The Design of A Rule-based and Event-driven Trust Management Framework

Yan Wang
Department of Computing
Macquarie University
Sydney, NSW 2109, Australia
yanwang@ics.mq.edu.au

Kwei-Jay Lin
Dept of EE and CS
University of California, Irvine
Irvine, CA 92697, USA
klin@uci.edu

Duncan S. Wong
Department of Computer Science
City University of Hong Kong
Hong Kong
duncan@cityu.edu.hk

Vijay Varadharajan
Department of Computing
Macquarie University
Sydney, NSW 2109, Australia
vijay@ics.mq.edu.au

Abstract

In both E-Commerce (EC) and Service-Oriented Computing (SOC) environments, sellers or service providers interact with customers or service clients for services or transactions. From the point view of customers or service clients, the trust status of the seller or service provider is a critical issue to consider, particularly when the seller or service provider is unknown to them. Typically, the trust evaluation is based on the feedback on the service quality provided by customers and clients. Traditionally, the trust evaluation method is based on formulas only. This might be rigid to some complex applications, like SOC. In this paper, we propose a novel integrated trust management framework that is event-driven and rule-based. In this framework, the trust computation is based on formulas. But rules are defined to determine which formula and arguments to use according to the event occurred during the transaction or service. In addition, we also propose some trust evaluation metrics and a formula. A set of empirical studies has been conducted to study the properties of the proposed formula and how to control the trust change trend in both trust increment and decrement cases. The proposed framework is more generic and suitable for different domains and complex trust evaluation systems.

1 Introduction

In Service-Oriented Computing (SOC) field, a variety of e-services across various domains can be provided to clients in a loosely-coupled environment via various technologies (such as Web services [1]). The diversity and complex structure of services, the loosely coupled system architecture, and the subjectiveness of trust ratings make trust evaluation/management a very challenging and critical issue to the fast developing service-oriented applications.

With respect to trust evaluation, the issue has been actively pursued in Peer-to-Peer networks (P2P). In general, P2P networks are used for information-sharing systems, such as Napster [2]. In such systems, each peer can act as a client or a server at the same time. Being a serving party, the peer can provide some files to the community. Other peers can retrieve information with interest and download from trustworthy peers [3] who provide complete files. Thus, in such an environment, it is quite natural for a client peer to doubt the trust status of serving peers prior to any download actions in order to find the right peer to interact. In particular, in Peer-to-Peer e-commerce environments, the trust issue is more prominent as neither a buyer nor a seller is willing to be cheated.

In both P2P (or P2P E-Commerce) and SOC fields, there are some common features in the study of trust evaluation. First, the trust status of a seller or service provider is important to a buyer or a service client. A trust management mechanism is necessary for trust request broadcast, trust data collection, and trust computation. Second, each rating is provided by buyers or service clients posterior to transactions.

On the other hand, there are some differences in both fields. First, the difference exists in the trust management organization. In general, in P2P environments, it advocates that the networks work without any central management. Therefore, in P2P trust evaluation, a typical process is that each peer can rate the other peer after an interac-

tion/transaction. This is the local rating. When a certain peer (referred to as *requesting peer*) is willing to know the trust status of a target peer (say peer X), it can send requests to other peers. A peer with interaction history with X can respond to this request with its ratings. This peer is referred to as a *responding peer* or a *recommending peer* as its ratings become recommendations when they are sent to the requesting peer. In contrast, in SOC environments, a central management server can be set up for trust management (e.g., bound to the central UDDI server. Service clients can report their ratings to the central server as transaction feedback after transactions [6]. In addition, in P2P trust evaluations, in general, it is the requesting peer to compute the final trust value subject to its trust metrics and preferences. However, in SOC trust evaluation, it is more feasible for the central trust management server(s) to compute the trust values and respond them as services to requesting clients.

