Purdue University
Purdue e-Pubs

Computer Science Technical Reports | Department of Computer Science

2003

Dynamic Trust Production Based on Based on Interaction Sequence

Yuhui Zhong

Yi Lu

Bharat Bhargava
Purdue University, bb@cs.purdue.edu

Report Number:
03-006

Zhong, Yuhui; Lu, Yi; and Bhargava, Bharat, "Dynamic Trust Production Based on Based on Interaction Sequence" (2003). *Computer Science Technical Reports.* Paper 1555.
http://docs.lib.purdue.edu/cstech/1555

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.

DYNAMIC TRUST PRODUCTION BASED ON INTERACTION SEQUENCE

Yuhui Zhong
Yi Lu
Bharat Bhargava

CSD TR #03-006
March 2003

Dynamic Trust Production Based on Interaction Sequence *

Yuhui Zhong Yi Lu Bharat Bhargava

Center for Education and Research in Information Assurance and Security
and
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907, U.S.A
{zhong, yilu, bb}@cs.purdue.edu

Abstract

New emerging applications such as e-commerce and peer-to-peer systems require autonomous agents to dynamically determine trust in open environments. Trust is a truster's opinion towards a trustee established through direct interactions or recommendations. Many research efforts are on trust propagation via recommendations. We investigate the trust production problem that determines trust from a sequence of interactions (i.e. rated interactions). This paper presents four trust production rules: "equal-weight-interaction", "dilute-with-time", "slow-trust-quick-distrust", and "slow-trust-quick-distrust-with-supervision". These rules can be implemented by using an update or an analysis algorithm. An update algorithm only maintains a trust state, which is modified when a new interaction becomes available. An analysis algorithm maintains the whole interaction sequence and evaluates trust based on it. Four types of user behaviours, "stable user", "repenting user", "cheater" and "smarter cheater", are modelled. The proposed rules are studied via simulation by using these models. The results demonstrate that these rules make similar judgments on a stable user who behaves consistently well or badly. All rules except "equal-weight-interaction" are able to react to sharp changes in behaviours (a "repenting user" or a "cheater") within 10 interactions. The "slow-trust-quick-distrust-with-supervision" rule can even catch a "smart cheater", who repeatedly misbehaves in short periods and then tries to cover the bad effect by good behaviours.

1. Introduction

New emerging applications such as e-commerce and peer-to-peer systems require autonomous agents to dynamically determine trust in open environments. Trust is a truster's opinion towards a trustee established through direct interactions or recommendations. Evaluating trust based on interactions between a trustee and truster (i.e. first-hand information) is called trust production. Determining trust according to recommendations from third parties (i.e. second-hand information) is called trust propagation among trusters [9]. A trust model consists of both trust production and propagation.

Although automatic trust production is critical for a trust model, it draws fewer attentions than trust propagation does. Most research efforts view interactions as an element set. A statistical characteristic such as mean or mode is used as trust values [1][4]. Trust production in real life is a complex and subjective process. It has the following properties.

- *Time-dependent:* It is more appropriated to thought interactions as a sequence than a set. A trustee whose interactions are {bad, good, good, good} may be considered more trustworthy than the one with {good, good, good, bad} although the numbers of "good" and "bad" are the same. The reason is in the former case "bad" occur long time ago while in the latter case "bad" just occurs. This example shows that both the interactions and their orders count in trust production. A trust production rule is time-dependent if it views interactions as a time sequence.
- *Interaction-dependent:* A bad interaction often has much more effect on trust values than a good one does. More efforts are needed to gain the same amount of trust than to loose it. This is identified as easy-destruction-hard-construction property. In order to satisfy this property, we differentiate interactions that increase trust from those that decrease trust and treat them differently. A trust production rule works in this manner is called interaction-dependent.
- *Trustee-dependent:* A truster is more cautious to a trustee who has done harm to her. It will be more difficult for the trustee to achieve the same amount of trust than an innocent one since her good behaviours are overlooked while bad behaviours are taken seriously. A trust production rule considering a trustee's history to decide the effect of an interaction is called trustee-dependent.

This research is supported by NSF grant IIS-0209059, CISCO URP and CERIAS security center.

- *Subjectivity:* Social science literature shows that trust production is a subjective process. Two trusters may form different opinion upon viewing the same interaction sequence. Subjectivity is accommodated by providing multiply choice of trust production rule or tuneable parameters for a rule.

We present four production rules and the corresponding algorithms that have some or all of the above properties. Four user models including stable user, repenting user, cheater, and smart cheater are identified. The performances of the production rules in terms of δ-transition phase are investigated analytically and experimentally under the models of repenting user and cheater. Their capabilities of catching smarter cheaters are studied via simulations.

The rest of this paper is organized as the follows. Section 2 introduces the related work. Trust production rules are realized by using update and analysis algorithms, which are presented in section 4. Four user behaviour models are identified in section 5. We discuss the theoretical analysis in section 6. Section 7 presents the simulation results. The conclusion and future work is in section 8.

2. Related work

The problem of establishing and maintaining trust without predefined trusted third parties has drawn much interest. One of the first works trying to formalize trust in computer science is that of Marsh [3]. The model includes the concepts widely adapted by other researchers, such as situation or context, blind trust. Although it has strong sociological foundation, the limitation of this model is that it only considers a truster's own interactions. Rahman and Hailes proposed a trust model that can be implemented in P2P systems [1]. Trust is established via truster's own interactions and recommendations from third parties. Semantic distance that characterizes the difference between previous recommendations from the provider and interactions of the truster is used to predict the "true meaning" of a recommendation. How to collect the recommendations and how well the model will scale is not clear. Yu and Singh proposed a model based on a social network among entities [4, 5]. The main contribution of this research effort is the approach that systematically propagates recommendations via social network. Aberer and Despotovic simplify the above model and use it to manage trust in a P2P system. They emphasize on the data management and retrieval problems for trust assessment.

None of above research efforts models the subjectivity of trust that is reflected as different trust attitudes and heuristics of trusters. The methods for determining trust based on own interactions do not capture certain unique properties of trust formation (e.g. the effects of interactions dilute with time and trust construction is difficult after mistrust event occurs).

