A Reputation and Trust Management Broker Framework for Web Applications

Kwei-Jay Lin[1], Haiyin Lu[1], Tao Yu[1], and Chia-en Tai[2]

[1]*Department of Electrical Engineering and Computer Science*
University of California, Irvine, CA, USA
[2]*Department of Computer Science and Information Engineering*
National Taiwan University, Taipei, Taiwan

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 service user is associated with a broker who may represent multiple users. A broker collects for its users the distributed reputation ratings about any web service. In return, a user provides its broker the transaction rating after every transaction with any service in order to build up the reputation database on all services. In addition, brokers form a trust network where they exchange and collect reputation data about services. By delegating trust management to brokers, individual users only need to ask their brokers about the reputation of a service before any transaction with a server. The only overhead for a user is the responsibility to share the reputation feedback with its broker. We present the distributed reputation and trust management framework and show the performance of the system by simulations.

1. Introduction

In the real world, *trust* is a relationship between two entities: it is one entity'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 some specific services.

3. *Consistency*: whether the subject entity is able to deliver the result with an acceptable certainty.

The first property is usually guaranteed 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 alone. The consistency of the 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 trust management framework is a discipline that has gained much attention in recent years [4] due to the growth of online transactions and e-business activities. Traditional distributed encryption and authentication mechanisms can only solve the identification and, to some extent, the qualification problems. However, it is not enough to simply trust that an entity will act consistently in all transactions. *Reputation* systems [1, 2, 3, 4] 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 provider have been implemented in many e-marketplaces. The success of Amazon and eBay proves that such reputation systems are helpful in fostering trust between parties.

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 [1, 3]. In a distributed trust system, reputation information is scattered among parties in the system. The distributed approach brings new challenges, including how to eliciting reputation information, how to evaluate the trustworthiness of a party with information gathered from potentially untrustworthy

parties, and how to propagate reputation information through 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 deployment of software trust brokers to manage the trust relationship for general web service users. In offline communities, people often rely on recommendations by word-of-mouth from personal experience of trusted acquaintances or reviews from 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, credit card customers, and cash customers. Our design is motivated to emulate these real-life trust building processes and reputation mechanisms by using software brokers as trusted experts.

We envision that many online users will have their online *trust brokers*, which may be implemented by a common, certified software package just like Microsoft® Passport or Liberty Alliance. The broker collects server reputation information for its users. Each user, after every transaction with a server (or a peer user), will produce a reputation rating on the server and send it to its broker so that brokers may use it to build up the reputation about that server. In this way, the reputation of any server can be collected from the report of all its previous clients. However, due to the distributed nature of brokers, it is impossible to collect all information from all brokers. We need efficient mechanisms to collect and manage the distributed reputation information in brokers.

Another issue that we study is how to build the referral network among brokers. Brokers interact and share reputation information. However, given possibly contradictory experience, brokers do not have the same level of trust on each other. In our design, brokers solicit server reputation information from brokers they trust more. The proposed mechanism allows brokers to build up different trust levels on one another, and to select only those they trust more 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 proposed trust system framework in Section 3 and a trust broker design in Section 4. Section 5 discusses some issues on *reputation authority*. The system simulation results are presented in Section 6.

2. Previous Work

Research activities in distributed trust management lie broadly in the following areas [4].

1. *Formalizing trust* [8]. 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. *Incentive mechanisms for eliciting honest feedbacks* [2]. 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.

3. *Mechanisms to guard against coordinated attack against the system* [3]. 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.

4. *Referral network systems where agents cooperate to propagate reputation information in the community* [1]. 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.

Our work is related to the referral network approach [1] by using a network of brokers to

propagate reputation information. However, we collect the reputation on both servers and brokers. Each server is assumed to have a constant probability that it may fail to deliver the requested service, due to hardware, network or server loads problems. Such inconsistency cannot be practically corrected due to cost and other circumstances. Our trust broker design is to identify the service failure probability, or the reputation, of each server.

3. System Architecture

This research proposes a distributed trust and reputation management framework that addresses the challenges on managing trust among e-services. As discussed earlier, reputation information in general is distributed within the community. The challenge for each 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.

