A DISTRIBUTED REPUTATION BROKER FRAMEWORK FOR WEB SERVICE APPLICATIONS

Kwei-Jay Lin
Department of Electrical Engineering and Computer Science
University of California, Irvine, California, USA
klin@uci.edu

Yue Zhang
Department of Electrical Engineering and Computer Science
University of California, Irvine, California, USA
yuez@uci.edu

Jane Y.J. Hsu
Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
yjhsu@csie.ntu.edu.tw

Tao Yu
Department of Electrical Engineering and Computer Science
University of California, Irvine, California, USA
tyu1@uci.edu

ABSTRACT

This paper presents a distributed reputation and trust management framework that addresses the challenges of eliciting, evaluating and propagating reputation for web applications. We propose a broker framework where every user is associated with a reputation broker who collects for its users the reputation ratings about any web service. In return, a user provides its broker the service rating after each transaction with any service in order to build up the reputation for the service. In our mechanism design, brokers form a trust network where they exchange and collect reputation data about all services. By delegating trust management to brokers, individual users only need to check with their brokers about the reputation of a service before any transaction. We present the distributed reputation and trust management framework and show the performance of the system by simulations.

Keywords: Trust, Reputation, Web Services, Distributed Systems

1. Introduction

In real world, *trust* is a relationship between two entities: it is one's belief on certain attributes about the other. There are several properties to be considered in a trust relationship:

1. *Identification*: whether the subject entity is what it claims to be.
2. *Qualification*: whether the subject entity is capable of performing specific services.
3. *Consistency*: whether the subject entity is able to deliver a service with an acceptable certainty.

The first property is usually ensured by requesting certain information from the subject and identifying that the subject is among a specific group that can be trusted to have certain privileges. The second property is often answered by asking for a proof about the subject's capability and performing some validation procedure using pre-defined policy. The last property is the most difficult one since it should not be claimed by the subject itself. The consistency of services or performances provided must be verified by others, either by a formal certification process or by feedbacks from peer clients. In this paper, we call the third property the *reputation* about a peer.

For e-services, the design of a trust management framework is a discipline that has gained much attention in recent years [Dellarocas 2003] due to the growth of online transactions and e-commerce activities [Chen 2003, Wang 2001]. Traditional distributed encryption and authentication mechanisms can only solve the identification and, to some extent, the qualification problems. However, it is unlikely that an entity will act consistently in all transactions. Reputation systems [Yu 2000, Jurca 2003, Xiong 2004, Dellarocas 2003] are thus far the most preferred mechanism addressing this problem. Reputations systems where feedback ratings are aggregated over a period of time to reflect the trustworthiness of a service have been implemented in many e-marketplaces. The success of Amazon and eBay proves that such reputation systems are helpful in fostering trust for vendors.

Both Amazon and eBay are examples of centralized reputation systems. With a single trust authority controls all reputation information, such systems may be vulnerable or inflexible. In addition, centralized authority may be subject to the scalability problem. To solve these problems, distributed trust management systems have been proposed and studied [Yu 2000, Xiong 2004]. In a distributed trust system, reputation information is scattered among parties in the system. The distributed approach brings new challenges, including how to elicit reputation information, how to evaluate the trustworthiness of a party with information gathered from potentially untrustworthy parties, and how to propagate reputation information throughout the community.

This paper presents the design of a general trust framework and the implementation of trust brokers. Due to the complexity of trust management, we propose the design of software trust brokers to manage the trust relationship for e-service users. In real world communities, people often rely on recommendations by word-of-mouth from trusted acquaintances or trusted experts to evaluate the trustworthiness of a service provider. Different trust levels exist in our real-life activities with close friends, friends of friends, people with known qualifications, and total strangers. Our design is motivated to emulate these real-life trust building processes and reputation mechanisms by using software brokers as trusted experts.

We propose that online users make use of *trust brokers*, which may be implemented using some common, certified software package like Liberty Alliance [Liberty Alliance 2004]. Like search engine sites and portals, trust brokers are independently maintained and operated; users may choose among many brokers either freely available or with paid services. A broker collects service reputation information for its users. Each user, after each transaction with an invoked service (from some service provider or vendor), will produce a reputation rating on the service and send it to a broker so that the broker may build up the reputation about that service. In this way, the reputation of a service can be produced from the report of all its previous clients.

Another issue that we study is how to build the referral network among trust brokers. Trust brokers interact and share reputation information just like real world travel agency brokers and stock brokers. Due to the distributed nature of trust brokers, it is infeasible to collect all reputation information from all brokers. We need efficient mechanisms to collect and manage the distributed reputation information in trust brokers. However, given possibly contradictory experience, brokers may not have the same level of trust on each other. Brokers usually solicit service reputation information from those brokers they trust more. Our proposed mechanism allows brokers to build up different trust levels on each another, and to select only those they trust to share reputation information.

The rest of this paper is organized as follows. Section 2 provides an overview of the current research on trust management. We introduce our trust system framework in Section 3 and a trust broker design in Section 4. Some simulation results are presented in Section 5. Section 6 presents an extended model when a service's behavior may be affected by the transaction value. The simulation study for the extended model is presented in Section 7.

2. Previous Work

Research activities in distributed trust management include broadly the following areas.

1. *Formalizing trust* [Dellarocas 2004]. There are many different ways to calculate trust. In practice, Amazon simply takes an average of product ratings based on customer reviews. BizRate compiles the average satisfactory index about the merchant in addition to product rating; while eBay presents the *feedback score* and the percentage of positive feedbacks. Researchers proposed various improvements, e.g. by giving higher weights to feedbacks from users with better reputation. A successful reputation system should make it hard to build up good reputation so that a user is less likely to abuse its hard earned reputation.
2. *Referral network systems where agents cooperate to propagate reputation information in the community* [Yu 2000]. Each agent is assumed to have neighbors, which are then connected to their own neighbors. An agent dynamically restructures its neighbors based on their trustworthiness. A direct neighbor is not as trustworthy as some indirect neighbors if the direct neighbor's opinions are not consistent with the agent's own experience.
3. *Using context factor to model the dynamic behavior of entities* [Xiong 2004]. It has been proposed that *Transaction Context* may be used as an important factor when aggregating the feedback from each transaction as transactions may differ from one another. For example, if a community is business savvy, the size of a transaction is an important context that should be incorporated to weigh the feedback for that transaction. However, their model mixes reputations with different context weights together. One major concern is this game rule allows the system to use multiple small context weight positive feedback to hide the effect of one big context weight negative feedback, which may need further research to prevent it from happening.
4. *Incentive mechanisms for eliciting honest feedbacks* [Jurca 2003]. Studies of eBay's reputation system have shown that it is difficult to elicit feedbacks. An important reason for such difficulty is the lack of incentives for the users. In some communities, users are reluctant to share information for fear that it will give competitive advantage to others. Incentive mechanisms address this issue by providing incentives to users that gives honest feedbacks through some side payment mechanism.
5. *Mechanisms to guard against coordinated attack against the system* [Xiong 2004]. Biased feedbacks can be filtered out with a large number of feedbacks. Even a simple approach such as to take the average of all

ratings is able to filter out subjective and biased ratings. In contrast, coordinated attacks on the system are much harder to guard against. A group of users might form a collusion giving only positive feedbacks to the members in the group and negative feedbacks to others outside the group.