3. Trust production rules

In this section, we present the data structure used for trust production, three initialisation strategies, and four production rules.

3.1 Data structure and initialisation

We assume that a truster gives a rating ranging from 0 to 1 to each interaction. The higher the rate is, the more trustworthy the interaction is. From the perspective of trust production, two interactions are identical if their rates are the same.

A trust production rule determines a trust state given an interaction sequence. A trust state is a 2-tuple <tValue, iNum>, where tValue is a real number between 0 and 1. The higher the tValue is, the more trustworthy the trustee is. *iNum* is the number of interactions between the truster and the trustee. A trust state is associated with a specific trustee and context. It must be initialised. There are three initialisation strategies.

- *Null initialisation:* Both tValue and iNum are set to 0. tValue is meaningless when iNum is 0. The trust state is analogous to the null value in database.
- *Static initialisation:* iNum is set to 1. tValue is set to a predefined value. The optimistic, realistic or permissive attitude of a truster can be implemented by using different initial values.
- *Dynamic initialisation:* iNum is set to 1. tValue is dynamically evaluated each time initialisation is required. There are three cases. (1) If the truster has interactions with the trustee in other contexts, the mean or mode of tValues in these contexts is used as the initial value. (2) If the truster has interactions with other trustees in the same context, the mean or mode of tValues towards these trustees in this context is used. (3) The mean or mode of tValues of all trustees in all contexts is used.

Compared with other strategies, dynamic initialisation is more adaptive to the environment at the cost of higher overhead.

3.2 Production rules

Based upon the dependencies on time, interaction, and trustee, four production rules are proposed. They are equal-weight-interaction (EWI), dilute-with-time (DWT), slow-trust-quick-distrust (STQD), and slow-trust-quick-distrust-with-supervision (STQD-S).

- *EWI*: All interactions contribute equally to the determination of the trust value. Interactions are considered as a set. A statistical characteristic such as mean or mode is used as the trust value. Trust state is the only information stored in a trustee's profile. There is no profile for a truster. This rule is independent of time, interaction and trustee. It is not tuneable since no parameter is used. EWI is the most simple but widely used rule in literatures [1],[4].

• *DWT*: Suppose the interaction sequence is $<Ob_1, Ob_2, \ldots Ob_n>$. Ob_n is the most recent one, which contributes to $(1 - \alpha)$ portion of the trust value. The rest α portion comes from the previous trust value that is determined in the same way. α is called *dilution factor*. It is specified by a truster and stored in her profile. This rule is time-dependent in the sense that the effect of an interaction on trust value reduces with time. The assumption of this rule is that the recent behaviour of a trustee reflects its trust values better than previous ones do.

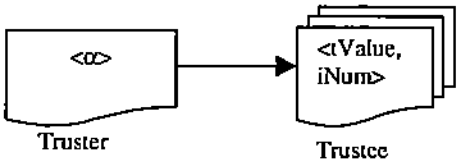

Figure 0 Data structure for DWT

Neither of the rules discussed above is interaction-dependent. The contribution of an interaction to the trust value is not related to how good or bad the interaction is. These rule do not have the easy-destruction-hard-construction property. Hence, two interaction-dependent rules, STQD and STQD-S, are proposed. We define trust destruction events for them. A trust destruction event is an interaction that decreases the current trust value.

• *STQD*: Like DWT this rule determines trust value by combining the rate of most recent interaction and previous trust value. Unlike DWT that uses one dilution factor for all interactions, STQD chooses one from a pair of factors (i.e. *construction factor* W_c and *destruction factor* W_d). If the most recent interaction is a trust destruction event, W_d is chosen. $(1 - W_d)$ is the current dilution factor. Otherwise, W_c instead of W_d is used. W_c and W_d are specified by the truster and stored in her profile. They satisfy the constraint $W_c < W_d$, which realizes easy-destruction-hard-construction. In STQD, W_c and W_d are fixed and used for all trustees. It does not differentiate interactions of a notorious user and one with good reputation.

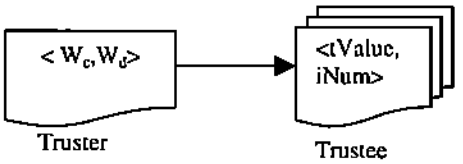

Figure 2 Data structure for STQD

• *STQD-S*: Trust value is evaluated as in STQD until a *foul event* occurs. A foul event is an interaction whose rating is lower than a threshold specified by a truster. When a foul event occurs, the trustee is put under supervision in the sense that her W_c is decreased and W_d is increased. If the trustee does not conduct any foul event during the supervision period, her W_c and W_d are restored to the original values. Otherwise, her W_c and W_d will be further decreased and increased respectively. The supervision period associated with a trustee will increase each time when she is put under supervision, so that she will be punished longer next time conducting a foul event. In this way, a trustee with worse history is treated harsher. The parameters stored in a truster's profile include γ, W_c, W_d, ρ_1, ρ_2, and ρ_3. γ is the threshold for foul events. W_c and W_d are initial construction and destruction factors. τ is the original supervision period. ρ_1 is the penalty ratio used to decrease construction factor. ρ_2 and ρ_3 are the penalty ratios used to increase destruction factor and supervision period respectively. The current construction factor W_c', destruction factor W_d' and supervision period τ' are stored with trust state in a trustee's profile.

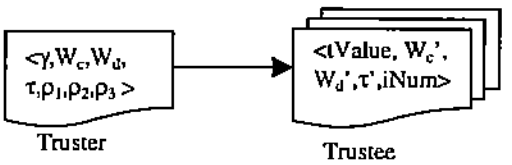

Figure 3 Data structure for STQD-S

Table 1 summarizes the rules discussed in this section. EWI is the most primitive rule while STQD-S is the most flexible one. The last three rules can be tuning by adjusting parameter. The rule used and the values of rule parameters reflect the subjectivity of the truster. The storage cost of STQD-S is higher than other rules. Here, N denotes the number of trustees.

