Analyzing the Robustness of CertainTrust

Sebastian Ries, Andreas Heinemann

Abstract Trust in ubiquitous computing is about finding trustworthy partners for risky interactions in presence of uncertainty about identity, motivation, and goals of the potential interactions partners. In this paper, we present new approaches for estimating the trustworthiness of entities and for filtering and weighting recommendations, which we integrate in our trust model, called CertainTrust. We evaluate the robustness of our trust model using an canonical set of population mixes based on a classification of typical entity behaviors. The simulation is based on user traces collected in the Reality Mining project. The evaluation shows the applicability of our trust model to collaboration in opportunistic networks and its advantages in comparison to a distributed variant of the Beta Reputation System.

1 Introduction

The main driving force behind the idea of ubiquitous computing (UC) is to support humans in their everyday life. UC environments are expected to be made up by a huge number of heterogeneous, loosely coupled devices. Thus, collaboration between devices, e.g., sharing information or resources, is an indispensable enabler for the evolution of UC.

Sebastian Ries
Technische Universität Darmstadt, Hochschulstrasse 10, 64289 Darmstadt, Germany,
e-mail: ries@tk.informatik.tu-darmstadt.de
The author's work was supported by the German National Science Foundation (DFG) as part of the PhD program "Enabling Technologies for Electronic Commerce"

Andreas Heinemann
Technische Universität Darmstadt, Hochschulstrasse 10, 64289 Darmstadt, Germany,
e-mail: aheine@tk.informatik.tu-darmstadt.de

Frequent collaboration requires frequent decisions about who to interact with, demanding for a non-intrusive way of decision making. This can be done based on certificates or by hard-coded policies, which, e.g., only allow for the interaction with a set of pre-defined partners. However, in an unmanaged domain we can neither expect pre-defined or certified partners to be available all the time, nor that all suitable interaction partners are certified. The shortcomings of both approaches become obvious, when considering opportunistic [7] or pocket switched networking [8] in which information is disseminated in absence of an end-to-end connectivity, but rather in an one hop manner from one mobile node (user) to the next one.

Thus, we favor the approach of selecting interaction partners based on trust, which is built on experiences from past interactions. Although, one may be uncertain about the identity, motivation or goals of potential interaction partners, direct experiences from past interactions are a good indicator whether to interact another time. As direct experiences may be rare, indirect experiences (recommendations) need to be considered as additional information. As recommendations may be more or less accurate, techniques for filtering and weighting recommendations are important.

In this paper, we show how our trust model improves the selection of interaction partners, and thus improves the quality of interactions, i.e., positive outcome and feedback. For that purpose, we evaluate the trust model against a canonical set of populations with respect to its users' behaviors and by varying stability of the users' behaviors, i.e., varying the adherence of the users to a behavior. To cope with bad interaction partners and lying recommenders we introduce new approaches for calculating the trustworthiness of entities, and provide a new way to handle recommendations.

The remainder of the paper is structured as follows: First, we present related work in Sec. 2. Then, we briefly describe the scenario for our analysis, i.e., content distribution in an opportunistic network (see Sec. 3). In Sec. 4, we explain the parts of our trust model, called CertainTrust, that are relevant for the evaluation. In Sec. 5 we introduce our classification of entity behaviors and derive the possible population mixes. In Sec. 6, we present our simulation setup and the gained results. We discuss the results and summarize this work in a conclusion.

2 Related Work

Our scenario is motivated by an application, called *musicClouds* [7], which allows for autonomous sharing of music files in opportunistic networks. It focuses on defining filters for specifying the meta information of files of interest, but not on the selection of the interaction partner. In [8], Hui et al. argue for the relevance of pocket switched networking since there are numerous scenarios in which local connectivity might be preferred over an internet-based

connection, due to bandwidth, latency or costs. Although, the authors focus on analyzing mobility patterns and consequences for data forwarding, they state that, among other aspects, establishing and maintaining trust relationships, as well as the provision of incentives for participating in the network are important.

Besides the seminal work on trust in [11], which focuses on modeling trust for only two agents, and centralized approaches [6], there is a growing number of trust models which allow for a decentralized computation of trust, being more suitable for UC (see below).

Distributed trust models usually compute trust based on direct experiences from past interactions in a certain context and indirect ones (via recommendations) [1, 9, 10, 12]. A few approaches integrate additional knowledge from similar contexts [2] or related aspects [14], which requires an additional modeling of ontologies expressing the relations between those contexts or aspects. In this paper, we focus on modeling trust in a single context, as this is the most appropriate for our scenario.