Our research adopts and integrates many of the ideas from past research of the first three topics. We propose a definition of service reputation based on past transaction outcome. We also integrate reputation data from distributed brokers using a mechanism similar to the referral network.

3. System Model and Assumptions

Using a distributed trust system, reputation information in general is distributed within the community. The challenge for a user is to gather enough information for making an informed judgment on the trustworthiness of a service. More specifically, the trust building process involves two separate problems:

1. how to gather reputation information, and
2. how to utilize the information gathered

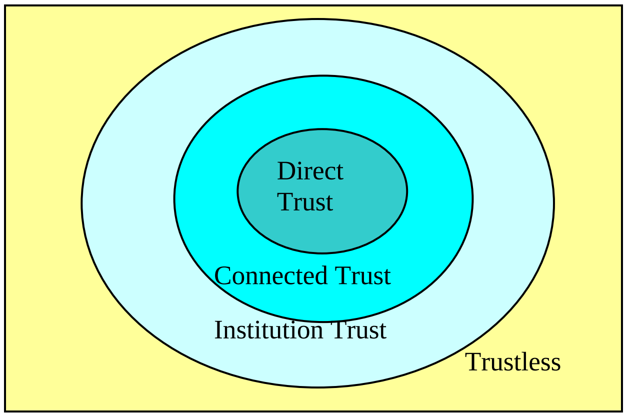

Figure 1: Trust Hierarchy

We have modeled the trust relationship into three levels as shown in Figure 1. The first level, the *direct trust* level, is for users that subscribe the same trust broker where there is a direct measurement of trust among all members. The second level, the *connected trust* level, is when two users utilize two different trust brokers. So users must find information about each other through some distributed trust collection protocols. If there is not enough trust that can be gathered at this level, the third level, the *institution trust* level, relies on a centralized trust authority to provide global certified trust service about each other for decision making. If the third level still cannot meet the trust policy, users will have to use some trustless protocol [Atallah 2003] to conduct business. Our work in this paper is mostly on the connected trust level.

Figure 2 shows our distributed trust and reputation management system consisting of three types of components: users, brokers, and reputation authorities. In our model, all users also function as servers themselves (just like agents or in P2P systems). A user in the role of a "client" can generate any request to initiate a transaction with another user, assuming the role of a "server". In this architecture, users rely on their trust brokers to collect reputation information. A broker typically works for multiple users who are willing to share reputation information among the group. Each broker maintains a reputation database that collects the reputation of all services that have had transactions with its user clients.

This project proposes a distributed trust and reputation management framework that addresses the challenges on managing trust among e-services. Our work is related to the referral network approach [Yu 2000] by using a network of brokers to propagate reputation information. However, we collect the reputation on both services and brokers. Each service is assumed to have a *consistency factor* (CF) which is the probability that it could deliver a requested service. The reason for failure to deliver a service may be due to hardware, network or system overload problems and may be difficult to correct for cost and other reasons. Our trust broker design is to identify the consistency factor, or *reputation*, of each service.

After each transaction, a user *A* sends a rating on a service *B* to *A*'s broker. The current system assumes users are diligent in providing honest feedbacks. Thus a broker will collect the complete and accurate ratings generated by its clients. In this way, a broker may accumulate enough reputation information (i.e. *direct trust*) about a service to support its clients. However, if a broker finds that its local reputation database is insufficient for making a recommendation to its users, it will contact the other brokers (i.e. information from *connected trust*) or reputation authorities (information from *institution trust*) to gather more information.

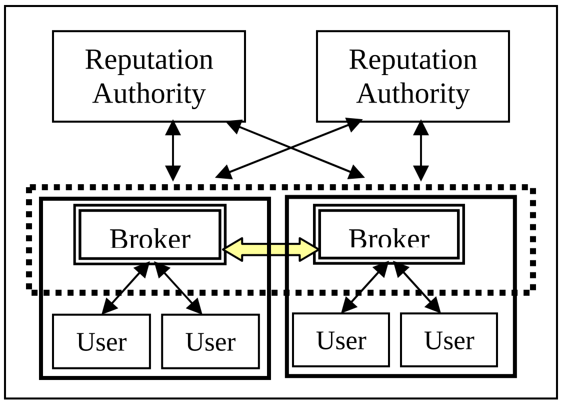

Figure 2: System Model

While users rely on their brokers to manage reputation information, brokers talk to each other to sum up the reputation about a subject server. In our model, brokers may decide not to share certain reputation information with another broker, but they cannot lie or produce false information. A server, however, may not provide consistent services or results. Therefore, some server may have a less-than-perfect reputation.

Reputation authority is the last resort for any broker if it cannot find sufficient information about a peer. Reputation authority, which may be set up by industrial consortium or by public institutions, maintains a global database about all servers. Due to its size, however, the ratings kept by any authority may be incomplete or out of date. For any given service, the rating from each reputation authority may be different, just like credit rating companies may have erroneous information in real life. Brokers therefore will consult a reputation authority only if there is no other option available.

4. Broker Architecture

A broker has two main components (Figure 3), *reputation manager* and *connection manager*. The reputation manager receives requests from client users and other brokers. It decides whether to ask the connection manager for collecting information from other brokers or reputation authorities. The connection manager takes requests from reputation manager and passes the requests to other brokers and reputation authorities. In this section, we present each component in details.

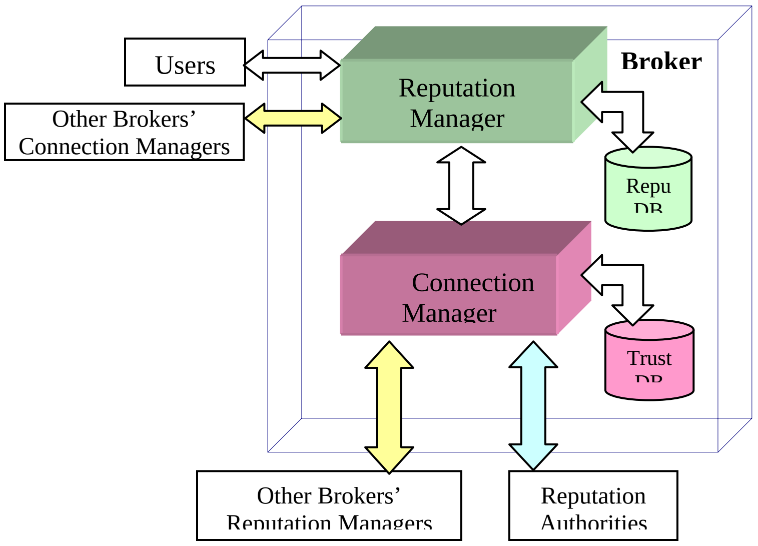

Figure 3: Broker Architecture

4.1. Reputation Manager

Reputation manager has three functionalities. First, it receives and responds to requests from its client users and other brokers. Second, it is responsible for integrating and maintaining its users' feedback information. Third, it forwards reputation-solicit requests to the connection manager when necessary.