Therefore, in SOC trust evaluation, some methods can be borrowed from P2P trust evaluation models. But due to the diversity and complex structure of services, the loosely coupled system architecture, and the subjectiveness of trust ratings, more complex mechanisms should be studied. In these studies, the first concern is the SOC-oriented trust management architecture. Traditionally, in most trust evaluation models, a binary or numerical rating system is adopted and a formula is proposed for the trust computation. This is simple and may be effective enough. But in SOC environments, as there are a variety of service providers and service clients across different domains, each domain may have its own *policy* to come up with an evaluation. Additionally, in trust evaluation, according to the transaction history and the quality of recent transactions, new trust values can be derived. Particularly, in a negative case, when an undesirable service happened (e.g., a bad quality or fraud service), corresponding penalty should be determined in the trust calculation. The penalty varies from event to event, from party to party, from policy to policy, and from domain to domain. So is the positive case. In most existing studies [3, 5, 14, 6, 7, 11], the trust computation relies on predefined formulas only. This is simple but might not be adaptable enough to reflect appropriate trust variations in response to events and policies in domains.

In this paper, we present a novel trust evaluation framework and a trust evaluation model. The proposed architecture is rule-based and event-driven. The rules are categorized corresponding to different events. Namely, an event can trigger a corresponding rule or a set of rules. Rules are maintained in rule base operated by the rule owners. The proposed framework also adopts formulas for trust computation and we also advocate defining formulas as less as possible to enable a simple and efficient system. But it is determined by the rules on which formula to use, and what are the arguments when applying a formula.

This paper is organized as follows. In section 2, we review some existing studies. Section 3 presents the rule-based and event-driven trust management framework. In Section 4, we discuss some trust evaluation metrics and propose a formula-based method for trust evaluation. Some empirical study results are illustrated in section 5. In section 6, we conclude our work.

2 Related Work

Trust evaluation is considered as an important issue in Peer-to-Peer information sharing networks as a client peer needs to know prior to download actions which serving peer can provide complete files. In [3], Damiani *et al* proposed an approach for evaluating the reputation of peers through distributed polling algorithm before downloading any information. The approach adopts a binary rating system and it is based on the Gnutella [1] query broadcasting method using TTL limit. EigenTrust [5] also adopts a binary rating system and aims to collect the *local trust values* of all peers to calculate the *global trust value* of a given peer. Some other earlier studies also adopted the binary rating system, such as [14]. In [7], Marti *et al* proposed a voting reputation system that collects responses from other peers on a given peer. The final reputation value is calculated combining the values returned by responding peers and the requesting peer's experience with the given peer. This seems more reasonable than the model in [3].

As pointed in [15], binary ratings work pretty well for file sharing systems where a file is either the definitive correct version or is wrong, but cannot accurately model richer services such as web services and e-commerce, where a boolean may not adequately represent a peer's experience of the quality of service (QoS) with other peers, e.g., the quality of products the peer sends and the expected delivery time [15]. In most later studies on trust evaluation (e.g., [15, 11, 12]), a numeral rating system is adopted, where, for example, the rating is a value in the scope of $[0, 1]$. Such a system is more suitable for complex applications, such as e-commerce or service-oriented applications while binary ratings work pretty well for file sharing systems where a file is either the definitive correct version or is wrong.

In the literature, trust issue also caused much attention which is not bound to the P2P networks only. In [8], Sabater and Sierra proposed a model discussing the trust development between groups. In [4], Griffiths proposed a multi-dimensional trust model which allows the agents to model the trustworthiness of others according to various criteria. In [6], Lin *et al* proposed a method of reputation-based trust evaluation in service-oriented environments based on the proposed architecture consisting of distributed trust management brokers. In [10], Vu *et al* proposed a model to evaluate and rank the trust and reputation of QoS-based ser-

vices, which is much valuable for service search and selection.

3 Trust Management Architecture

In this section, we present an integrated, event-driven and rule-based architecture for trust management. This architecture inherits the features of formula-based trust evaluation method. But it is more suitable for service-oriented applications.

The proposed architecture is a centralized management architecture. Service clients or buyers can report to the trust management authority after transactions. Alternatively, a protocol can be designed where reporting feedback is an embedded step prior to the completion of the whole transaction. But it is out of the scope of this paper. The trust management server manages the data of service providers and service clients as well as the trust data of service providers. The proposed architecture is different from the one in [10], which adopts a decentralized (Peer-to-Peer) architecture. A central management architecture is more efficient with less communication cost. In the P2P-based architecture, it doesn't have the cost to set up separate central servers. But once a peer needs to know the trust status of a service provider, in general, it has to broadcast a request to other peers. Hereafter, the requesting peer will collect some volume of trust data for trust computation. This process has to repeat whenever a peer wants to know the trust status of a target peer. Thus it is costly in terms of network communication. The decentralized architecture is more scalable but less reliable as when the requesting peer broadcasts the request, it is not likely for all peers, who have the transaction history with the target service provider under investigation, to be online and respond. In contrast, in a centralized management architecture, the requesting client can simply communicate with the central trust management server, which stores trust history data, compute the trust value accordingly and respond to the clients. It is also feasible to adopt unified trust management and evaluation policies in a certain domain. In addition, such an architecture makes it easier in terms of security management, such as, authentication.

