Comparison of the Beta and the Hidden Markov Models of Trust in Dynamic Environments

Marie E.G. Moe, Bjarne E. Helvik, and Svein J. Knapskog

Centre for Quantifiable Quality of Service in Communication Systems,
Norwegian University of Science and Technology,
O.S. Bragstads plass 2E, N-7491 Trondheim, Norway
{marieeli,bjarne,knapskog}@q2s.ntnu.no

Abstract. Computational trust and reputation models are used to aid the decision-making process in complex dynamic environments, where we are unable to obtain perfect information about the interaction partners. In this paper we present a comparison of our proposed hidden Markov trust model to the Beta reputation system. The hidden Markov trust model takes the time between observations into account, it also distinguishes between system states and uses methods previously applied to intrusion detection for the prediction of which state an agent is in. We show that the hidden Markov trust model performs better when it comes to the detection of changes in behavior of agents, due to its larger richness in model features. This means that our trust model may be more realistic in dynamic environments. However, the increased model complexity also leads to bigger challenges in estimating parameter values for the model. We also show that the hidden Markov trust model can be parameterized so that it responds similarly to the Beta reputation system.

1 Introduction

Trust is a fundamental part of social and commercial relationships, both in the real-life and the virtual world. Complex dynamic environments, like the Internet, makes it extremely hard to obtain perfect information about potential interaction partners. In e-commerce and other electronic transactions and services, where the assets of interaction partners might be at risk, trust mechanisms may facilitate the decision-making process and lower the risk. Since trust management can be assumed to decrease risk, it can also be assumed that it will increase security and can be considered as a *soft security* mechanism [15]. Soft security accepts the fact that it is possible to circumvent the implemented security mechanisms, given enough time, effort and money. Since we might have users with malicious intentions in a system, the challenge is to detect them and find a way to monitor their behavior and possibly influence their actions, in order to prevent them from causing any harm. Trust management serves this purpose by evaluating the trustworthiness of users and offering different service levels to users based on a trust policy. If services are denied to untrusted users, an incentive for users not to misbehave, is created.

Computational trust and reputation models seek to quantify trust as a value derived from previous direct experiences and/or second-hand information, such as recommendations, and suggest mathematical and logical expressions for how to combine several

E. Ferrari et al. (Eds.): TM 2009, IFIP AICT 300, pp. 283–297, 2009.

opinions about trustworthiness into reputation values. Such models are clearly needed in the virtual world where non-human agents are making trust-based decisions. But also when the human end-user is making the decisions, such calculated trust values can be very useful as decision support. For this reason a number of different trust models have been proposed. The modeling complexity varies, ranging from very simple eBay-like models to more sophisticated models based on probability theory, e.g. the Bayesian trust and reputation models [12,7,2,13,6].

In this paper we will present a comparison of our previously proposed hidden Markov trust model [11] to a binomial Bayesian reputation system [7]. The comparison is done with the help of simulations of trust scenarios. The objectives of this paper is to discuss probabilistic measurement of trust, outline the models and compare their performance in a dynamic environment where the (un)trusted objects may change behavior. We show that the hidden Markov trust model performs better when it comes to the detection of changes in behavior of agents, this means that our trust model may be more realistic for dynamic environments. We also show that the hidden Markov trust model can be parameterized so that it responds similarly to the Bayesian reputation system.

The remainder of this paper is organized as follows. Section 2 discusses the challenges related to modeling dynamic trust and how the different models can be evaluated. Section 3 gives a brief introduction to Bayesian trust models, in particular the Beta reputation system, Section 4 discusses the hidden Markov trust model, the simulation results are presented in Section 5, and Section 6 discusses the simulation results and concludes the paper.

2 The Dynamic Trust Modeling Problem

Since trust and reputation are active fields of ongoing research, numerous different models for quantification and evaluation of trust have been proposed. A review on some of these computational trust models can be found in [17]. However, there seems to be no single agreed upon model that can be used for benchmarking and comparison of the different trust and reputation algorithms.

Reputation models used for electronic commerce are often based on very simple mathematical formulas for combining opinions. One example is the reputation system implemented in eBay, where a feedback score is calculated as a sum of ratings that can be either positive, corresponding to a value of $+1$, negative with a value -1, or neutral with 0 value. A survey of trust and reputation systems that are currently used in online services can be found in [8].

In [4], several desirable qualities of reputation systems are listed. According to the authors a reputation system should be efficient, robust against attacks, easily understandable and verifiable. It should also be *weighted toward current behavior*, meaning that it responds quickly to changes in behavior so that an entity which has performed well consistently over a long time but then suddenly changes its behavior will be detected and maybe no longer trusted. This feature is missing in many trust and reputation models as trust is modeled as a static property, not taking the time component into consideration. For some applications of reputation and trust the time dependency and response to dynamic behavior are very important, as the behavior of agents could

be assumed to be highly dynamic. One example of such an application is when trust metrics are used in ad hoc routing protocols to counter malicious nodes, see for instance [9,3,2,1]. The common approach is that every node in the network monitors its neighbors and measures the frequency of packet dropping, misrouting and other potentially malicious behavior, and keeps a trustworthiness rating or reputation value recorded for all other nodes based on these observations. The underlying routing protocol is then modified with a trust component which selects routing paths and makes routing decisions based on the reputation values.