The trust model in [1] focuses on virtual organizations, and provides a label-based approach for representing trust, but includes only a rudimental approach for treating uncertainty. In [12], Quercia et al. provide a trust model that allows for a discrete, non-binary representation of trust. The model is capable of expressing the confidence of a trust value, mainly based on the variance derived from the data about experiences in past interactions, and the number of past experiences. A general approach to model trust, called "Subjective Logic", proposed by Jøsang [9], integrates the Bayesian update mechanism together with a representation reflecting a human notion of belief, capable of expressing uncertainty. This model also provides a method for integrating continuous feedback [10]. Both approaches do not allow to explicitly define how many experiences are necessary to reach the maximal level of certainty. Thus, they treat (un-)certainty equally for all contexts, and in a rather static manner. Furthermore, their filtering and weighting techniques focus on scenarios with a majority of positive recommendations. The trust model provided by Buchegger in [3] proposes a filtering of recommendations based on the similarity of the recommendations to the direct experiences, which may be circumvented if direct experience is missing, or by a repeated stepwise providing of misleading recommendations. The approach introduced in [15] is close to our approach, as it uses the past accuracy of past recommendations per entity for weighting its recommendations. But it takes a long time until it totally excludes misleading recommendations from a recommender. Furthermore, it introduces the assumption that it is necessary to learn the trust in recommenders depending on the actual values of the recommendations.

3 Scenario

For the evaluation of the impact of our trust model, we choose a scenario in which humans with mobile devices share music in an opportunistic network. In the following we refer to users with their mobile devices as entities. As these entities move, they will meet other entities, and interact with each other, e.g., they exchange music (mp3 files) or recommendations, in a spontaneous manner. Due to different goals or motivation, the users will show different behaviors when providing files to others. The goal of a typical user is to interact only with trustworthy interactors, i.e., interactors from which he expects to receive a good file (correct file, no viruses, complete song, and expected quality).

We assume that a user appreciates the support of a software component including a trust model, which supports him with information about the trustworthiness of the available candidates for an interaction, or is even capable of making decisions and interacting on its own. This will be especially true, if it allows for increasing the quality of a user's interactions, i.e., the number of received good files.

After an interaction, the *quality of the interaction* is determined by feedback. The generation of the feedback does not necessarily require user interaction. In some cases this can also be done automatically, e.g., by scanning the mp3 file for viruses, checking the size, the bitrate, and noise. This allows the software component to create histories about interactors and recommenders.

4 Trust Model

In this section, we introduce our system model. For self-containment of this paper, we briefly describe the representational trust model (for details see [13]), which defines how trust is modeled. Then, we present our approach of deriving an expectation value from the collected evidences. Furthermore, we present a new approach for filtering and weighting recommendations. At last we introduce the update mechanism based on feedback on past interactions. For an overview of the different steps in establishing trust between entities and selecting entities based on trust see Fig. 1.

4.1 System Model

The participants in the system are called `entities`. Entities can be either `active` or `passive`. Active entities can be humans with mobile devices, or autonomous software components, e.g., web services. Active entities have a behavior which can be more or less trustworthy. A passive entity is any kind

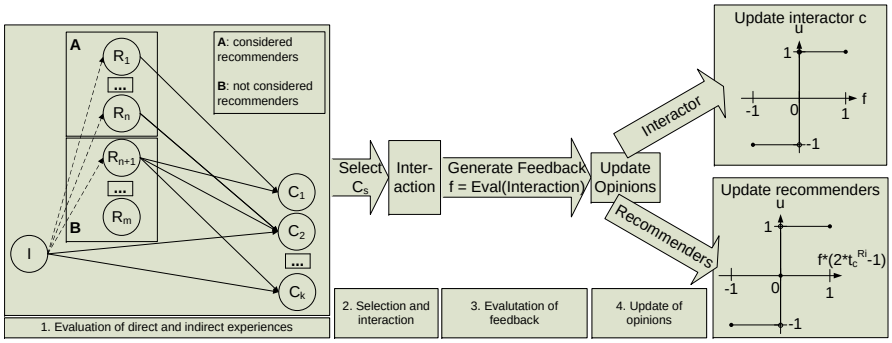

Fig. 1 Main steps in establishing trust between entities and selecting entities

of object, which can be rated, but does not have any behavior, e.g., a music file. For the evaluation in this paper, we only consider active entities.

Furthermore, we define the following roles for an entity (see Fig. 1): `initiator (I), candidate (C), recommender (R)`. The initiator is an entity which wants to initiate an interaction. The candidates – sometimes referred to as *interactors* – are the potential partners for this interaction. The entities which pass recommendations about the candidates to the initiator have the role of the recommenders. The initiators as well as the recommenders will always be active entities. The candidates may be active or passive.

4.2 Representational Trust Model

The representational trust model of CertainTrust has already been published in [13]. In general, this trust model can be used for different contexts. However, for simplifying the notation, we assume there is only one.

Let entities be denoted by capital letters $A, B, \ldots$. The opinion of entity A about the trustworthiness of entity B as *candidate* is denoted as o_b^A. The opinion of an entity A about the trustworthiness of B as *recommender* will be denoted as o_B^A. Furthermore, a parameter, called the *maximal number of expected evidences*, expresses how many evidences an entity expects to reach the maximal certainty of an opinion (see below) and is denoted as e.