In our proposed framework, we assume there are a set of software agents with corresponding tasks. There are several databases storing the data of service providers, clients and trust data respectively. The framework serves for the trust management in service-oriented environments, which is an integrated system combining both rules and formulas for trust computation. These rules are used to define the policy of the trust management authority and categorize events that may occur in transactions. The policy can reflect the nature of the domain. Thus the proposed framework is more generic and can be applied to various domains of service-oriented applications.

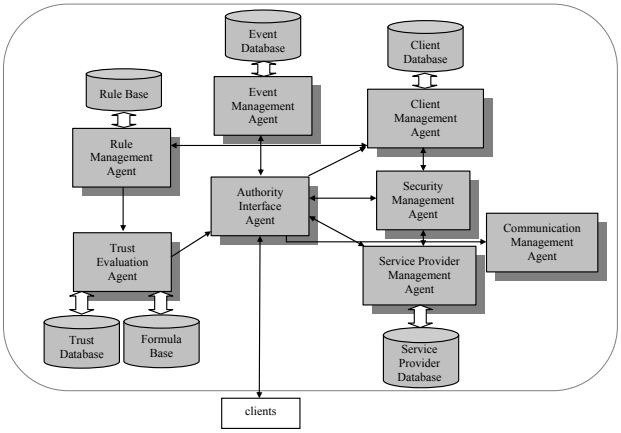

Figure 1. Trust Management Authority

3.1 Authority Interface Agent

This is the agent interacting with clients. It receives the request from a client and send it to the Event Management Agent in order to start the trust evaluation process. Finally this agent sends the result to clients.

3.2 Client Management Agent and Service Provider Management Agent

The two agents manage clients and service providers respectively and manage corresponding databases.

3.3 Security Management Agent

The Security Management Agent manages the security mechanisms of the trust management authority. It is also responsible of the authentication of service providers and service clients. Alternatively, it can cooperate with other servers to build up a Kerberos-like distributed authentication architecture [9]. We will not discuss this issue as it is out of the scope of this paper.

3.4 Event Management Agent

This agent is responsible of categorizing the events in transactions in a domain and managing the event database. These events are first categorized as *positive* and *negative* events. In the proposed framework, a service client or a buyer reports the feedback together with the event occurred. In particular, if the event is a *negative* one, the agent will communicate with the Rule Management Agent to determine the the penalty in trust evaluation. Namely, negative events should be further categorized. Each category includes events with the same nature and corresponds to the same degree of penalty in trust evaluation.

3.5 Rule Management Agent

This agent manages the rules for trust evaluation in the domain. There rules reflect the nature of the trust management in the domain. Rules are stored in the rule database. They can be inserted and updated. For example, when the feedback is a positive for a satisfactory service, the corresponding formula and arguments for trust evaluation should be determined. If the feedback is negative with a unsatisfactory event, the corresponding formula and arguments for penalty will be selected, which thereafter are sent to the Trust Evaluation Agent.

3.6 Trust Evaluation Agent

This agent is responsible of evaluating the target service provider as requested by a service client. Trust data are stored in the Trust Database, which is operated by this agent. The agent is also responsible of managing the formula base.

3.7 Trust Management Process

In general, a trust evaluation request is sent to the Authority Interface Agent, which transfers it to the Event Management Agent. The reported event, positive or negative, triggers the process of trust evaluation. The Event Management Agent communicates with the Rule Management Agent about the event. Corresponding rules will be determined for the formula and argument selection. These are sent to the Trust Evaluation Agent, which computes the trust result based on both history trust data and the reported feedback. The result is sent to the client via the Interface Agent.

