Incremental Trust in Grid Computing

Brinkløv, Michael Hvalsøe; Sharp, Robin

Published in:
Seventh IEEE International Symposium on Cluster Computing and the Grid -- ccGrid07

Link to article, DOI:
10.1109/CCGRID.2007.63

Publication date:
2007

Document Version
Publisher's PDF, also known as Version of record

Citation (APA):
Brinkløv, M. H., & Sharp, R. (2007). Incremental Trust in Grid Computing. In Seventh IEEE International Symposium on Cluster Computing and the Grid -- ccGrid07 (pp. 135-144). IEEE. DOI: 10.1109/CCGRID.2007.63

Incremental Trust in Grid Computing

Michael Brinkløv Robin Sharp
Informatics & Mathematical Modelling
Technical University of Denmark
DK-2800 Kongens Lyngby, Denmark
{mhb,robin}@imm.dtu.dk

Abstract

This paper describes a comparative simulation study of some incremental trust and reputation algorithms for handling behavioural trust in large distributed systems. Two types of reputation algorithm (based on discrete and Bayesian evaluation of ratings) and two ways of combining direct trust and reputation (discrete combination and combination based on fuzzy logic) are considered. The various combinations of these methods are evaluated from the point of view of their ability to respond to changes in behaviour and the ease with which suitable parameters for the algorithms can be found in the context of Grid systems.

1 Introduction

In distributed computer systems, authorising a user to use a resource involves two vital issues of trust: the user and resource must be able to identify themselves to one another in a trustworthy manner, and they must trust one another not to misuse the system (by accident or design). We denote the two types of trust involved *identity trust* and *behavioural trust* respectively. Some classifications [7, 9] introduce further subdivisions of behavioural trust, but we shall not need these here.

In many Grid systems today, the mechanisms of identification and behavioural trust are combined. Typically, an identity is proved by presentation of a certificate, and used to classify its owner as belonging to a particular Virtual Organisation (VO), whose members are authorised to perform certain operations. This is satisfactory in systems which are static, in the sense that the set of entities is relatively constant over time, and the entities generally behave well (within limits set by the system). This is a common situation in Grid computing today, where typically a research community is given access to run similar applications on a given set of machines, and the members of the community all know and want to help one another.

In the future, the situation is likely to change, as more open Grid environments appear, where there may be thousands of users and computers, who dynamically come and go, do not know one another, and maybe do not have a friendly relationship. It will then no longer be realistic for an administrator to set up pre-defined, static behavioural trust relationships, and new computer-based mechanisms for determining the degree of trust are needed. These mechanisms should as far as possible emulate the activity of human administrators in deciding whether to adjust the rights given to the individual entities. Trust systems based on human notions of trust have in recent years been investigated by a number of groups, and the general principles involved elucidated (see for example [5]).

In this paper we consider some mechanisms for handling behavioural trust dynamically, so that the degree to which, say, A trusts B is determined by what information A has accumulated about B's behaviour. We assume anonymous malware attacks are detected by other means. The trust mechanisms considered are *incremental:* Good behaviour is rewarded by a higher degree of trust (leading, say, to authorisation to do more things), and bad behaviour by a lower degree. Although the mechanisms considered have all been presented in the literature, we are not aware of any systematic attempt to analyse them within comparable scenarios of use. The contribution of this paper is therefore to give a comparison of these mechanisms, based on a series of simulations.

The structure of the paper is as follows: In Section 2 we give a review of the trust mechanisms considered. In Section 3 we present and discuss the simulation experiments (which included both small and large distributed systems) and their results. In Section 4 we consider how such trust mechanisms can be efficiently introduced in a practical Grid environment, where issues of scalability become important, and finally in Section 5 we evaluate the various mechanisms.

Seventh IEEE International Symposium on Cluster Computing and the Grid(CCGrid'07)
0-7695-2833-3/07 $20.00 © 2007 IEEE

2 The Models Considered

In this study we assume that the identities of entities are undisputed and that accountability is perfect. We therefore use the unqualified term "trust" in the sense of "behavioural trust". We consider the trust which A has in B to be made up of two components:

Direct trust: based on A's experience of B's behaviour.

Reputation: evaluated from *ratings* communicated by third parties, based on their experiences of B's behaviour.

Different models differ in the ways in which direct trust and reputation are combined, how ratings are combined to give the reputation, the extent to which old information is discarded as time passes, etc. A large number of models of trust have been described in the literature, and this is not the place for a complete review. Instead we focus on some central issues which differentiate the models.

2.1 Combining Direct Trust and Reputation

There are two predominant ways of combining direct trust and reputation: so-called *discrete models* and models based on *fuzzy logic*. Discrete models have in particular been promoted by Azzedin and Maheswaran [2, 1], who used a linear combination of a direct trust function and a reputation function to evaluate the trust $\Gamma(x, y, t)$ of x in y at time t:

$$\Gamma(x, y, t) = w_d * \Theta(x, y, t) + w_r * \Omega(x, y, t)$$

where w_d and w_r $(= 1 - w_d)$ are constant weights, Θ is the direct trust function:

$$\Theta(x, y, t) = DTT(x, y) * \Upsilon(t - t_{xy})$$

and Ω is the reputation function:

$$\Omega(x, y, t) = \frac{\sum_{z \in \mathcal{D}-x} RTT(z, y) * R(z, y) * \Upsilon(t - t_{zy})}{|\mathcal{D} - x|}$$

where $\mathcal{D}$ is the domain of entities. Υ is a *decay function*, expressing the rate at which trust depreciates, where t_{zy} is the time at which z most recently interacted with y. DTT and RTT are tables of the current direct trust and reputation values respectively, and $R(z, y) \in [0, 1]$ is a *recommender trust factor*, expressing our confidence that there is no collusion between z and y. Azzedin and Maheswaran have presented several variations on this basic model, including variable contexts and the use of brokers [3], and have evaluated their performance in a Grid context.