Evidence Model: The evidence based representation allows to derive beta probability density functions from the collected evidence. The beta distribution, denoted as $Beta(\alpha, \beta)$, can be used to model the posteriori probabilities of binary events; typically using $\alpha = r + 1$ and $\beta = s + 1$, where $r \geq 0$ and $s \geq 0$ represent the number of collected positive and negative evidence, respectively. The *number of collected evidence* is represented by $r + s$. If an opinion is represented by the parameters r and s, we use the notation

$o = (r,s)^{rs}$. This representation allows for easily integrating feedback in the trust model (see Sec. 4.6).

Human Trust Interface: In the HTI an opinion o is a 2-dimensional expression, represented by a 2-tuple $o = (t,c)^{HTI} \in [0,1] \times [0,1]$, where the superscript refers to the representational model. The opinion $o_b^A = (t_b^A, c_b^A)^{HTI}$ expresses the opinion of A about the trustworthiness of entity B as *interactor*. The value of t_b^A represents the past experience of A with B as interactor, calculated as relative frequency of good interactions. This value is referred to as the *trust value*. The value c_b^A is referred to as *certainty value*. This value expresses, which certainty the provider of an opinion assigns to the trust value, e.g., a low certainty value expresses that the trust value can easily change, whereas a high certainty parameter indicates that the trust value is rather fixed, and thus will be a good estimate for the future behavior. The certainty increases with the ratio between the number of collected evidences and the number of expected evidences. For opinions with certainty $c_b^A = 0$ the trust value is initialized with $t_b^A = 0.5$. For details of the calculation see [13].

4.3 Expectation Value

The opinion about the trustworthiness of an entity is based on information collected from past interactions. As the trust model is to support users in future interactions, we extend our model with new approaches to derive an expectation from the opinions. The certainty of an opinion is to indicate whether the trust value is expected to be a good prediction or not; thus, both values need to be included in the expectation value.

In case that certainty $c = 1$, the trust value should be used as expectation value; if the certainty $c = 0$, i.e., complete uncertainty, the expectation value should be an appropriate initial value; in-between it seems natural, that the expectation value moves from this initial value towards the trust value with increasing certainty. For an opinion $o = (t,c)^{HTI}$, this can be expressed as:

$$E(o) = c \cdot t + (1-c) \cdot f \text{ ,where } 1-c \text{ is the uncertainty}$$

The parameter f can be used to determine the initial expectation value in case of complete uncertainty and influences the expectation value until complete certainty $c = 1$ is reached. Thus f can be used to express a user's general attitude (dispositional trust) or depend on additional knowledge about the distribution of trustworthy and untrustworthy entities. In the following, we briefly introduce several strategies to initialize f.

Pessimistic ($f = 0$): The pessimistic strategy for calculating an expectation value uses only the evidences on which the opinion was based. According to this strategy, the expectation value is 0.0, if there has not been collected any evidence at all (complete uncertainty). This reflects a user's attitude like

"I believe entities are untrustworthy, unless I know the opposite with high certainty".

$$E_{pessimistic}(o) = t \cdot c$$

Moderate ($f = 0.5$): The moderate approach is especially suitable for binary decisions, in the case that positive and negative results happen with the same probability. According to this strategy the expectation value is 0.5, if there has not been collected any evidence at all (complete uncertainty). This reflects a user's attitude like "I believe there are as many trustworthy as untrustworthy entities."

$$E_{moderate}(o) = t \cdot c + (1 - c) \cdot 0.5$$

Optimistic ($f = 1$): The optimistic behavior reflects a user's attitude like "I believe entities are trustworthy, unless I know the opposite with high certainty".

$$E_{optimistic}(o) = t \cdot c + (1 - c)$$

Dynamic Recalculation of f: As an entity's experiences within a community grows it might be reasonable that it dynamically updates the parameter f that expresses its initial believes. To evaluate whether a dynamic expectation value may have positive effects on the trust model, we designed an ad-hoc heuristic. We define a new variable *community factor* cf, that each entities calculates based on its own experiences and is used as the initial parameter f.

The basic idea is to derive the community factor from the opinions about the known entities. For the community factor used for the expectation value about recommenders we provide the update mechanism in pseudo code:

$r = 0$; $s = 0$; for (Entity E: known Recommenders){
    $u = trustValue(o_E^A)$;
    $r = r + u$; $s = s + (1 - u)$; }
$o_{cf} = (r,s)^{r,s}$; $cf = E_{moderate}(o_{cf})$;

For the interactors the calculation is similar. But due to the fact, that all entities prefer to interact with good interactors, we adjust the value u in the pseudo code according to $u = trustValue(o_e^A) * (1 - certaintyValue(o_e^A)/2)$.

Comparison with the Mean Value of the Beta Distribution: For the Beta distribution $Beta(r+1, s+1)$ the expectation value (mean) E_{mean} is defined as $(r+1)/(r+s+2)$. This mean value only depends on the number of collected evidences. It gets closer to the mode of the distribution as the number of collected evidences grows, but this relation is static. This means, it does not depend on the number of maximal expected evidences, and thus, it does not properly integrate the certainty of an opinion in our model.

