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, such as those based on the Grid paradigm. 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 computing systems.

1 Introduction

In distributed computer systems, the task of authorising a user to use a resource involves two separate considerations: Firstly, that the user and the resource can identify themselves to one another in a trustworthy manner, and secondly, that they trust one another to behave correctly and not to misuse the system (either by accident or on purpose). There are thus two types of trust involved, known as *identity trust* and *behavioural trust* respectively. Some classifications [6, 8] introduce further subdivisions of behavioural trust, but we shall not do so here.

In many systems today, the mechanisms of identification and behavioural trust are combined, so that a proof of identity enables the user or resource to be classified as belonging to a group of entities which are trusted to a certain degree – and therefore can be authorised to perform a certain set of operations. A typical example can be seen in many systems based on the Grid paradigm, where identification is proved by presentation of a certificate, and the identity is used to classify its owner as belonging to a particular Virtual Organisation (VO), whose members are all authorised to perform certain operations.

This is generally satisfactory in systems which are relatively static, in the sense that the set of users and resources is relatively constant over time, and can be trusted to behave nicely within the limits set by the system. This is a common situation in present-day Grid computing, 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, this situation is expected to change, however, 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 do not necessarily have a friendly relationship. This means that

it will no longer be realistically possible to set up pre-defined, static behavioural trust relationships, and new mechanisms for determining the degree of trust for authorisation purposes are needed.

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. The trust mechanisms considered are *incremental:* Good behaviour is rewarded by a higher degree of trust (leading for example to authorisation to do more things), and bad behaviour by a lower degree of trust. 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 short 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 incremental 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.

2 The Models Considered

In this study we assume that the identities of entities are undisputed, so in the remainder of this paper the unqualified term "trust" will be used 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 personal 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 third party ratings are combined to give the reputation, the extent to which old information is discarded as time passes, and so on. A very large number of models of trust have been described in the literature, and we do not have space for a complete review. Instead we concentrate 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:

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

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:

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

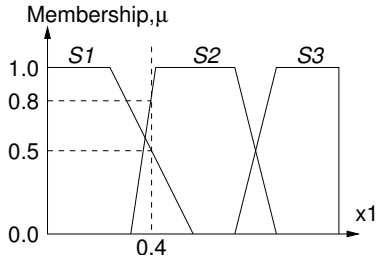


Figure 1: Fuzzy sets and membership functions

where $\mathcal{D}$ is the domain of entities. The function Υ is a *decay function*, expressing the rate at which trust depreciates, 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 the confidence that there is no collusion between z and y. Azzedin and Maheswaran have presented several variations on this basic model, including for example variable contexts and the use of brokers [3], and have evaluated their performance in the context of Grid systems.

The main alternative to such discrete models for combining direct trust and reputation is to use combination based on *fuzzy logic*. The rationale for this is that trust is a linguistic concept, which is only poorly described by simple numerical values, say between -1 (for 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, for example that a particular behaviour will give rise to a "moderately high" degree of trust. The basic idea is that for each behavioural parameter, a set of (possibly overlapping) *fuzzy sets* is defined in terms of so-called *membership functions*, which specify the degree of membership of the various sets for the possible values of the parameter [11, 14]. 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:

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

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

$$\mu_{Ok}(y) = \max[\min[\mu_{Sk}(x_1), \mu_{Tk}(x_2), \ldots]], \quad k = 1, 2, \ldots, r$$

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 for trust evaluation in Grid systems has been proposed by Hwang and Song [12, 13], although their approach had a different focus from ours, as they were interested in combining evaluations of *job success rate* and *self-defence capability* to find an overall trust value.

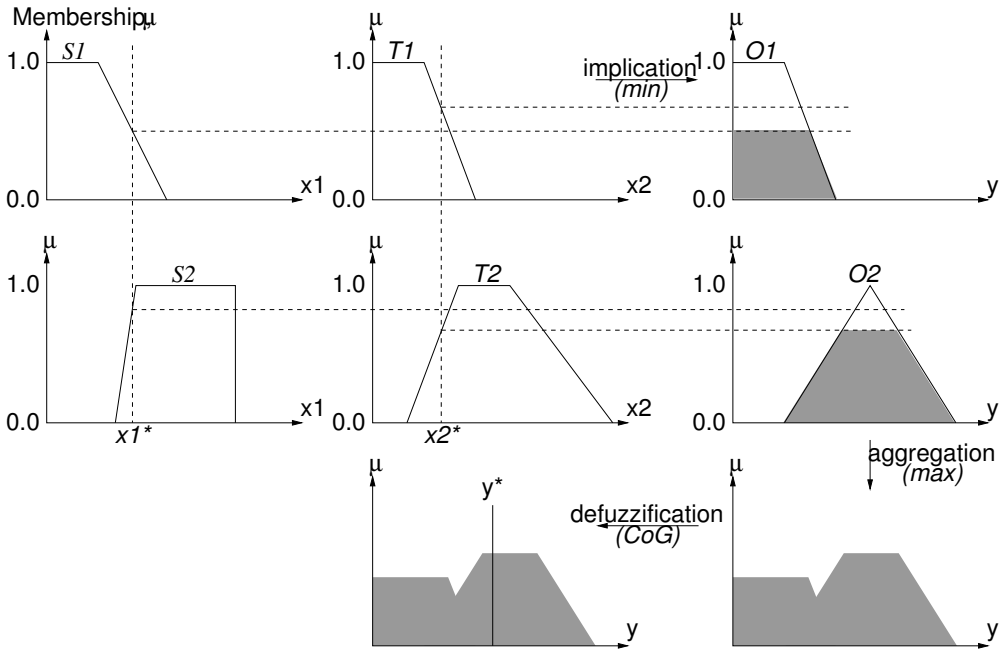


Figure 2: Fuzzy inference and defuzzification

2.2 Evaluating Reputation

As in the case of combining direct trust and reputation, methods proposed for evaluating reputation from the information provided by third parties largely fall into two classes: *discrete* methods and methods based on *Bayesian* statistics. The TRUMMAR system introduced by Derbas et al. [5] is a typical example of the use of discrete methods. The reputation of y from x's point of view is evaluated as:

$$\Omega(x,y) = w_o * \Omega(x,y)_{old} + w_n * \frac{\sum_i \Omega(x,i) * \Omega(i,y)}{\sum_i \Omega(x,i)} + \ldots$$

