiment demonstrates that the choice of training strategy does not have significant impacts on performance.

## 6.2 Performance due to trust models

A major benefit of EM is its *flexibility* in handling different models. While the Simple method needs to be re-coded, EM can take a new model without much effort. In this experiment, the Bayesian network trust model without bias in Fig. 7 is adopted.

Figure 8 shows the simulation results for both Simple and EM under different bias distributions. The

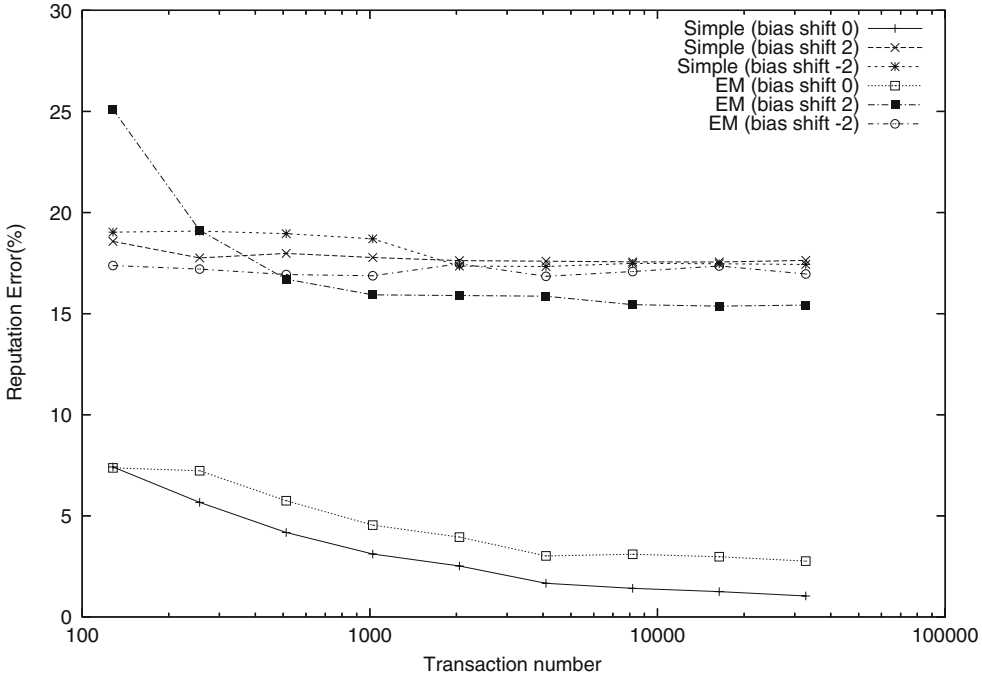
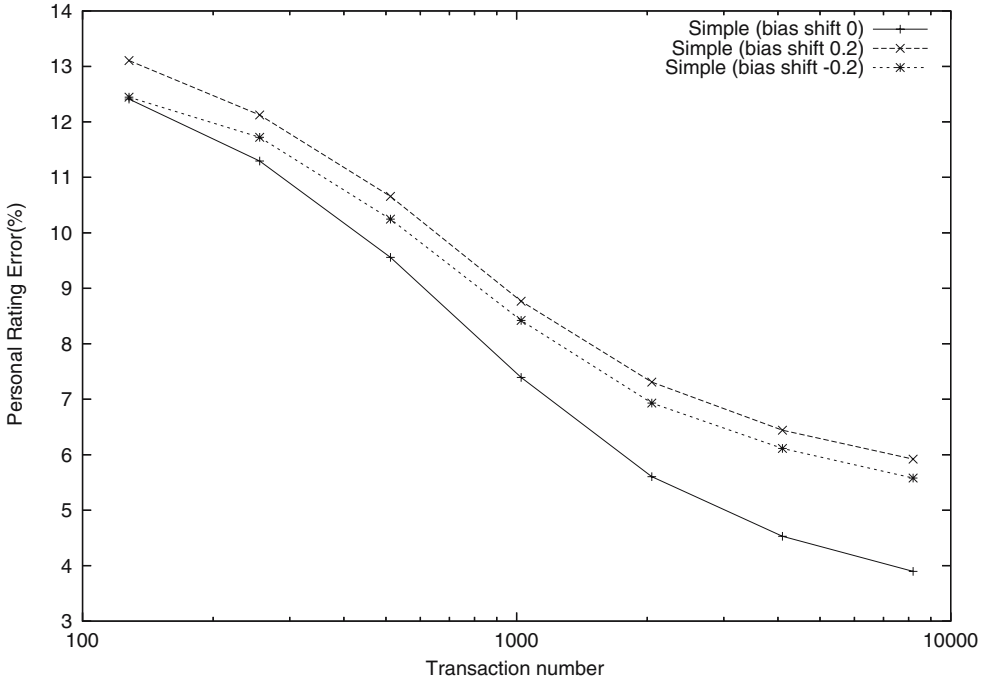