Our moderate approach for calculating the expectation value produces the most similar results to E_{mean}. We are currently working on a mapping of our strategies for determining the expectation value to adequate counterparts

derived from the Beta distribution. This can be done using $\alpha = r + r_0$ and $\beta = s + s_0$, adjusting r_0, s_0 depending on the value of f, the number of collected evidences, and the number of expected evidences.

4.4 Computational Trust Model

The task of the computational trust model is trust propagation, i.e., to aggregate the direct and indirect evidences. As proposed in [13] we propagate trust based on the two operators *consensus* – summing up several opinions to a single one – and *discounting* – weighting recommendations based on the opinion about the recommender.

Consensus: For the consensus of the opinions $o_c^{B_1}$, ..., $o_c^{B_n}$ we use: $consensus(o_c^{B_1},...,o_c^{B_n}) := \sum_{i=1}^{n} o_c^{B_i} := o_c^{B_1} \oplus ... \oplus o_c^{B_n} := (\sum_{i=1}^{n} r_c^{B_i}, \sum_{i=1}^{n} s_c^{B_i})^{rs}$.

Discounting: Let the opinion of entity A about the trustworthiness of B_i as recommender be $o_{B_i}^A$, and $o_c^{B_i}$ is B's recommendation about entity C as candidate. For the discounting we propose: $discounting(o_{B_i}^A, o_c^{B_i}) := o_{B_i}^A \otimes o_c^{B_i} := (E(o_{B_i}^A) * r_c^{B_i}, E(o_{B_i}^A) * s_c^{B_i})^{rs}$. Thus, using the pessimistic strategy for the expectation value, is equal to the discounting operator defined in [13].

Simple Trust Propagation: Let the direct experience of entity A with candidate C be o_c^A; furthermore A has collected indirect experiences from the recommenders $B_1,...,B_n$. The aggregated opinion of A about C is denoted as $\widetilde{o}_c^A$, and can be calculated as:

$$\widetilde{o}_c^A = o_c^A \oplus \sum_{i=1}^{n} o_{B_i}^A \otimes o_c^{B_i} \tag{1}$$

This variant of trust propagation has some shortcomings:

1. The opinions of recommenders which are known to provide bad recommendations are still considered.
2. All available recommendations are used. Thus, it would be possible, just by creating a huge number of "fake" entities to dominate the resulting opinion, even if the weight of a single recommendation is very low.

More Robust Trust Propagation: For the more robust variant of the trust propagation we propose a few enhancements to overcome the shortcomings pointed out above. To deal with the first issue, it seems reasonable that the initiator I only considers recommendations from recommenders which provided mostly accurate recommendations in the past, i.e., the trust value of o_R^I is greater than or equal to 0.5. Furthermore, we also consider recommendations of unknown recommenders.

To overcome the second issue, we use another feature of our trust model to limit the considered recommendations. We sort the recommendations descending by the expectation value of calculated by the initiator for the recom-

menders. Then, we consider only recommendations as long as the certainty of the aggregated opinion is less or equal to 1. Thus, we only use the best recommendations, until the sum of direct evidences and weighted indirect evidences is equal to the number of maximal expected evidences.

These arrangements together are supposed to improve the robustness of our model to false recommendations (either false praise or false accusation), since we use only the recommenders which have been known to be the best recommenders from their past recommendations. Furthermore, in the case that we have enough direct evidences and good recommendations, this makes our model also quite robust to sybil attacks [4], since it is no longer possible to overtake an opinion based on sufficient direct experiences and good recommendations only by providing an arbitrary huge number of recommendations using specially created recommenders.

Although, it is possible to further increase the robustness to false recommendations and sybil attacks by introducing further criteria for considering recommendations, e.g., excluding unknown recommenders, we are not going to extensively evaluate this aspects, since it would be beyond the scope of this paper.

4.5 Selection of a Candidate

The selection of the most trustworthy candidate for an interaction of a set of candidates C_1, ...,C_n is done based on the expectation value. Let $\widetilde{o}^A_{c_i},...,\widetilde{o}^A_{c_n}$ be the aggregated opinions of A about the candidates. The most trustworthy candidate is selected as an entity with maximal expectation value. Depending on the initiator the expectation value is calculated by one of the strategies proposed above.

4.6 Update Mechanism

After each interaction, the entity that initiated the interaction updates the opinions about its interaction partner (selected candidate) and the recommenders (see Fig. 1). In the case of a dynamical recalculation of the expectation value, the community factor also is updated after an interaction. The quality of an interaction, which for now is equal to the generate feedback fb, can be in $[-1;1]$. Here, -1 is the worst possible feedback, 1 is the best one. The feedback for an interaction, can be either user generated or by an external software component.