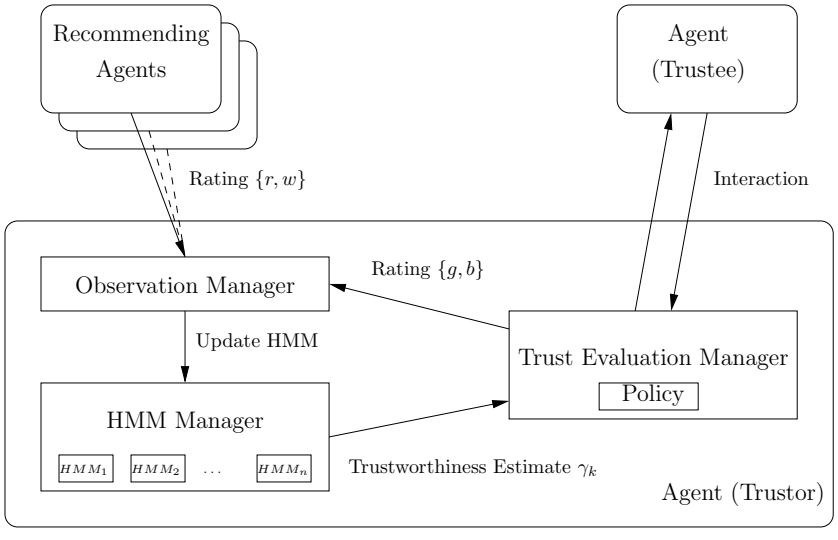

Fig. 1. The architecture of a reputation system using hidden Markov trust modeling

The computational trust algorithm used to calculate the reputation value varies. In [3], a simple eBay-like scheme is used, where reputation is a sum of recommendations of $+1$ whenever a packet reaches its destination, and -1 if a packet is dropped, a node is considered untrusted if its reputation value falls below a certain threshold. In [2] a more sophisticated scheme, based on a Bayesian reputation system, is used. A trust-based ad hoc routing protocol based on hidden Markov modeling of trust was proposed by the authors in [10]. The architecture of the trust component used in this approach, presented as a more general decentralized reputation system, is illustrated in Figure 1. Every agent in the system keeps and updates hidden Markov models (HMMs), that are modeling the trust state of all the other agents. Before an agent (trustor) initiates an interaction with another agent (trustee), it looks up the trustworthiness value derived from the HMM belonging to the trustee. The trustor then decides according to a policy whether or not to interact with the trustee. After an interaction, the HMM belonging to the trustee is updated with a rating, good g, or bad b, based on the outcome of the interaction. The HMMs are also updated with observations in the form of ratings from other agents in the system, so called second-hand opinions, which are either in the form

of recommendations r, or warnings w. In this study we do not include trust transitivity between different contexts, for the simplicity of the comparison, so we assume that the HMMs are only updated with observations related to the same context. We also do not consider chains of recommendations.

With this paper we would like to compare our hidden Markov modeling of trust to the Bayesian approach, in particular with regard to performance of the modeling of the dynamic aspect of trust, since this is a very important feature for applications in dynamic networking environments. A quantitative approach to comparing trust models can be found in [13]. In this paper it is proposed to use the information theoretical measure *relative entropy*. However, this approach is only applicable to probabilistic trust models that share the same fundamental assumptions about the underlying probability distributions. In cases where a direct mathematical comparison of models is difficult, comparison by the help of simulations seems to be the most viable approach. Different trust scenarios can be simulated in order to see which trust and reputation system performs best with regard to reliability of the calculated values under various hostile agent strategies.

3 Bayesian Trust Modeling

Bayesian trust models, for calculating reputation scores from ratings, are based on the assumption that the behavior of an agent can be described according to a probability distribution. The trust value is a function of the expected value of the probability distribution, which gets updated with every new rating received according to Bayes' Theorem. Binomial Bayesian reputation systems, where ratings can be expressed by two values, *good* or *bad*, are modeled with the Beta probability density function [12,7,2]. Multinomial Bayesian reputation systems, that allow for ratings with graded levels, are modeled with the Dirichlet probability density function [13,6]. In this paper we will focus on the binomial case, and evaluate the performance of our proposed trust model compared to the Beta reputation system proposed by Jøsang et al. [7].

3.1 The Beta Reputation System

The Beta reputation system models the reputation formation for a trustor as a sequence of observations, where each observation is the outcome of the rating done by a trustee, based on the outcome of an interaction. A reputation centre collects ratings from all the agents, and updates each agent's reputation score.