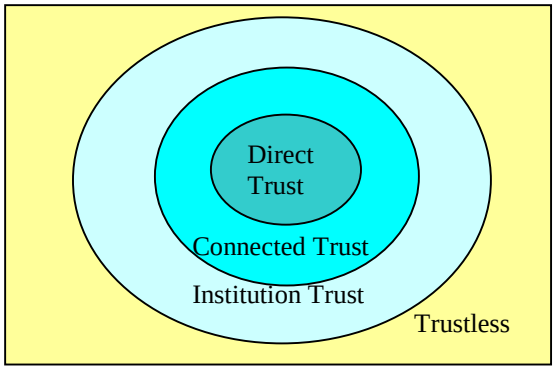

Figure 1 Trust Hierarchy

We divide the trust relationship into three levels as shown in Figure 1. The first level, the *direct trust* level, is for users that belong to a trust broker where there is a direct measurement of trust among all members. The second level, the *connected trust* level, is when two users belong to 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. There will usually be some cost associated with using the trust authority at this level. If the third level still cannot meet the trust policy, users will have to use some trustless protocol [5] to conduct business.

Figure 2 shows our distributed trust and reputation management system consisting of three types of components: users, brokers, and reputation authorities. In the model, all users may 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 servers that have had transactions with its users.

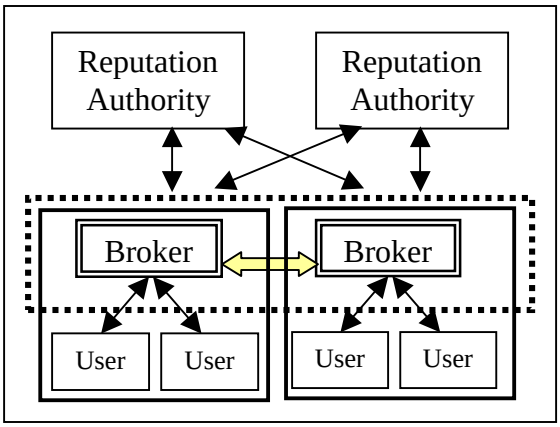

Figure 2 System Architecture

After each transaction, the client user *A* sends a rating on the server user *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 users. This way, a broker has a chance to accumulate enough reputation information (i.e. *direct trust*) about a server to support its users. However, if a broker finds its local reputation database inadequate for making a recommendation to its users, it will contact the other brokers (i.e. info from *connected trust*) or reputation authorities (info from *institution trust*) to gather more information.

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 for its user sufficient information about a peer. Reputation authority 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 server, the rating from each reputation authority may be different, just like credit rating companies may have erroneous information in real life. Moreover, reputation authorities may utilize some incentive program [2] to collect user reputations, and may charge a fee for its services.

4. Broker Design

A broker has two major components (Fig. 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 look at each component in details.

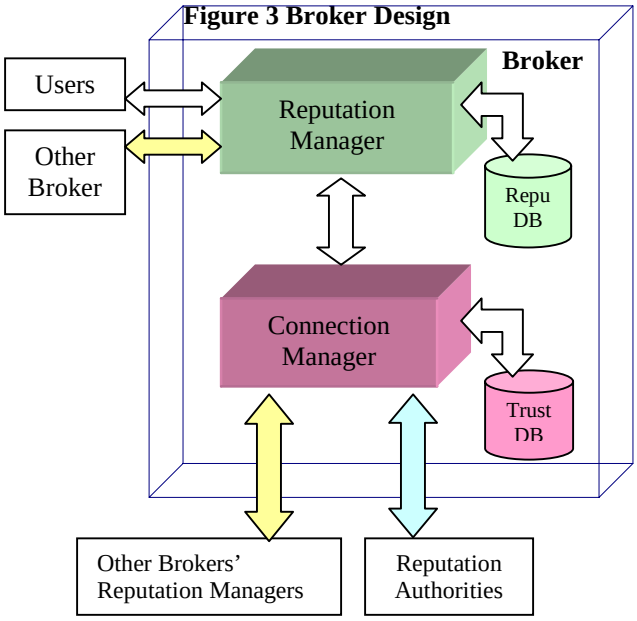

Figure 3 Broker Design

4.1. Reputation Manager

Reputation manager has three functionalities. First, it handles requests from client users and other brokers. Second, it forwards requests to the connection manager when necessary. Third, it is responsible for saving its users' feedback information.

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. Each reputation record has the following fields:

1. UserID: the ID of the server
2. Rating: a *reputation value* between 0 and 1
3. Size: the number of transactions used 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 size in its local database, and returns the server's rating only if the size is greater than repuTrans. Otherwise, the broker will forward the request message to the connection manager, which will use 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 after each transaction. Let N be the transaction size 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 using the formula:

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

The difference in feedback time between r and R_{old} is denoted by Δt, while $e^{-\beta\Delta t}$ specifies the discount factor of R_{old}. The non-discounted 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 cannot rely on its own resources to rate all servers when interacting with these servers. Collaboration among brokers is extremely important. Only through collaboration can the system identify untrustworthy servers promptly thus reducing the risk for everyone in the community. For this reason, brokers have a strong motivation to cooperate.