Update the Opinion for the Selected Candidate: Let the opinion (direct experiences) of A about the selected candidate C be $o^A_c = (r^A_{c_{old}}, s^A_{c_{old}})^{rs}$;

for the feedback f, the direct experiences are updated to $(r^A_{c_{new}}, s^A_{c_{new}})^{rs}$ using (with $u := fb$):

$$(r^A_{c_{new}}, s^A_{c_{new}})^{rs} = (r^A_{c_{old}} + (u+1)/2), s^A_{c_{old}} + (1-u)/2)^{rs} \tag{2}$$

Update the Opinions for the Recommenders: The update of the opinions about the recommenders is performed according to the accuracy of their recommendations. For this reason, the trust value of the recommendation of B about the candidate C is compared with the feedback with which A rated the interaction. If both have the "same tendency", then the recommendation is supposed to be positive and the opinion of A about B as recommender is updated positively, else there is a negative update.

More formally: If the opinion of A about entity B as recommender was $o^A_B = (r^A_{B_{old}}, s_{B_{old}})^{rs}$ and the recommendation by B about the candidate C was $o^B_c = (t^B_c, c^B_c)^{HTI}$ with $(c^B_c > 0)$, and $A's$ feedback for the interaction with C is fb, we calculate u as:

$$u := \begin{cases} 1 & \text{, if } (2 * t^B_c - 1) * fb > 0 \\ -1 & \text{, if } (2 * t^B_c - 1) * fb < 0 \\ 0 & \text{, else .} \end{cases} \tag{3}$$

The update of o^A_B is done using u in Eq. 2. For example, if the trust value of the recommendation was in $]0,1]$ and the interaction was positive $(f > 0)$, then the recommendation is considered to be good $(u = 1)$, and the positive evidences of the opinion about the recommender are increased by 1; the negative evidences are kept unchanged.

In the case the interaction behavior of C depends on the initiator of the interaction, and C shows different interaction behavior towards R and I, the recommendations of R will be misleading, and I will negatively update the recommender trust for R. This is due to the fact that I is not capable of distinguishing between R is lying and C interaction behavior is interactor dependent.

Furthermore, we point out that the *normalization* as described in [13] introduces an implicit aging of the evidences, if the collected evidences exceed the maximal number of expected evidences.

5 Basic Types of Behavior and Population Mixes

Entities may be recommenders or interactors. In both roles, an entity can be good (+) or bad (-). This means a good interactor provides good interactions, leading to positive feedback $(fb = 1)$, a bad interactor provides interactions leading to negative feedback $(fb = -1)$. A good recommender provides recommendations which reflect its real experiences. The model for

bad (lying) recommenders is derived from [15]. Bad recommenders try to provide recommendations with a maximal misleading expectation value, i.e., if $E_{mean}(o_x^B)$ is the expectation value calculated by recommender B for interactor x, the recommendation of B would be an opinion with the expectation value $1 - E_{mean}(o_x^B)$. This can be achieved by switching the positive and negative evidences. Thus, we identified 4 basic types of behaviors, see Fig. 2.

Combining these basic types of behaviors, we can derive 15 canonical population mixes: *h*, *m*, *s*, *w*, *hm*, *hs*, *hw*, *ms*, *mw*, *sw*, *hms*, *hsw*, *hmw*, *msw*, *hmsw*. The numbers of entities with a specific behavior within a population are set to be equal, e.g., the population mix *h* contains only entities with honest behavior, the population mix *hm* contains 50% entities with honest behavior and 50% malicious, and so on.

Basic entity behaviors		Recommendation behavior	
		+	-
Interaction behavior	+	honest (h)	selfish (s)
	-	malicious (m)	worst (w)

Fig. 2 Basic entity behaviors

Furthermore, we believe that the interaction behavior of an entity may be unstable, i.e., good interactors may sometimes provide bad interactions and vice versa. Therefore, we introduce an additional parameter called stability y. In the case of stability $y = 1$ an entity totally adheres to its assigned interaction behavior. In the case the stability of entity is set to 0.9 it adheres only in 90% of its interactions to the behavior it has been assigned, in the other 10% it will do the opposite. Knowing the stability of an entity and its behavior, it is easy to derive the probability for positive interactions with this entity. For simplicity, we assume stability only influences the interaction behavior, the recommendations behavior is assumed to be stable.

We use two different settings for the stability factor per population mix. In the first setting we set the stability factor to 1 for all entities, in the second one the stability factor is randomly (and uniformly distributed) chosen from the interval $[0.5; 1]$. In case of the population *hm*, a stability of 1 leads to a population in which 50% of all entities provide only good interactions and 50% provide only bad interactions. Using the same population but choosing the stability factor from the interval $[0.5; 1]$ per entity, the probabilities for good interactions over all entities are uniformly distributed in $[0; 1]$.

6 Simulation

The simulation is based on the scenario described in Sec. 3. For having realistic user traces as mobility model, we use the user traces collected in the Reality Mining project [5]. The data provides information about 97 users of mobile phones and their location as the ID of the cell tower the mobile phones were connected to.

