2011 Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops

The Neighbor-Trust Metric to Measure Reputation in Organic Computing Systems

Rolf Kiefhaber*, Stephan Hammer*, Benjamin Savs†, Julia Schmitt*,
Michael Roth*, Florian Kluge*, Elisabeth André*, Theo Ungerer*
Department of Computer Science
University of Augsburg, Germany
*{*kiefhaber, hammer, schmitt, roth, kluge, andre, ungerer*}@*informatik.uni-augsburg.de*
†{*benjamin.savs@student.uni-augsburg.de*}

***Abstract*—Reputation is an important aspect of trust. If no direct trust experiences are available, one needs to rely on reputation data from other sources. In this paper we present the Neighbor-Trust metric that exploits these communication capabilities of a network by directly asking all neighbors of a target communication partner for reputation trust data. This results in a reputation path of length one, but also in a vulnerability to attacks by unknown, lying entities that try to promote not trustworthy entities. However, by adding weights for reputation data given by entities and a learning mechanism the Neighbor-Trust metric is able to identify and adapt to lying participants in the network by reducing the weight their reputation data has in future reputation calculations. We present an evaluation for the metric and show how to exclude lying participants from the network.**

***Keywords*-Organic Computing, Trust, Reputation**

I. Introduction

Trust is an important aspect of cooperation, it even is the basis for cooperation. When no implicit trust exists in a system, i.e. heterogeneous nodes in Organic Computing systems, trust has to be build over time. Reputation is used when no direct trust information is available and other participants have to provide their direct trust information. Reputation metrics are thoroughly researched in current literature. Maresch [1] gives an overview over different reputation metrics in the context of the Semantic Web and illustrates that reputation metrics can be divided into two types: global and local reputation metrics. *Global reputation metrics* know the reputation data of the entire network and globally calculate a single reputation value per user. In general, systems in which users have to trust in the honesty or the skills of other user's global reputation metrics are used to rate the trustworthiness of the individual users. The buyer/seller rating systems of eBay or the reviewer rating system of Amazon are good examples for such systems. *Local reputation metrics* on the other hand just use the local knowledge of a user within the system. If a user wants to know the reputation of another user he asks around to gather information from users that had direct interaction with the target user. The gathered data is then aggregated to a reputation trust value of the target user.

Organic Computing systems [2] were developed to handle the growing complexity of distributed systems and make them controllable. To achieve this OC systems introduce the so called self-x properties, self-configuration, self-optimization, self-healing and self-protection to autonomously react on changed situations. In Organic Computing systems global reputation metrics are not appropriate to use, since there is no entity that has a global view over the network. Instead each node gathers its own experience and relays this information to asking nodes. Organic Computing systems typically consist of heterogeneous nodes, whose capabilities, e.g. their connection to the network, may fluctuate over time. This results in a different subjective view each node gathers about the network and its participants. Overall the nodes will build their own subjective view of others in the network thereby adhering to an important property of trust: subjectivity [3].

An important difference between Organic Computing systems and systems involving users is the confinement of OC systems. An OC system as a purely technical system consists of a number of nodes that can be communicated with. Nodes can enter and leave the system but each node can be equally communicated with. Equally here means that there exist no social rules compared to human societies. Every node can initiate a communication with a completely unknown node without the need to adhere to social rules. Nodes in typical reputation systems often ask their neighbors, directly known communication partners, about their opinion of a target node which will get relayed to their neighbors and so on, creating a chain of requests that gets less meaningful the longer it gets. The TidalTrust metric [4] is a good example for this.

The *Neighbor-Trust reputation metric* presented in this paper exploits the ability of equally communicable nodes by identifying the direct neighbors of a node and gathers their direct trust data. This reduces the reputation chain to one. The resulting threat of attacks by lying entities that try to promote not trustworthy entities is reduced by a weight for trust statements and a learning metric. The neighbor relationships are an overlay network presenting the direct communication experiences, and therefore the existence of direct trust values, between different nodes. More precisely a *neighbor* of an entity is an entity that has direct trust values about the target entity. An entity can be either a user (User-to-User Trust) or a node in a network (Device-to-

Device Trust). In Organic Computing systems the entities are nodes. These neighbor relations build a overlay network over the underlying communication network. The communication channel topology can be entirely different and is independent of the neighbor relationship topology.