Again, w_o and w_n are constant weights. The summation includes all the "neighbours" of x who contribute information. If necessary, futher terms can be added, to give a more graduated rating from "close neighbours", "friends", "strangers" or other categories, and a decay of reputation with time can be included in the model. The model takes the point of view that x's trust in i influences x's opinion on the rating of y given by i, and combines the contributions from all sources in a simple linear manner. A similar approach is taken in Eigentrust [9] and other systems.

Bayesian methods evaluate a post-observation reputation value from the pre-observation value and the result of an observation, just as when posterior odds are evaluated in Bayesian statistics from prior odds and the result of an observation. Such methods have achieved some prominence as a result of the work of Jøsang and his co-workers in developing the *Beta reputation system* [7]. This was originally targeted towards e-commerce, but can be applied in other areas as well [8]. A similar general approach, also focussed on e-commerce, was used by Mui et al. [10]. The calculation of reputation makes use of the beta probability

density function, defined as the function f of two parameters α and β:

$$f(p \mid \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1}, \quad \text{where } 0 \leq p \leq 1 \text{ and } \alpha, \beta > 0$$

Here $\Gamma(x)$ is the usual gamma function, and we apply the restriction that the probability $p \neq 0$ 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 $s(x,y)$ represent the total amount of respectively positive and negative feedback about y provided by x. Sometimes it is preferable to provide a single feedback rating, say $v \in [-1,+1]$, instead of a pair of ("good","bad") ratings; v is related to (r,s) by $r = w_t \cdot (1+v)/2$ and $s = w_t \cdot (1-v)/2$ respectively, where w_t is a *transaction weight* reflecting the importance of the activity whose outcome is considered.

The above calculation assumes, as in most statistical approaches, that the behaviour of the systems is *stationary*, so our evaluation of the ratings can be based on all observations back to "the beginning of time". Often this is not the case, as the systems' behaviour may change with time. In fact, our main interest may lie in discovering exactly when a change of behaviour takes place. In the Beta system, this issue is effectively 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 remembered and used in the reputation calculation, while if $\lambda = 1$ everything is remembered. In practice, it is not necessary to keep all the old feedback values, as $r(x,y)$ can be worked out 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 similarly 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

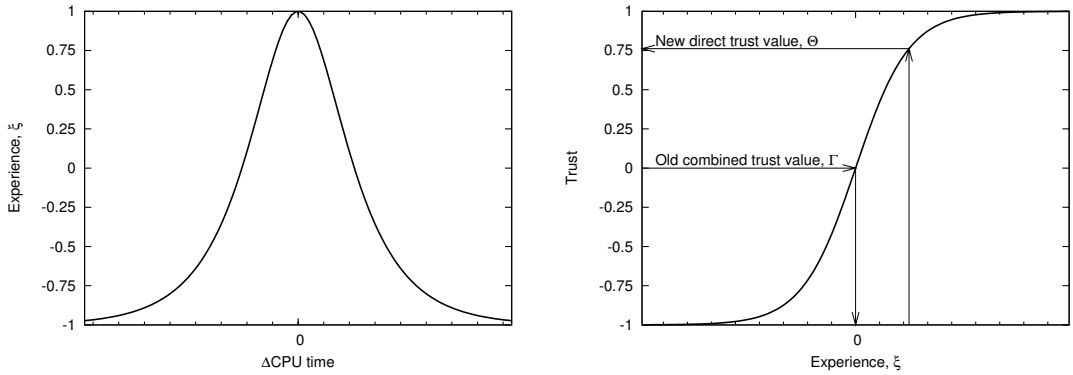
For the purpose of experimenting with the various models, we have created a discrete event-driven simulator [4] in C++ using the Free Fuzzy Logic Library [15] and the GNU scientific library. The simulations model a number of users and resources that interact by sending and receiving jobs. Job submission times are determined from a Poisson distribution with mean ν. When a job is submitted, a random user, i, and a random resource, j, are selected. If the user's trust in the resource exceeds the user's trust threshold, τ_i, the user submits a job to the resource, which will accept the job if its trust in the user exceeds its threshold, τ_j.

When the job finishes, the user and the resource each calculate a degree of satisfaction (the *experience*, ξ) for the interaction, based on the difference between the job parameters and what actually occurred. The value of ξ is then used to calculate a new direct trust value. Afterwards a small selection of the entities in the system are asked to give a rating for calculation of a new reputation value. A rating should express the entity's trust in the particular user (or resource). However, users and resources may lie by submitting inaccurate ratings – the worse an entity behaves, the more inaccurate it tends to be.

The reputation can be calculated using either of the reputation models described earlier. Finally, the direct trust and the reputation values are combined to one trust value using the discrete or the fuzzy logic

approach. When referring to the different methods for calculating trust, we use the notation *Discrete-Beta* to denote discrete combination of trust where the reputation is calculated using the Beta reputation system. The notation *Fuzzy-Discrete* denotes fuzzy logic combination of trust where the reputation is calculated using the discrete reputation system and similarly for *Discrete-Discrete* and *Fuzzy-Beta*.