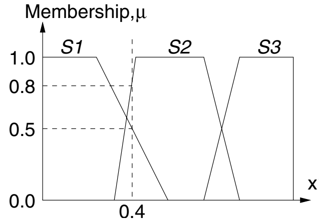


Figure 1. Fuzzy sets and membership functions

The main alternative to discrete models is to use *fuzzy logic*. The rationale for this is that trust is a linguistic concept, which is poorly described by simple numerical values, say between -1 (total distrust) to +1 (absolute trust). Fuzzy inference should be a useful theoretical apparatus for dealing with the lack of precision inherent in describing trust and reputation, e.g. that a given behaviour will lead to a "moderately high" degree of trust. For each behavioural parameter, a set of (possibly overlapping) *fuzzy sets* is defined in terms of *membership functions*, which specify the degree of membership of each fuzzy set for the possible values of the parameter [12, 15]. For example, in Figure 1, if parameter x_1 has value 0.4, then it has degree of membership 0.5 in set S_1, 0.8 in set S_2 and 0.0 in set S_3. When the result of using several parameters $(x_1, x_2, \ldots, x_n)$ has to be combined to a result y, a set of rules is used of the form:

$$\textbf{if } (x_1 \tilde{\in} S_i \wedge x_2 \tilde{\in} T_j \wedge \ldots) \textbf{ then } y \tilde{\in} O_k$$

where $S_i, T_j, \ldots O_k$ are fuzzy sets and $\tilde{\in}$ indicates (a non-zero degree of) *fuzzy* membership. The membership functions μ_k for the output fuzzy sets $O_k, k = 1, \ldots, r$ are commonly given by the so-called min-max formula:

$$\mu_{Ok}(y) = \max[\min[\mu_{Sk}(x_1), \mu_{Tk}(x_2), \ldots]]$$

as shown in Figure 2. For a given set of input parameter values, this gives a fuzzy output set. The actual output value, $y*$, is found from the membership function of this set (shaded) by *defuzzification*. A variety of methods are available for this. Two of the most widely used are:

Center of Gravity (CoG): The weighted average of the output membership function.

Mean of Maximum (MoM): The mean of the highest points of the output membership function.

The use of fuzzy logic to evaluate trust in Grid systems has been proposed by Hwang and Song [13, 14], although their approach had a different focus from ours, as they aimed at combining evaluations of *job success rate* and *self-defence capability* to find an overall trust value.

0-7695-2833-3/07 $20.00 © 2007 IEEE

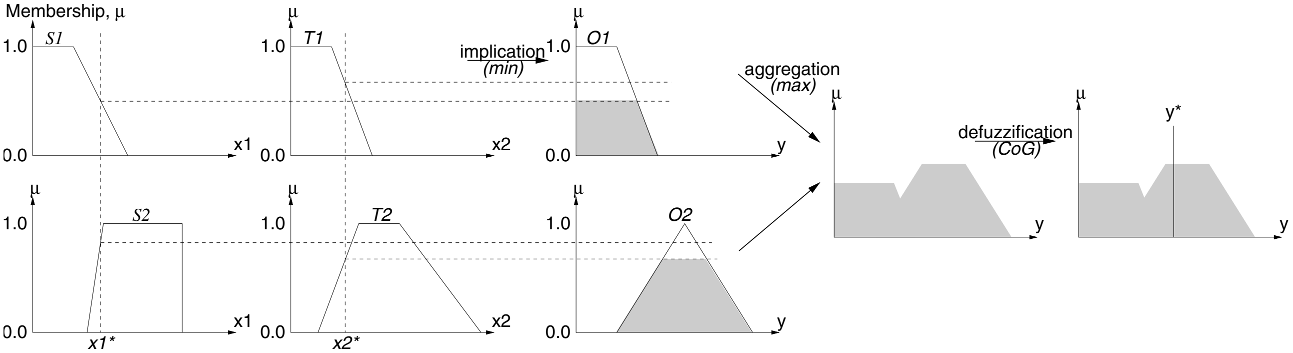


Figure 2. Fuzzy inference and defuzzification

2.2 Evaluating Reputation

Methods for evaluating reputation from ratings provided by third parties also fall largely into two classes: *discrete* methods and methods based on *Bayesian* statistics. The TRUMMAR system introduced by Derbas et al. [6] is a typical example of a discrete method, based on the simple idea that x's trust in z influences x's opinion on z's rating of y. Then y's reputation as seen by x at time t is:

$$\begin{aligned}\Omega(x,y,t) \quad = \quad & w_o * \Omega(x,y,t_{xy}) * \Upsilon(t - t_{xy}) + \\ & w_n * \frac{\sum_z \Omega(x,z,t_{xz}) * \Omega(z,y,t_{zy})}{\sum_z \Omega(x,z,t_{xz})} + \ldots\end{aligned}$$

w_o and w_n are constant weights, and the summation includes all "neighbours", z, of x who contribute information. A similar approach is taken in Eigentrust [10] and other systems.