Every time a broker receives a feedback rating from one of its users, it updates the reputation information about the service in its reputation database. Each reputation record has the following fields:

1. UserID: the ID of the service
2. Rating: a *reputation value* between 0 and 1
3. Count: the number of transactions used as the basis to generate the reputation
4. Timestamp: the time of the last feedback

The timestamp, which records the time when a feedback rating was last submitted, is necessary in order to value more recent ratings with higher weights. The total number of transactions used to generate the rating is an important indication on the accuracy of the reputation.

4.1.1. User Request

A user sends a transaction request and a threshold to its broker in the form of <myID, serverID, repuTrans>. The broker first checks the total transaction count in its local database, and returns the server's rating if the count is greater than repuTrans. Otherwise, the broker will forward the request message to the connection manager, which uses the broker-broker protocol to contact other brokers for reputation data.

4.1.2. User Rating

The reputation manager collects feedback ratings from its clients about each transaction. Let N be the transaction count of the current rating R_{old}. After a client submits a rating r regarding *x*, the reputation manager updates its reputation value of peer *x* stored in its local database from R_{old} to R_{new}. The reputation value of *x* is updated as follows:

$$R_{new} = e^{-\beta\Delta t}\frac{N}{N+1}R_{old} + (1-e^{-\beta\Delta t}\frac{N}{N+1})r \qquad (1)$$

The difference in feedback time between r and R_{old} is denoted by Δt, and $e^{-\beta\Delta t}$ specifies the discount factor of R_{old}. Eq. 1 considers that more recent feedbacks provide more accurate ratings on the service's behavior. This is true if some services are subject to recent server hardware problems, network connection problems, or software upgrade issues. In general, the result from more recent transactions allows users to have a better expectation on the current service performance. On the other hand, if service quality is not time dependent we could simply set β as 0. In that case, the non-time dependent formula is simply $R_{new} = \frac{N}{N+1}R_{old} + (\frac{1}{N+1})r$ that takes the average of all past ratings.

4.2. Connection Manager

The connection manager provides two important functions. It maintains a list of trusted brokers as well as a list of trusted reputation authorities. It acts as the interface between the broker and the trust network, and is responsible for sending requests to other brokers and reputation authorities.

4.2.1. Broker-Broker Trust Protocol

In a distributed system, brokers should not rely only on their own experiences to rate all services. Collaboration among brokers is extremely useful. Only through collaboration can the system identify untrustworthy servers promptly and reduce the risk for its clients. For this reason, brokers have a strong motivation to cooperate.

However, given that the objective of a broker is to provide a unique service to its users, some brokers may choose not to share its data with other brokers for fear of competition for clients. Therefore, each broker maintains a list of trusted brokers and their *trust values* in its trust database. Trust information is *not* static. The trust value for a peer broker is based on the number of accurate recommendations that peer has provided earlier. It is updated each time after a recommendation is received and compared with the actual transaction outcome. At the beginning, all peer brokers are given a neutral trust value of *X*=0.5. After each transaction experience, if the recommendation from the peer is correct, the trust value is updated using the formula:

$$X = X + F*(1-X) \qquad (2)$$

Otherwise, X is updated using the formula

$$X = X*(1-F) \qquad (3)$$

In the above equations, F is a positive index with a value less than 1. For example, if F is 0.2 and X was 0.6, the new value of X is 0.68 when the recommendation received is good; while a bad recommendation will reduce X to a value of 0.4. The update equations are designed in such a way that X always has a value between 0 and 1. Moreover, it is more difficult to gain additional trust than to lose trust when X has a large value.

The connection manager of each broker maintains a list of fellow brokers sorted by their trust values. When the reputation about a specific peer is requested, the broker will contact the first m brokers with a trust value higher than a threshold value T. m is called the fan-out value. In addition, a *depth* parameter may be specified in the recommendation request. A trusted broker will forward the request to its own trusted brokers with the depth value decremented by 1. The forwarding requests form a recursive recommendation chain until the depth value reaches 0. The length of the chain is bounded by the *depth* parameter, which is in turn decided by how much local reputation is already there, specified by the original requestor. For example, suppose that the local reputation database currently has a size of 100 transactions, but the user client wants to have a size of 1000 transactions, the broker may want to define $m = 3$ and depth = 2. A total of 12 brokers will be contacted by the request (3 in the first level and 9 in the second level) that most likely will return recommendations based on the experience of 1200 transactions. If there are already a large number of local transactions, the broker should use a smaller depth with a larger m such that the recommendation collection can be done more efficiently with fewer indirect requests.

A threshold value T is used to filter out any low-trust broker from the reputation solicitation. The threshold value should be used by all connected brokers at all levels for the specific request.

4.2.2. Aggregating Reputation Recommendations

A connection manager keeps all recent recommendations from the broker chain in its database. All recommendations received for the same request share the same request ID. The broker uses the following method to aggregate the recommendations.

Each recommendation R_i is weighed by the number of transactions N_i, the time differential factor $F(\Delta t)$, and the trust value on that broker X_i. Each broker uses a differential threshold to decide whether the recommendation should be taken at the full value. If R_i was reported with a time differential less than the threshold, the value of the time differential factor $F(\Delta t)$ is 1; otherwise, it is $e^{-\beta\Delta t}$. We have

$$R = \sum \frac{X_i * N_i * R_i}{N} * F(\Delta t) \qquad (4)$$

where $N = \sum X_i * N_i * F(\Delta t)$.

The connection manager forwards the reputation recommendation to the reputation manager, which in turn forwards it to the user. In the recommendation, the locally recorded reputation is combined with the recommendations received from all brokers. If the user decides to act on the recommendation, it will send a feedback report to its broker after the transaction. The broker's reputation manager will forward the user's rating to the recommendation manager. The connection manager checks its recommendation database for all foreign recommendations of the target server. If the user's rating is the same as the recommendation within an acceptable margin, the connection manager will update the trust value for the broker who sent the recommendation. The connection manager then rearranges the order of the trusted brokers list according to the new trust value.

4.2.3. Responding to Other Brokers

As we have discussed earlier, trusted brokers will cooperate with each other in the community. However, if broker B asks C, which has a small trust value on B, for a recommendation on some service, C may decide not to comply willingly. This is only fair if B has not given too much credible information to C in the past. The behavior of a broker on another broker is affected by their mutual trust values. A broker will return the recommendation with the size discounted by the trust value. For example, if the local reputation has a size of 1000 transactions, and the trust value is 0.6, the reputation record reported will reflect only 600 transactions.

It should be clear that the trust value is not symmetric between two brokers. One broker may be rated highly by another broker but not vice versa. For this reason, the local trust values for all brokers should be kept by a broker carefully and privately. Since the values are updated dynamically, the trust relationship among brokers is time-varying and unpredictable. However, the long-term relationship among good brokers should prevail so that they will all belong in a trusted cluster.