However, given that the objective of a broker is to provide service to its own users, some brokers will choose not to share its data with all other brokers for fear of competition for customers, etc. 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 of a broker is based on the number of accurate recommendations that have been provided. It is updated each time after a recommendation is received and compared with the actual transaction result. At the beginning, all fellow brokers are given a neutral trust value of X=0.5. After each transaction experience, the trust value is updated using the formula

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

if there is a match; and using the formula

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

if the recommendation does not match the actual experience. In the equations, F is a positive index with a value of 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 is good. On the other hand, a bad recommendation will reduces X to a new 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 difficult to gain additional trust but easy 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. Each fellow broker contacted will send back its reputation record about the peer.

Moreover, 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 500 transactions, the broker may want to define m = 2 and depth = 2. A total of 6 brokers will be contacted by the request (2 in the first level and 4 in the second level) that most likely will return recommendations based on the experience of 600 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.

The threshold value T is used to filter any low-trust broker from the 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 user, *C* may decide not to do the favor. This is only fair since *B* has not given too much credible information to *C* in the past. The behavior of a broker on another broker can be classified in three cases:

1. Always cooperate: if the trust value is higher than *H*.
2. Partially cooperate: if the trust value is between *H* and *L*.
3. Do not cooperate: if the trust value is less than *L*.

For fellow brokers with high trust values, a broker will return the complete reputation record on the requested user. For brokers with low trust values, nothing will be returned (that is, a zero reputation record is reported). For medium trust brokers, a recommendation with the size discounted by the trust value is returned. 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.

5. Reputation Authority

In the case when a broker and its trust network together still do not have enough evidence about a potential client, a broker may consult a reputation authority. A reputation authority collects reputation information from all brokers and produces a global rating on all users. 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. In addition, some reputation authority may impose service charges to those brokers requesting for information.

To prevent its reputation data from being discredited by coordinated attacks and conspired brokers that provide false ratings, a reputation authority may use robust mechanisms to detect and filter inaccurate reports, as well as to reward accurate reports. For example, an incentive mechanism [2] can be used to encourage all brokers to report feedback honestly. Another approach is to use feedback and community factors to produce a more correct trust report [3].

6. System Performance Study

To model reputation, every user in our system has a randomly assigned consistency factor (CF) that defines its capability to deliver a service. For example, if a user has a CF of 0.8, it may fail to deliver its service 20% of the time. Every broker keeps in its reputation database its local CF values of all servers. The local CF values are generated from its users' rating report. To evaluate the system's reputation knowledge about a server, we compute the total standard deviation between all brokers' local CF values on the server and the server'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.

6.1. 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 about every 1 msec. Therefore the same transaction pair (A, B) will be generated every 360,000 msec 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 36,000 msec. We decide that any of B's execution history outside of the window of last 100 transactions with A should no longer be meaningful to A. We can thus derive the ß value to be $1/(36*10^5)$ or around $2.7*10^{-7}$ 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 most trusted brokers in the trust network and the search for reputation data in the trust network can go as deep as 5 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.

6.2. Simulation Result

Figures 4-7 show the results from those simulations. 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 perfect knowledge on every user's CF initially, they are more likely to keep the system in a reasonably correct state. If the initial reputation data are wrong 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 value is 1-CF.

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.