The remaining paper is structured as follows. Section II gives an overview over related work on reputation metrics in the current literature. Section III presents our reputation metric that is evaluated and discussed in section IV. Finally section V concludes the paper and presents future work.

II. Related Work

Several reputation metrics exist and a general overview can be found in [1]. We restrict the related work to some local trust metrics that influenced the development of our metric.

The trust metrics TidalTrust [4] and Moletrust [5] are very similar and were developed for the use in Trust-Aware Recommender Systems. The aim was to improve the quality of, e.g., product recommendations by taking into account that people feel more confident in recommendations made by friends than made by strangers. Both metrics use trust networks and predict trust values of users that are no neighbors of the active user in a breadth-first respectively a depth-first fashion. The predicted trust statements afterwards are used as a weight in a recommender system. However, TidalTrust as well as Moletrust only take short paths with a minimum strength into account because ratings of users get less precise and useful the more distant they are. By this means, both metrics cannot predict precise trust values of users outside a specific range in the trust network and for this reason, they are not suitable in Organic Computing systems in which all devices, independent of their position inside the network, can interact with each other. Both metrics also require a dense network to work effectively. In Organic Computing systems no such network can be assumed, since the network structure is dynamically fluctuating.

The Regret [6] reputation metric is a local trust metric that also is strongly affected by human and social behavior and especially by the saying “Tell me who you associate with and I will tell you who you are”. Therefore, it is split into three parts: *Witness Reputation, Neighborhood Reputation and System Reputation*. Witness Reputation refers to the direct trust values of the neighbors of a target node, the witnesses. Neighborhood Reputation refers to the reputation the neighbors of the target nodes have. System Reputation is defined as the reputation of the entire system the target node is part of. Thus, Regret is a reputation metric that is not limited to a special part of a trust network. However, it strongly relies on highly subjective thoughts that are typical for human societies but cannot be provided in pure physical systems.

FIRE [7] is a trust and reputation model used in open multi-agent systems and therefore developed to deal with the requirements of an open system. It combines several trust types, namely *Interaction trust, Role-based trust, Witness reputation and Certified reputation*. Interaction trust and Role-based trust refer to direct trust between two participants and are not considered in this paper. (A metric to measure direct trust for the reliability or nodes on middleware level, the Delayed-Ack metric, was already introduced in [8].) Witness reputation and Certified reputation are the reputation mechanics in FIRE. Witness reputation similar to TidalTrust and MoleTrust uses locally close neighbors and searches only up to a maximal length of a reputation chain before aborting. The component we focus on is the Certified reputation component. With this component an agent A is able to ask an agent B to prove its trustworthiness and receives a Certified reputation that consists of ratings agent B received from other agents in past direct interactions. Because this kind of reputation in general can be provided by almost all agents, devices or people this approach is eligible for our system, too. A big drawback of the Certified reputation component is that any entity has to store the ratings received from its neighbors. Our Neighbor-Trust reputation metric solves the problem with long reputation chains by exploiting the ability to identify and to contact an entity’s direct neighbors. Furthermore, the Certified reputation component is vulnerable to attacks from malicious entities or groups of entities that offer false ratings to promote not trustworthy entities.

To solve this problem, we need a trust metric comparable to the Eigen-Trust metric [9]. This metric, for example, is used in file-sharing networks to identify and isolate manipulating people or entities so that the trust values inferred for every person or component reflect the community’s real opinion. We want to extend this metric by using seperated trust values for the direct interaction with entities and for the reputation entities provide about other entities. The reason is, that a bad interaction partner nevertheless could be a good informant and vice versa.

With the Neighbor-Trust metric we will present a local trust metric that is able to identify and isolate manipulating entities, too.

III. Neighbor-Trust reputation metric

The Neighbor-Trust metric is based on metric 2 evaluated by Satzger et al. in [10]. Satzger et al. gathered reputation data from the neighbors of a node, but instead of using a weighted mean metric he used a normal mean metric without weights. Therefore, the Neighbor-Trust metric enhances Satzger’s metric by adding weights and a learning mechanism and is able to identify and isolate malicious entities.