4 Trust Evaluation

In this section, we will study a trust evaluation method. Though a complex application oriented trust management framework has been proposed, it doesn't indicate that there should be as more formulas as possible. A good formula can incorporate both positive and negative cases with arguments adjusted flexibly to adapt to domains and respond to events.

4.1 Trust Rating System

Here we assume a trust result calculated by the trust management authority is a numerical value in the scope of $[0, 1]$, where 1 means the most trustworthy and 0 implies the worst reputation. The rating given by a client or a buyer is also a numerical value in the scope of $[0, 1]$. For a client A, if it has an interaction with a service provider B, it can give a local trust rating $R^{(k)}_{A\to B} \in [0, 1]$ for the interaction occurred at time period t_k. The value can be calculated considering multiple aspects of the quality of the service provided by B [13]. These aspects exist in the concerns of the availability of services, the efficiency of service or product delivery, and the conformance of the service or product with the advertisement. As discussed in [15], this rating system is more appropriate for service-oriented environments, instead of file-sharing systems.

4.2 Trust Evaluation Metrics

A good trust evaluation system should be a fair system to reflect the trust status of different parties according to the quality of services or transactions. Namely, the trust result difference should reflect the service quality difference of sellers or service providers. This is radically important for buyers or service clients to make decisions when a set of potential sellers or service providers are available.

In general, reputation establishment is a long period with the accumulation of good ratings. The whole process can be divided into three periods.

Period 1 - Initial Reputation Establishment: Initially, a new service provider or seller has no reputation. Thus it can attract clients by offering good, even extremely good services. But meanwhile, the reputation improvement is not quick as the reputation improvement relies on long-term good services. That is, in this period, the trust may improve quickly, but the trust value is not in the relatively high level.

Period 2 - Reputation Improvement: After the first period improvement, the trust level has reached to a good level. During this period, the trust improves slowly before reaching the highest trust level.

Period 3 - Reputation Stabilization: After the accumulation of good services for some periods, the trust value maintains in the high level. Thus the improvement is not significant.

The three periods are depicted in Figure 2. The plotted function is as follows ($x \in [0, +\infty),\ y \in [0, 1]$), which is a transformation of Hyperbolic Tangent - $tanh(x)$.

$$y = \frac{e^{3x} - e^{-3x}}{e^{3x} + e^{-3x}} \tag{1}$$

In the above discussion, we assume each rating obtained is very good (e.g., R=1.0) leading to the curve plotted in Figure 2. In order to show the long period of trust improvement, we set the maximal time to 100 units. But in practice, this can be set by arguments. For instance, formula (1) can be generalized as formula (2), where $\alpha = 3$ and $\beta = 1$. Such a function is referred to as a *basic curve function*,

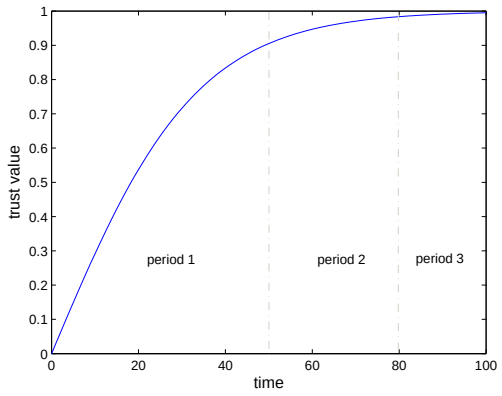

Figure 2. Trust Improvement Curve ($formula(2),\ \ \alpha = 3\ \ \beta = 1$**)**

which will be used for determining the trust increment in later section. In Figure 2, the x-axis stands for time. Each unit is a time period. In this work, we consider 'abstract' time unit. It may be an hour, a day, a week or a month, which is application dependable.

$$bf(x) = \frac{e^{\alpha x} - e^{-\alpha x}}{(e^{\alpha x} + e^{-\alpha x})\beta} \qquad (\alpha \geq 1 \quad \beta \geq 1) \qquad (2)$$

In addition, there may be some negative cases that should lead to penalty in trust evaluation. Prior to presenting the detailed trust evaluation formula, we first propose a set of trust evaluation principles as follows.

Principle 1: The initial trust value of a party should be set to a low value;

This principle enforces the party being evaluated to provide honest and good services to customers in order to obtain good trust ratings and upgrade its trust level.