Bayesian methods evaluate a post-observation reputation value from the pre-observation value and the result of an observation. Such methods have achieved prominence as a result of the work of Jøsang and his co-workers in developing the *Beta reputation system* [8]. This was originally targeted at e-commerce, but can be applied in many other areas [9]. A similar general approach, also focussed on e-commerce, was used by Mui et al. [11]. The calculation of reputation makes use of the beta probability density function, defined as the function f of parameters α and β:

$$f(p \mid \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1}$$

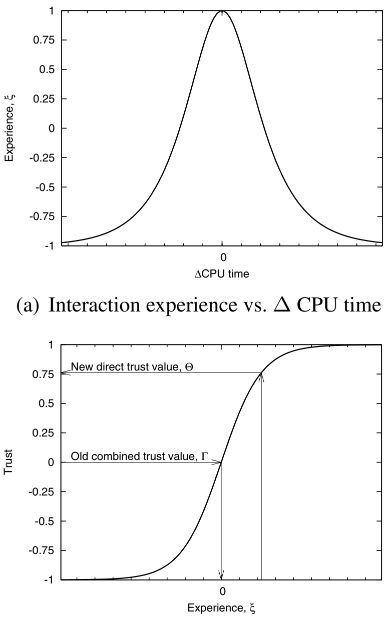
where $0 \leq p \leq 1$ and $\alpha, \beta > 0$. Here $\Gamma(x)$ is the usual gamma function, and we apply the constraints that $p \neq 0$ if $\alpha < 1$ and $p \neq 1$ if $\beta < 1$. With the beta function, the expectation value for the probability is $\mathcal{E}(p) = \alpha/(\alpha+\beta)$.

If we have observed a process with two outcomes, say $\{x, \bar{x}\}$, and have seen r outcomes x and s outcomes $\bar{x}$, then the probability density function for seeing outcome x in the future is expressed by f with $\alpha = r + 1$ and $\beta = s + 1$. This has a maximum at $\mathcal{E}(p) = (r+1)/(r+s+2)$. Then y's reputation as seen by x is $\Omega(x,y) = f(p \mid r(x,y), s(x,y))$, where $r(x,y)$ and $s(x,y)$ are respectively the total amount of positive and negative feedback about y provided by x (all at time t).

As in most statistical approaches, the calculation assumes that the behaviour of the systems is *stationary*, so the ratings can be based on all observations back to "the beginning of time". But often the systems' behaviour changes with time, and our main interest is to discover when such changes take place. In the Beta system, this issue is dealt with by introducing a *forgetting factor*, $\lambda \in [0,1]$, such that $r(x,y) = \sum_{i=1}^{n} r(x,y,i) \cdot \lambda^{n-i}$. Here $r(x,y,i)$ is the i'th value of the sequence of n positive feedback values about y provided by x. A similar formula holds for s. If $\lambda = 0$, only the most recent value is used in the calculation, while if $\lambda = 1$ all previous values are used. It is not necessary to store all the old feedback values, as $r(x,y)$ can be found from a recursion formula: $r(x,y) = r_{old}(x,y) \cdot \lambda + r_{obs}$, where r_{obs} is the latest positive feedback received, and likewise for s. Effectively this restricts the reputation calculation to a *window* of width $\sim 1/(1-\lambda)$ observations up to the current instant.

3 Simulation Experiments

To experiment with the various models, we created a discrete event-driven simulator [4] in C++ using the Free Fuzzy Logic Library [16] and the GNU scientific library. The simulations model a number of users and resources that interact by sending and receiving jobs. Job submission times are determined from a Poisson distribution with mean ν. For each job, a user, i, and a resource, j, are selected at random. If the user's trust in the resource exceeds the user's *trust threshold*, τ_i, the user submits the job to the



(a) Interaction experience vs. Δ CPU time

(b) Direct trust vs. Experience

Figure 3. Determining direct trust

resource, which accepts it if its trust in the user exceeds its own threshold, τ_j.

When the job finishes, the user and resource each calculate a degree of satisfaction (the *experience*, ξ) for the interaction, based on the difference between the job parameters and what actually happened. We assume here that the resource can determine if the user specified the true requirements for the job, and the user can check whether the resource provided what was asked for. The value of ξ is then used to calculate a new direct trust value. Afterwards a number of entities in the system are asked to give a *rating* for use in calculating a new reputation value. A rating should express the entity's trust in the particular user (or resource). However, badly behaving entities may lie by submitting inaccurate ratings.

The reputation can be calculated using either of the reputation models, and the direct trust and reputation values are combined to a final trust value using the discrete or the fuzzy logic approach. We use the notation *Discrete-Beta* for discrete combination of trust where the reputation is calculated using the Beta reputation system, *Fuzzy-Discrete* for fuzzy logic combination of trust where the reputation is calculated using the discrete reputation system, and similarly for *Discrete-Discrete* and *Fuzzy-Beta*.

In the experiments, ξ was for simplicity evaluated just from differences in CPU time, using the heuristic shown in Figure 3(a). When Δ CPU time is small, both user and resource had an experience matching what was promised, so ξ is large. As Δ increases, ξ falls: the user experienced that the resource did not complete the job or overcharged for it, or the resource experienced that the user under- or overstated the job's CPU time. The calculation of the direct trust value, Θ, is illustrated in Figure 3(b). The previous combined trust value, Γ, is used in calculating the new value of Θ, since until new evidence is uncovered the old value of Γ should give the best indication of the current level of trust. These rules give a high trust value quickly if the user or resource behaves well, while many good experiences are needed for a very high trust value. Likewise, bad behaviour quickly leads to a low trust value, but many bad experiences are needed for a very low trust value.

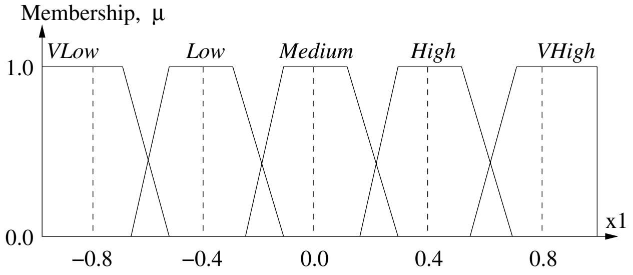


Figure 4. Fuzzy membership functions

The behaviour of an entity (user or resource) in the simulation is described by two separate activities: the request (or allocation) of CPU time and the return of a rating. When a simulated user has a job that requires CPU time t_0, he will request $t_r = t_0 + \delta_u$, where δ_u is a randomly chosen deviation. When δ_u is small, the resource has a good experience ξ (cf. Figure 3(a)), while large δ_u gives a bad experience. Similarly, when a resource allocates CPU time to the job, it will in fact allocate $t_a = t_r + \delta_s$. For entity i, values of δ_u and δ_s are drawn from a Laplace distribution with width a_i. If a_i is small, there is a higher probability of a deviation close to zero, corresponding to good behaviour, while large a_i gives bad behaviour. Similarly, when i returns a rating for j, a deviation δ_r is added to the true reputation value to emulate the effect of i lying. δ_r is also drawn from a Laplace distribution with width a_i, but is divided by 100 to ensure that the rating seems credible.