In these experiments, the value of ξ was for simplicity evaluated just from differences in CPU time. For a given job, we assume that a user first requests an amount of CPU time from the resource, and the resource then allocates a certain amount of CPU time and inserts the job into the job queue. When the processing is done, we assume that the resource can determine if the user information about CPU time was correct and then return the job to the user. The user can likewise check whether the resource provided the CPU time that was requested.



(a) Interaction experience from Δ CPU time

(b) Calculating direct trust

Figure 3: Determining direct trust

This heuristic is illustrated in figure 3(a). When Δ CPU time is small, this will result in a high degree of satisfaction from the interaction and consequently a large value of ξ. From the resource's point of view it did not waste time on a deceiving user, and from the user's point of view it did not waste time on a resource that did not deliver what was promised. As Δ increases, ξ falls. From the user's point of view, the resource either did not complete the job or overcharged the job processing, while from the resource's point of view, the user either understated or overstated the processing time of the job. The calculation of the direct trust value is illustrated in Figure 3(b). The previous combined trust value is used to calculate the new direct trust value. The argument for using the old combined trust value is that it should be a more accurate indication of the current level of trust. As illustrated in the figure, a high trust value should be brought about quickly if the user or resource is behaving properly but it should take a long time to get a very high trust value. Likewise a low trust value should be given quickly with bad behaviour but a very low trust value should be based on many bad experiences.

The behaviour of an entity (user or resource) in the simulation is determined by two separate actions. Namely 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, this will result in a good experience ξ from the resource's point of view, as illustrated in Figure 3(a). If δ_u is large, it will result in a bad experience. Similarly, when a resource allocates CPU time to the job, it will in fact allocate a CPU time $t_a = t_r + \delta_s$. When δ_s is small, this will result in a good experience from the user's point of view, whereas when δ_s is large, the experience will be bad. The values of δ_u and δ_s are drawn from a Laplace distribution with width a_i, where the value of a_i for entity i determines that entity's behaviour. When a_i is small, then the deviation has a higher probability of being close to zero, corresponding to good behaviour, while when a_i is large, there is a higher probability that the deviation is (numerically) large, corresponding to bad behaviour. In a similar way, when a rating is returned, a deviation δ_r is added to the actual trust value. Again, δ_r is drawn from a Laplace distribution with width a_i, but the resulting deviation is in this case divided by 100 to ensure that the rating seems credible.

We have used the fuzzy logic membership functions shown in figure 4. The functions are based on Song and Hwang's functions from [12, 13], except that we have chosen to use trapezoidal curves, whereas they use sigmoid curves. Each of our fuzzy sets has the same area, and the same membership functions are used

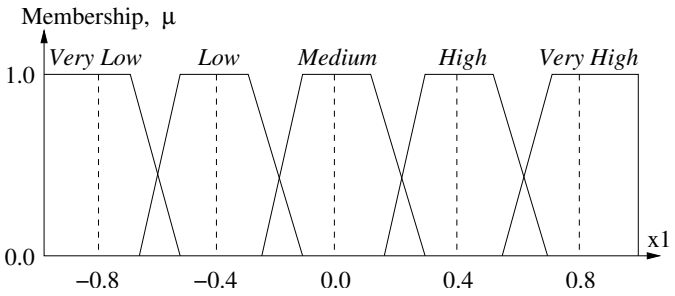


Figure 4: Fuzzy membership functions

for input (direct trust, Θ and reputation, Ω) and output (trust, Γ). The default fuzzy rule base will enforce a trust policy where direct trust and reputation are (as far as possible) considered to be equally significant, for example as shown in the following rules:

$$\textbf{if } (\Theta \in Medium \wedge \Omega \in Very\,High) \quad \textbf{then} \quad \Gamma \in High \qquad (1)$$
$$\textbf{if } (\Theta \in Low \wedge \Omega \in High) \quad \textbf{then} \quad \Gamma \in Medium \qquad (2)$$

When the two types of trust contain conflicting information, however, the fuzzy rule base will enforce a policy where the extremes are chosen. For example as illustrated in the following rules:

$$\textbf{if } (\Theta \in High \wedge \Omega \in Very\,High) \quad \textbf{then} \quad \Gamma \in Very\,High \qquad (3)$$
$$\textbf{if } (\Theta \in Very\,High \wedge \Omega \in High) \quad \textbf{then} \quad \Gamma \in Very\,High \qquad (4)$$

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 $\Gamma \in Very\,High$ then the maximum defuzzified value is 0.8 since this is the center of gravity of the $Very\,High$ set. Likewise when $\Gamma \in Very\,Low$, the minimum defuzzified value is -0.8 since that is the center of gravity of the $Very\,Low$ set. This limits the output trust Γ interval to $[-0.8 : 0.8]$. The trust scale is arbitrary, so for ease of comparison the Discrete combination method was also limited to this interval. Since the policy of the fuzzy rule base predominantly weights direct

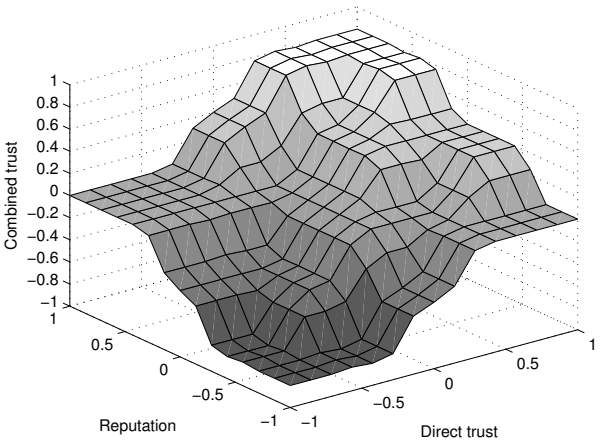


Figure 5: Control surface

Parameter	Value
Forgetting factor, λ	0.4
Transaction weight,w_t	6
$w_d,\ w_r$	0.5
w_0	0.0
w_n	1