For our evaluation we choose a single week of this data set, in which a big number of users were connected to a small number of cell towers; thus, we expected to have a big number of possible interactions. Based on [7], we assume that a group of users is in proximity to each other if the users are connected to the same cell tower within a 15 minute time interval. For the evaluation, we consider a so-called *meeting* to happen in intervals in which six or more users are connected to the same cell tower. The reason is, that we want to evaluate the trust model's capabilities in selecting the most trustworthy candidate from a set of candidates. The set of candidates is determined randomly as half of the available entities, i.e., an initiator has at least 3 candidates for an interaction. In the restricted data set, there are 68 distinct users (entities), which met each other in 556 meetings. In average an entity took part in 59.94 meetings, and met 46.76 distinct entities. The average number of entities per meeting is 7.33. By repeating the selected week three times in our simulation, we are able to evaluate the value of direct experiences as well as indirect ones. Although these assumptions might look a little simplistic, we believe that using real world user traces allows for a more realistic evaluation than using artificially created user profiles as a basis for the simulation.

We do the simulation for all 15 populations, each with stability $y = 1$ and $y \in [0.5; 1]$. Each simulation was repeated 20 times per trust model and population mix using the same seeds for the comparison of the different models and baselines.

6.1 Meeting Procedure

Each meeting proceeds as follows. In each meeting each entity has to interact with one candidate, i.e., each entity is the initiator of one interaction. The candidates for an interaction are randomly chosen as the half of the entities which are part of the meeting, i.e., we expect that half of the entities in the meeting can provide a specific mp3-file. If the trust model includes recommendations, the initiator asks all entities that are part of the meeting for providing recommendations about the candidates. Then, the initiator evaluates the trustworthiness of the candidates, and selects the most trustworthy one, i.e., the one with the greatest expectation value. We chose this setting in contrast to a setting in which each entity has the choice whether to interact or not, since we want to evaluate the trust model and not the decision making component. After each interaction, the initiator updates the opinions about its interaction partner (selected candidate) and the recommenders based on the outcome of the interaction as described in Sec. 4.6.

6.2 Baselines and Models

The first baseline is the random strategy (referred to as *const_0.5* in Fig. 4). This strategy selects the partner for the interaction randomly – expecting positive interactions from any candidate with probability 0.5. Furthermore, we use the *perfect strategy* that always selects the best candidate based on the behavior it has been assigned by the simulation environment. In a way, this is similar to a "best possible" selection process in a hypothetic world, in which all entities have labels on their forehead stating the behavior (and the probability for a positive interaction). We compare the following trust models in our evaluation:

1. CT_M: CertainTrust using the mean as expectation value and $e = 20$.
2. CT_C: CertainTrust using the expectation value based on a dynamically calculated expectation value and $e = 20$.
3. BetaRepSys: The beta reputation system was proposed in [10]. Since the goal of our research is to provide a trust model for UC, we use a distributed variant of this reputation system in which each entity is its own reputation centre. The reputation centre stores only direct experiences. The expectation value (reputation) for an interaction partner is calculated using the consensus operator for combining the direct experiences with the available recommendations.

6.3 Evaluation Metrics

The evaluation is based on three metrics.

1. Each entity B is assigned a characteristic probability for providing good interactions (denoted as p_B) at the beginning of the simulation (as described in Sec. 5). For the first metric, we calculate the mean absolute error $err(A)$ an entity makes when estimating this probability for all entities in the population P. For entity A this error is calculated as: $err(A) = (\sum_{B \in P} | E(o_b^A) - p_B |) / | P |$. For the calculation of $E(o_b^A)$ entity A may ask all entities in P for recommendations. The *average error in estimating the trustworthiness* err is defined as: $err = (\sum_{A \in P} err(A)) / | P |$. The average error should be close to 0.

2. We define the reputation $R(A)$ of an entity A as the average of the expectation values calculated by each entity B in the population P for entity A: $R(A) = (\sum_{B \in P} E(o_a^B)) / | P |$. Again, entity A may ask all entities in P for recommendations. As the average reputation over all entities in the population depends on the population mix, we calculate the average only over the entities belonging to the same behavior.

$$average\_R(behavior_i) = \frac{\sum_{A \text{ is assigned } behavior_i} R(A)}{| A \text{ is assigned } behavior_i |} \tag{4}$$

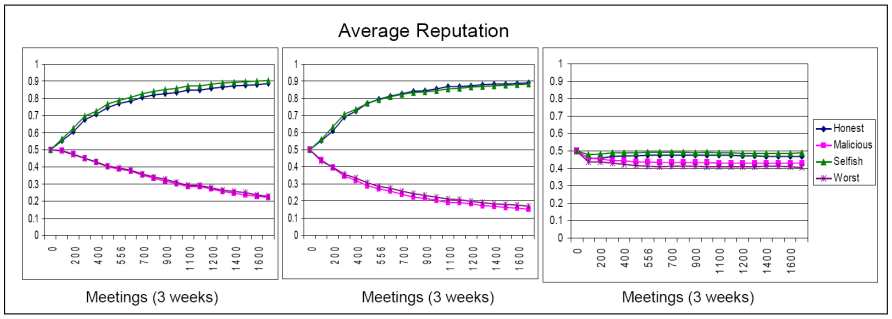

Fig. 3 Reputation evaluation over time in population *hmsw* with stability 1 (order: CT_C, CT_M, BetaRepSys)