	EWI.	DWT	STQD	STQD-S
Time-dep.		√	√	√
Interaction-dep.			√	√
Trustee-dep.				√
Tuneable		√	√	√
Profile of truster		√	√	√
Storage cost	2N	2N+1	2N+2	5N+7

Table 1 Summarization of trust production rules

In this paper, trust value is represented by a real number. Complicate representations, such as opinion and fuzzy expression, are proposed [1],[4],[6],[12]. They can be used directly by EWI and DWT because these rules do not pose any constraint on trust representation. STQD and STQD-S require that the domain of a trust representation be totally ordered.

4. Realization of rules

The production rules discussed in section 3 can be realized by using trust update or trust analysis algorithms.

4.1 Trust update vs. Trust analysis realization

A trust update algorithm only maintains the current trust state. Trust state TS_{n+1} is constructed by using TS_n and interaction Ob_{n+1}. Trust update algorithm has the following forms:

$$TS_1 = f_1(Ob_1)$$
$$TS_{i+1} = f_i(TS_i, Ob_{i+1}) \qquad 2 \le i$$

where $\{f_1, \ldots f_n \ldots\}$ is a family of functions. It is impossible for an update algorithm to evaluate trust based on a subsequence like "recent n interactions" since interactions are not saved.

A trust analysis algorithm saves interaction sequence and computes trust states based on them. An analysis algorithm with an infinite memory model (i.e. all interactions are stored) is more expressive than an update algorithm. We can always derive the corresponding analysis algorithm for a given trust update algorithm [8]. In practice, analysis algorithms with a sliding window in stead of infinite memory model are used. The window size determines the maximum interactions that can be kept. When the window is full, a new interaction will replace the oldest one. In this case, trust state is computed on the sequence starting from the second oldest interaction. Trust analysis algorithm with window size n has the following forms:

$$TS_{1,i} = f_i(Ob_1, \ldots Ob_i) \qquad \text{for } 1 \le i \le n-1$$
$$TS_{k,n} = f_n(Ob_k, \ldots Ob_{n+k-1}) \qquad 1 \le k$$

where $\{f_1, \ldots f_n\}$ is a family of functions. f_i has i parameters, here $1 \le i \le n$. $TS_{k,n}$ represents the trust state evaluated based on the interaction sequence of length n starting from Ob_k.

4.2 Realization using trust update algorithms

The rules proposed in section 3 are realized by using trust update algorithms. Analysis algorithms can be devised similarly.

EWI Algorithm

$TS_{(k,c)}$ denotes the trust state of trustee k in context c. It has two fields: iNum and tValue.

```
1:  Initialise TS(k,c).tValue based on the strategy
    specified by the truster
2:  if "null initialisation" strategy is used then
3:      TS(k,c).tValue = 0
4:      TS(k,c).iNum = 0
5:  else
6:      TS(k,c).iNum = 1
7:  end if
8:  While there exists new interaction Ob
9:      TS(k,c).iNum = TS(k,c).iNum + 1
10:     W = 1/ TS(k,c).iNum
11:     TS(k,c).tValue = TS(k,c).tValue×(1−W)+Ob×W
12: end while
```

Initialisation is done by the first to the seventh lines. Line 10 ensures each interaction has equal weight. Line 11 evaluates trust value based on the new interaction and the previous tValue.

DWT Algorithm

$TS_{(k,c)}$ denotes the trust state of trustee k in context c. It has two fields: iNum and value.

```
Input:  Dilution factor α∈(0,1)
1:      Initialise TS(k,c) as EWI algorithm does.
2:      while there are new interaction Ob
3:          if TS(k,c).iNum = 0 then
4:              TS(k,c).tValue = Ob
5:          Else
6:              TS(k,c).tValue = TS(k,c).tValue×α+Ob×(1−α)
7:          end if
8:          TS(k,c).iNum = TS(k,c).iNum + 1
9:      end while
```

Line 6 combines the new interaction and previous tValue using α. Dilution factor in this DWT algorithm is fixed. A potential improvement is to let $\alpha = f(Time_{Ob_n} - Time_{Ob_{n-1}})$. In this way, α is adjusted based on the time interval between two consecutive interactions.

STQD Algorithm

$TS_{(k,c)}$ denotes the trust state of trustee k in context c. It has two fields: iNum and value.

```
Input:  Construction factor Wc, destruction factor Wd
        such that Wc < Wd
1:      Initialise TS(k,c) as EWI algorithm does.
2:      while there exists new interaction Ob
3:          TS(k,c).iNum = TS(k,c).iNum + 1
4:          if Ob < TS(k,c).tValue then
5:              W = Wd
6:          else
7:              W = Wc
8:          end if
9:          TS(k,c).tValue = TS(k,c).tValue×(1−W)+Ob×W
10:     end while
```

Lines 4 to 8 determine whether the interaction is a destruction event and choose the factor correspondently.

STQD-S algorithm

$TS_{(k,c)}$ denotes the trust state of trustee k in context c. It has six fields: iNum, value, W_c, W_d, period (i.e. τ'), rest. The first five fields are discussed in section 4 and rest is an auxiliary field.

```
Input:  Foul event threshold γ. Initial construction factor
        Wc, destruction factor Wd such that Wc < Wd.
        Initial supervision period τ. Penalty ratios ρ1, ρ2.
        ρ3 such that ρ1, ρ2∈ (0, 1) and ρ3>1.
1:      Initialise iNum and value as EWI does.
2:      TS(k,c).Wd = Wd
3:      TS(k,c).Wc = Wc
4:      TS(k,c).period = τ
5:      TS(k,c).rest = 0
```

```
6:   while there are new interaction Ob
7:       if Ob ≤ γ then
8:           TS(k,c).Wd = TS(k,c).Wd + ρ1 × (1 − TS(k,c).Wd)
9:           TS(k,c).Wc = ρ2 × TS(k,c).Wc
10:          TS(k,c).rest = TS(k,c).rest + TS(k,c).period
11:          TS(k,c).period = ρ3 × TS(k,c).period
12:      end if
13:      TS(k,c).iNum = TS(k,c).iNum + 1
14:      if Ob ≤ TS(k,c).tValue then
15:          W = TS(k,c).Wd
16:      else
17:          W = TS(k,c).Wc
18:      end if
19:      TS(k,c).tValue = TS(k,c).tValue×(1−W)+Ob×W
20:      if trustee is under supervision and Ob > γ then
21:          TS(k,c).rest = TS(k,c).rest − 1
22:          if TS(k,c).rest = 0
23:              TS(k,c).Wd = Wd
24:              TS(k,c).Wc = Wc
25:          end if
26:      end if
27:  end while
```

The first to the fifth lines initialise a trustee's profile. Line 7 determines if a foul event occurs. If so, the trustee is put under supervision. As discussed in section 4, her current construction factor is decreased and destruction factor is increased (lines 8-9). Line 10 computes how long she will stay under supervision. Line 11 increments the supervision period that will be used when she conducts a foul event next time. The thirteenth to the nineteenth lines update tValue and iNum as SQDT does. If a trustee is under supervision and the new interaction is not a foul event, her rest supervision period is reduced by one (lines 20-21). The construction factor and destruction factor are restored when the supervision period ends (lines 22-25).