Parameter	Value
Initial trust value	0.0
Mean of Poisson distribution, ν	25
a_i, entity with "good" behaviour	2
a_i, entity with "bad" behaviour	20
Trust threshold, τ_i	-1.0

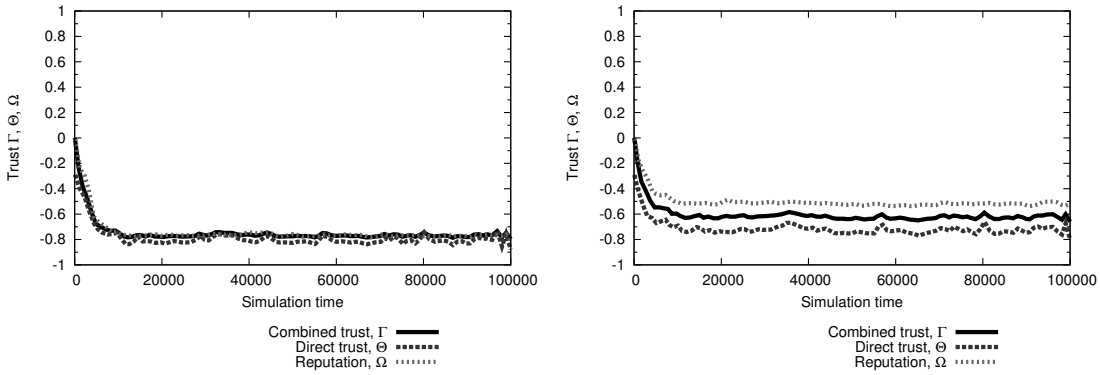
Table 1: Simulation parameters

trust and reputation equally, the Discrete combination method also weights the two types of trust equally. Relevant simulation parameters are shown in Table 1.

The experiments were performed for systems with a wide range of sizes, from small clusters with four users and four resources, up to large systems with 1000 users and resources. Each experiment has been 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 three other entities.

3.1 Detecting bad behaviour

In the first simulation scenario, a single resource exhibits potentially bad behaviour, while the remaining users and resources exhibit good behaviour. This does *not* mean that the "bad" resource will always misbehave. It merely indicates that there is a non-zero probability that the resource will allocate too much or too little CPU time. Occasionally the resource will allocate the exact amount of CPU time, and thus receive a high direct trust value. The experiment could naturally also have been performed with a user exhibiting potentially bad behaviour instead of a resource. A user's perception of the resource is illustrated in Figure 6.



(a) Discrete-Discrete

(b) Discrete-Beta

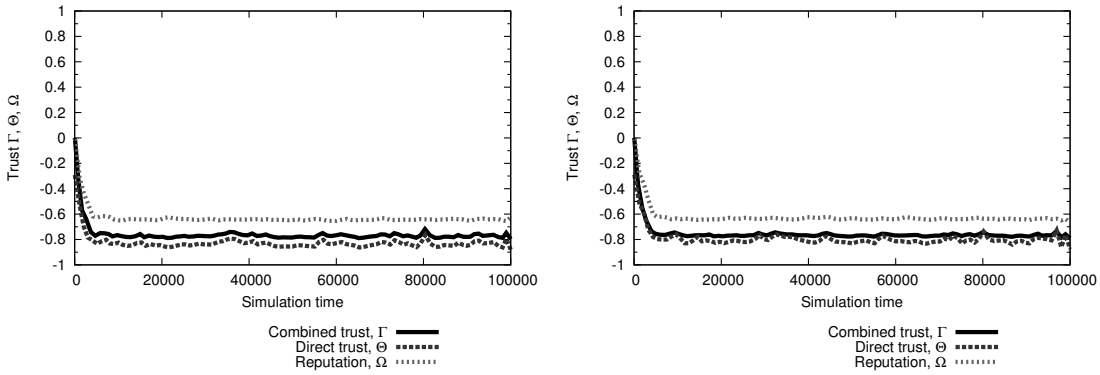
Figure 6: Bezier approximations of a user's trust in a badly behaving resource.

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

Table 2: Average number of events before a user's trust in the resource decreases below -0.5 when $w_d = w_r = 0.5$

As expected with the Discrete reputation system, the reputation in Figure 6(a) almost instantly decreases to the minimum value. The Beta reputation system in Figure 6(b) does not reach the low values that the Discrete reputation system does. In fact it appears that the reputation value does not decrease significantly below -0.5. To show the rate at which trust evolves, Table 2 lists the number of interactions ("events") needed to reach a trust value below -0.5. The slower progression of the reputation when using the Beta reputation system is very noticeable. To investigate this further, the experiment was repeated with $w_d = 0.8$ and $w_r = 0.2$ for the calculation of the combined trust value. Figure 7(a) illustrates that, when the weight for direct trust, w_d, is increased, then the combined trust and reputation decreases further. The change in w_d can be viewed as a change in security policy, so that the policy states that the direct trust will be $w_d/w_r = 4$ times more significant than the reputation. Figure 7(b) shows for comparison the same experiment when the direct trust and reputation are combined using the fuzzy logic method.