In the case when a broker and its trust network together do not have enough evidence about a potential client, a broker may consult a reputation authority. A reputation authority collects reputation information from public on a voluntary basis, and produces a global rating on all services. Since a reputation authority is an independent service provider, its data may be more unbiased. On the other hand, as in the case of any big organization, the data from a global reputation authority may not be as accurate and timely as some local user groups.

5. Base Model Performance Study

To test the efficiency of our trust formation and system design, we have conducted simulation on reputation collections. Each service in our simulation has a randomly assigned consistency factor (CF) that defines its capability to deliver a service consistently. To evaluate the system's reputation knowledge about a service, we compute the total standard deviation between all brokers' local CF values on the service and the service's true CF value. The average deviation of all servers defines the system's overall reputation correctness (SC). A system is said to have perfect reputation knowledge when SC is 0.

5.1. System Parameters

We have conducted simulations of a reputation system with 600 users using 60 brokers. Each broker collects transaction feedbacks from its 10 users and interacts with other brokers to gather global reputation about other users. In our simulation, the system generates a new transaction every 1 msec. Therefore the same transaction pair (A, B) will be generated every 360 seconds on average. Since each broker is maintaining reputation database for 10 users, we should have one new reputation rating on B received by A's broker every 36 seconds. We decide that any of B's execution history outside of the window of the last 100 transactions with A is no longer meaningful to A. We can thus derive the ß value to be $1/(36*10^5)$ or about $2.7*10^{-7}$ in Eqs. (1) and (4).

In our simulations, the F value in Eqs. (2) and (3) is randomly selected for each broker in the range of [0.2, 0.5]. Initially, each broker has a trust value of 0.8 on 4 other brokers, and 0.5 on the rest of the brokers. We also set the broker search fan-out m = 2 and depth = 5. In other words, each broker will connect to the two brokers trusted most in the trust network and the search for reputation data in the trust network can go as deep as 5 indirect levels.

In each of our simulations, we generate $6*10^6$ transactions between users and servers, and compute the system correctness after every 60,000 transactions and thus have 100 data points from each simulation.

5.2. Simulation Result

Figures 4-7 show the simulation results. The x-axis represents the data points and the y-axis represents the SC, which is the average of standard deviation. In each figure, we have 3 curves, for different initial values on users in the broker's reputation database; the values are set to 1, actual CF, and 1-CF respectively. The ß value is set to 10^{-6} or $2.7*10^{-7}$. Another parameter, the reputation threshold, is used to decide whether to proceed with the transaction. If the reputation return from a broker is below the threshold, *A* will not conduct the transaction with *B*. In that case, no update on *B*'s reputation can be reported to the broker.

The result shows that the initial reputation value has a big impact on the system correctness. When all brokers have a correct initial value on every user's CF, they are more likely to keep the system in a reasonably correct state. If the initial reputation data are opposite to the true values (initial values are 1-CF) in brokers' reputation databases, the system will improve slowly with time, if the reputation threshold is 0. In all cases, the system correctness is the worst if the initial reputation values are 1-CF. If all brokers assume all services are perfect (CF=1) initially, the SC will be large initially but improve significantly over time. This shows our trust network is working and converging toward a more informed system.