In STQD-S, the increase of destruction factor w_d needs to satisfy the following constraints:

1. *for any* n, $w_n^d \le 1$ *and* $w_n^d \le w_{n+1}^d$
2. $\lim_{n\to\infty} w_n^d = 1$

w_n^d is the destruction factor when a trustee is put under supervision n times without being released even once. The first constraint ensures that w_n^d is monotonic increasing with n. The upper bounded by 1. The second constraint indicates that the destruction factor can be close to 1 to any extent if n is large enough. The function we use is as follows.

$$w_{n+1}^d = w_n^d + \rho_1 \times (1 - w_n^d) \quad (4.2.1)$$

By solving equation 4.2.1, we get:

$$w_n^d = 1 - (1 - w_0^d)(1 - \rho_1)^n$$

where, w_0^d is the initial destruction factor.

∴ Equation 4.2.1 satisfies the above constraints.

Symmetrically, the decrease of construction factor w_c must satisfy the following constraints:

1. *for any* n, $0 \le w_n^c$ *and* $w_{n+1}^c \le w_n^c$
2. $\lim_{n\to\infty} w_n^c = 0$

w_n^c is defined similarly as w_n^d is. The function we use is as follows.

$$w_{n+1}^c = \rho_2 \times w_n^c \quad (4.2.2)$$

By solving equation 4.2.2, we get:

$$w_n^c = \rho_2^n \times w_0^c$$

where, w_0^c is the initial value of construction ratio.

∴ Equation 4.2.2 satisfies the above constraints.

4.3 Incremental trust analysis algorithms

For trust analysis algorithms, we are interested in finding their incremental versions that avoid the computation from the scratch by using the previous result and fixed number of individual interactions. Such algorithms may not exist when sliding window memory model is used. We prove the existence of incremental algorithms for EWI and DWT. We assume that the window size is n in the following discussion. The notation $TS_{k,m}$ is used to represent the trust state constructed based on an interaction sequence of length m starting from the k-th interaction..

Lemma 4.3.1: An incremental trust analysis algorithm exists for EWI rule.

Proof: If window is not full, $TS_{1,i+1}$ is computed by using equation 4.3.1

$$TS_{1,i+1} = (i \times TS_{1,i} + Ob_{i+1}) / (i+1) \qquad (i < n) \qquad (4.3.1)$$

If window is full, $TS_{k+1,n}$ is computed by using the following equation.

$$TS_{k+1,n} = TS_{k,n} + (Ob_{k+n} - Ob_k) / n \qquad (k \geqslant 0) \qquad (4.3.2)$$

Correctness of the algorithm:

Case 1: Equation 4.3.1 holds when an interaction is added and the window is not full.

$$TS_{1,i+1} = (\textstyle\sum_{j=1}^{j=i+1} Ob_j)/(i+1) =$$
$$(i \times TS_{1,i} + Ob_{i+1})/(i+1)$$

Case 2: Equation 4.3.1 holds when an interaction is added and the window is full.

$$TS_{k+1,n} = (\textstyle\sum_{j=k+1}^{j=k+n} Ob_j)/n = TS_{k,n} + (Ob_{k+n} - Ob_k)/n$$

In this algorithm, the oldest and the most current interactions are used. It agrees with the intuition that an incremental algorithm undoes the effect of the oldest interaction and appends that of the most current one.

Lemma 4.3.2: An incremental trust analysis algorithm exists for DWT rule.

Proof: If window is not full, $TS_{1,i+1}$ is directly computed according to the rule. If window is full, equation 4.3.3 is used.

$$TS_{k+1,n} = \alpha \times TS_{k,n} + (1 - \alpha) \times Ob_{k+n} - \alpha^n \times Ob_k - \alpha^n \times Ob_{k+1} \qquad (4.3.3)$$

Correctness of the algorithm:

By applying mathematic induction, it is easy to prove that for any $1 \le i$, we have:

$$\left.\begin{array}{l} TS_{i,n} = \alpha^{n-1} Ob_i + (1-\alpha)\times(\sum_{j=2}^{j=n} \alpha^{n-j} Ob_{i+j-1}) \\ TS_{k,n} = \alpha^{n-1} Ob_k + (1-\alpha)\times(\sum_{j=2}^{j=n} \alpha^{n-j} Ob_{k+j-1}) \\ TS_{k+1,n} = \alpha^{n-1} Ob_{k+1} + (1-\alpha)\times(\sum_{j=2}^{j=n} \alpha^{n-j} Ob_{k+j}) \end{array}\right\} \Rightarrow$$

$TS_{k+1,n} = \alpha \times TS_{k,n} + (1-\alpha) \times Ob_{k+n} - \alpha^n \times Ob_k - \alpha^n \times Ob_{k+1}$

In this algorithm, two oldest and the most recent interactions are used in the recurrence formula.

5. User behaviour models

In order to study the proposed trust production rules, interaction sequences representing different user behaviours need to be generated. A user's behaviour is determined by her trustworthiness as well as some unpredictable factors. We model a user's behaviours using random variables with normal distribution. Two characteristic functions $<f_m(n), f_v(n)>$ are used. The mean function $f_m(n)$ determines the mean value of the normal distribution at interaction n. It ranges over [0, 1] and characterizes the effect of trustworthiness. The variance function $f_v(n)$ determines the variance of the normal distribution at interaction n. It characterizes the effect of unpredictable factors. The rating of the interaction n is generated from the normal distribution with mean $f_m(n)$ and variance $f_v(n)$.