3. For each interaction, the feedback can be either 1 or −1. The accumulated sum of feedback (*acc_sum*) is calculated for each entity as sum of the feedback over its past interactions. This value strongly depends on the population mix. In a population with stability $y = 1$ and only honest entities are only positive interactions; in a population with only malicious entities are only negative ones. Therefore, we define the *average percentage of accumulated sum of feedback* as the portion of the accumulated sum of feedback achieved using the considered trust model relative to the accumulated sum achieved using the *perfect strategy*:

$$percentage\_acc\_sum(model\_X) = \frac{\sum_{A \in P} acc\_sum(entity\_A, model\_X)}{\sum_{A \in P} acc\_sum(entity\_A, perfect\ strategy)} \tag{5}$$

The *average percentage of accumulated sum of feedback* is the third metric for comparing the different trust model. The closer the average percentage of accumulated sum of feedback is to 1.0, the more positive interactions had an entity, and the closer is trust model to the perfect selection strategy.

6.4 Results

Reputation Evaluation Over Time: The three diagrams in Fig. 3 show the evaluation of the reputation over time for the models CT_M, CT_C, and the BetaRepSys. As space is limited we provide these diagrams only for the population *hmsw*. As we can see from these diagrams both CertainTrust variants are capable of detecting the different behaviors of the entities. The true reputation of honest and selfish entities would be 1 and for malicious and worst entities it would be 0. The BetaRepSys is hardly capable of detecting differences as 50% of the recommendations are misleading.

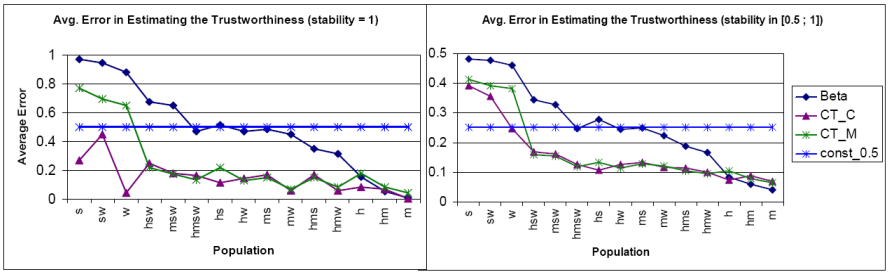

Fig. 4 Average percentage of accumulated sum per population (at the end of the simulation) - Populations sorted according to the percentage of good recommenders (the lines are only for ease of reading)

Average Error in Estimating the Trustworthiness: Fig. 4 shows the results of the error in estimating the trustworthiness at the end of each simulation (averaged over all runs) per population and stability of behaviors.

From the results we can see that whenever there are 33% or more misleading recommendations, our CertainTrust variants have a considerable better performance than the BetaRepSys. Only in case of populations with only good recommenders the BetaRepSys has slight advantages. The CT_C model produces the best results in populations with only bad recommenders.

For stability in $[0.5;1]$ the absolute numbers for the error are less than for stability $y = 1$. This can be explained as in the first case the average probability of good interactions per entity is in the interval $[0.25;0.75]$ and in the latter case in the interval $[0;1]$.

Average Percentage of the Accumulated Sum: Fig. 5 shows the results of the average percentage of the accumulated sum at the end of each simulation (averaged over all runs) per population and stability of behaviors. In the populations *m*, *w*, and *mw* the accumulated sum for the perfect strategy as well as the accumulated sum for all other models is negative. Therefore, we omitted the results here.

In the case of stability $y = 1$ the BetaRepSys only produces similar results as the CT variants in the populations *h*, *s*, *hs* (this is trivial as there are 100% good interactors), and in the population *hm*. In the other populations the CT variants outperform with similar results.

In the case of stability in $[0.5;1]$ the BetaRepSys can only compete in the populations *hm* and *h*. Both CT variants show similar performance.

7 Discussion

In contrast to the simulations in [12,15], our simulation presents results over a huge set of populations and uses a mobility model as basis for the interactions and recommendations. The population mixes are derived from our

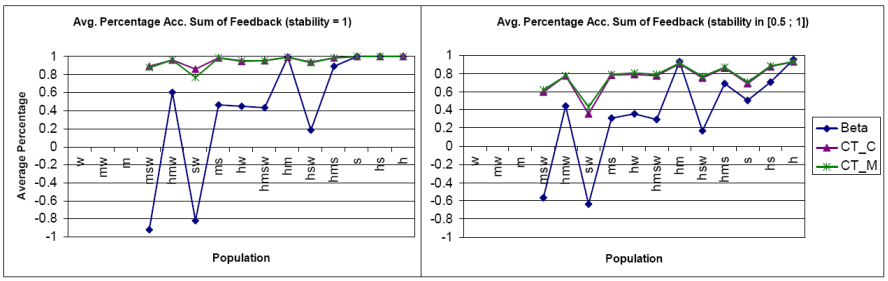

Fig. 5 Average percentage of accumulated sum per population (at the end of the simulation) - Populations sorted according to the percentage of good interactors (the lines are only for ease of reading)

classification of the basic types of behaviors. As we consider all kinds of combinations the results present an good overview of the performance of the considered models. The possibilities for interactions and recommendations between entities are based on real world user trace. This is especially important as we develop a trust model for opportunistic collaboration and UC applications.