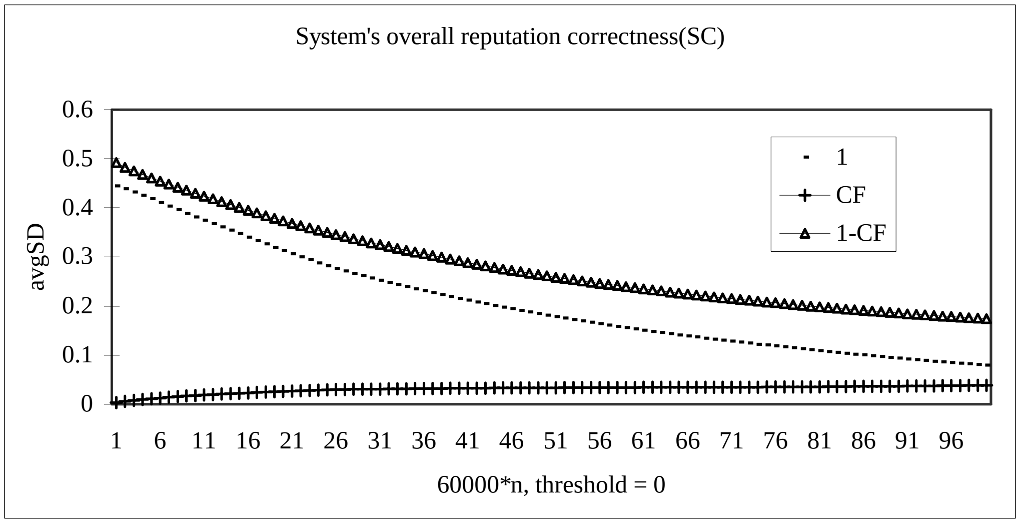

Figure 4: SC for threshold=0 and ß=10^{-7}

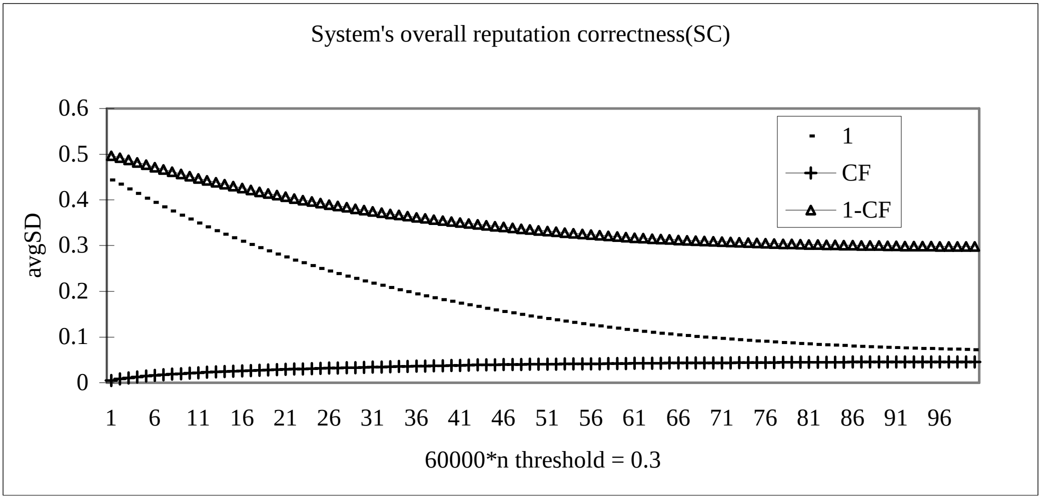

Figure 5: SC for threshold=0.3 and ß=10^{-7}

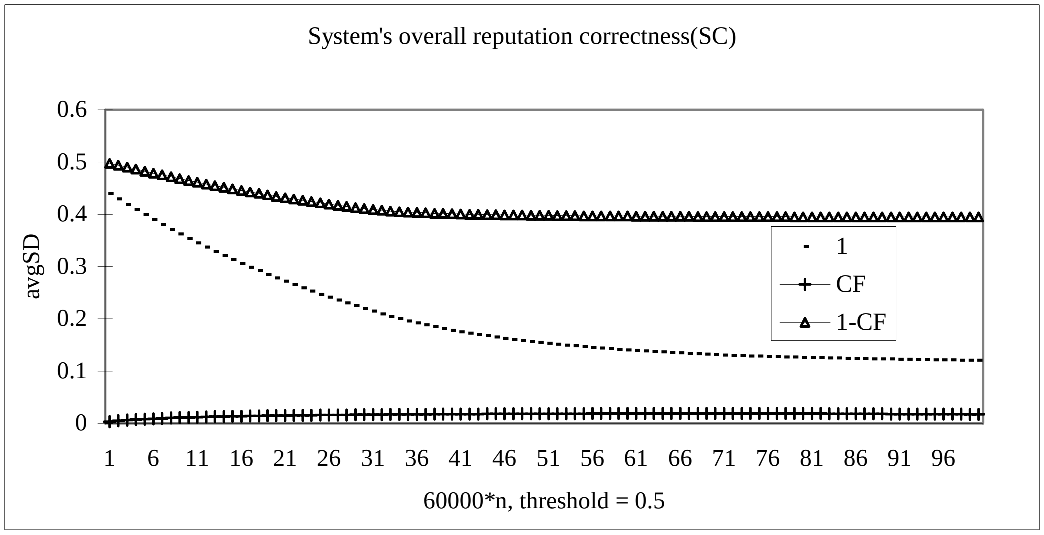

Figure 6: SC for threshold=0.5 and ß=10^{-7}

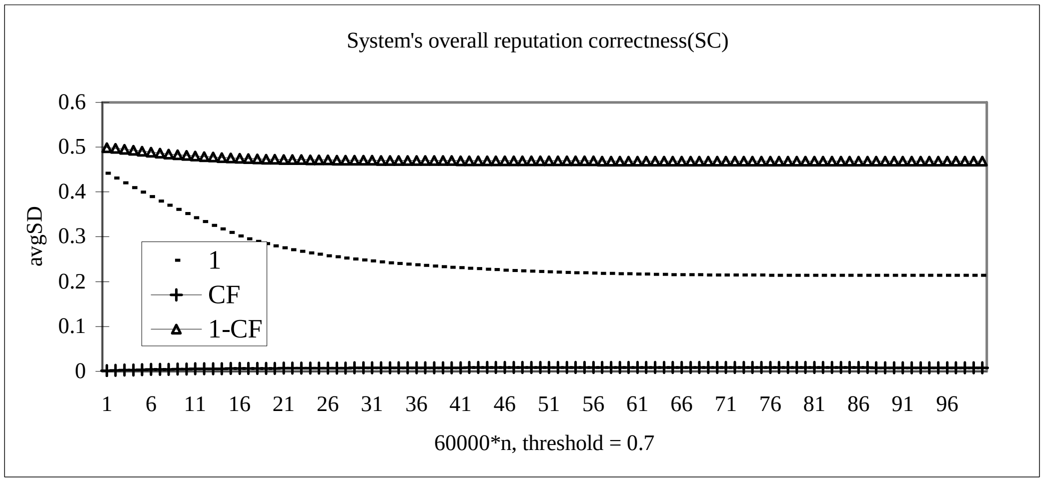

Figure 7: SC for threshold=0.7 and ß=10^{-7}

When the reputation threshold is high, a user is more likely not to conduct a transaction with a supposedly "bad" server and thus will not be able to generate new reputation data to improve the current system knowledge, even when that knowledge is inaccurate. This is an issue in our current system design. We will need additional mechanisms to test if the current reputation data is correct even when the reputation reported is very bad. This is like asking someone to taste a food item even if he knows that most people do not like it. There should be some way for the person to be prepared for the worst, or even be rewarded for the courage. How to verify or correct a bad reputation is an interesting problem for our future study.

6. Extending the Trust Model with Transaction Context

[Dellarocas 2003] suggests that trust may not be a fixed value associated with a service entity but rather is subject to the entities' behavior and applies only within a context at a given time. Entities may vary their service qualities based on the benefits they can receive. Intelligent malicious entities may adaptively deploy a strategy in order to maximize their expected payoff given the rules of the game. There are a number of ways in which such peers can attempt to fool the system and obtain higher benefits. In e-commerce settings, if reputation is not weighed by the transaction size (i.e. total dollar value), sellers could build a good reputation by selectively satisfying many low-valued transactions and then cheating occasionally on transactions that involve a high value, but still allows them to maintain an acceptable reputation.

6.1. Entity's behavior

In earlier sections, a service entity's behavior, which is represented by CF, is assumed to be a constant, no matter how context changes. However, this may not be true for more sophisticated services. The probability of an entity to deliver a requested service successfully may be related to the context, such as the benefit it can get. Therefore, the trust management model should incorporate various context values in evaluating the trustworthiness of services in different communities and transactions.

In this study, we assume that a seller may try to build up a good reputation by providing good services for small-size transactions and then commit fraud on large-size transactions to make a big profit. We therefore suggest that context information such as ***transaction size*** must be taken into account. We assume that a seller's service quality continuously degrades as the transaction size increases and the service quality does not vary as time varies.

The consistency factor of such sellers is shown in Figure 8 as a *negative exponential distribution* function which satisfies the features of CF that (1) entity's CF value decreases gradually as the context weight increases and (2) the range of CF is [0, α], where $0 < \alpha \le 1$. Suppose TB is the size of the transaction, $CF(TB)$, the probability to deliver a requested service with a transaction size ***TB***, is defined as

$$CF(TB) = \alpha \times \exp(-\beta \times TB), \text{ where } 0 < \alpha \le 1, \beta > 0 \quad (5)$$

where α is the parameter to describe the maximum CF value the service can achieve and β is the parameter of the speed that CF decreases as TB increases. The greater the β is, the faster the CF value decreases as the TB increases.

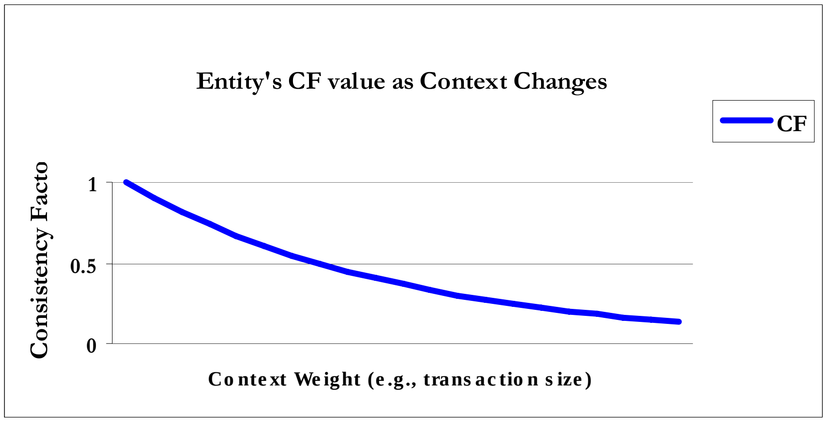

Figure 8: Entity's CF value and context weight

In our design, each reputation record includes the following fields:

- UserID: ID of the server
- The maximum and minimum values of transaction size

- α, β : the two parameters to define the CF function (Eq. 5) which specifies the entity's dynamic behavior. The rating of an entity at certain TB value can be obtained once α, β are known.
- Count: number of transactions used to generate α, β

6.2. User's Reputation Inquiry

Similar to the system design in the base model, a user sends a transaction request and a threshold to its broker in the form of <myID, serverID, repuTrans, TB>. ***TB*** is the new parameter which is the value of the transaction size. The broker first checks the total transaction count in its local database, and returns the server's rating if the size is greater than repuTrans. Otherwise, the broker forwards the request message to the connection manager, which will use the broker-broker protocol to request other brokers for reputation data.