Four typical user behaviour models are identified. They are called "stable user", "repenting user", "cheater" and "smart cheater". Because we are interested in the trustworthiness of a user (the mean function $f_m(n)$), in the following discussion, we let $f_v(t)$ be a constant 0.51 such that ratings of interactions fall into $[f_m(n) - 0.1, f_m(n) + 0.1]$ with probability of 95%.

- *Stable user*: A stable user is the one whose trustworthiness does not change over time. The mean function of a stable user is defined as $f_m(n) = m$, where m is a constant. The ratings of interactions of a stable user obey the same normal distribution. Figure 4 shows the ratings of interactions of a stable user with $f_m(t)=0.8$. The fluctuation of ratings results from the unpredictable factors that affect the user's behaviors.

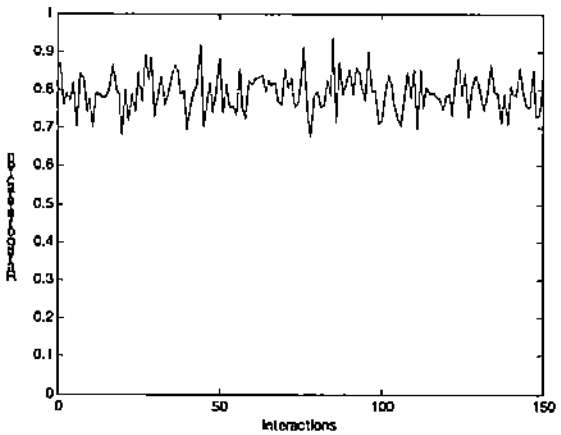

Figure 4 Behaviours of a stable user

- *Repenting user*: The behaviours of a repenting user can be divided into two phases: misbehaving followed by repenting. The characteristic of a repenting user is that her trustworthiness is low in the first phase, but high afterwards. This is demonstrated by a series of low rating interactions followed by suddenly shift of interactions to high ratings. The mean function of a repenting user is defined as:

$$f_m(n) = \begin{cases} m_{low} & n \le n_0 \\ m_{high} & otherwise \end{cases}$$

where n_0 is called the turning point.

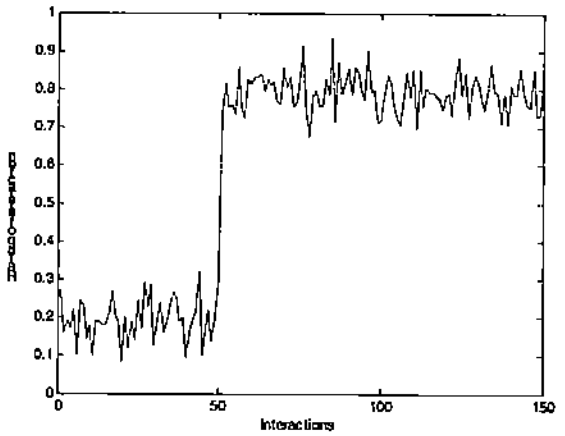

Figure 5 Behaviours of a repenting user

Figure 5 shows the ratings of interactions of a repenting user whose mean function $f_m(t)$ is 0.2 for the first 50 interactions. Then, $f_m(t)$ changes to 0.8.

- *Cheater*: A cheater behaves in the way opposite to a repenting user. The behaviours of cheaters can be divided into two phases: trust-building and trust-abusing. A user behaves well in the trust-building phase to achieve a trustworthy image. Then, he abuses the gained trust by misbehaving. The mean function of a cheater is defined as:

$$f_m(n) = \begin{cases} m_{high} & n \le n_0 \\ m_{low} & otherwise \end{cases}$$

where n_0 is the turning point.

Figure 6 shows the ratings of interactions of a cheater

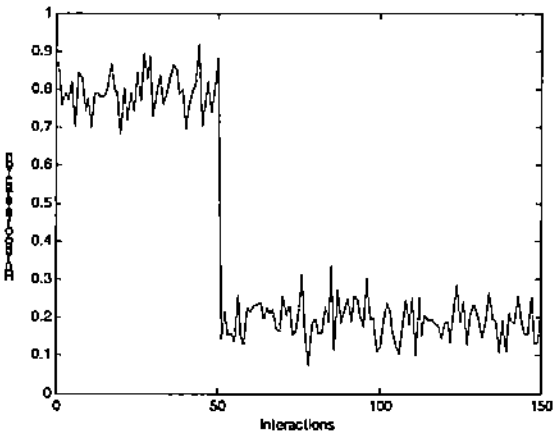

Figure 6 Behaviours of a cheater

whose characteristic function $f_m(t)$ is 0.8 for the first 50 interactions and 0.2 afterwards.

- *Smart cheater*: A smart user does bad things. However, instead of misbehaving continuously, he tempts cover the bad effects by intentionally doing something good after misbehaviours. He repeats the process of trust-building and trust-abusing. For a smart cheater, $f_m(t)$ is a periodic function. For simplicity, we assume the period is N, the mean function is defined as:

$$f_m(n) = \begin{cases} m_{high} & (n \bmod N) < n_0 \\ m_{low} & otherwise \end{cases}$$

Figure 7 shows the ratings of interactions of a smart cheater whose period is 20. In the first 15 interactions of each period, $f_m(t)$ is 0.8. For last 5 interactions, $f_m(t)$ drops to 0.2.

We investigate the above four user behaviour models in this paper. Other models can be defined and analyzed as well by using similar approaches.

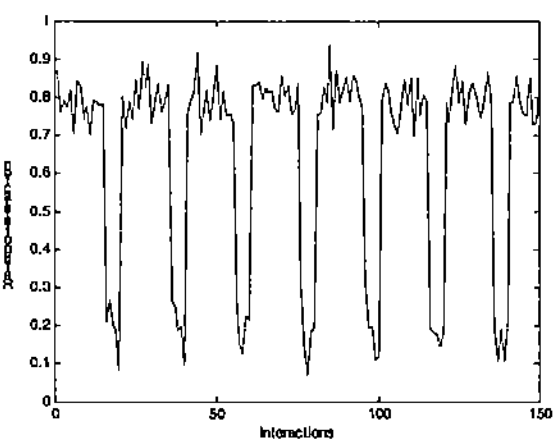

Figure 7 Behaviours of a smart cheater

6. Theoretical analysis

For a production rule, we denote T_k by the trust value evaluated based on the first k interactions. T_k is a random variable, whose expectation $E[T_k]$ summaries the value the rule may produce based on the k interactions.