We used the fuzzy logic membership functions shown in Figure 4. These are based on Song and Hwang's functions from [13, 14], but use trapezoidal instead of sigmoid curves. Each of our fuzzy sets has the same area, and the same membership functions are used for input (direct trust, Θ, and reputation, Ω) and output (trust, Γ). The default fuzzy rule base will enforce a trust policy where di-

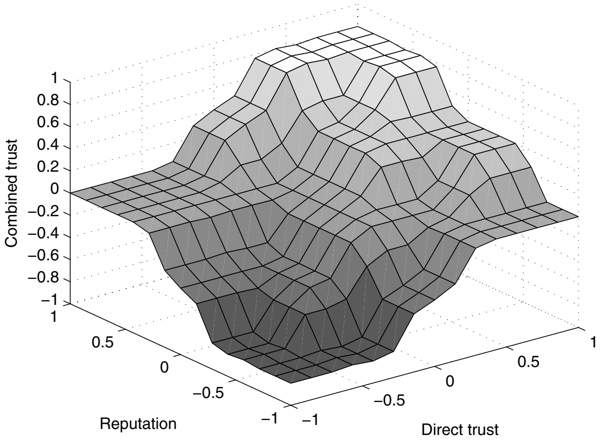


Figure 5. Control surface

rect trust and reputation are as far as possible considered to be equally significant, as in the rules:

$$(\Theta \tilde{\in} Medium \wedge \Omega \tilde{\in} VHigh) \Rightarrow \Gamma \tilde{\in} High \quad (1)$$
$$(\Theta \tilde{\in} Low \wedge \Omega \tilde{\in} High) \Rightarrow \Gamma \tilde{\in} Medium \quad (2)$$

When there is no output class midway between the two, however, the fuzzy rule base will enforce a policy where the most extreme class is chosen, as in rules like:

$$(\Theta \tilde{\in} High \wedge \Omega \tilde{\in} VHigh) \Rightarrow \Gamma \tilde{\in} VHigh \quad (3)$$
$$(\Theta \tilde{\in} VHigh \wedge \Omega \tilde{\in} High) \Rightarrow \Gamma \tilde{\in} VHigh \quad (4)$$
$$(\Theta \tilde{\in} Low \wedge \Omega \tilde{\in} VLow) \Rightarrow \Gamma \tilde{\in} VLow \quad (5)$$

Figure 5, showing the control surface for the fuzzy membership functions and rule base, gives an overview of the security policy. It should be noted that when $\Gamma \tilde{\in} VHigh$, the maximum defuzzified value is 0.8, as this is the center of gravity of the *VHigh* set. Likewise when $\Gamma \tilde{\in} VLow$, the minimum defuzzified value is -0.8, the CoG of the *VLow* set. This limits the output Γ's range to $[-0.8 : 0.8]$. Since the trust scale is arbitrary, the Discrete combination method was for ease of comparison also limited to this interval. As the policy of the fuzzy rule base predominantly weights direct trust and reputation equally, they are also weighted equally ($w_r = w_d$) in the Discrete combination method. The most important simulation parameters are summarised in Table 1.

The experiments were performed for systems with a wide range of sizes, from small clusters with 4 users and 4 resources, to large systems with 1000 users and resources. Each experiment was repeated 20 times using different random number seeds, for each of the combination methods, each using the Discrete or the Beta reputation system. Ratings were in all cases collected from 3 other entities.

Parameter		Parameter	
Mean job submission time, ν	25.0	Initial trust value	0.0
		Forgetting factor, λ	0.4
$w_d,\ w_r$	0.5	a_i, "good" behaviour	2.0
w_o	0.0	a_i, "bad" behaviour	20.0
w_n	1.0	Trust threshold, τ_i	-1.0

Table 1. Simulation parameters

Combinaton method – Reputation system	Trust, Γ	Direct trust, Θ	Reputation, Ω
Discrete-Discrete	8.4	4.6	8.55
Discrete-Beta	9.9	4.2	21.75

Table 2. Average no. of events before a user's trust in a resource falls below -0.5

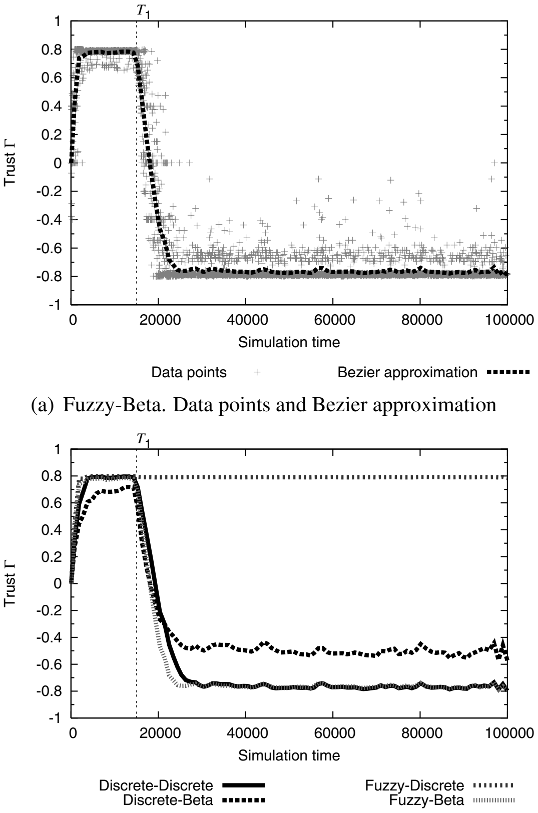
3.1 Detecting bad behaviour

In the first simulation scenarios, we investigated how rapidly trust in a single resource will depreciate when this resource exhibits bad behaviour, while the remaining entities all behave well. The experiment could of course equally well have been done with a user exhibiting bad behaviour. Note that a "bad" resource does not misbehave continuously – but there is a *non-zero probability* of it allocating too much or too little CPU time for any job.