**Fig. 8** Performance based on trust model without bias ($n = 1, m = 100, T = 35,000$)

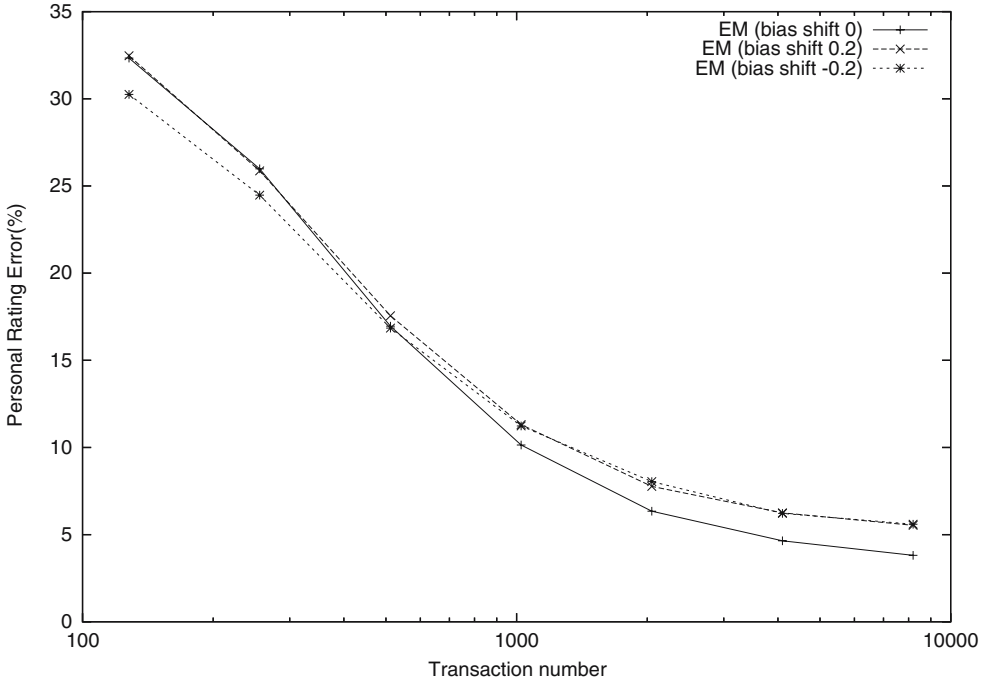
*reputation error* converges to less than 5% within 1,000 transactions when there is no bias shift. However, the error remains above 17% when the bias shift is either 0.2 or −0.2. This experiment highlights the importance of adopting the right model. Performance suffers when the notion of bias is not included in the model.

### 6.3 Distributed trust management

The last experiment evaluates the performance of the two-tier broker-based trust framework in a real distributed environment. The simulations are set up with 10 brokers, 1,000 simulated users, and 10,000 transactions.