The results for the average accumulated sum of feedback (see Fig. 5) show that our trust model achieves good results in all considered populations. Compared to the perfect selection strategy *CT_C* is capable of achieving more than 75% of maximally reachable results in 21 of 24 populations. Compared to the BetaRepSys we see that CertainTrust variants outperform in most population mixes. This is especially important as a user will evaluate not the trust models itself, but the improvement of the quality of his interactions. The reason for the improved overall quality of interactions can be the smaller error in the estimated trustworthiness (see Fig. 4) for most of the populations. Furthermore, we have shown that our CertainTrust model CT_C using the dynamically updated community factor may have advantages in estimating the trustworthiness of interactors.

We learn that the main drawback of the distributed variant of the BetaRepSys is that it does not weight recommendations. The discounting, as proposed in [10], is based on the assumption that the behavior of an entity as recommender is equal to the behavior of this entity as interactor and may still be misleading. Furthermore, the discounting would discard the recommendations by unknown interactors. Thus, it heavily depends on direct experiences. The filtering techniques for the BetaRepSys as proposed in [16] will only work if the majority of recommenders provides good recommendations. This is also true for the filtering techniques proposed in [12]. As the representational model of the BetaRepSys as well as our model are based on collected evidences, it should be possible to define a mapping between both representations. Thus, the filtering mechanisms presented in this paper can easily be transferred to the BetaRepSys and other models based on collected evidences.

8 Conclusion

We presented and evaluated new approaches for discounting opinions and estimating the trustworthiness of entities. We showed that our trust model allows to significantly improve the quality of interactions especially in presence of lying recommenders. Furthermore, we showed our trust model's robustness over different populations and varying stability of the users' behaviors. The dynamical calculation positively influences the trust models' performance. We will investigate on this in future work.

References

1. Abdul-Rahman, A., Hailes, S.: Supporting trust in virtual communities. In: Proceedings of Hawaii International Conference on System Sciences (2000).
2. Billhardt, H. et al.: Trust-based service provider selection in open environments. In: ACM SAC '07, pp. 1375–1380. ACM Press, New York, NY, USA (2007).
3. Buchegger, S., Le Boudec, J.Y.: A Robust Reputation System for Peer-to-Peer and Mobile Ad-hoc Networks. In: P2PEcon 2004 (2004)
4. Douceur, J.R.: The sybil attack. In: IPTPS '01: Revised Papers from the 1st Int. Workshop on Peer-to-Peer Systems, pp. 251–260 (2002)
5. Eagle, N., Pentland, A.S.: Reality mining: sensing complex social systems. Personal Ubiquitous Comput. **10**(4), 255–268 (2006).
6. Golbeck, J.: Computing and applying trust in web-based social networks. Ph.D. thesis, University of Maryland, College Park (2005)
7. Heinemann, A.: Collaboration in Opportunistic Networks. Ph.D. thesis, Technische Universität Darmstadt (2007).
8. Hui, P. et al.: Pocket switched networks and human mobility in conference environments. In: WDTN '05: Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking, pp. 244–251 (2005).
9. Jøsang, A.: A logic for uncertain probabilities. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **9**(3), 279–212 (2001)
10. Jøsang, A., Ismail, R.: The beta reputation system. In: Proceedings of the 15th Bled Conference on Electronic Commerce (2002)
11. Marsh, S.: Formalising trust as a computational concept. Ph.D. thesis, University of Stirling (1994).
12. Quercia, D., Hailes, S., Capra, L.: B-Trust: Bayesian trust framework for pervasive computing. In: 4th Int. Conf. on Trust Management, pp. 298–312 (2006)
13. Ries, S.: CertainTrust: A trust model for users and agents. In: ACM SAC '07, pp. 1599 – 1604 (2007)
14. Sabater, J., Sierra, C.: Reputation and social network analysis in multi-agent systems. In: Proceedings of the AAMAS, pp. 475–482 (2002).
15. Teacy, W.T. et al.: TRAVOS: Trust and reputation in the context of inaccurate information sources. In: Proceedings of the AAMAS **12**(2), 183–198 (2006).
16. Whitby, A., Jøsang, A., Indulska., J.: Filtering out unfair ratings in bayesian reputation systems. Icfain Journal of Management Research **4(2)**, 48 – 64 (2005)