The underlying mathematical model of the Beta reputation system considers the ratings as a sequence of trials with binomial outcomes, for each trial there is a probability p of getting a *good* rating (recommendation) and a probability $(1-p)$ of getting a *bad* rating (warning). The parameter p belonging to a trustor is initially unknown, so due to lack of information it is assumed that it is drawn from a uniform distribution on $[0,1]$. As ratings concerning this trustor start to arrive, there is more information available and we can update the distribution of p. In accordance with Bayesian inference we have a *prior* hypothesis X about the outcome of a trial, which is updated *a posteriori* to the actual outcome Y in accordance with Bayes' Theorem

$$P(X \mid Y) = \frac{P(X)P(Y \mid X)}{P(Y)}. \tag{1}$$

The Beta distribution

$$Beta(\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1} \tag{2}$$

is a *conjugate prior* for binomial trials (Bernoulli process). This means that if we assume that the prior X hypothesis is described by $Beta(\alpha,\beta)$, and Y is a sequence of ratings, out of which r is the number of good ratings (recommendations) and w is the number of bad ratings (warnings), then the posterior $P(X \mid Y)$ is also described by a Beta distribution $Beta(\alpha+r,\beta+w)$. The initial prior is given by $Beta(1,1)$, which corresponds to the uniform distribution on $[0,1]$. The reputation value is given as a function of the expectation value of the Beta distribution $E(p) = \alpha/(\alpha+\beta)$, for the posterior hypothesis the expectation is found by setting $\alpha = r+1$ and $\beta = w+1$. This results in a very simple calculation of the probability expectation value. Let (r_k, w_k) denote the ratings received at iteration step k, we then get the following recursion for deriving the Beta parameters:

$$\alpha_k = \alpha_{k-1} + r_k, \quad \beta_k = \beta_{k-1} + w_k, \quad \alpha_0 = \beta_0 = 1. \tag{3}$$

For finding the probability expectation value at iteration step k we get:

$$E(p_k) = \frac{\alpha_k}{\alpha_k + \beta_k}. \tag{4}$$

The probability expectation value given in Equation 4 gives a reputation rating in the range $[0,1]$, where the value 0.5 represents a neutral reputation value. To make the reputation model more realistic, several modifications to the calculation of the reputation value are introduced. These variations include *discounting* of ratings based on the reputation of the agent providing the rating, *forgetting* old ratings by giving old ratings less weight than more recent ratings, and *weighting* of ratings according to the value of the rated transaction.

3.2 Evaluation of the Beta Model

The Beta reputation system without forgetting factor is efficient, easily understandable and verifiable, but it is not weighted toward current behavior. This is due to the underlying Bayesian framework, which assumes that the behavior of agents can be approximated by a *fixed* probability distribution. Since agents may change behavior over time, this static modeling is not realistic. The forgetting factor $0 \leq \phi \leq 1$ was introduced in [7] to overcome this problem. It is used to scale the parameters (α,β) in every update of the Beta distribution, so that the we get

$$\alpha_k^* = \alpha_{k-1}^* \phi + r_k, \quad \beta_k^* = \beta_{k-1}^* \phi + w_k, \quad \alpha_0^* = \beta_0^* = 1. \tag{5}$$

A forgetting factor $\phi = 1$ means that all ratings are weighted equally, and nothing is forgotten, with $\phi = 0$ only the last rating is remembered. In Figure 2 we can see how

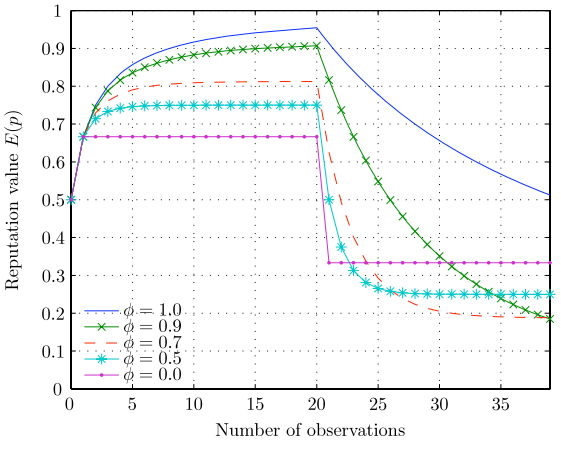

Fig. 2. The Beta model with different forgetting factors, the observations are 20 good ratings followed by 20 bad ratings

the Beta model responds to a sequence with 20 good ratings followed by 20 bad ratings, with different forgetting factors.

As noted by the authors of [13], the forgetting factor is a form of exponential decay on the parameters of the Beta model giving an effective bias towards newer information, but it is unclear if this fading mechanism is really modeling dynamic behavior of the agents. If agents were likely to change their behavior in such a way that the probability p of getting good ratings slowly increases or decreases, this fading of parameters seems like a good modeling approach. However, if we consider a disruptive agent that follows a strategy where it behaves good for a certain amount of time, building up a good reputation value, and then suddenly starts to misbehave taking advantage of its reputation, this slowly adapting model might not be good enough.