The security policy when using fuzzy logic will, when possible, weight the direct trust and the reputation equally. Put another way, when it is possible to select a fuzzy output set that lies exactly between the direct and reputation fuzzy sets, then that is done. When that is impossible, and the direct trust and reputation conflict, the rule base will enforce a security policy where trust value extremes are selected. This is illustrated



(a) Discrete-Beta where w_d=0.8 and w_r=0.2

(b) Fuzzy-Beta

Figure 7: Bezier approximations of a user's trust in a badly behaving resource (2)

in the following fuzzy rules.

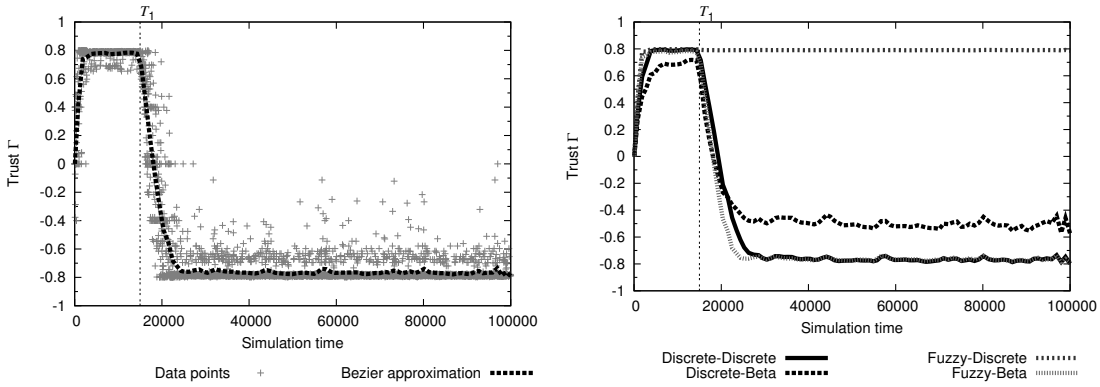
$$\textbf{if } (\Theta \in High \wedge \Omega \in Very\,High) \quad \textbf{then} \quad \Gamma \in Very\,High \quad (3)$$
$$\textbf{if } (\Theta \in Low \wedge \Omega \in Very\,Low) \quad \textbf{then} \quad \Gamma \in Very\,Low \quad (5)$$

When the direct trust has weight $w_d = 0.8$ and the reputation weight $w_r = 0.2$ for the Discrete combination method, then direct trust will have preference in all calculations. That might not be a reasonable decision in every scenario. One could naturally design Discrete w_d and w_r that depend on the actual value of Θ and Ω. However it might be conceptually difficult to determine a security policy based on the numerical weights for the two types of trust for every possible scenario. The linguistic concept of fuzzy logic makes designing the policy more straightforward.

3.2 Detecting changing behaviour 1

In a real world scenario it cannot be expected that an entity behaves either well or badly all the time. Here we investigate the effect of a sudden change in behaviour, where a resource changes from good to potentially bad behaviour at time T_1. Figure 8(a) contains the data points and the corresponding Bezier approximation for the combined trust when using the Fuzzy-Beta method for $T_1 = 15000$. Although there are fluctuations for the combined trust value, the Bezier approximation curve still captures the general progression of trust. Figure 8(b) contains Bezier approximation curves for the two combination methods and the two reputation systems. Again, only the combined trust is plotted, since in a real world scenario only the combined trust would be used to make a decision.

From Figure 8(b) it is noticeable that the Fuzzy logic method using the Discrete reputation system does not pick up on the change in behaviour. It arrives at the wrong conclusion since, in the eyes of the user, the resource exhibits good behaviour. The reason for this lies in the fact that when $T_1 = 15000$ then the reputation has the value 0.7 (or above). The fuzzy membership functions in Figure 4 show that the reputation,



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

(b) Bezier approximations

Figure 8: Change from good to bad behaviour at time $T_1 = 15000$

Ω, will then only have non-zero membership of the *Very High* set. Three fuzzy rules where $\Omega \in Very\,High$ are:

$$\textbf{if } (\Theta \in Very\,High \wedge \Omega \in Very\,High) \quad \textbf{then} \quad \Gamma \in Very\,High \quad (6)$$
$$\textbf{if } (\Theta \in High \wedge \Omega \in Very\,High) \quad \textbf{then} \quad \Gamma \in Very\,High \quad (3)$$
$$\textbf{if } (\Theta \in Medium \wedge \Omega \in Very\,High) \quad \textbf{then} \quad \Gamma \in High \quad (7)$$

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

To avoid this undesired behaviour, more emphasis could be put on the direct trust value. Specifically rule (3) above must be changed into:

$$\textbf{if } (\Theta \in High \wedge \Omega \in Very\,High) \quad \textbf{then} \quad \Gamma \in High \quad (3')$$

This change implies that when a user/resource receives ratings that lead to a reputation value conflicting with the direct trust value, then direct trust will influence the combined trust value to a higher degree. This can be understood as a change in security policy, whose effect becomes noticeable when a user or a resource has calculated a $Very\,High$ reputation value but has only experienced $High$ direct trust. Figure 9 illustrates the effect of changing the rule. In this case the change in behaviour is correctly detected.

In short, when using a fast reputation system, such as the Discrete system, then extra consideration must go into the fuzzy rule base design. Otherwise it may result in artificially high ratings when the experienced behaviour changes, and the rule base may enforce an undesired security policy. The experiment in Section 3.3 will further illustrate this.