Each of the two reputation systems was used, and direct trust and reputation were combined using the Discrete method with the weights $w_d = w_r = 0.5$. The rate at which trust evolves is indicated in Table 2, which lists the number of interactions ("jobs") needed to reach a trust value below -0.5. One noticeable feature is the much slower change in the reputation when using the Beta reputation system. We shall look more closely into this below.

As it cannot be expected that an entity behaves either well or badly all the time, we next investigated what happens when a resource changes from good to bad behaviour at time T_1. Figure 6(a) contains the data points and the corresponding Bezier approximation for the combined trust for the Fuzzy-Beta method with $T_1 = 15\,000$. Figure 6(b) contains Bezier approximation curves for the two combination methods and the two reputation systems. Only the combined trust is plotted, since in the real world only the combined trust would be used to make a decision.

In Figure 6(b) we see that the Fuzzy logic method using the Discrete reputation system does not detect the change in behaviour. The reason for this lies in the fact that when $T_1 = 15\,000$, the reputation $\Omega > 0.7$. From Figure 4 we see that Ω will then only have non-zero membership of the *VHigh* set. Now there are three fuzzy rules where



(a) Fuzzy-Beta. Data points and Bezier approximation

(b) Bezier approximations

Figure 6. Change from good to bad behaviour at time $T_1 = 15\,000$

$\Omega \tilde{\in} VHigh$:

$$\begin{array}{rcll} (\Theta \tilde{\in} VHigh \wedge \Omega \tilde{\in} VHigh) & \Rightarrow & \Gamma \tilde{\in} VHigh & (6) \\ (\Theta \tilde{\in} High \wedge \Omega \tilde{\in} VHigh) & \Rightarrow & \Gamma \tilde{\in} VHigh & (3) \\ (\Theta \tilde{\in} Medium \wedge \Omega \tilde{\in} VHigh) & \Rightarrow & \Gamma \tilde{\in} High & (7) \end{array}$$

In order for the trust value, Γ, to have a non-zero membership of the set *High*, the direct trust, Θ, must have some degree of membership of the set *Medium*. If not, the result will be $\Gamma \tilde{\in} VHigh$. Since ratings are based on Γ, this will in turn lead to high ratings and consequently a reputation value in the set *VHigh*.

This odd behaviour can be avoided by putting more emphasis on the direct trust value. Similar consisderations apply to rules where $\Omega \tilde{\in} VLow$. Specifically, rules (3) and (5) should be changed to:

$$\begin{array}{rcll} (\Theta \tilde{\in} High \wedge \Omega \tilde{\in} VHigh) & \Rightarrow & \Gamma \tilde{\in} High & (3') \\ (\Theta \tilde{\in} Low \wedge \Omega \tilde{\in} VLow) & \Rightarrow & \Gamma \tilde{\in} Low & (5') \end{array}$$

This can be understood as a change in security policy, whose effect becomes apparent when a user or a resource has calculated a *VHigh* (/ *VLow*) reputation value but has only experienced *High* (/ *Low*) direct trust. Figure 7 shows that with this new policy the change in behaviour is correctly detected.

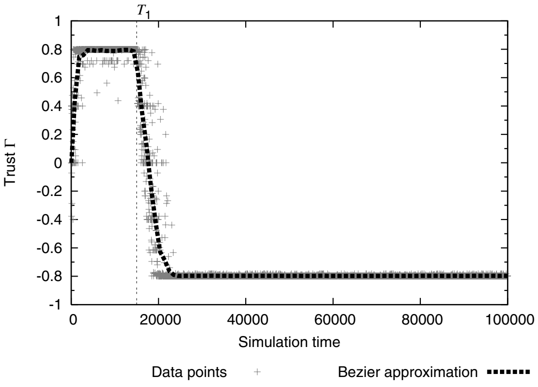


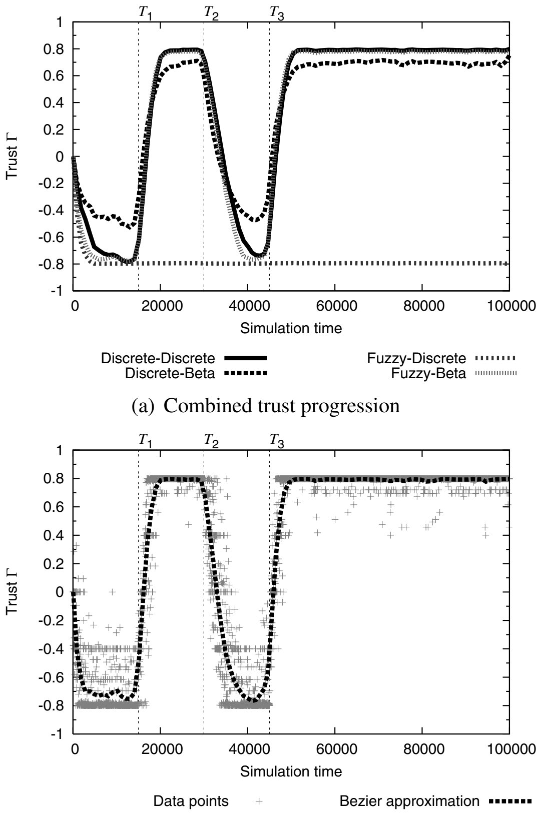
Figure 7. Fuzzy-Discrete

Combination method – Reputation system	Trust, Γ	Direct trust, Θ	Reputation, Ω
Discrete-Discrete	19.5	17.3	22.05
Fuzzy-Discrete*	12.6	11.35	13.95
Discrete-Beta	26.35	15.15	73.8
Fuzzy-Beta	14.3	12.55	18.75