If an entity wants to interact with another entity but does not have any direct experiences yet it asks others in the system about their opinions, their *reputation*. Figure 1 displays the relationship of the different entities.

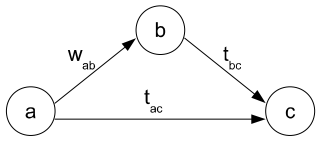

Figure 1. nodes in the reputation graph

Alice (a) wants to get information about Carol (c), so she asks Bob (b) about his opinion about Carol. w_{ab} is the trust Alice gives the information Bob provides respectively the weight she gives his information and t_{bc} is the direct trust value Bob has about Carol. Later Alice might have a direct experience with Carol, displayed by t_{ac}. To get an accurate value, Alice will ask more entities than just Bob. She will ask all neighbors of Carol.

To calculate a total reputation value r the following formula is used:

$$r_{ac} = \frac{\sum_{i \in neighbors(c)} w_{ai} \cdot t_{ic}}{\sum_{i \in neighbors(c)} w_{ai}}$$

This formula corresponds to a weighted mean metric. The values t_{ic} are the direct trust values of the neighbors to the target node c, and the weights w_{ai} represent the personal opinion of the requesting entity a, how much to trust the direct trust values the neighbors provide. These weights are normally independent of the context of the direct trust value the neighbor provides. In a computational environment, where entities are nodes in a network this assumption can hold. In comparison to that, users sometimes vary their weights according to the context, mostly because of “gut feeling”. For example, if Alice asks Bob’s opinion about Carol’s ability to repair cars as well as her ability to craft tables, Alice might weight both information from Bob differently because she subjectively thinks Bob knows more about cars than tables. It is hard to find an exact specification about this “gut-feeling” and since these nuances are highly subjective and tend to be irrational as well, “gut-feeling” is not a mechanic that can be transferred to a computational system. Therefore, in our work the weight will be adjusted independent of the context of the trust request. This also prevents a high fragmentation of information, when a weight is saved for every context.

Because the metric weights received trust statements to reduce the threat of attacks by lying entities we want to learn the weights over time by judging previous statements of neighbors. If a neighbor gave information that corresponded with the node’s own experience then the future statements of the neighbor will be weighted higher than before. Correspondingly, if the experience differs from the recommendation, the weight will be lowered. Figure 2 displays the relationship of the different values.

t_{ac} is the direct trust experience A had with C. t_{bc} is the trust value B told A. If both of these values are close enough then B gave a correct statement about $C's$ ability. Close enough is defined by the threshold value τ. If t_{ac} and t_{bc} differ by maximally τ then the weight w will be gradually increased, with the most increase if both value exactly match. If they differ by more than τ, w will be decreased, with a gradually stronger decrease the further t_{bc} differs from $t_{ac} \pm \tau$. In addition a single transaction should only change w by a maximal amount, called θ. Limiting the adjustment of w prevents extreme jumps in w. A single bad recommendation should not set w close to 0. For this the threshold τ^* is introduced. It defines an area, where w will be decreased up to θ. If t_{bc} is even beyond t_{ac} +/- τ^* then w will be fully decreased by θ. This leads to the following formula:

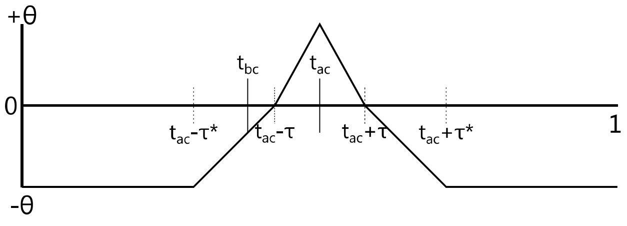

Figure 2. thresholds

$$w_{ab}^{n+1} = w_{ab}^{n} \begin{cases} +(\frac{\tau - |t_{ac}^{n} - t_{bc}^{n}|}{\tau}) \cdot \theta, & \text{if } 0 \leq |t_{ac}^{n} - t_{bc}^{n}| \leq \tau \\ -(\frac{|t_{ac}^{n} - t_{bc}^{n}| - \tau}{\tau^* - \tau}) \cdot \theta, & \text{if } \tau < |t_{ac}^{n} - t_{bc}^{n}| \leq \tau^* \\ -\theta, & \text{otherwise} \end{cases}$$

where w_{ab}^{n} means the weight a has of b at time n (analog for t).