Another problem with the Beta model is the lack of time component. The reputation formation is only depending on the number of ratings, without taking the time between ratings into account. If we assume that ratings are not received at regular intervals, the claim that the forgetting factor takes care of adjusting the model towards *new* information may not be valid anymore. A simple way of rectifying this is by introducing a time stamp on the ratings, like suggested in [19].

4 The Hidden Markov Trust Model

The hidden Markov trust model takes the time between observations into account, it also distinguishes between system *states* and uses methods previously applied to intrusion prevention [5] for the prediction of which state an agent is in. The hidden Markov trust model was originally proposed by the authors as a component in a trust-based ad hoc routing protocol [10]. It was further developed with a parameter learning component for multiagent environments [11].

4.1 Hidden Markov Modeling

A hidden Markov model (HMM) consists of a finite set of N hidden states $S = \{s_1, \ldots, s_N\}$ with an associated probability distribution. The state of the monitored agent is described by a discrete time Markov chain $\boldsymbol{x}_k = x_1, x_2, \ldots$ where $x_k \in S$ is the possibly hidden state of the agent at sampling instant k. $\boldsymbol{P}_k = \{p_{ij}^k\}$ is the set of state transition probabilities, $p_{ij}^k = P(x_{k+1} = s_j \mid x_k = s_i)$, $1 \leq i, j \leq N$, where x_k is the current state of the system. $\pi = \{\pi_i\}$ is the initial state distribution, where $\pi_i = P(x_1 = s_i)$, $1 \leq i \leq N$. The output from the agent ratings is classified by the set of observation symbols $V = \{v_1, \ldots, v_M\}$. Let $\boldsymbol{y}_k = y_1, y_2 \ldots$ denote the sequence of observations, where $y_k \in V$ is the observation made at sampling instant k. The HMM consists of two stochastic processes; the hidden process $\boldsymbol{x}_k$, and the observable process $\boldsymbol{y}_k$ that depends on $\boldsymbol{x}_k$. The relation between $\boldsymbol{x}_k$ and $\boldsymbol{y}_k$ is described by the probability distribution matrix $\boldsymbol{B} = \{b_j(m)\}$, where $b_j(m) = P(y_k = v_m \mid x_k = s_j)$, for $1 \leq j \leq N$, $1 \leq m \leq M$. See for instance [14] for a more extensive introduction to HMMs.

In the hidden Markov trust model considered in this paper, we choose to use two hidden states $\{trusted, untrusted\}$, and four observation symbols $\{g, b, r, w\}$, corresponding to the observations *good, bad, recommendation* and *warning*. The reason for choosing two states is to make it easier to compare the model with the binomial Bayesian model. A comparison of our model with more states and more observation symbols to a multinomial Bayesian model would be an interesting topic for our future work. An agent is in an untrusted state if it has been behaving in a malicious way in previous interactions, it is in a trusted state if it has shown good behavior. We model trust as a dynamic variable, changing with time. This allows us to capture the behavioral characteristics of agents that are behaving good for a certain time, but then suddenly start misbehaving. Since an agent's behavior can be changing with time it is not necessarily the case that an agent is in the same state as it were at the last encounter. An agent can only do its best guessing about the trustworthiness state of an other agent based on its own previous direct experiences, which were either *good* or *bad*, and *recommendations* or *warnings* from other agents in the system. This means that the system state is hidden, and hence we use the HMM approach.

We consider a decentralized reputation system where each agent updates its own trust value for the other agents based on its own direct experiences, and from feedback in the form of ratings communicated from other agents in the multiagent system. We model the agent interaction as a stochastic process. This means that we assume that there is a random time interval between each agent interaction and that the behavior of an agent is only dependent on its current state. When using a Markov model to model the state of an agent, we make the following assumptions; all information about the agent is contained in the state, observations are independent given the current state, and state occupation time is negatively exponentially distributed.

4.2 State Probability Distribution

From the HMM we can derive a prediction of the probability distribution over the states, and we use the probability of being in the trusted state as reputation value. Our modeling approach is different from the Beta model as we do not assume that there is an

underlying fixed probability p of getting a good rating. Instead we assume that an agent is in one of the hidden states, and that the ratings are characterized by different values of p dependent on the current state of an agent. The rating process is similar to the monitoring process in an intrusion detection system, and the challenge is to predict the current state of an agent and detect a possible state change.

We have not made any assumptions about time between observations, and there is no direct relation between observations and state-changes. As a consequence the system could have made zero, one or more transitions during the time between to successive observations. The time when observation number k is produced is denoted t_k. Time between observation $k-1$ and observation k is denoted $\delta_k = t_k - t_{k-1}$.