For the three remaining models, it can be concluded that each of them arrives at the correct conclusion, namely that the behaviour of the resource changes from good to bad. With the Discrete method and the

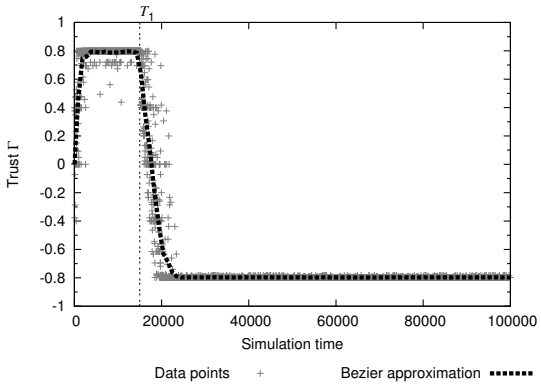


Figure 9: Fuzzy-Discrete

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

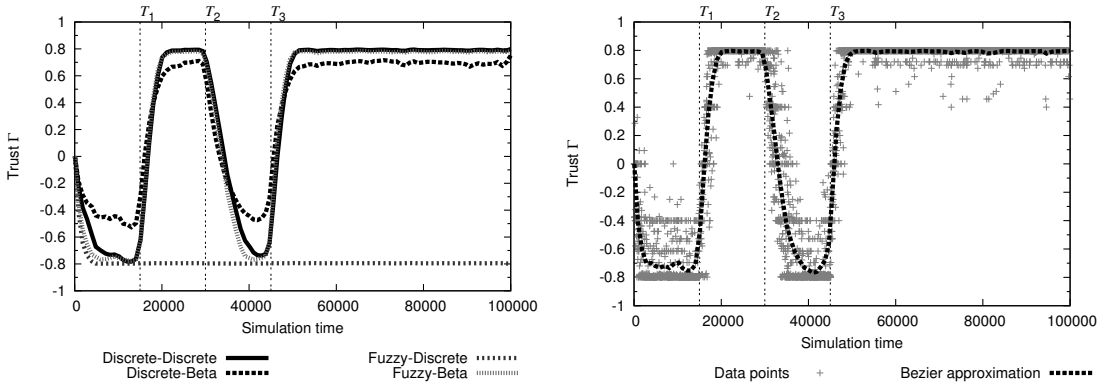
Table 3: Average number of events after T_1 before a user's trust in the resource is below -0.5

Discrete reputation system, the values initially increase rapidly to the maximum. After the simulation time reaches T_1, the trust also quickly decreases to the minimum values.

Table 3 shows the number of events that are needed for the trust values to decrease to -0.5 or below. Note that the Fuzzy-Discrete combination uses the alternative security policy (rule $(3')$) described above. It is particularly noticeable how much slower the reputation changes with the Discrete method using the Beta reputation system. The Fuzzy method using the Beta reputation system, on the other hand, is comparable with the Discrete. Interestingly it can be seen that fuzzy logic combined with the Beta reputation system is slightly faster than the Discrete method with the Discrete reputation system. Or put in another way the effect of the Beta reputation system is not apparent. The reason for this is that until time T_1 the number of observations is rather limited, which means that the knowledge in the Beta reputation is fairly limited.

3.3 Detecting changing behaviour 2

The next experiment reflects a more extreme turn of events. Here the behaviour of the resource will change three times. Initially the behaviour will be (potentially) bad. When the simulation time reaches T_1 (here 15000) it will start to behave correctly, at T_2 $(= 30000)$ it will potentially begin to misbehave again, and finally at T_3 $(= 45000)$ it will behave correctly again. The Bezier approximation curves for the two combination methods using the two reputation systems are illustrated in Figure 10(a). As in the previous experiment, only a single user's view of the combined trust values for the resource has been plotted. It can again be seen that the Fuzzy logic method using the Discrete reputation system does not pick up on the change in



(a) Combined trust progression

(b) Fuzzy-Discrete. Alternative rule base

Figure 10: When behaviour varies

Method - reputation system	Trust, Γ	Direct trust, Θ	Reputation, 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	11.05

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

behaviour. To avoid this, the policy must be changed so that rules $(3')$ and $(5')$ below are used instead of (3) and (5), placing a higher emphasis on direct trust when $\Omega \in Very\,High$ and when $\Omega \in Very\,Low$:

$$\begin{array}{lll} \textbf{if } (\Theta \in High \wedge \Omega \in Very\,High) & \textbf{then} & \Gamma \in High \quad (3') \\ \textbf{if } (\Theta \in Low \wedge \Omega \in Very\,Low) & \textbf{then} & \Gamma \in Low \quad (5') \end{array}$$

Figure 10(b) demonstrates the Fuzzy-Discrete approach using the alternative rule base. Again this experiment is intended to illustrate that, when a fast reputation system is used in connection with fuzzy logic, then special care should be put into the design of the rule base. Otherwise it may enforce an undesired security policy.

The remaining methods and reputation system in Figure 10(a) illustrate the expected progression of trust. Table 4 shows the number of events needed for the trust values to increase above +0.5 after the simulation time has passed T_3. Note that Fuzzy-Discrete uses the alternative rule base, meaning that it uses a different security policy. We expected the Beta reputation system to require a significantly larger amount of 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 will be investigated in the next experiment.

3.4 Influence of the forgetting factor

In the next experiment we considered the same scenario, but adjusted the value of the forgetting factor λ to be closer to 1, so that reputation values going back further in time would be included in the calculation. This is only relevant for the Beta reputation system. Figure 11 illustrates the results in this scenario with different values for λ. The random seeds are the same for this and the previous experiment.