For stable users, we are interested in whether the trust values evaluated by a trust production rule are also stable. For repenting users and cheaters, we would like to investigate how fast a rule responds to the sharp changes of the behaviours. In this section, we analytically investigate these issues by studying $E[T_k]$ for the rules proposed in section 3.

6.1 Stable user

Definition 6.1.1: A trust production rule *converges* for a stable user if $\lim_{k\to+\infty} E(T_k)$ exists. $\lim_{k\to+\infty} E(T_k)$ is called *theoretical trust* for a stable user.

Lemma 6.1.1: EWI rule converges for any stable user. The theoretical trust value is M, which is the mean of ratings of interactions.

Prove: $T_k = \frac{\sum_{i=1}^{k} Ob_i}{k} \Rightarrow E(T_k) = M \Rightarrow \lim_{k\to+\infty} E(T_k) = M$

Lemma 6.1.2: DWT rule converges for any stable user. The theoretical trust value is M, the mean of ratings of interactions.

Prove: Let α be the dilution factor.

$T_k = \alpha^{k-1} Ob_1 + (1-\alpha)\sum_{i=0}^{k-2} \alpha^i Ob_{k-i} \Rightarrow$

$E(T_k) = \alpha^{k-1} M + (1-\alpha)\sum_{i=0}^{k-2} \alpha^i M \Rightarrow E(T_k) = M$

$\therefore \lim_{k\to+\infty} E(T_k) = M$

6.2 Repenting user

Definition 6.2.1: A trust production rule with parameter set P *converges* for a repenting user if $\lim_{k\to+\infty} E(T_k)$ exists. $\lim_{k\to+\infty} E(T_k)$ is called *theoretical trust* for a repenting user.

Definition 6.2.2: Assume a trust production rule with parameter assignment P converges for a repenting user. Given 0<δ<1, the *δ-Transition phase* is defined as:

$$min\left\{ n \middle| n \geq n_0 and \left| \lim_{k\to+\infty} E(T_k) - E(T_n) \right| < \delta \right\} - n_0$$

where n_0 is the turning point.

δ-Transition phase is the minimum interaction number n_1 greater than the turning point such that the difference between the theoretical trust and expected trust value at n_1 is smaller than δ. It characterises how fast a production rule converges.

Lemma 6.2.1: EWI rule converges for a repenting user with means of m_{low} and m_{high} and turning point n_0. The theoretical trust value is m_{high}. δ-Transition phase of EWI is

$$max(\left\lceil \frac{n_0(m_{high}-m_{low})}{\delta} \right\rceil - n_0, 0)$$

Proof: $T_{n_0+k} = (T_{n_0} \times n_0 + \sum_{i=1}^{k} Ob_{n_0+i}) / (n_0+k)$

$$= \frac{T_{n_0} \times n_0}{n_0+k} + \frac{k}{n_0+k} \times \frac{\sum_{i=1}^{k} Ob_{n_0+i}}{k}$$

$$E(T_{n_0+k}) = E(\frac{T_{n_0} \times n_0}{n_0+k}) + \frac{k}{n_0+k} \times E(\frac{\sum_{i=1}^{k} Ob_{n_0+i}}{k})$$

$$= \frac{n_0}{n_0+k} E(T_{n_0}) + \frac{km_{high}}{n_0+k} = \frac{n_0 m_{low}}{n_0+k} + \frac{km_{high}}{n_0+k}$$

$$\lim_{k \to +\infty} E(T_k) = \frac{km_{high}}{n_0+k} = m_{high}$$

$$|\lim_{k \to +\infty} E(T_k) - E(T_n)| = \frac{n_0(m_{high}-m_{low})}{n} < \delta$$

$\therefore$ δ-Transition phase is $max(\left\lceil \frac{n_0(m_{high}-m_{low})}{\delta} \right\rceil - n_0, 0)$

Lemma 6.2.2: DWT converges for a repenting user with means of m_{high} and m_{low} and turning point n_0 for any dilution factor between 0 and 1. The ideal trust value is m_{high}. δ-Transition phase of dilute-with-time is

$$\max(\left\lceil \log_\alpha (\frac{\alpha^{n_0+1}\delta}{\alpha m_{high} - (\alpha^{n_0} - \alpha^{n_0-1}+1)m_{low}}) \right\rceil - n_0, 0)$$

Proof: $T_{n_0+k} = \alpha^{n_0+k-1} Ob_1 + (1-\alpha) \sum_{i=2}^{n_0+k-1} \alpha^{n_0+k-1-i} Ob_i$

$$= \alpha^{n_0+k-1} Ob_1 + (1-\alpha) \sum_{i=2}^{n_0} \alpha^{n_0+k-1-i} Ob_i$$

$$+ (1-\alpha) \sum_{i=n_0+1}^{n_0+k-1} \alpha^{n_0+k-1-i} Ob_i$$

$$E(T_{n_0+k}) = (\alpha^{n_0+k-1} + \alpha^{k-1}(1-\alpha^{n_0-1}))m_{low} + (1-\alpha^k)m_{high}$$

$$\lim_{k \to +\infty} E(T_k) = m_{high}$$

$$|\lim_{k \to +\infty} E(T_k) - E(T_n)| = \alpha^n (\frac{m_{high}}{\alpha^{n_0}} - \frac{(\alpha^{n_0} - \alpha^{n_0-1}+1)m_{low}}{\alpha^{n_0+1}})$$

$\therefore$ δ-Transition phase is

$$\max(\left\lceil \log_\alpha (\frac{\alpha^{n_0+1}\delta}{\alpha m_{high} - (\alpha^{n_0} - \alpha^{n_0-1}+1)m_{low}}) \right\rceil - n_0, 0)$$