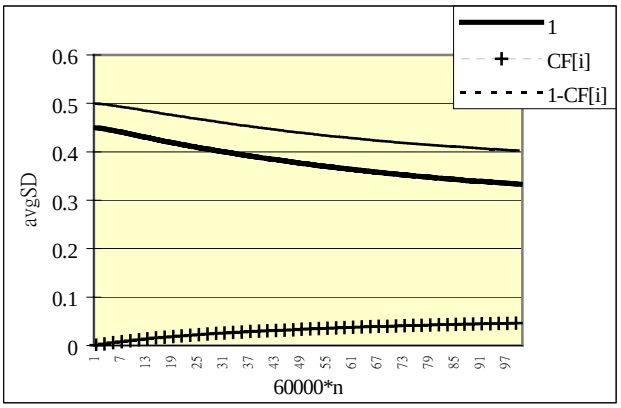

Figure 4. SC for repu threshold=0 and ß=10^{-6}

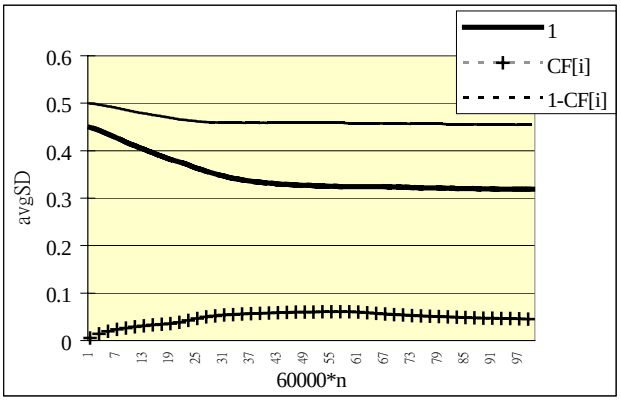

Figure 5. SC for threshold=0.3 and ß=$2.7*10^{-7}$

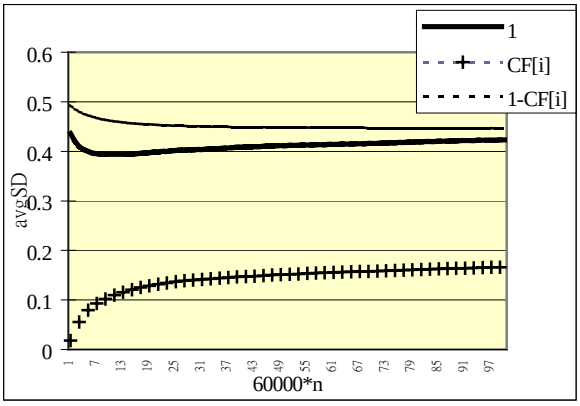

Figure 6. SC for threshold=0.5 and β=10^{-6}

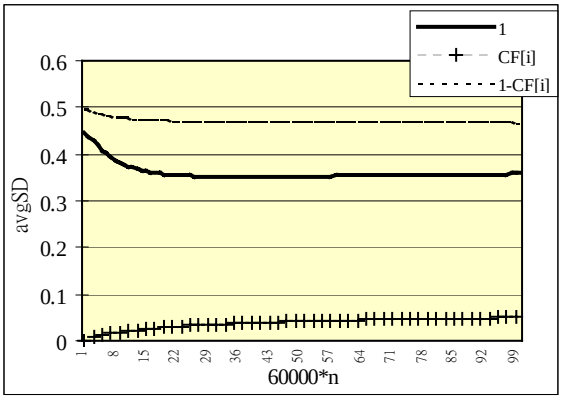

Figure 7. SC for threshold=0.7 and β=10^{-6}

7. Conclusions

With the expansion of broadband Internet and the growing adoption of Web services standards, we expect a continuing growth of e-services and e-commerce. Due to the on-line nature of e-services, it is important to check the trustworthiness of any service before it is invoked. However, most e-service users will be too naïve to design a trust measure themselves. On the other hand, they may connect to a trust community that can provide them with valuable experiences on a potential server. 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. Our framework combines three levels of trust and utilizes security broker, trust network, and reputation authority at each level respectively. 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. We believe this is an effective way to manage trust and reputation in the e-service environment.

8. References

[1] B. Yu and M. Singh. "A Social Mechanism of Reputation Management in Electronic Communities" Proceedings of 4th Int. Workshop on Cooperative Information Agents, pp. 154 - 165, 2000.

[2] R. Jurca and B. Faltings. "An Incentive Compatible Reputation Mechanism", Proc. IEEE Conf. on E-Commerce, pp. 285-292, Newport Beach, CA, June 2003.

[3] L. Xiong and L. Liu. "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities", to appear in IEEE Transactions on Knowledge and Data Engineering, Special Issue on Peer to Peer Based Data Management.

[4] C. Dellarocas 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

[5] M.J. Atallah, H.G. Elmongui, V. Deshpande, L.B. Schwartz, "Secure supply-chain protocols", Proc. IEEE Conf. on E-Commerce, pp. 293-302, Newport Beach, CA, June 2003.

[6] Chen, H., Yu, T., & K.J. Lin. "QCWS: An Implementation of QoS-Capable Multimedia Web Services." Proceedings of the Fifth International Symposium on Multimedia Software Engineering, pp. 38-45, Taichung, Taiwan, Dec 2003.

[7] T. Yu 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.

[8] C. Dellarocas, "The Design of Reliable Trust Management Systems fro Electronic Trading Communities", Working paper, MIT 2001, http://ccs.mit.edu/dell/trustmgt.pdf.