In the extended model, the approach to aggregate the recommendations is similar to Eq. (4) except that the time factor is removed. This is because in this new model, we assume R_i is independent of time. Therefore, each recommendation R_i is weighted by the number of transactions N_i, and the trust value on that broker X_i. Each broker uses a differential threshold to decide whether a recommendation should be taken at the full value. We have

$$R = \sum \frac{X_i * N_i * R_i}{N} \tag{6}$$

where $N = \sum X_i * N_i$

6.3. Reputation Update

6.3.1. The Least Squares Fitting—Exponential technique

The ultimate goal of reputation update is to make the $\alpha_{local}, \beta_{local}$ values to be as close as to $\alpha_{real}, \beta_{real}$ as possible. We use the *Least Squares Fitting—Exponential* technique to derive $\alpha_{real}, \beta_{real}$ as users' feedbacks are accumulated. Suppose $(x_1, y_1), (x_2, y_2) ... (x_n, y_n)$ are the sample points that sit on the curve $y = \alpha \times \exp(-\beta \times x)$. To fit this curve, suppose $\alpha \equiv \exp(A)$ and $\beta \equiv B$, the best-fit values of *A* and *B* are:

$$A = \frac{\sum_{i=1}^{n} (x_i^2 y_i) \sum_{i=1}^{n} (y_i \ln y_i) - \sum_{i=1}^{n} (x_i y_i) \sum_{i=1}^{n} (x_i y_i \ln y_i)}{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} (x_i^2 y_i) - (\sum_{i=1}^{n} x_i y_i)^2} \tag{7}$$

$$B = \frac{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} (x_i y_i \ln y_i) - \sum_{i=1}^{n} (x_i y_i) \sum_{i=1}^{n} (y_i \ln y_i)}{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} (x_i^2 y_i) - (\sum_{i=1}^{n} x_i y_i)^2} \tag{8}$$

Every time a user submits a feedback, the *Least Squares Fitting—Exponential* technique is used to amend $\alpha_{local}, \beta_{local}$. The more feedbacks users submit, the closer $\alpha_{local}, \beta_{local}$ will be close to $\alpha_{real}, \beta_{real}$, the actual parameters of the entity.

6.3.2. Reputation Update Procedures

Every time a broker receives a feedback rating from one of its users, it updates the reputation information about the server in its reputation database. After a client submits a rating r regarding entity *x* with the transaction size *tb*, the reputation manager updates its reputation value of peer *x* stored in its local database from R_{old} to R_{new}.

Assume R_{old} is the old rating for the transaction size *tb*. $\alpha_{local}, \beta_{local}$ are the latest approximations of $\alpha_{real}, \beta_{real}$ that are saved in the broker. N is the transaction count of the current reputation rating $\alpha_{real}, \beta_{real}$. Then R_{old} is calculated by $R_{old} = \alpha_{local} \times \exp(-\beta_{local} \times tb)$, and the reputation value for transaction size *tb* is updated using the formula: $R_{new} = \frac{N}{N+1} R_{old} + \frac{1}{N+1} r$. (tb, R_{new}) is the new data point to be used by the *Least Squares Fitting* process to produce new $\alpha_{local}, \beta_{local}$ values.

Theoretically, the best way to perform *Least Squares Fitting* is to save all historical sample data points in the database. However, from the engineering perspective this approach will cause a large memory overhead. Once users and feedbacks increase, this overhead could be very expensive for the server which is running the trust management broker program.

Therefore, an alternative solution is adopted in our solution without saving actual sample data points in the database. In our approach, the total range of the context is equally divided into fixed intervals, as shown in Figure 9. For each interval, only the boundaries (the coordinates of start point and end point) and the count of the interval (i.e., how many feedbacks have be submitted with the context falling in that interval) are saved. We calculate the *interval data point* (x, y) for each interval for fitting process as follows. For an interval that has received just one feedback, $(x, y) = (tb, R_{new})$; for other intervals, $(x, y) = (x_{avg}, y_{avg})$, where (x_{avg}, y_{avg}) is a point on the function

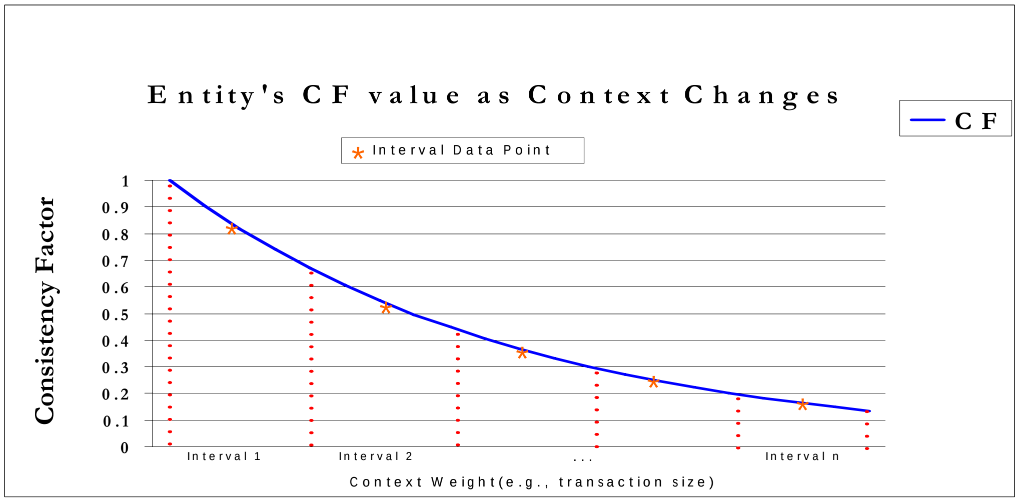

Figure 9: Exponential Least Squares Fitting using sample data points

$y = \alpha_{local} \times \exp(-\beta_{local} \times x)$. y_{avg} is the average value of the function $y = \alpha_{local} \times \exp(-\beta_{local} \times x)$ on the interval range from $tb1$ to $tb2$, and can be found as follows:

$$y_{avg} = \frac{\int_{tb1}^{tb2} \alpha_{local} \times \exp(-\beta_{local} \times x)}{Interval_{length}}$$

Given y_{avg}, x_{avg} can be calculated since (x_{avg}, y_{avg}) is a point on the negative exponential curve.