Principle 2: A good reputation is the result of accumulative good services and behaviors, and consequent good ratings in a relatively long period;

Namely, the trust deviation is a long process, which indicates the difficulty of trust upgrade. The process may be different from domain to domain. In a formula-based trust evaluation system, arguments can be used to control the curve.

Principle 3: The trust evaluation system should be punitive to dishonest and unsatisfactory services. In a good trust evaluation system, it is relatively easier to downgrade the trust value than to upgrade trust values.

In contrast to the trust level upgrade, dishonest services and unsatisfactory services will lead to penalty and trust level downgrade. In addition, given two services with the same nature (e.g., the same transaction amount) and different feedback (i.e. one satisfactory and one unsatisfactory), the trust level increment should be not greater than the trust level decrement. In particular, a dishonest transaction will lead to severe penalty (e.g., reset the trust value to 0).

A trust evaluation system is radically an incentive system to encourage honest and satisfactory services or transactions, and dis-encourage and penalize dishonest and unsatisfied services or transactions. For instance, after having a honest and satisfactory transaction, the seller can get a trust value increment of 0.01. On the contrary, if the transaction is dissatisfactory, the seller will get a decrement of 0.2 or more.

Principle 4: The trust evaluation result should reflect the service history. Namely, the good trust level is the result of good services and good feedback for a certain period.

Principle 5: Trust computation can be based on formulas. But rules can be defined for the selection of formulas and arguments.

In different domains, there may be different policies for trust evaluation. Thus, though some formulas can be applied in different domains, arguments may be different. In addition, in the case of penalty, the decrement is determined by arguments, which are selected by rules according to the nature of negative events.

4.3 Trust Evaluation Method

Definition 1: Let $T_x^{(k)}$ denote the trust value of target service provider X at time period t_k, and $R_x^{(k+1)}$ is the rating of target party X at time period t_{k+1}. $\Delta = R_x^{(k+1)} - T_x^{(k)}$. *The trust value of* X *at time period* t_{k+1} is

$$T_x^{(k+1)} = \begin{cases} min(1,\ \ T_x^{(k)} + \theta \cdot \Delta) & \text{if } \Delta \geq 0 \\ max(0,\ \ T_x^{(k)} + \theta \cdot \Delta) & \text{if } \Delta < 0 \end{cases} \qquad (3)$$

where

$0 \leq \theta < 1$ is the *impact factor* determining the impact of recent change (i.e. Δ) on the trust calculation;

Formula (3) results in a value in the scope of $[0, 1]$. To obtain the trust value in period t_{k+1}, the trust value in period t_k is used. Thus, the new trust value results from the old trust value and the latest rating. This follows *Principle 4* proposed in section 4.2.

To calculate θ, we propose an impact factor function based on the basic curve function $bf(x)$.

Definition 2: If the current trust value is $T_x^{(k)}$, the *impact factor function* θ can be

$$\theta = \lambda \cdot bf'(T_x^{(k)}) \tag{4}$$

where $\lambda > 0$ is the scale control factor; $bf'(x) \geq 0$ is the derivative of the basic curve function $bf(x)$.

Definition 3: We define the *scale control factor* $\lambda^{(k+1)}$ as follows.

$$\lambda^{(k+1)} = \begin{cases} \lambda_+ \leq 1 & \text{if } \Delta \geq 0 \\ \lambda_- \geq 1 & \text{if } \Delta < 0 \end{cases} \tag{5}$$

According to formulas (3) and (4), in equation (5), when $\Delta \geq 0$, there will be an increment in the trust calculation. Namely, $T_x^{(k+1)} \geq T_x^{(k)}$. In this case, normally $\lambda^{(k+1)} = \lambda_+ = 1$. Thus the increment will be $bf'(T_x^{(k)}) \cdot \Delta$. When $\Delta < 0$, there will be a decrement. By default, $\lambda^{(k+1)} = \lambda_- = 2$. Assuming the same $|\Delta|$, the decrement is $2 \cdot bf'(T_x^{(k)}) \cdot \Delta$ at least. This indicates that it is harder to improve the trust value than worsening it. Therefore, relatively it takes longer time to reach a high level trust value (e.g., 0.95) than to drop from a high level to a low level. This follows *Principle 3* in section 4.2. The value of λ is determined by the pre-defined rules (refer to Section 3). In addition, when a severely negative event happened, $\lambda > 2$ will be applied for decrement cases. This also indicates a harder trust improvement constraint.