The formula is divided into three cases:

1) **positive case**: w will be increased by the percentage of how close t_{ac} and t_{bc} are together. If $t_{ac} = t_{bc}$ then w will be increased by θ. If $t_{bc} = t_{ac} \pm \tau$ then w remains unchanged.
2) **negative case**: Analog to the case before, w will be decreased by an amount of θ if t_{bc} is located between $t_{ac} \pm \tau$ and $t_{ac} \pm \tau^*$. Being in the middle between both thresholds would decreases w by $\frac{1}{2}\theta$.
3) **extreme negative case**: In this case both values differ so much from each other that w will be maximally decreased by θ.

By adjusting the weights over time neighbors that give bad recommendations are identified and their opinions weighted down. All further reputation requests will return more trustworthy and accurate recommendations than the first requests, since the opinion of lying nodes was rated down. A node learns the trustworthiness of its communication partners over time, similar to persons who enter a new community and have to get to know their communication partners.

To exploit the new-found information the initial weight of new nodes can be adjusted. By starting with $w = 1$

every node is trusted until proved otherwise. After some nodes are identified as trustworthy nodes, keeping a high weight w after several transaction, an exploitation of these nodes compared to new nodes is desirable. By starting at $w = 0.5$ a node would start with mixed feelings towards a new node. This decreases the probability to gain bad recommendations by newcomers that are not known so far. This is highly relevant in Organic Computing systems that are characterized by highly dynamic configurations. By reducing the initial weight, nodes with knowingly high trust, will have higher weight and their recommendations will be preferred, while at the same time information about new nodes can be learned with a low risk of getting a bad interaction later on.

This leads to another problem. By leaving the network, assuming a new identity and rejoining the network, a node with bad reputation can start anew. This is especially problematic if gaining a new identity is cheap[11]. This is a fundamental problem with identification. In real life building up a new identity is a costly operation in therefore not used by normal society members. So far, we do not consider the id problem and presume that a node is always correctly identified.

IV. Evaluation

To evaluate the reputation metric we created a network of 10 nodes. Evaluation with more nodes yielded similar results, but with 10 nodes more specific effects where visible. A specific percentage of these nodes were malicious, i.e., they were always rating their direct communication partners badly, independent of their response. These malicious nodes are returning unsatisfactory results to service request and try to sabotage the network. By giving bad ratings to honest nodes they try to reduce their reputation value and attract working requests.

To find the neighbors of a target node, all reputation requests are originally sent to the target node that relays the request to all its neighbors. A node identifies a neighbor by observing its message flow. Every node that received a message from the target node is considered a neighbor, so all nodes that were sent messages are saved as neighbors by the target node.

We investigated how good our reputation metric is able to identify the malicious nodes and how successful their trustworthiness of giving correct reputation information about other nodes gets rated down, thereby isolating them from the network in the process. The trustworthiness matches the weight of the node in our metric.

To identify the limits of the metric we ran simulations with different percentage of malicious nodes: 30%, 50% and 70%. As evaluator we used the Trust-Enabling Middleware (TEM) [12], an organic middleware developed in the OC-TRUST project[1]. The TEM is a middleware system that enables services to run on it, to save their experiences into the middleware and to initiate direct trust and reputation calculations. A malicious node is a TEM node with a malicious service running it. The TEM middleware is based on the OCμ[13] middleware, an organic middleware developed in the Deutsche Forschungsgemeinschaft (DFG) priority program 1183[2] on Organic Computing.

In every timestep a node selects another node as communication partner and sends it a message. As selection metric the node asks about the reputation of all other nodes and takes the one with the highest reputation. If several nodes are tied, e.g. in the start when no information yet exists, the communication partner is selected randomly. The node replies with a positive (honest node) or negative (malicious node) result. After the transaction the result is rated and the rating is saved into the TEM. A honest node saves the corresponding value, either 1.0 for successful or 0.0 for unsuccessful transactions, whereas the malicious nodes always rate its communication partner with 0.0.