The transition rate matrix $\Lambda = (\lambda_{ij})$ is describing the dynamics of the system. To simplify the notation we will use i and j instead of s_i and s_j. The relation between system states and the transition rates is given by

$$\lambda_{ij} = \begin{cases} \lim_{dt \to 0} \dfrac{P(\boldsymbol{x}(t+dt) = j | \boldsymbol{x}(t) = i)}{dt} & \text{if } i \neq j \\ \sum_{j \neq i, j=1}^{N} -\lambda_{ij} & \text{if } i = j \end{cases}. \tag{6}$$

Since observations are received at irregular intervals, the running transition probabilities $p_{ij}^k = P(x(t+\delta_k) = j | x(t) = i)$ depend on the time since last observation δ_k, and have to be calculated each time an observation is received. The running transition probability matrix $\boldsymbol{P}_k = (p_{ij}^k)$ can be derived from Kolmogorov's equations [16] as follows

$$\boldsymbol{P}_k = e^{\Lambda \delta_k}. \tag{7}$$

There are several analytical and numerical methods for solving these ordinary differential equations, in our case the state space is very small, so calculations are inexpensive. Let $\gamma_k = (\gamma_k(i))$ denote the prediction of the state probability distribution at time t_k given all observations received until time t_k, $\gamma_k(i) = P(x_k = i | \boldsymbol{y}_k)$ where $\boldsymbol{y}_k = y_1, \ldots, y_k$. The algorithm for calculating γ_k is given in [5], it is an on-line algorithm derived from the *forward-backward* procedure described in [14], and is very efficient. It does not require the agents to keep any history of past observations in memory.

4.3 Parameter Estimation

The parameters that need to be set in the HMM are the initial state distribution π, the observation symbol probabilities $\boldsymbol{B}$ and the state transition rates Λ. In [11] we describe a method for learning the model parameters by the combination of the machine learning technique *reinforcement learning* [18] and the forward-backward procedure, which finds the maximum likelihood parameter estimate from a training sequence of observations. As this parameter learning technique is not the main focus of this paper, we will assume that these parameters are available to the model and perform the simulations with a few different representative values for the parameters.

We will set the initial state distribution π to be uniform over the states, so we have that $\pi_1 = 0.5$ and $\pi_2 = 0.5$, in order to get the same starting condition as the Beta model. However, to overcome the problem of agents changing their identities and re-entering the system frequently, it might be better to change the starting condition so that a newcomer to the system is most likely not in a trusted state.

The state transition rates can be calculated from estimated expected state sojourn times $H = (h_1, h_2)$, the relation between transition probabilities and transition rates is given by

$$\lambda_{ij} = \frac{p_{ij}}{h_i} \quad \text{for } i \neq j. \tag{8}$$

The transition rate models the tendency of the agent to change its trustworthiness over time, large state transition rates will lead to faster response to indications of state changes in the model.

The observation symbol probabilities models the uncertainty of the observations. If we for instance have the parameter $b_1(g) = 0.9$, this means that we have a probability of 0.1 of getting a good observation even though the agent really is in an untrusted state. In other words we have a certainty of 90% of getting correct observations. Figure 3 shows how the hidden Markov trust model responds to an input of 20 good followed by 20 bad observations for different observation symbol probabilities. The time between observations is fixed, and we have used the estimated state sojourn times $h_1 = 100$, and $h_2 = 100$. We have used symmetric observation probabilities in this example, i.e. if $b_1(g) = 0.9$ we also have that $b_2(b) = 0.9$.

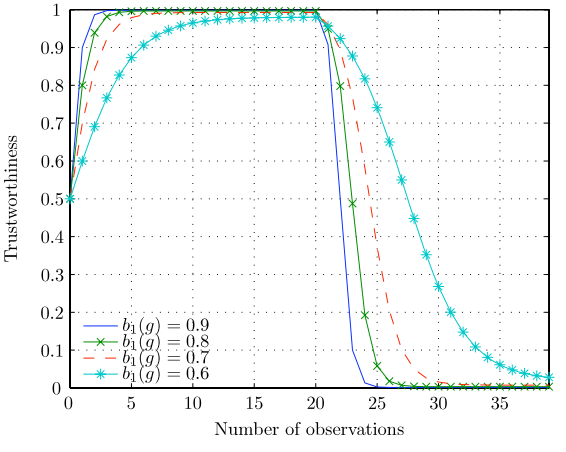

Fig. 3. The hidden Markov trust model with different observation probabilities, the input is 20 good direct observations followed by 20 bad direct observations

As we can see from Figure 3, the observation symbol probabilities influence the response to state transitions in the model. This is natural, since if the observations are unreliable, we would like to have more observations indicating a state change before we believe that an actual state change has occurred. It would make sense to assign a higher observation symbol probability to the first-hand observations $\{g, b\}$ than to the second-hand observations $\{r, w\}$. For the second-hand observations we could choose to model each recommender separately, this means that we assign different observation symbol probabilities $\{r, w\}$ to every recommender. If we have a history of previous

recommendations and warnings coming from a specific recommender, we can learn the parameters from this sequence of observations.

5 Simulation Results

In this section we will present some simulation results from our comparison study of the hidden Markov trust model and the Beta reputation model. We describe a selection of trust scenarios and compare the performance of the models in these situations.

5.1 Simulation Assumptions and Parameters

When we do the comparison of the Beta model and the hidden Markov trust model in the following, we will consider a decentralized version of the Beta reputation system, where we let each agent calculate its own reputation value for the other agents instead of calculating the reputation values in a reputation centre. We assume that there is a trusted reliable communication protocol in place that allows agents to obtain feedback from other agents in the form of ratings.