During the fitting process, each *interval data point* is duplicated by the count of that interval in order to give each *interval data point* a different importance weight. Finally, if there are n intervals, a total of $Count_1 + Count_2 + ... + Count_n$ data points will be used for the fitting process based on Eqs. (7) and (8).

7. Performance Study for System with Transaction Context

To model the entities' context dependent behavior, every entity in a system has a randomly assigned $\alpha_{real}, \beta_{real}$ pair that defines its dynamic capability to deliver a service. The probability of the entity to deliver its service at transaction *tb* is defined as $y = \alpha \times \exp(-\beta \times tb)$. Every broker keeps in its reputation database $\alpha_{local}, \beta_{local}$ of all servers. $\alpha_{local}, \beta_{local}$ are generated from users' rating reports. To evaluate the system's reputation knowledge about a server, we compute the integral of square values between all brokers' $\alpha_{local}, \beta_{local}$ on the server and $\alpha_{real}, \beta_{real}$. Assume there are m users and n brokers in total, the square integral is:

$$SI = \frac{\sum_{i=0}^{n} \int_{Context_{\min}}^{Context_{\max}} (a_{local\,i} \times \exp(-\beta_{local_i} x) - a_{real_i} \times \exp(-\beta_{real_i} x))^2}{n}$$

The average square root of square integral of all servers defines the system's overall reputation correctness (SC) where $SC = \sum_{i=0}^{m} \sqrt{SI_i}$. A system is said to have perfect reputation knowledge when SC is 0.

7.1. System Parameters

All of the simulation parameters, except the initial CF value ($\alpha_{real}, \beta_{real}$) assigned to the entity and the initial CF value assigned ($\alpha_{local}, \beta_{local}$) in the local broker, remain the same as in Section 5.1.

In this simulation, α_{real} is assigned a random value from the set $\{0.1, 0.2 ... 1.0\}$, β_{real} is randomly selected from the set $\left\{\frac{0.1}{Context_{\max}}, \frac{0.2}{Context_{\max}} ... \frac{1.0}{Context_{\max}}\right\}$. In the local broker, the initial ($\alpha_{local}, \beta_{local}$) is set to ($\alpha_{real}, \beta_{real}$), ($1-\alpha_{real}, \beta_{real}$), and ($1, 0$) for three testing cases of initial broker's CF knowledge of actual CF, 1-CF, and 1 respectively. The minimum context is set to 1 and the maximum context is set to 100,000. The range of the context is equally divided into 5 intervals for the least square fitting process.

7.2. Simulation Result

Figures 10-13 show the results from our simulations. The x-axis represents the data points and the y-axis represents the SC, which is the average of standard deviation. Each figure has 3 curves, using different initial values on users in the broker's reputation database; the values are set to 1, actual CF, and 1-CF respectively. Another parameter, the reputation threshold, is used to decide whether to proceed with the transaction. If the reputation return from a broker is below the threshold, *A* will not conduct the transaction with *B*. In that case, no update on *B*'s reputation can be reported to the broker.

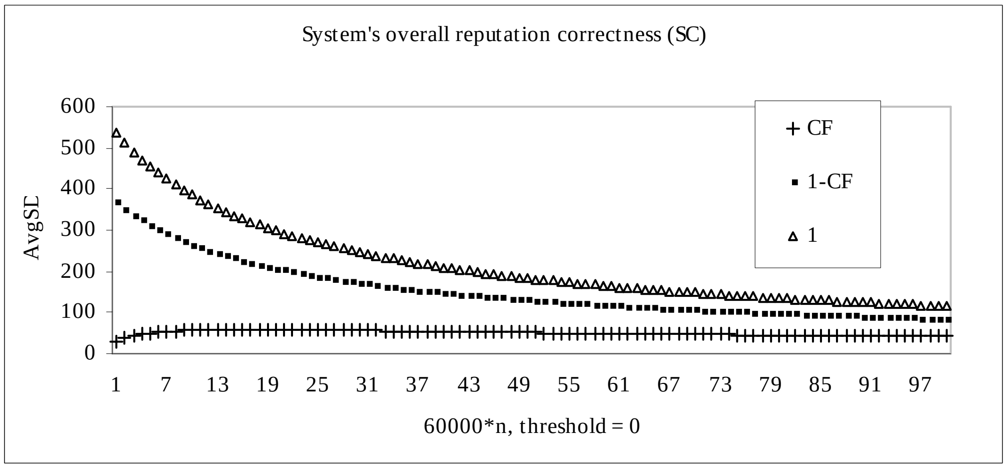

Figure 10: SC for repu threshold=0

All of the simulation results show that the SC curves in the extended model follow the same trend as in the base model: the higher the threshold is set, the slower the *AvgSD* decreases. This matches the statistical nature of reputation that the more feedbacks are accumulated, the more accurate the simulated parameters approach the actual service behavior.

At the same time, we observe no signification simulation performance degradation in the extended service model even when the dynamic service behaviors need to be deal with in the presence of context change. This demonstrates the efficiency of our *Exponential Least Squares Fitting* approach. It does not save any actual sample data points in the database and still can effectively evaluate the dynamic service model's parameters with reasonable costs.