For example, if we adopt formula (6) as $bf(x)$ (plotted in Figure 3), which is a specific function of formula (2) , where $\alpha = 2$ and $\beta = 20$.

$$bf(x) = \frac{e^{2x} - e^{-2x}}{(e^{2x} + e^{-2x}) * 10} \tag{6}$$

Its derivative function is:

$$bf'(x) = \theta(x) = \frac{\alpha}{\beta} - \frac{\alpha(e^{\alpha x} - e^{-\alpha x})^2}{\beta(e^{\alpha x} + e^{-\alpha x})^2} \; (\alpha \geq 1 \quad \beta \geq 5) \tag{7}$$

An example of formula (7) is plotted in Figure 4, where $\alpha = 2$ and $\beta = 20$.

According to the above proposed formulas, some properties are listed as follows.

Property 1: θ is in reverse proportion to the current trust value $T^{(k)}$.

This indicates an incentive mechanism. When two parties A and B, if $T_A^{(k)} < T_B^{(k)}$, given the same $\Delta > 0$, $\theta_A \cdot \Delta > \theta_B \cdot \Delta$. This is incentive to new parties with low trust values.

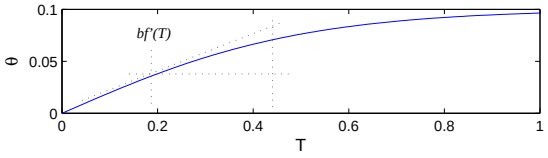

Figure 3. Basic Curve Function ($Formula\ (6)$**)**

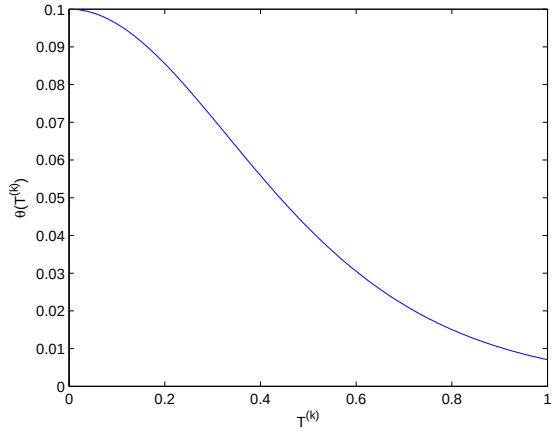

Figure 4. Impact Factor Function ($Formula\ (7)$**,** $\alpha = 2\ \beta = 20$**)**

Property 2: The maximal of the impact factor is 0.2.

$\theta_{max} = 0.1$ when $T = 0$. Namely, in this case, it is quite incentive for the service provider (probably new) to improve its trust level. But meanwhile, the trust improvement is a long process (*Principle 2*). Thus, even if $T^{(0)} = 0$ and $R^{(1)} = 1$, the increment is 0.1*1=0.1. Thus the new trust value is $T^{(1)}$=0.1, which is still in the low level.

5 Performance Study

5.1 Study 1

In this section, we study how to control the scale of trust changes. Basically, it can be controlled by setting different values of α and β. Here we first $\beta = 20$, $T^{(0)} = 0.1$, $R^{(k+1)}$=1, $\lambda_+ = 1$, and α is set to 1, 2 and 3 respectively. The result is plotted in Fig. 5. We can observe that the larger the value of α, the stricter the trust evaluation. That is, it takes longer time to reach a high trust level.

In another case, we set $\alpha = 2$ and set β to 5, 10, 20 and 30 respectively. The result is plotted in Fig. 6. We can

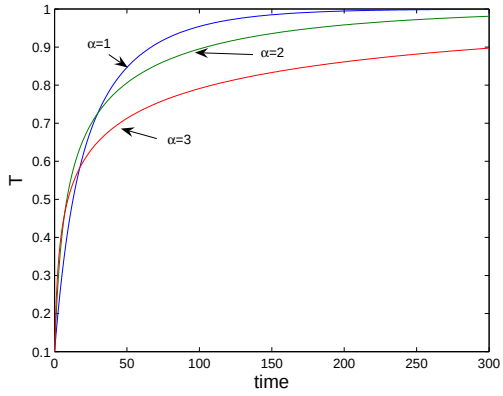

Figure 5. Study 1 with different α ($\beta = 20$)