6.3 Cheater

Converge and *δ-Transition phase* is defined for cheater in the similar way as those for repenting user. We can prove the following lemma as we do in section 6.2

Lemma 6.3.1: EWI rule converges for a cheater with means of m_{high} and m_{low} and turning point t_0. The theoretical trust value is m_{low}. δ-Transition phase of equal-weight-interaction is

$$\max(\left\lceil \frac{t_0(m_{high}-m_{low})}{\delta} \right\rceil - t_0, 0)$$

Lemma 6.3.2: DWT rule converges for a cheater with means of m_{high} and m_{low} and turning point t_0 for any dilution factor between 0 and 1. The ideal trust value is m_{low}. δ-Transition phase of DWT is

$$\max(\left\lceil \log_\alpha (\frac{\alpha^{n_0+1}\delta}{(\alpha^{n_0} - \alpha^{n_0-1}+1)m_{high} - \alpha m_{low}}) \right\rceil - n_0, 0)$$

The analytical study shows that EWI and DWT rules converge for stable user, repenting user, and cheater. For the last two user models, DWT (whose δ-Transition phase is $O(\log_\alpha \delta)$) converges faster than EWI (whose δ-Transition phase is $O(1/\delta)$) does.

The smart cheater user model, STQD and STQD-S production rules are too complicated to theoretically analyze. Their characteristics are studied via simulations in the next section.

7. Experimental study

Four sets of experiments are conducted to investigate how the rules proposed in section 3 perform for the four user models. The rules are implemented by using update algorithms.

The parameters for each rule are determined as follows.

- There is no parameter for EWI algorithm.
- DWT algorithm has one parameter: dilute factor that is 0.9. Hence, 90% of the current trust value comes from the previous trust value and 10% from the rating of the current interaction.
- Two parameters are used for STQD algorithm: construction factor and destruction factor, which are 0.05 and 0.1 respectively. The functionality of these parameters are discussed in section 4.
- The parameters used for STQD-S are shown in table 2. W_c and W_d represent the initial construction and destruction factors respectively. ρ_1, ρ_2 and ρ_3 are penalty ratios for construction factor, destruction factor, and supervision-period. The threshold of foul event is 0.18. The meanings of these parameters are introduced in section 4.

w_c	w_d	ρ_1	ρ_2	ρ_3	γ
0.05	0.1	0.9	0.1	2	0.18

Table 2 Parameters used in STQD-S algorithm

The results of the experiments are illustrated in figure 8 – 13. The x-axis of each figure is the sequence of interactions. The y-axis represents the evaluated trust value.

7.1 Experiments on stable users

This set of experiments is to investigate whether the rules produce stable trust values for stable users. 500 interactions are generated for a "stably good" user and a

"stably bad" user respectively. For the "stably good" user, $f_m(t) = 0.8$. For "stably bad" user, $f_m(t) = 0.2$.

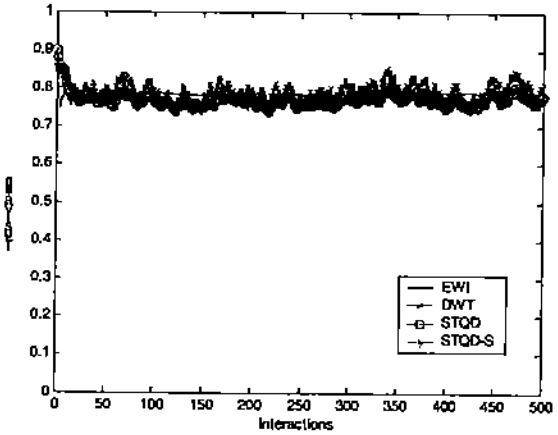

Figure 8 Results for a stably good user

Stably good user: The behaviours of the "stably good" user is shown in figure 4. Figure 8 demonstrates the results of applying the four algorithms to the user. The mean and variance of the evaluated trust values are shown in table 3. For a "stably good" user, all algorithms make the similar judgements (i.e., the mean of the trust values is close to the mean of the ratings of interactions). STQD-S and STQD algorithms have the same results because they are the same if no foul event occurs. The possibility of conducting a foul event for a "stably good" user with mean = 0.8 and variance = 0.51 is close to 0.

	EWI	DWT	STQD	STQD-S
Mean	0.7872	0.7943	0.7722	0.7722
Variance	0.007439	0.02251	0.01775	0.01775

Table 3 Simulation results for a stably good user

Stably bad user: The results of applying the algorithms to the "stably bad" user are illustrated in figure

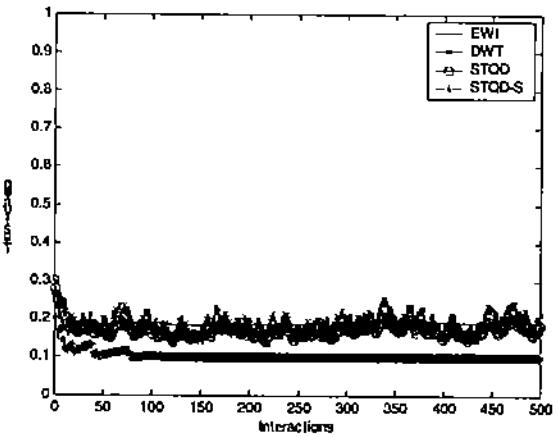

Figure 9 Results for a stably bad user

9. Table 4 shows the statistical results of the evaluated trust values. The values vary little for all algorithms as in the previous case. The trust values produced by the first three algorithms are still close to 0.2, the mean of ratings of interactions. STQD-S gives much lower values (about 0.1) than others do.

	EWI	DWT	STQD	STQD-S
Mean	0.1872	0.1943	0.1722	0.1072
Variance	0.007439	0.02251	0.01775	0.02243

Table 4 Simulation results for a stably bad user

Since the possibility for the "stably bad" user to conduct foul events is high, he is under supervision at most of the time. The construction and destruction factors become close to 0 and 1 respectively because of the punishment for foul events. Deconstruct events are the dominant interactions used for trust evaluation. Thus, the trust values are close to the minimum rating of interactions that is 0.1.