Table 3. Average no. of events after T_1 before a user's trust in a resource falls below -0.5

Evidently, if we use a fast reputation system, such as the Discrete system, extra consideration must go into the fuzzy rule base design. Otherwise we may get excessive ratings when the behaviour changes, and effectively an undesired security policy. The experiment in Section 3.2 will further illustrate this. Each of the three remaining models arrives at the correct conclusion: that the behaviour of the resource changes from good to bad. With the Discrete method and the Discrete reputation system, the trust values initially increase rapidly to the maximum, and after time T_1 they quickly decrease to the minimum values.

Table 3 shows the number of events needed for the trust values to decrease below -0.5. The Fuzzy-Discrete combination uses the alternative security policy (rules $(3')$ and $(5')$). With the Discrete-Beta system, the reputation changes very slowly, whereas the Fuzzy-Beta system is comparable with Fuzzy-Discrete. Interestingly, both of these are slightly faster than the Discrete-Discrete system, the reason for this being that the number of observations before T_1 is rather small, so only limited history is incorporated into the Beta reputation.



(a) Combined trust progression

(b) Fuzzy-Discrete. Alternative rule base

Figure 8. Oscillating behaviour

3.2 Oscillating behaviour

The next experiment used a more extreme setup, where the behaviour of a resource oscillates, starting with bad behaviour. At time T_1 (here $15\,000$) it starts to behave correctly, at T_2 $(30\,000)$ it begins to misbehave again, and finally from T_3 $(45\,000)$ it behaves correctly. The Bezier approximation curves for the two combination methods using the two reputation systems are shown in Figure 8(a). As in the previous experiments, only a single user's view of the combined trust values for the resource is shown.

Once again, the Fuzzy logic method using the Discrete reputation system does not pick up the change in behaviour, and to avoid this, the security policy expressed by rules $(3')$ and $(5')$ needs to be used. Figure 8(b) demonstrates the Fuzzy-Discrete approach using this alternative rule base. This experiment again illustrates that, when a fast reputation system is used together with fuzzy logic, care needs to be put into the design of the rule base. Otherwise it may enforce an undesired security policy.

Combination method – Reputation system	Trust, Γ	Direct trust, Θ	Reputa- tion, Ω
Discrete-Discrete	9.15	6.35	10.3
Fuzzy-Discrete*	7.15	4.5	7.65
Discrete-Beta	9.1	4.25	15.15
Fuzzy-Beta	8.85	5.0	11.05

Table 4. Average number of events after T_3 before a user's trust in the resource is above +0.5

Combi. method	Forgetting factor, λ				
– reput. system	0.0	0.4	0.9	0.99	1.00
Discrete-Beta	7.15	9.10	17.15	105	∞
Fuzzy-Beta*	6.75	8.25	14.85	118	∞

Table 5. Average no. of events after T_3 before a user's trust in a resource exceeds +0.5. Fuzzy-Beta uses rules $(3')$ and $(5')$.

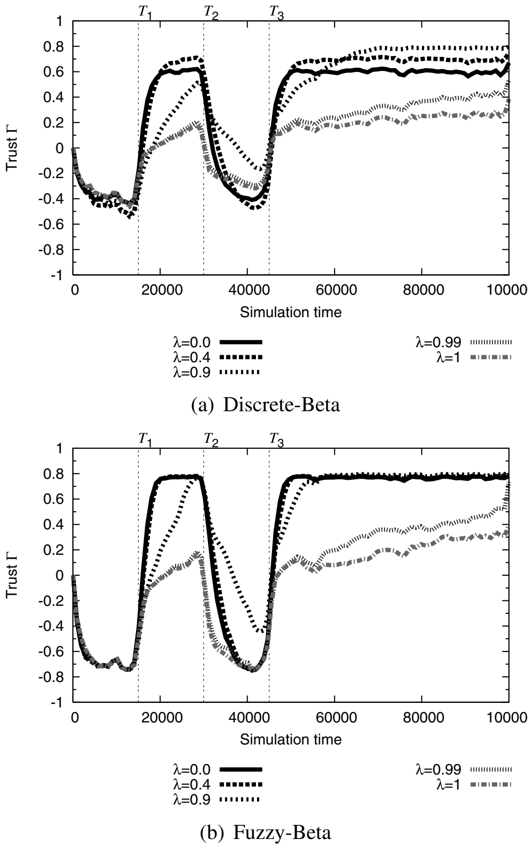
The remaining methods and reputation system in Figure 8(a) show the expected progression of trust. Table 4 shows the number of events needed for the trust to increase above +0.5 after time T_3. Note that Fuzzy-Discrete uses the alternative rule base. As expected, the Beta reputation system required significantly more events before reaching the +0.5 level, because it includes historical data in the trust calculation, and is therefore slower to react to changes in behaviour. The rate of reaction should be determined by the value of the forgetting factor (here, $\lambda = 0.4$); this was investigated in the next experiment.

3.3 Influence of the forgetting factor

This experiment was based on the same scenario, but the forgetting factor λ in the Beta reputation system was varied, so that reputation values from various periods of time would be included in the calculation. The random seeds were the same as in the previous experiment. Results for five values of λ are shown in Figure 9.