(a) Simple Method




(b) EM

**Fig. 9** Performance in a distributed environment ($n$=10, $m$=1,000, $T$=10,000)

Each broker manages 100 users, and the processes can run on any number of networked machines. The trust model in Fig. 3 is used. The EigenTrust mechanism is deployed at the broker level with a threshold of 0.05 and a maximum iteration of 50.

Figures 9(a) and 9(b) show the simulation results of both Simple and EM under different bias distributions. This experiment illustrates the scalability of trust management to multiple brokers running in a real distributed environment. As in the single-broker case, the Personal Rating Error falls below 5% within 10,000 transactions. The two-tier trust management works well in a distributed environment.

## 7 Conclusion

This paper presents a two-tier broker-based framework for distributed trust management. A Bayesian network is defined to model the combined trust from objective server performance and subjective user bias. The probabilistic personalized trust model can be learned using expectation maximization or alternatives training strategies.

The trust brokers aggregate feedbacks from local users while supporting personalized services. When the number of feedbacks collected by a given broker is insufficient to make justifiable recommendations, the broker may request for additional information from trusted brokers. Instead of combining trust ratings from multiple brokers as in Lin et al. (2005), the EigenTrust mechanism (Kamvar et al., 2003) is adopted to compute a global trust vector. At the broker level, such a P2P trust mechanism avoids malicious attacks from uncooperative brokers. At the individual level, the broker-based trust mechanism fosters user community and their willingness to share feedbacks.

Research presented in this paper has improved over previous work in several ways. First, a general two-tier broker-based trust management framework is proposed. Second, Bayesian networks can be used to model personalized trust. Third, a robust implementation of the distributed trust simulator, which is configurable for a wide range of simulations, has been built. In addition, our experiments have demonstrated that personal rating error converges to below 5% consistently within 10,000 transactions (i.e. 10 transactions per user or 1,000 transactions per broker) regardless of the specific training strategy or bias distribution. However, the choice of trust model has a significant impact on the performance of reputation prediction. We've also shown that the two-tier trust framework scales well to distributed environments without much overhead.

**Acknowledgements** This research is supported in part by the National Science Council of Taiwan #NSC-94-2218-E-002-057, Institute for Information Industry #94-CS-0457, UC MICRO project #04-0511 and GeoSpatial Technologies, Inc. The authors would like to thank Chia-en Tai and Haiyin Lu for sharing earlier versions of the previous trust simulator and valuable lessons gleaned from earlier experiments. Special thanks go to the anonymous reviewers for their constructive suggestions that helped improve the paper.

## References

Dellarocas, C., & Resnick, P. (April 2003). Online reputation mechanisms: A roadmap for future research. In *Summary Report of the First Interdisciplinary Symposium on Online Repurtation Mechanisms*. Retrieved from http://ccs.mit.edu/dell/papers/symposiumreport03.pdf.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, *B39*, 1–37.

Fernandes, A., Kotsovinos, E., Ostring, S., & Dragovic, B. (2004). Pinocchio: Incentives for honest participation in distributed trust management. In: *Proceedings of the Second International Conference on TrustManagement* (pp. 63–77). Oxford, UK.

Jurca, R., & Faltings, B. (2003). An incentive compatible reputation mechanism. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems* (pp. 1026–1027). New York: ACM.

Jurca, R., & Faltings, B. (2004). Eliciting truthful feedback for binary reputation mechanisms. In: *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence* (pp. 214–220). Beijing, China.

Kamvar, S.D., Schlosser, M.T., & Garcia-Molina, H. (2003). The eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the Twelfth International World Wide Web Conference*. Budapest, Hungary.

Lin, K. J., Lu, H., Yu, T., & Tai, C. e. (2005). A reputation and trust managment broker framework for web applications. In: *Proceedings of the IEEE International Conference on e-Technology, e-Commerce, and e-Service* (pp. 262–269). Hong Kong, China.