For the model parameters, we have used the Beta model with a forgetting factor $\phi = 0.9$, and the hidden Markov trust model with state sojourn times $h_1 = 100$, $h_2 = 100$ and observation symbol probabilities $b_1(g) = 0.8$, $b_2(b) = 0.8$. For the Beta model, we can see from Figure 2 that a high value of ϕ gives the best response to state changes as it gives the largest variation in the reputation value. Small values of ϕ seems to give quicker response, but leads to a convergence of the reputation score to a less extreme value. This means that the reputation value becomes more average and does not clearly distinguish between states. Since we want to model these state changes with our hidden Markov trust model, we have used the Beta model with a high value of ϕ. For the hidden Markov trust model, we have used relatively high observation probabilities following the same reasoning. In the last simulation we have used other parameters for the hidden Markov trust model, because we want to illustrate the flexibility of the model by showing how we can adjust the parameters so that it responds similarly to the Beta model.

5.2 Response to State Changes

We have already seen from Figures 2 and 3 how the two models respond to a state change, we have 20 good observations followed by 20 bad observations. In Figure 4 we see the difference between the models more clearly. Such an input set of observations could come from a trust scenario where an agent builds its reputation value by behaving good for a certain amount of time, and then decides to take advantage of its good reputation by suddenly changing its behavior. From Figure 4 we see that the slope of our model is much steeper than the slope of the Beta model. The Beta model has a lower reputation value for the first observation, but this is due to the slower convergence of the Beta model to the good state. If we for instance had a threshold for detecting state changes at the reputation value 0.5, the hidden Markov trust model would detect this already at the third bad observation while the Beta model would detect it after six bad observations.

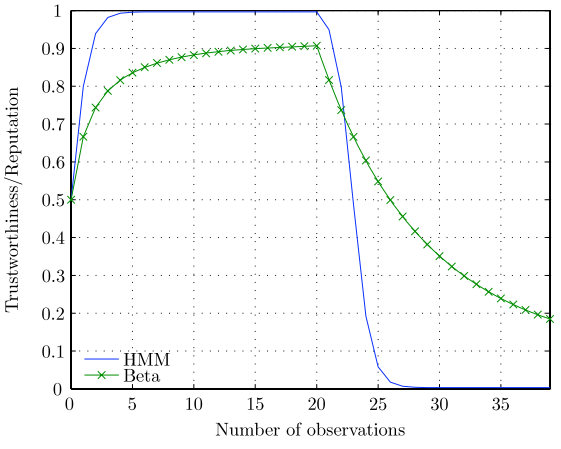

Fig. 4. The hidden Markov trust model compared to the Beta model, the input is 20 good observations followed by 20 bad observations

5.3 Time Component

The Beta model does not take the time component into consideration, it only models the reputation value in terms of number of ratings. In the hidden Markov trust model we include the time between observations in our model. To illustrate the advantage of including the time aspect, we consider the following scenario. We assume that an agent has been compromised, i.e. 'taken over' by a malicious agent. The agent then proceeds with a strategy of 'laying low', meaning that is waits for a long time without acting malicious, so that when it starts to show malicious behavior it can take full advantage of the good reputation that the previous owner of the agent had built up. From Figure 5 we can see an example of such a scenario, where we have 9 good observations, then one bad observation at time $t = 10$, then no observations until time $t = 35$, followed by 5 bad observations. We can observe from the plot that the hidden Markov trust model gives a steeper slope and continues the negative trend over time, while the Beta model is just stretched at the x-axis.

5.4 Disruptive Behavior

We want to see how the models react to a disruptive agent that changes it strategy in order to adapt to the rules of threshold-based intrusion detection. In particular, we consider an agent that follows a pattern of misbehavior adapted to a detection rule of 'three strikes and you're out'. In Figure 6 we have an example of this scenario, where an agent is showing good behavior for 10 observations to build up its reputation, and then proceeds with the disruptive behavior giving a pattern of 2 bad observations, one good observation, 2 bad observations, and so on. We can see from the plot that the Beta model picks up this behavior with a decreasing reputation value, but the hidden Markov trust model detects the state change faster and converges to much lower trustworthiness values.