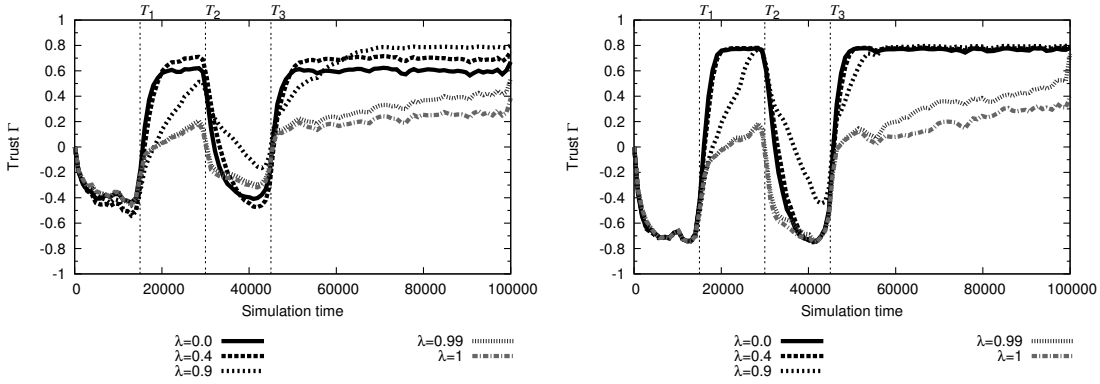
As λ increases, and more and more historical information about the reputation ratings is included in the trust calculation, we observe two effects. As long as λ is still relatively small (in fact $\lesssim 0.6$), the combined trust values, as expected, react more slowly to changes in behaviour. When 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. From a statistical point of view, we are including observations from periods where the behaviour is not *stationary*, and the results cannot be relied on. Table 5 shows the number of events needed for the

Method - reputation system	$\lambda = 0.0$	$\lambda = 0.4$	$\lambda = 0.9$	$\lambda = 0.99$	$\lambda = 1$
Discrete-Beta	7.15	9.1	17.15	105.36	∞
Fuzzy-Beta*	6.75	8.25	14.85	118	∞

Table 5: Average number of events after T_3 before a user's combined trust in the resource is above +0.5. The Fuzzy-Beta combination uses the fuzzy rules $(3')$ and $(5')$.

combined trust to be above +0.5 after the time has passed T_3 for different forgetting factors. It is clearly noticeable that an increasing forgetting factor causes a slower progression of the reputation.

It is plainly important to choose a suitable value for λ when using the Beta reputation method. With λ close to 0, historical knowledge is forgotten immediately, giving a rapid reaction to changes in behaviour,



(a) Discrete-Beta

(b) Fuzzy-Beta

Figure 11: Different forgetting factors

and making the Beta method 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 reasonably rapid response to changing behaviour. Values of λ close to 1 require a large number of observations of "new" behaviour before the trust rating will change signficantly, and there is a serious risk that observations from different epochs of behaviour will be combined, rendering the trust evaluation statistically dubious.

3.5 Larger scale experiments

The previous experiments were performed using only a small number of simulated users and resources (in fact, 4 of each). The purpose of the next experiments was to investigate 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 3.3. However the average number of interactions between a user and a resource is varied and 100 users interact with 100 resources. Again a single resource varies its behaviour. Initially the behaviour of the resource will be (potentially) bad. When the time reaches T_1 (here $7\,500\,000$) it will start to behave correctly, when the time reaches T_2 $(= 15\,000\,000)$ it will potentially misbehave again, and finally at T_3 $(= 22\,500\,000)$ it will start to behave correctly again. As previously, only three ratings are used to calculate the reputation. To achieve rapid reputation progression, the users and resources are able to provide references to other entities with which they have recently had contact. This means that the ratings which a user/resource receives are fairly updated. A single user's perception of the resource is shown in Figure 12.

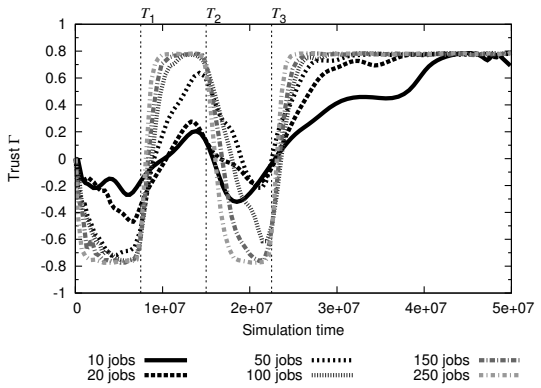


Figure 12: Fuzzy-Beta. Varying the number of interactions

Only the results for Fuzzy combination using the Beta reputation system are shown. The remaining trust calculation methods showed similar results. Different numbers of interactions between the user and the resource result in different progressions of trust. In particular, we see that the span of trust values is reduced when the number of interactions is reduced. Put in another way, when the number of interactions is reduced, then the number of "good" or "bad" experiences is naturally also reduced, leading to fewer trust value extremes. Changing the number of interactions does however not change the general trust observation. 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 the progression obviously reflects the true situation more closely after more interactions.

The second experiment increases the number of users and resources even further. Again a single resource varies its behaviour and Figure 13 contains a user's point of view for different system sizes. Only the

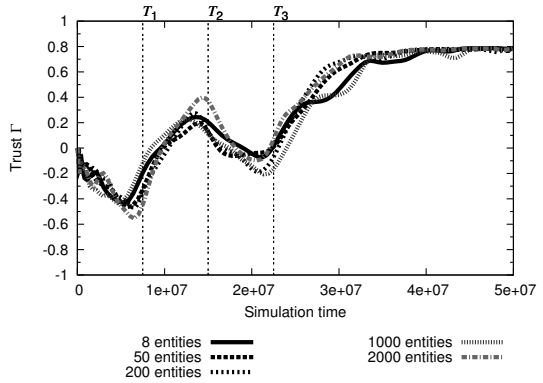


Figure 13: Fuzzy-Beta. Different sizes of system.