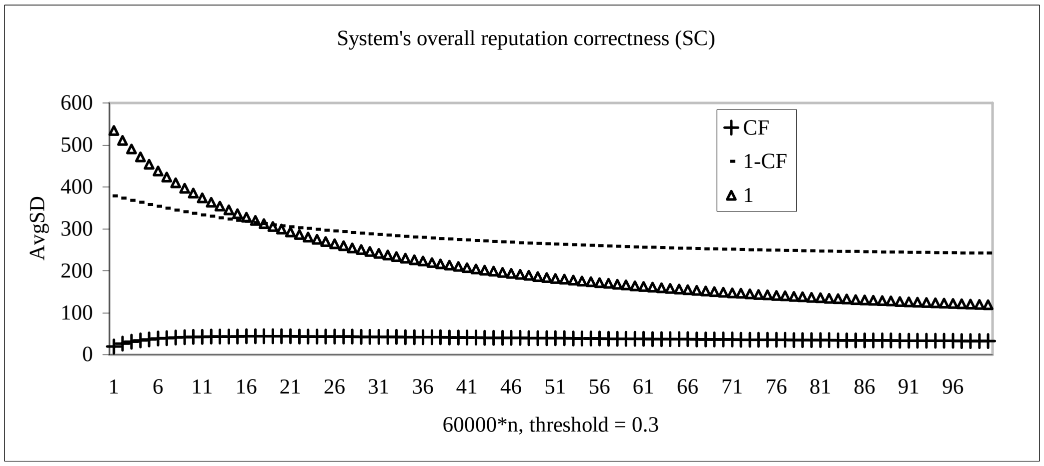

Figure 11: SC for repu threshold=0.3÷

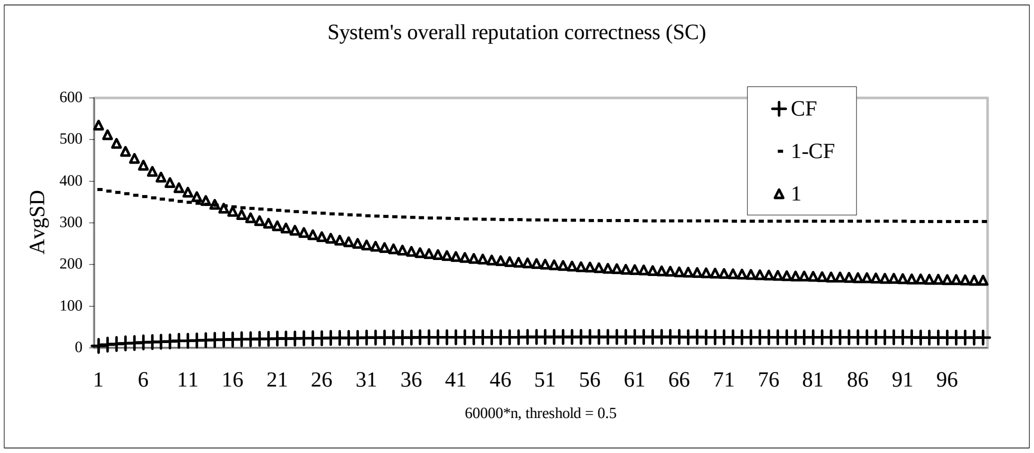

Figure 12: SC for repu threshold=0.5

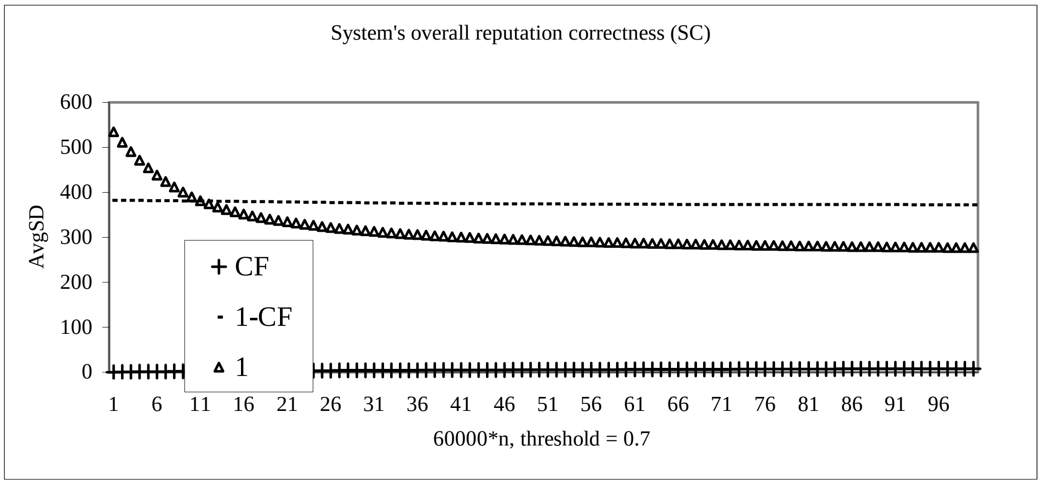

Figure 13: SC for repu threshold=0.7

8. Conclusions

With the expansion of online services and the growing adoption of Web services standards, we expect a continuing growth of e-services and e-commerce. Due to the unreliable nature of e-services, it is important to check the trustworthiness of any service before it is invoked. However, most e-service users may not be capable or even bother to deploy a trust measure themselves. On the other hand, they may be connected to a trust community that can provide them with valuable experiences on a potential service. It is both effective and efficient for a user to use and to share reputation information in such a friendly distributed trust network.

In this paper, we have presented a distributed trust framework where service brokers manage trust information for users. A broker keeps a trust value on each of its fellow brokers in the network and updates the trust value after checking their recommendation against the actual experience. A broker also maintains the reputation on e-servers using the feedback from clients and from other brokers. Our simulation shows that this is an effective way to manage trust and reputation in the e-service environment. We plan to continue this study in the future, exploring further the trust and reputation management in a more complicated context environment that both the changes of time and transaction size can trigger the change of service behavior, as well as more effective mechanisms to eliminate the impact of dishonest feedbacks.

REFERENCES

Atallah, M.J., H.G. Elmongui, V. Deshpande, and L.B. Schwartz, “Secure Supply-chain Protocols”, *Proc. of IEEE Conf. on E-Commerce*, pp. 293-302, Newport Beach, CA, June 2003.

Chen, H., Yu, T., and K.J. Lin, “QCWS: An Implementation of QoS-Capable Multimedia Web Services.” *Proceedings of the 5th Int. Symposium on Multimedia Software Engineering*, pp. 38-45, Taichung, Taiwan, Dec 2003.

Chen M., A.N.K. Chen, and B.B.M. Shao, “The Implications and Impacts of Web Services to Electronic Commerce Research and Practices”, *Journal of Electronic Commerce Research*, volume 4, number 4, 2003

Chen M., and M.J. Meixell, “Web Services Enabled Procurement in the Extended Enterprise: An Architectural Design and Implementation”, *Journal of Electronic Commerce Research*, volume 4, number 4, 2003

Dellarocas, C. and P. Resnick, “Online Reputation Mechanisms: A Roadmap for Future Research” *Summary report of the First Interdisciplinary Symposium on Online Reputation Mechanisms*, April 26-27, 2003. http://ccs.mit.edu/dell/papers/symposiumreport03.pdf

Dellarocas, C., “Building Trust On-Line: The Design of Robust Reputation Mechanisms for Online Trading Communities.” Chapter VII, *Information Society or Information Economy? A combined perspective on the digital era*, Doukidis, G., Mylonopoulos, N. and Pouloudi, N. (Eds.), Idea Group Publishing, 2004

Jurca, R. and B. Faltings, “An Incentive Compatible Reputation Mechanism”, *Proc. of IEEE Conf. on E-Commerce*, pp. 285-292, Newport Beach, CA, June 2003.

Liberty Alliance, http://www.projectliberty.org/, 2004

Wang Y.S., T.I. Tang, and J.T.E. Tang, “An Instrument for Measuring Customer Satisfaction Toward Web Sites That Market Digital Products and Services”, Journal *of Electronic Commerce Research*, volume 2, number 3, 2001

Xiong, L. and L. Liu, “PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities”, *IEEE Transactions on Knowledge and Data Engineering*, Special Issue on Peer to Peer Based Data Management, Vol. 16, No. 7, July 2004.

Yu, B. and M. Singh, “A Social Mechanism of Reputation Management in Electronic Communities”, *Proc. of 4th Int. Workshop on Cooperative Information Agents*, pp. 154 -165, 2000.

Yu, T. and K.J. Lin, “Service Selection Algorithms for Web Services with End-to-end QoS Constraints” *Proc. of IEEE Conference on E-Commerce Technology*, San Diego, CA, July 2004.