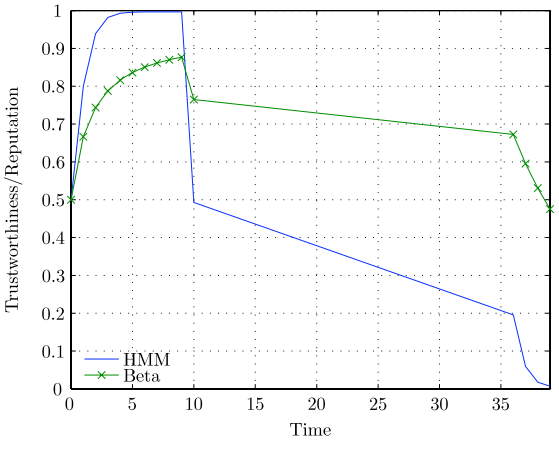

Fig. 5. The hidden Markov trust model compared to the Beta model, the input is 9 good observations, 1 bad observation at time $t = 10$, then no observations until time $t = 35$ followed by 5 bad observations

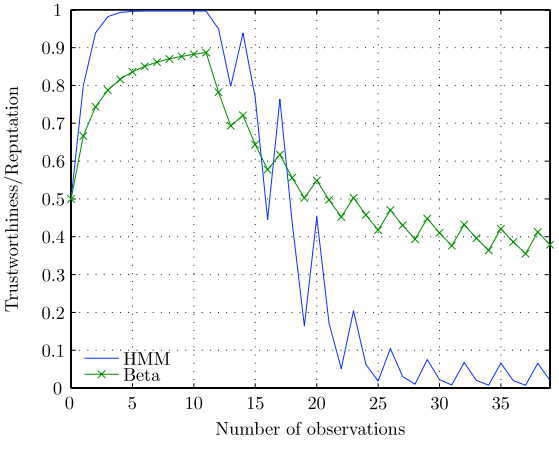

Fig. 6. The hidden Markov trust model compared to the Beta model, the input is 10 good observations, followed by a disruptive behavior giving a pattern of 2 bad observations, one good observation, 2 bad observations, and so on

5.5 Model Flexibility

We have shown some examples where the hidden Markov trust model performs better than the Beta model in detecting state changes. This is not very surprising as the Beta model is not based on the assumption that agents can be in different *states* when it comes to trustworthiness. The performance of both models is of course dependent on the model parameters. The Beta model in the variant that we used in our simulations has fewer parameters than the hidden Markov trust model, albeit we have included the

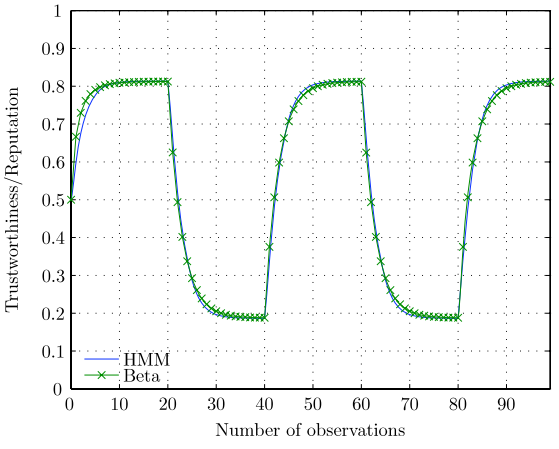

Fig. 7. The hidden Markov trust model compared to the Beta model, the parameters of the models have been adjusted to make them respond similarly, input is 20 good observations followed by 20 bad observations, 20 good observations and so on

forgetting factor as a parameter in order to study the variant of the Beta model which is most sensitive to dynamic behavior. We used the parameters that seemed to give the most beneficial results for both models.

Now we want to illustrate the flexibility of the hidden Markov trust model, by adjusting its parameters so that it responds similarly to the Beta model. The results of this adjustment can be seen in Figure 7. We have used the Beta model with a forgetting factor of $\phi = 0.7$, and adjusted the parameters of our model to make it respond close to the Beta model. For the hidden Markov trust model we have used the observation symbol probabilities $b_1(g) = 0.6$, $b_1(b) = 0.4$, $b_2(b) = 0.6$ and $b_2(g) = 0.4$. We also adjusted the state sojourn times to $h_1 = 8$ and $h_2 = 8$. The hidden Markov trust model with these parameters describes a situation where observations are very uncertain and state transition rates are high. With such parameters we could say that the state modeling aspect of our model has been suppressed.

6 Discussion and Conclusion

We have seen from the simulated examples that the Beta model and the hidden Markov trust model performs differently. We will now explain the fundamental differences between the two models, and discuss the findings from the simulations in this light. The hidden Markov model assumes an underlying state, observations are uncertain and we have an uncertainty of which state an agent is in. The Beta model does not assume that an agent is either good or bad, but rather seeks to pinpoint the trustworthiness of an agent on a continuous scale from 0 to 1. The interpretations of the observations in this model are deterministic. The difference between the models can be seen as an analogy to the difference between fuzzy sets and probabilities. In fuzzy logic an agent can be

partially trusted, in the sense that he is 70% honest and 30% dishonest. This is different from a situation where we are 70% certain that an agent is 100% honest. This fundamental difference between the two models explains why the hidden Markov model performs better when it comes to the detection of changes in behavior of the agents over time. While the hidden Markov model recognizes a state transition, the Beta model is instead modeling an agent that gradually becomes partially more dishonest. This difference is clearly demonstrated in the simulation illustrated in Figure 6, where we consider an agent with a disruptive strategy. Additionally, we have the effect of the different time constants in the models. While the Beta model is relying on the 'lifetime' of old observations, the time constant in the hidden Markov trust model is associated with the underlying state transition process.