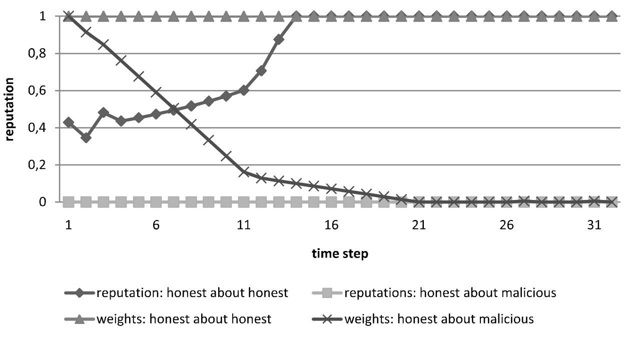

Figure 3. 30% malicious nodes

Figure 3 shows a simulation with 30% malicious nodes. The reputation and weight development of honest nodes can be seen over time. The values are averaged over the honest and malicious nodes. "Reputation: honest about malicious", for example, means the average reputation honest nodes have about malicious nodes. All weights start at 1, which is the initial value, if no information is available. The weight of honest nodes remains constantly at 1, since these nodes always give correct answers and therefore get never penalized. Since malicious nodes lie about other nodes their weights get reduced until reaching 0, effectively removing their reputation information from the aggregated value.

The reputation value changes accordingly to the weights. In the start the reputation of good nodes about other good nodes is lower than 1, since the malicious nodes give false and bad reputations. When the weight of the malicious nodes is reduced, their recommendations get more and more ignored until all are identified and only the honest nodes are considered, rising their reputation up to 1. Malicious nodes

[1]http://www.informatik.uni-augsburg.de/lehrstuehle/swt/se/projects/oc-trust/

[2]http://www.organic-computing.de/spp

have constantly bad reputation of 0, because their work is not satisfying, which results of bad ratings from honest nodes. Malicious nodes always rate other nodes bad, which leads to bad ratings from all nodes for malicious nodes. This explains the constant bad reputation of these nodes.

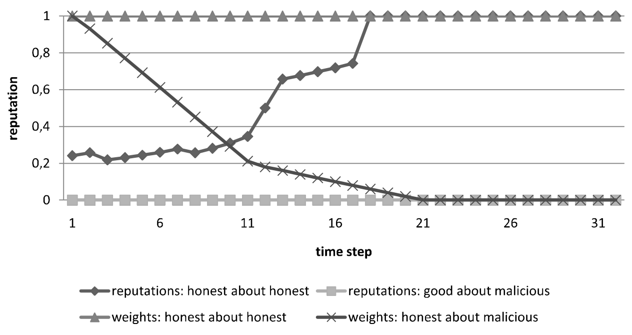

Figure 4. 50% malicious nodes

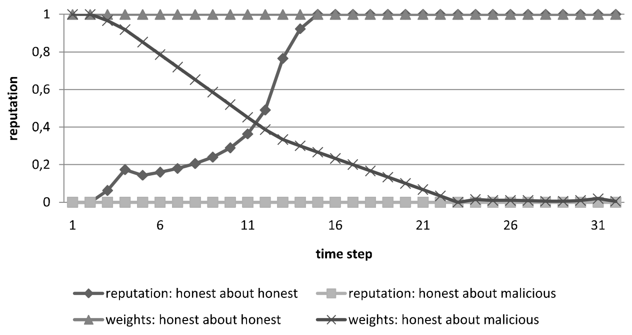

Figure 5. 70% malicious nodes

In figure 4 and 5 we increased the amount of malicious nodes to 50% and 70% respectively. The development of the curves is similar to the 30% case, which shows that our metric also works for systems with a high percentage of malicious nodes, even with over 50% malicious nodes. This demonstrates the robustness of the metric. Since the reputation of malicious nodes is going to 0, they will not be chosen anymore as interaction partner, thereby isolating them from the network.