In each experiment, 50% of the entities are users and 50% are resources.

results for Fuzzy combination using the Beta reputation system are shown. Once again, the remaining trust calculation methods showed similar results. It is directly noticeable in the figure that the progression of trust is not the same, but they do share similarities regardless of the number of users and resources present. Most importantly they all arrive at the same conclusion. Namely that the behaviour of the resource varies before ending with good behaviour. In this experiment, each user interacted with each resource approximately 20 times. This means that in the grid scenario with 8 entities (4 users and 4 resources) around 4000 jobs are simulated. When the system consisted of 1000 users and 1000 resources, approximately 20 million jobs were simulated.

Table 6 further illustrates that the progression of trust is similar, regardless of the number of users and resources present. Although a significantly larger number of jobs is processed in the larger scale experiments, this does not change the observed progression of trust. Approximately the same number of events are needed for the trust values to increase above +0.5 after the time has reached T_3. Since the behaviour of a

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 $T_3 = 22\,500\,000$ before a user's trust in the resource is above +0.5

user or a resource depend on the random deviations δ, and since new δ values are used for each interaction, then the behaviour of each user and each resource can not be considered similar. However, since the plots in Figure 13 show strong similarities and the average numbers of events in Table 6 are very comparable, it seems fair to conclude that the trust models are able to detect the correct progression of trust, regardless of the size of the system.

4 Including Trust in a Grid Environment

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

1. Evaluation of trust must be efficient with respect to use of storage space, CPU time and network bandwidth.
2. It must be possible to calculate a meaningful value for the experience, ξ.

This second requirement implies a need for the design of appropriate security policies that depend on trust. For example fuzzy logic membership functions, rule bases and the significance of an experience ξ in relation to the current trust value.

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. (Of course, on a given node, it is only necessary to store the slice of this information which refers to "the others".) The factor of proportionality depends slightly on the method chosen – for example for the Beta reputation method, two values (r and s) are stored, whereas the Discrete reputation method only needs one.

With respect to CPU time, Discrete combination of direct trust and reputation only requires a simple arithmetic calculation, whereas the use of fuzzy logic is much more demanding, due to the need for fuzzy inference and defuzzification. In particular, a choice of Gaussian membership functions combined with the CoG defuzzification method may lead to poor performance if the fuzzy logic implementation is not optimized for this. During our experiments we did however not notice any performance issues when trapezoidal membership functions were used.

Network performance is mainly affected by the submission and retrieval of ratings. In our experiments, we have calculated the reputation from three ratings, regardless of the number of users and resources in the Grid. (This in fact gave an accurate progression of reputation when a user or a resource was allowed to provide references to other entities.) Network activity is proportional to the square of the number of ratings exchanged each time, so in practice a compromise between network performance and reputation accuracy must be chosen. Monitoring a real world Grid environment using trust would provide valuable information on this subject.

The second requirement relates to how the experience ξ is evaluated and interpreted. In the experiments reported here, we evaluated ξ 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 of course the resources needed to store and calculate this information would be significantly larger than in the scenarios shown here.

The significance of a particular (good or bad) experience depends, as we have seen, on the policy which is in operation, and which ultimately determines the degree of reward or punishment which are appropriate to this experience. This is a "political" question within the community concerned, related to the acceptance of risk (or the degree of paranoia), and the extent to which a rapid 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 the various Discrete weighting factors, depending on the model chosen. In this respect, Grid computing is somewhat different from e-commerce, which is the area where most attention has been paid to trust-based authorisation systems. In Grid computing, bad experiences are (at least for the time being) as likely to be due to system failures and user incompetence, as they are to deliberate attempts to cheat the system for the sake of personal gain. The policy for reacting to bad experiences must reflect this.

5 Concluding Remarks

To satisfy most people's intuition about trust relationships, we want the evaluated trust to follow the observed and/or reported behaviour closely, but not so closely that a single failure or misbehaviour will cause a total breakdown in trust (which in the present context would imply a total refusal of authorisation). 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 the combined trust is evaluated (such as the relative weights of direct trust and reputation, the forgetting factor, the width of the experience function and so on) are chosen suitably.

In general terms, discrete methods seem to offer the potential of a more rapid reaction to improved or degraded behaviour than methods based on Bayesian techniques or fuzzy logic. However, it is more difficult to find the exact parameter values which are needed, in order for this potential to be realised. If the parameters are chosen badly, discrete methods appear to give trust evaluations which do not correspond to our intuitive requirements. It is not at all clear how the "correct" parameters should be found in a practical situation in a large distributed system. The methods based on Bayesian techniques and fuzzy logic tended to be more conservative, in the sense that they tended to preserve a previously held view of the opposite party until considerable evidence had been accumulated that this view was mistaken. However, they demonstrated intuitively correct behaviour over a large range of parameter values. We must therefore expect that they are much easier to apply in a practical situation.

We intend to 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 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 wish to express their gratitude for this support.

References

[1] Farag Azzedin and Muthucumaru 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] Farag Azzedin and Muthucumaru 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] Farag Azzedin and Muthucumaru 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. 10 pages.

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

[5] Ghada Derbas, Ayman I. Kayssi, Hassan Artail, and Ali 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.

[6] Tyrone Grandison and Morris Sloman. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials*, 4(4), 2000.

[7] Audun Jøsang and Roslan Ismail. The Beta reputation system. In *Proc. 15th Bled Conference on Electronic Commerce*, June 2002.

[8] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 2006. In the press.

[9] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *WWW '03: Proc. 12th International Conference on World Wide Web*, pages 640–651, New York, NY, USA, 2003. ACM Press.

[10] Lik Mui, Mojdeh Mohtashemi, and Ari 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. 9 pages.

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

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

[13] Shanshan Song, Kai Hwang, and Mikin 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, October 2004.

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

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