As λ increases, we observe two effects. As long as λ is still relatively small ($\lesssim 0.6$), the combined trust values, as expected, react more slowly to changes in behaviour. When λ approaches 1, on the other hand, so much historical information is used that the calculation does not correctly follow the changes in behaviour at all – there is simply too much "historical baggage" from epochs with significantly different behaviour. Table 5, showing the number of events needed after time T_3 for Γ to exceed +0.5 for different forgetting factors, confirms these observations.

It is plainly important to choose a suitable value for λ when using the Beta reputation system. With λ near 0, historical knowledge is forgotten immediately, giving a fast



(a) Discrete-Beta

(b) Fuzzy-Beta

Figure 9. Different forgetting factors

reaction to changes in behaviour, so the Beta method is more or less equivalent to the Discrete method. Moderate values of λ include a certain amount of historical knowledge, ensuring that the trust rating will only change noticeably after a certain number of observations of "new" behaviour have been made, but still giving a fairly rapid response. Values of λ close to 1 require a large number of observations of "new" behaviour before the trust rating changes significantly, and there is a serious risk of including observations from different epochs of behaviour, rendering the trust evaluation statistically invalid.

3.4 Larger scale experiments

In the previous experiments, only 4 users and 4 resources were considered. Further experiments were therefore conducted to see whether the conclusions still apply in larger systems, where the effect of a small amount of misbehaviour might get overlooked.

The first experiment resembles that in Section 3.2, but we now investigate what happens as we change the number of interactions between entities when there are 100 users and 100 resources. Again, a single resource varies its behaviour, starting with bad behaviour. At T_1 (here $7\,500\,000$) it will start to behave correctly, at T_2 $(15\,000\,000)$ it will start to misbehave again, and finally from T_3 $(22\,500\,000)$ it will behave correctly again. As before, only three ratings are used to calculate the reputation, as this seems adequate in practice. To achieve rapid reputation progression, users and resources are able to provide references to other entities with which they have recently had contact. This means that the ratings which an entity receives are reasonably updated. A user's perception of the resource is shown in Figure 10.

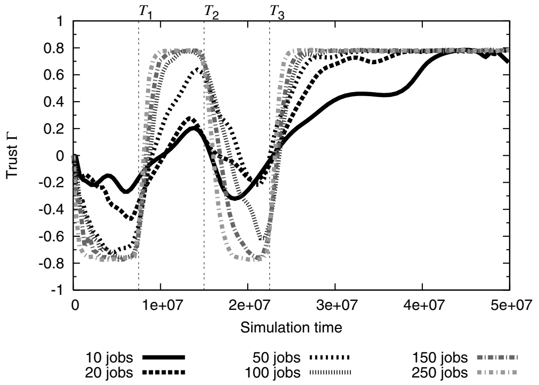


Figure 10. Fuzzy-Beta. Varying numbers of interactions

Only the results for Fuzzy combination using Beta reputation are shown; results for the other methods were similar. As the number of interactions is reduced, the span of trust values falls. However, in general terms the user still sees the resource varying its behaviour before ending with good behaviour. Most importantly, even with an average of only 10 interactions the correct general progression of trust is observed, though it obviously reflects the true situation more closely after more interactions.

In the second experiment, the number of entities was increased even further. In all cases, 50% of the entities were users and 50% resources, and a single resource varied its behaviour. Although they are not identical, the trust progression curves (not shown here) for all four methods showed strong similarities, regardless of the number of entities present. Most importantly, all cases arrived at the same conclusion: that the behaviour of the resource varies before ending with good behaviour. In this experiment, each user interacted with each resource approx. 20 times. Thus in the scenario with 8 entities (4 users, 4 resources) around 4000 jobs were simulated. With 1000 users and 1000 resources, about 20 million jobs were simulated.

Users	Resources	Trust, Γ	Direct trust, Θ	Reputation, Ω
4	4	3.9	2.55	5.68
25	25	3.3	2.5	4.85
100	100	3.35	2.85	4.75
500	500	4.25	3.4	4.5
1000	1000	3.05	2.1	4.6

Table 6. Average number of events after T_3 before a user's trust in the resource exceeds +0.5

Table 6 offers further evidence that the progression of trust is insensitive to the number of entities, even though significantly more jobs are processed in the larger scale experiments. It should be remembered that the behaviour of each user and resource is chosen independently in each experiment. Since both the trust progression plots and the average numbers of events in Table 6 are very similar, it seems fair to conclude that the trust models can detect the correct progression of trust, regardless of system size.

4 Practical deployment considerations

For trust mechanisms to be useful in a real Grid environment, two important requirements must be met:

1. Evaluation of trust must make economical use of storage, CPU time and network bandwidth.
2. It must be easily possible to calculate a meaningful value for the experience, ξ.

Since each user (subject) potentially needs to have access to information about all resources and vice versa, the total space requirements for each of the methods considered here are proportional to the product of the number of subjects and the number of resources in the system. (On a given node, it is of course only necessary to store the slice of this information which refers to "the others".) The factor of proportionality depends on the reputation system – for example in the Beta system, two values (r and s) are stored, whereas the Discrete system only needs one.

With respect to CPU time, Discrete combination of direct trust and reputation only requires a simple arithmetic calculation, whereas Fuzzy logic requires the much more demanding fuzzy inference and defuzzification. In particular, use of Gaussian membership functions combined with CoG defuzzification may lead to poor performance if the fuzzy logic implementation is not optimized for this. Our experiments, using trapezoidal membership functions, showed no performance issues.

Network usage is mainly affected by the submission and retrieval of ratings. In our experiments, we calculated reputation from three ratings, regardless of the number of users and resources. (This gave an accurate picture of reputation when an entity could provide references to other entities.) Network traffic is proportional to the square of the number of ratings exchanged, so a compromise between network performance and reputation accuracy must be found. Practical monitoring of a real trust-based Grid environment is needed to investigate this question.