In figure 6 we noticed an interesting effect. A single honest node, in this case node7, had a constant reputation value of 0 from other honest nodes. This happened because by chance all honest nodes chose another node, node8 here, as communication partner and continued to do so, since they received satisfactory results. This resulted in a situation, where no honest node had a direct interaction with the shown node7. This lead to no honest trust ratings about node7, just negative results of 0 from malicious nodes. In total this kept the reputation of node7 at a constant value of 0. The reason is the selection algorithm used to select a communication partner. The selection algorithm was choosing a random

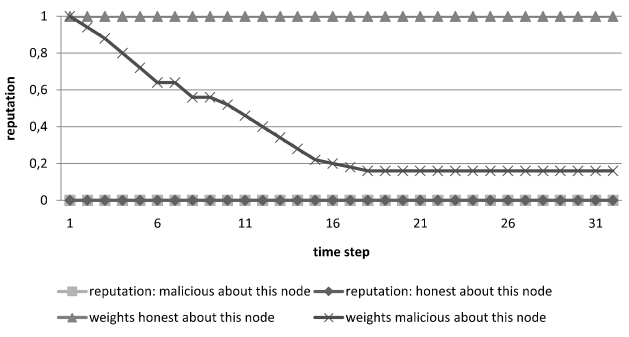

Figure 6. An isolated node with 50% malicious nodes

node if several nodes had the same reputation value. If a satisfactory node was chosen no more exploration for further possible communication partners was done. To counter this, nodes have to restart exploring, e.g. with not so important tasks, to create a good mix of direct trust values.

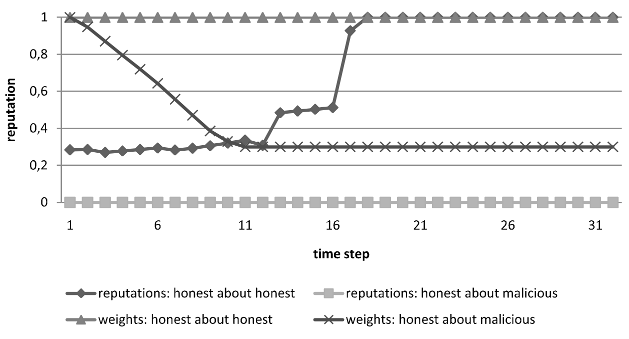

Figure 7. 30% malicious nodes with threshold 0.5

Another point we noticed was specific by using a middleware system. Since the TEM middleware provided the capacities to save the gathered measurements and calculate trust values from them, a malicious service, running on a TEM node, was unable to distinguish its wrongly written observations from rightly written observations from other honest services. The malicious service wrote its wrong observation into the middleware but when a reputation value was calculated these wrong results were indistinguishable from honest observations. This lead to disadvantages for the malicious services, that would require a lot of extra work to circumvent. Since the reputation calculations happen within the TEM middleware an extra malicious TEM node would have to be created, instead just a malicious service.

Figure 7 shows a related effect. We can see that the average weight of honest nodes about malicious nodes stays stable above 0. In this example the following situation occurred: Honest and malicious nodes interacted with a different set of nodes in the end. This resulted in no more updates to weights, since no additional comparison between direct interaction and reputations were possible. Apart from that some malicious nodes never interacted with the interaction partners of the honest nodes and thereby no more reputation information about them are given, leaving the mean above 0. In total this was a combination of several factors: Malicious nodes set their weights to honest nodes to 0 after some time. Since reputation calculations and weight adjustment happen internally in the TEM the malicious nodes cannot influence these calculations. Therefore, honest nodes return a reputation value of 1 while malicious nodes rate all direct interactions with 0, which leads, being different, to a weight reduction down to 0. Since malicious nodes always return 0 as trust value for all other nodes this resulted for all nodes to be considered untrustworthy by all malicious nodes. All honest nodes chose only two other nodes to interact with and stayed with them. The malicious nodes stayed with different nodes. In this situation the malicious nodes interacted with different nodes than the honest nodes. Therefore, no more weight adjustments are done, since no more additional direct comparisons that bring new results are performed. This situation was favored by the fact, that unknown nodes were seen as equivalent to nodes with 50% reputation by the selection algorithm. In previous examples we set unknown nodes equivalent to 0% nodes.