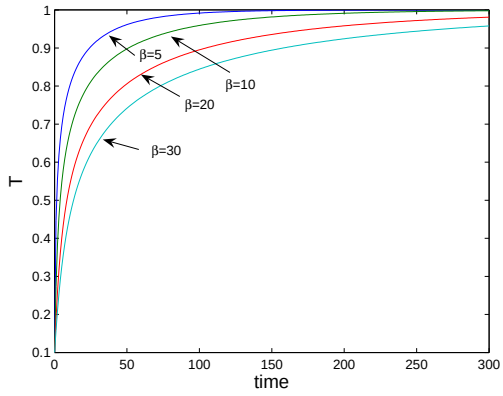

Figure 6. Study 1 with different β ($\alpha = 2$)

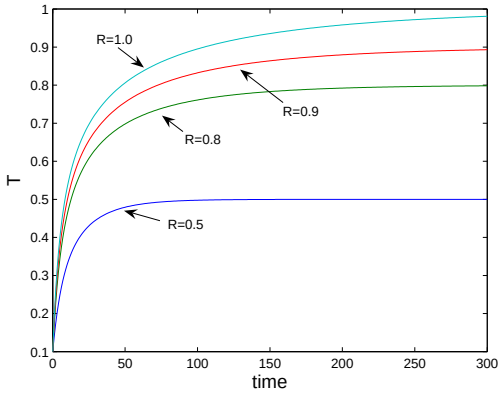

Figure 7. Study 2 ($\alpha = 2$ $\beta = 20$)

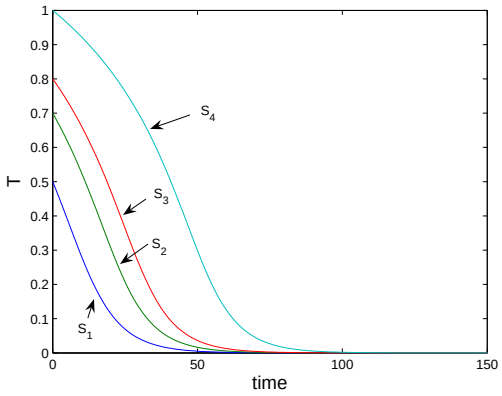

Figure 8. Study 3 ($\alpha = 2$ $\beta = 20$)

observe that the larger the value of β, the stricter the trust evaluation.

Therefore, based on our proposed framework, α and β are part of the system parameters. Their values can be determined by predefined rules according to the nature of the domain and applications.

5.2 Study 2

In this section, we study how a new service provider establishes its trust level. The initial trust value is set to 0.1. We assume there are 4 service providers. They are S_1, S_2, S_3 and S_4. In each period, they get static ratings of 0.5, 0.8, 0.9 and 1.0 respectively. Formula (7) is adopted as the impact factor function.

The results are plotted in Figure 7. We can observe that the trust value of each party increases incrementally from 0.1 When $R_A^{(k+1)} > R_B^{(k+1)}$, $T_A^{(k+1)} > T_B^{(k+1)}$. This is rational as it is incentive to good services and good ratings. In addition, when the trust value is static in each period, e.g., $R^{(k+1)} = a \in [0, 1]$, the final trust value is approaching a. Namely,

$$lim_{t_k \to \infty} T^{(k)} = a$$

5.3 Study 3

In this section, we study how a party's trust value drops. The basic curve function is based on formula (2), where $\alpha = 2$ and $\beta = 20$. We assume there are four parties S_1, S_2, S_3 and S_4. Their initial trust values $T^{(0)}$ are 0.5, 0.8, 0.9 and 1.0 respectively. We also assume static $R^{(k+1)} = 0$ and $\lambda_- = 1$. The results are plotted in Figure 8. We can observe that the trust value drops from $T^{(0)}$ approaching 0 after some period. In practice, if $R^{(k+1)} = 0$, it might be accompanied with a severe event occurred. This may lead to $\lambda_- \geq 2$. Namely, the trust value may drops more quickly.

This feature will be studied in the next study.