Pavlov, E., Rosenschein, J.S., & Topol, Z. (2004). Supporting privacy in decentralized additive reputation systems. In: *Proceedings of the Second International Conference on Trust Management*. Oxford, UK.

Wang, Y., & Vassileva, J. (2003). Bayesian network trust model in peer-to-peer networks. In: *Proceedings of Second International Workshop on Peers and Peer-to-Peer Computing* (pp. 23–34). Berlin Heidelberg New York: Springer.

Yu, B., & Singh, M.P. (2000). A social mechanism of reputation management in electronic communities. In: *Proceedings of the 4th International Workshop on Cooperative Information Agents IV, The Future of Information Agents in Cyberspace* (pp. 154–165). Boston, Massachusetts.

Yu, B., & Singh, M.P. (2002). An evidential model of distributed reputation management. In: *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems* (pp. 294–301). New York: ACM.

Zacharia, G., & Maes, P. (2000). Trust management through reputation mechanisms. *Applied Artificial Intelligence Journal*, *14*(9), 881–908.

**Jane Hsu** received her PhD in Computer Science from Stanford University in 1991. She is an associate professor on Computer Science and Information Engineering at National Taiwan University. Her research interests include intelligent multi-agent systems, data mining, service oriented computing and web technology. Prof. Hsu is on the editorial board of the International Journal Service Oriented Computing and Applications (published by Springer). She has served on the editorial board of Intelligent Data Analysis–An International Journal (published by Elsevier and IOS Press) and the executive committee of the IEEE Technical Committee on E-Commerce. She is a Program co-Chair for the 2005 IEEE International Conference on e-Technology, e-Commerce, and e-Service, as well as the 2004 Conference on Artificial Intelligence and Applications. In addition, she is actively involved in many key international conferences as organizers and members of the program committee. She is a member of AAAI, IEEE, ACM, Phi Tau Phi, and has been an executive committee member of TAAI.

**Tsung-Hsiang Chang** is a graduate student in the department of Computer Science and Information Engineering at National Taiwan University. His research interests include intelligent systems, software engineering, and human-computer interaction.

**Kwei-Jay Lin** received the BS in Electrical Engineering from National Taiwan University, the MS and Ph.D. in Computer Science from the University of Maryland, College Park. He is a Professor in the Department of Electrical Engineering and Computer Science at the University of California, Irvine. Prior to joining UCI, he was an Associate Professor in the Computer Science Department at the University of Illinois at Urbana-Champaign. His research interests include service-oriented systems, e-commerce and enterprise computing, real-time systems, scheduling theory, and distributed computing. Dr. Lin is an Editor-in-Chief of the International Journal of Service Oriented Computing and Applications (published by Springer), and the Editor-in-Chief of the Software Publication Track, Journal of Information Science and Engineering (published by Academia Sinica, Taiwan). He has served on the editorial boards of IEEE Transactions on Parallel and Distributed Systems and the IEEE Transactions on Computers. He is a Co-Chair of the IEEE Technical Committee on E-Commerce since 2004. he has chaired many international conferences, including Conference Chairs for the 2006 IEEE Conference on E-Commerce Technology in San Francisco, the 2004 IEEE Conference on e-Technology, e-Commerce and e-Service in Taipei, the 2003 IEEE Conference on E-Commerce in Newport Beach, CA, and the 1998 IEEE Real-Time Systems Symposium in Madrid, Spain.

**Chien-Ju Ho** is a graduate student in the department of Computer Science and Information Engineering at National Taiwan University. His research interests include intelligent learning systems, machine learning, and human-computer interaction.

**Han-Shen Huang** is a postdoctoral research fellow at the Institute of Information Science of Academia Sinica in Taiwan. His research interests include machine learning and applications of probabilistic models.

**Wan-rong Jih** is a PhD candidate in the Department of Computer Science and Information Engineering at National Taiwan University. She has conducted research on multi-agent systems, optimization algorithms for dynamic vehicle routing and protein secondary structure prediction. Her current research focuses on context-aware technology and service-oriented computing.