In these experiments, the experience, ξ, was evaluated from the difference between the specified and actual CPU times for a job. In a more realistic setting, more parameters, such as main storage usage, disc storage usage and amounts of data transferred via the network would be included. This makes no difference in principle to the results observed here, although the resources needed to store and calculate this information would obviously be larger.

The significance of a particular experience depends, as we have seen, on the policy in use, which determines the appropriate reward or punishment. This is a matter for the community concerned, related to the acceptance of risk and the extent to which a fast reaction to small changes is desired. The policy needs to be reflected in such factors as the form of fuzzy logic membership functions, fuzzy rule bases, and discrete weighting factors, depending on the model chosen. In this respect, Grid computing is different from e-commerce, the area where most attention has been paid to trust-based authorisation. In Grid computing, bad experiences are (at present) more likely to be due to system failures and user incompetence than to deliberate attempts to cheat the system for personal gain. The policy for reacting to them must reflect this.

5 Concluding Remarks

To satisfy most people's intuition about trust relationships, we want the evaluated trust to follow the observed behaviour closely, but we do not want a single failure to cause a total breakdown in trust (implying in the present context a total refusal of authorisation). The type of aggressive malware attack which should cause such a refusal should be handled by more traditional IDS/IPSs, rather than the mechanisms described here. We see from the current experiments that this can be achieved with any of the models discussed here, provided that:

1. The parties whose mutual trust is being evaluated interact a sufficient number of times for them to build up a reasonable picture of one another's behaviour.

2. The parameters which describe how combined trust is evaluated (such as the relative weights of direct trust and reputation, the forgetting factor, the width of the experience function, etc.) are chosen suitably.

In general terms, discrete methods seem to offer a more rapid reaction to improved or degraded behaviour than methods based on Bayesian techniques or fuzzy logic. However, it is necessary to find the right parameter values, such as the weights w_r and w_d, in order to achieve trust evaluations which correspond to our intuitive requirements. It is not clear how the correct parameters can be found in a practical situation in a large distributed system, except by repeated experiment. The methods based on Bayesian techniques and fuzzy logic tended to be more "self-adjusting". However, they are also more conservative, in the sense of tending to preserve a previously held view until considerable evidence had been accumulated that this view was mistaken. Nevertheless, they demonstrated intuitively correct behaviour over a large range of parameter values, and we expect them to be much easier to apply in practice.

The results presented here are only a small part of an ongoing study, in which the effect of changing other parameters of the trust and reputation system, such as the rating system, have also been studied. We shall continue to experiment with the trust-based methods described here in a practical setting, to see whether the results obtained here can be exploited in real-life systems, where there are more elements of uncertainty than in a simulator. The simulator created for these experiments [4] is a general and flexible tool, which has also been used to experiment with other contexts where trust-based decisions have to be made, in particular for finding a just price for services in Grid-based computing systems.

Acknowledgments

This work has taken place within the framework of the Danish Center for Grid Computing, funded by the Danish Natural Sciences Research Council. The authors express their gratitude for this support.

References

[1] F. Azzedin and M. Maheswaran. Integrating trust into grid resource management systems. In *Proc. 2002 International Conference on Parallel Processing (ICPP'02)*, pages 47–54. IEEE Computer Society, 2002.

[2] F. Azzedin and M. Maheswaran. Towards trust-aware resource management in grid computing systems. In *ccGrid'02: Proc. 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 452–457. IEEE Computer Society, 2002.

[3] F. Azzedin and M. Maheswaran. A trust brokering system and its application to resource management in public-resource grids. In *Proc. 18th International Parallel and Distributed Processing Symposium (IPDPS'04)*. IEEE Computer Society, 2004.

[4] M. Brinkløv and R. Sharp. Trustsim: A simulator for trust relationships in grid systems. Web manual, URL: `http://www.imm.dtu.dk/~robin/grid/trustsim/userguide.pdf`, February 2006.

[5] V. Cahill et al. Using Trust for Secure Collaboration in Uncertain Environments. In *Pervasive Computing Magazine*, volume 2, pages 52–61. IEEE, 2003.

[6] G. Derbas, A. I. Kayssi, H. Artail, and A. Chehab. TRUMMAR – A trust model for mobile agent systems based on reputation. In *Proc. 2004 IEEE/ACS International Conference on Pervasive Services (ICPS'04)*, pages 113–120, 2004.

[7] T. Grandison and M. Sloman. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials*, 4(4), 2000.

[8] A. Jøsang and R. Ismail. The Beta reputation system. In *Proc. 15th Bled Conference on Electronic Commerce*, June 2002.

[9] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 2006. In the press.

[10] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *WWW '03: Proc. 12th International Conference on World Wide Web, Budapest*, pages 640–651. ACM Press, 2003.

[11] L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation for e-businesses. In *Proc. 35th Annual Hawaii International Conference on System Science*. IEEE Computer Society, 2002.

[12] T. J. Ross. *Fuzzy Logic with Engineering Applications*. McGraw-Hill, New York, 1995.

[13] S. Song and K. Hwang. Trusted grid computing with security assurance and resource optimization. In *Proc. 17th International Conference on Parallel and Distributed Computing Systems (PDCS-2004)*, Sept. 2004.

[14] S. Song, K. Hwang, and M. Macwan. Fuzzy trust integration for security enforcement in grid computing. In *Network and Parallel Computing*, volume 3222 of *Lecture Notes in Computer Science*, pages 9–21. Springer Verlag, Oct. 2004.

[15] L. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[16] M. Zarozinski. Free Fuzzy Logic Library. URL: `http://ffll.sourceforge.net`, 2003.