5.4 Study 4

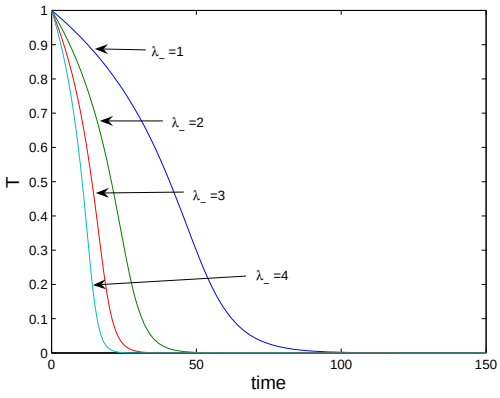

Figure 9. Study 4 ($\alpha = 2$ $\beta = 20$)

In this section, we study how the trust value drops with different λ values. The initial trust value is set to 1.0. In each period, $R^{(k+1)} = 0$. λ_- is set to 1, 2, 3 and 4 respectively. Results are plotted in Figure 9. Unsurprisingly, a larger λ_- value results in more decrement. In real applications, as we can adopt the rule-based framework. Rules should be predefined to determine the negative event category, where each category corresponds to a λ value.

6 Conclusions

In this paper, we present a novel framework for complex service oriented trust management, which is rule-based and event-driven. A generic method and a formula are also proposed for trust evaluation. They can be applied to different applications by using different *system arguments* - α, β and λ. In addition, both analytical and empirical studies have been conducted for illustrating the properties of the proposed method.

For future work, detailed event category and rules should be studied. Moreover, some more formulas should be proposed serving for different requirements of different applications.

References

[1] *GNutella*. http://www.gnutella.com/.

[2] *Napster*. http://www.napster.com/.

[3] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation based approach for choosing reliable resources in peertopeer networks. In *Proceedings of ACM CCS'02*, pages 207–216, Washington DC, USA, November 2002.

[4] N. Griffiths. Task delegation using experience-based multidimensional trust. In *Proceedings of the 4th International Joint Conference on Autonomous Agents in Multi-Agent Systems (AAMAS-05)*, pages 489–496, 2005.

[5] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th International WWW Conference*, Budapest, Hungary, May 2003.

[6] K.-J. Lin, H. Lu, T. Yu, and C. en Tai. A reputation and trust management broker framework for web applications. In *Proceedings of The 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05)*, pages 262–269, March 2005.

[7] S. Marti and H. Garcia-Molina. Limited reputation sharing in P2P systems. In *Proceedings of ACM EC'04*, pages 91–101, New York, USA, May 2004.

[8] J. Sabater and C. Sierra. REGRET: A reputation model for gregarious societies. In *Proceedings of the First International Joint Conference on Autonomous Agents in Multi-Agent Systems (AAMAS-02)*, pages 475–482, 2002.

[9] J. G. Steiner, C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of the USENIX Winter 1988 Technical Conference*, 1988.

[10] L.-H. Vu, M. Hauswirth, and K. Aberer. Qos-based service selection and ranking with trust and reputation management. In *Proceedings of 13th International Conference on Cooperative Information Systems (CoopIS 2005)*, Oct 31-Nov 4 2005.

[11] Y. Wang and V. Varadharajan. Interaction trust evaluation in decentralized environments. In K. Bauknecht, M. Bichler, and B. Pröll, editors, *Proceedings of 5th International Conference on Electronic Commerce and Web Technologies (EC-Web04)*, volume LNCS 3182, Springer-Verlag, pages 144–153, Zaragoza, Spain, August-September 2004.

[12] Y. Wang and V. Varadharajan. $Trust^2$: Developing trust in peer-to-peer environments. In *Proceedings of 2005 IEEE International Conference on Services Computing (SCC 2005)*, pages 24–31, Orlando, Florida, USA, July 2005.

[13] Y. Wang and V. Varadharajan. Two-phase peer evaluation in P2P e-commerce environments. In *Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Serivce (EEE-05)*, pages 654–657, Hong Kong, China, March 29-April 1, 2005.

[14] L. Xiong and L. Liu. PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Trans. on Knowlege and Data Engineering*, 16(7):843–857, 2004.

[15] B. Yu, M. P. Singh, and K. Sycara. Developing trust in large-scale peer-to-peer systems. In *Proceedings of 2004 IEEE First Symposium on Multi-Agent Security and Survivability*, pages 1–10, August 2004.