7.2 Experiments on repenting user

These experiments are carried out to study how the production rules respond to the change of a user's behaviours. 500 interactions are generated for a repenting user. The following equation defines her behaviours, which are illustrated in figure 5.

$$f_m(n) = \begin{cases} 0.2 & n \le 50 \\ 0.8 & 50 < n \end{cases}$$

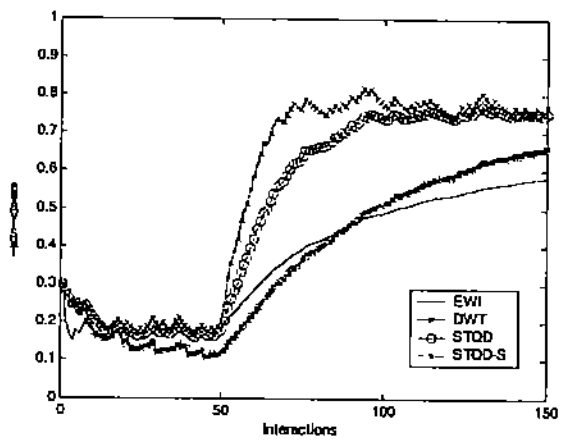

Figure 10 Results for a repenting user

Figure 10 shows the results for the first 150 interactions. The trust values of the first 50 interactions are like those of the "stably bad" user studied earlier. We are interested in how the algorithms respond to the sudden change of user behaviours. After the mean function $f_m(n)$ changes from 0.2 to 0.8, trust values produced by all algorithms start to increase until they become stable around 0.8. The trust value evaluated by DWT increases fastest. It is 0.2015 at the 50th interaction. It rises to 0.7169 at the 67th interaction. The trust value produced

that the smart cheater's effort to cover misbehaviours with good behaviours has less and less effect with the number of misbehaviours. The larger the number is, the faster the effect decreases. As shown in figure 13, after about 400 interactions (20 repetitions of the behaviour pattern), the trust value becomes stable at about 0.1 (i.e., the smart cheater is caught). The process may be speeded up by adjusting the parameters ρ_1, ρ_2 and ρ_3.

8. Conclusion

Trust production is significant for a computational trust model. We propose four trust production rules, equal-weight-interaction (EWI), dilute-with-time (DWT), slow-trust-quick-distrust (STQD), and slow-trust-quick-distrust-with-supervision (STQD-S) to automate this process. They partially or completely satisfy the time-dependent, interaction-dependent, trustee-dependent, and subjectivity properties. Four typical user behaviour models, stable user, repenting user, cheater, and smart cheater are developed. Analytical study shows that EWI and DWT converge for the first three user behaviour models, with DWT having a shorter transition phase. A series of experiments are conducted to investigate the performance of the trust production rules in terms of stable results, converging speed, and capability of catching a smarter cheater. The results indicate: (1) All rules make the same judgments for a stably good user. For a stably bad user, the trust value produced by STQD-S is lower than those produced by other rules. (2) EWI is the slowest one to respond to sharp change of user behaviors. (3) STQD and DWT have the similar results for all user models. (4) Only STQD-S can catch a smart cheater.

Trust production based on interaction sequences is the first step in the development of a computational trust model. We are designing algorithms that generate and interpret recommendations to propagate trust information among trusters. The next task is to investigate user behavior prediction and trust-related decision making based upon direct or indirect information. The smart cheater model is being extended to characterize more sophisticated user behaviors, which is of benefit to both trust modeling and fraud detection. Our ultimate objective is to establish an application-independent trust computing framework that will serve as a solid foundation to advance the security research in database and distributed systems.

REFERENCE

[1] A. Abdul-Rahman and S. Haites. Supporting trust in virtual communities. In Proc. of the 33rd Hawaii International Conference on Systems Science, 2000.

[2] K. Aberer and Z. Despotovic. Managing trust in a peer-2-peer information system. In Proc. of the 10th International Conference on Information and Knowledge Management (CIKM), pp 310-317, 2001.

[3] S. Marsh. Formalising trust as a computational concept. PhD thesis, Department of Computing Science and Mathematics, University of Stirling, April 1994.

[4] B. Yu and M. Singh. Distributed reputation management for electronic commerce. Computational Intelligence, 18(4):pp. 535-549, 2002.

[5] B. Yu and M. Singh. A social mechanism of reputation management in electronic communities. In Proc. of 4th International Workshop on Cooperative Information Agents, pp. 154-165, 2000.

[6] A. Josang. A logic for uncertain probabilities. International Journal of Uncertainty, Fuzziness and Knowledge-Based System, 9(3), 2001.

[7] M. Huhns and D. Buell. Trusted autonomy. IEEE Internet Computing, 6(3):pp 92-95, May/June 2002.

[8] C. Jonker and J. Treur. Formal analysis of models for the dynamics of trust based on experiences. In Proc. of the 9th European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW), 1999.

[9] V. Swarup and J. Fabrega. Trust: benefits, models, and mechanisms. Secure Internet Programming, Springer Verlag, 1998.

[10] T. Beth, M. Borcherding, and B. Klein. Valuation of trust in open networks. In Proc. of the European Symposium on Research in Computer Security (ESORICS), pp. 3-18, Nov. 1994.

[11] D. McKnight, V. Choudhury and C. Kacmar. Developing and validating trust measures for e-Commerce: an integrative topology. Information Systems Research, 13(3): pp. 334-359, Sep. 2002.

[12] J. Carbo, J. Molina and J. Davila. Trust management through fuzzy reputation. International Journal of Cooperative Information Systems, 12(1): pp. 135-155, Mar. 2003.

[13] S. Buchegger and J. Boudec. Performance analysis of the CONFIDANT protocol: cooperation of nodes - fairness in dynamic ad-hoc networks. In Proc. of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC), June 2002.

[14] D. Manchala. E-Commerce trust metrics and models. IEEE Internet Computing, 4(2): pp. 36-44, Mar./Apr. 2000.

[15] S. Jones, M. Wilikens, P. Morris and M. Masera. Trust requirements in e-business. Communications of the ACM, 43(12): pp. 81-87, Dec. 2000.

[16] D. Schoder and P. Yin. Building firm trust online. Communications of the ACM, 43(12): pp. 73-79, Dec. 2000.