The hidden Markov trust model has more parameters then the Beta model, thus it can be more fine-tuned and adaptable to dynamic environments. However, this also leads to challenges related to the parameter estimation. In [11] it is discussed how its parameters can be learned using a combination of the machine learning technique *reinforcement learning* [18] and the forward-backward procedure [14], which finds the maximum likelihood parameter estimate. Both the Beta model and the hidden Markov trust model can be further refined by introducing more dimensions or states. The multinomial Bayesian models, which allow for graded ratings, introduce more dimensions to the Bayesian modeling. It would have been interesting comparing a multinomial Bayesian trust model to a hidden Markov trust model with more states and more observation symbols. However, such a comparison would be challenging due to the big number of parameters that would need to be managed in the simulations. Including trust transitivity between different contexts is also an important issue that should be addressed in future work.

We have presented a comparison of the hidden Markov trust model and the Beta reputation system. Due to its larger richness in model features, the hidden Markov trust model shows a better ability to deal with dynamic environments, where we are unable to obtain perfect information and agents can be assumed to change their behavior over time. However, the increased model complexity also leads to larger challenges in finding representative parameters for the model. A disadvantage of both models might be that they are not easily understandable to human users, since they build on much more advanced mathematics than the simple eBay-like systems. These models are therefore maybe better suited for applications in multiagent systems, routing protocols and other distributed networking environments with non-human interpreters of the trustworthiness calculations.

References

1. Balakrishnan, V., Varadharajan, V., Tupakula, U., Lucs, P.: TEAM: Trust Enhanced Security Architecture for Mobile Ad-hoc Networks. In: ICON 2007: 15th IEEE International Conference on Networks, pp. 182–187. IEEE, Los Alamitos (2007)
2. Buchegger, S., Le Boudec, J.: A Robust Reputation System for P2P and Mobile Ad-hoc Networks. In: Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems (2004)

3. Dewan, P., Dasgupta, P., Bhattacharya, A.: On using reputations in ad hoc networks to counter malicious nodes. In: ICPADS 2004: Proceedings of the Tenth International Conference on Parallel and Distributed Systems. IEEE, Los Alamitos (2004)
4. Dingledine, R., Freedman, M., Molnar, D.: Accountability. In: Oram, A. (ed.) Peer-to-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly, Sebastopol (2001)
5. Haslum, K., Moe, M.E.G., Knapskog, S.: Real-time Intrusion Prevention and Security Analysis of Networks using HMMs. In: Proceedings of the Fourth IEEE LCN Workshop on Network Security (WNS 2008). IEEE, Los Alamitos (2008)
6. Jøsang, A., Haller, J.: Dirichlet Reputation Systems. In: Proceedings of the Second IEEE International Conference on Availability, Reliability and Security (ARES 2007). IEEE, Los Alamitos (2007)
7. Jøsang, A., Ismail, R.: The Beta Reputation System. In: Proceedings of the 15th Bled Electronic Commerce Conference (2002)
8. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decision Support Systems 43(2), 618–644 (2007)
9. Marti, S., Giuli, T.J., Lai, K., Baker, M.: Mitigating routing misbehavior in mobile ad hoc networks. In: MobiCom 2000: Proceedings of the 6th annual international conference on Mobile computing and networking, pp. 255–265. ACM, New York (2000)
10. Moe, M.E.G., Helvik, B.E., Knapskog, S.J.: TSR: Trust-based Secure MANET Routing using HMMs. In: Proceedings of the 4th ACM International Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet 2008). ACM, New York (2008)
11. Moe, M.E.G., Tavakolifard, M., Knapskog, S.J.: Learning Trust in Dynamic Multiagent Environments using HMMs. In: Proceedings of the 13th Nordic Workshop on Secure IT Systems (NordSec 2008) (2008)
12. Mui, L., Mohtashemi, M., Halberstadt, A.: A Computational Model of Trust and Reputation. In: Proceedings of the 35th Annual Hawaii International Conference on System Sciences, pp. 2431–2439 (2002)
13. Nielsen, M., Krukow, K., Sassone, V.: A Bayesian Model for Event-based Trust. Electronic Notes on Theoretical Computer Science (ENTCS) 172, 499–521 (2007)
14. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. In: Readings in speech recognition, pp. 267–296 (1990)
15. Rasmusson, L., Jansson, S.: Simulated Social Control for Secure Internet Commerce. In: Proceedings of the 1996 Workshop on New Security Paradigms (NSPW 1996). ACM, New York (1996)
16. Ross, S.M.: Continuous-Time Markov Chains. In: Introduction to Probability Models, 8th edn., pp. 349–390. Academic Press, London (2003)
17. Sabater, J., Sierra, C.: Review on computational trust and reputation models. Artificial Intelligence Review 24(1), 33–60 (2005)
18. Sutton, R., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
19. Whitby, A., Jøsang, A., Indulska, J.: Filtering out unfair ratings in bayesian reputation systems. In: Proceedings of the 7th International Workshop on Trust in Agent Societies (2004)