V. Conclusion and Future Work

We have shown that the Neighbor-Trust reputation metric can identify lying malicious nodes and isolate them from the network. Their further recommendations get weighted lower and lower until they are no longer taken into account. We have also seen that the selection algorithm for possible interaction partners, as well as a recurring exploration phase, is important to get meaningful reputation values in the long run.

In future work we want to explore different neighbor finding algorithms. By relaying reputation requests to its neighbors the target node is able to lie about its neighbors and only relay the request to nodes that will return a good rating. This is related to a modification to malicious nodes: malicious communities. Malicious communities consist of malicious nodes that know each other and rate each other positively while lying about all others. We want to evaluate our reputation metric using malicious communities.

Finally we want to investigate the addition of direct trust values into the selection of communication partners. In this paper we only took reputation into consideration, but existing direct trust values should be included into the decision as well. This also could include the trustworthiness, in our case the weight, of the other nodes. By adding such information not only nodes that return satisfying results but are also honest can be selected. We want to examine possible ways, how to weight reputation to direct trust and explore possible aggregation mechanisms.

References

[1] O. M. Maresch, "Reputationsbasierte Trust Metriken im Kontext des Semantic Web," Master's thesis, Technische Universität Berlin, 2005.

[2] U. Richter, M. Mnif, J. Branke, C. Müller-Schloer, and H. Schmeck, "Towards a generic observer/controller architecture for Organic Computing," in *GI Jahrestagung (1)*, 2006, pp. 112–119.

[3] S. P. Marsh, "Formalising Trust as a Computational Concept," Ph.D. dissertation, University of Stirling, 1994.

[4] J. A. Golbeck, "Computing and applying trust in web-based social networks," Ph.D. dissertation, University of Maryland, College Park, MD, USA, 2005, chair-Hendler, James.

[5] P. Massa and P. Avesani, "Trust metrics on controversial users: balancing between tyranny of the majority and echo chambers," *International Journal on Semantic Web and Information Systems*, 2007. [Online]. Available: http://www.gnuband.org/papers/trust_metrics_on_controversial_users_balancing_between_tyranny_of_the_majority_and_echo_chambers-2/

[6] J. Sabater Mir, "Trust and reputation for agent societies," Ph.D. dissertation, Universiat Autònoma de Barcelona, 2002.

[7] T. D. Huynh, N. R. Jennings, and N. R. Shadbolt, "An integrated trust and reputation model for open multi-agent systems," *Autonomous Agents and Multi-Agent Systems*, vol. 13, no. 2, pp. 119–154, 2006.

[8] R. Kiefhaber, B. Satzger, J. Schmitt, M. Roth, and T. Ungerer, "Trust Measurement Methods in Organic Computing Systems by Direct Observation," in *Proceedings of the 8th International Conference on Embedded and Ubiquitous Computing (EUC 2010)*. IEEE, 2010, pp. 105–111.

[9] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the 12th international conference on World Wide Web*, ser. WWW '03. New York, NY, USA: ACM, 2003, pp. 640–651. [Online]. Available: http://doi.acm.org/10.1145/775152.775242

[10] B. Satzger, F. Mutschelknaus, F. Bagci, F. Kluge, and T. Ungerer, "Towards trustworthy self-optimization for distributed systems," in *Software Technologies for Embedded and Ubiquitous Systems, 7th IFIP WG 10.2 International Workshop, SEUS 2009, Newport Beach, CA, USA, November 16-18, 2009, Proceedings*, 2009, pp. 58–68.

[11] E. J. Friedman and P. Resnick, "The social cost of cheap pseudonyms," *Journal of Economics and Management Strategy*, vol. 10, pp. 173–199, 2000.

[12] R. Kiefhaber, F. Siefert, G. Anders, T. Ungerer, and W. Reif, "The Trust-Enabling Middleware: Introduction and Application," Universitätsbibliothek der Universität Augsburg, Augsburg, Tech. Rep. 2011-10, 2011.

[13] M. Roth, J. Schmitt, R. Kiefhaber, F. Kluge, and T. Ungerer, "Organic Computing Middleware for Ubiquitous Environments," in *Organic Computing - A Paradigm Shift for Complex Systems*. Springer Basel, 2011, pp. 339–351.