Trust Management for Service Composition in SOA-based IoT Systems

Ing-Ray Chen, Jia Guo and Fenye Bao
Virginia Tech
Department of Computer Science
{irchen, jiaguo, baofenye}@vt.edu

Abstract — **An Internet of Things (IoT) system connects a large amount of tags, sensors, and smart devices often with mobility to facilitate information sharing, enabling a variety of attractive applications. On the one hand, the service oriented architecture (SOA) can provide connectivity and interoperability among heterogeneous IoT devices in the physical network. On the other hand, IoT devices are virtually connected via social networks. In this paper, we analyze the notion of adaptive trust management to support reliable service composition applications in SOA-based IoT systems. Each device records user satisfaction experiences toward devices with which it has interacted, and collects trust feedbacks from other devices sharing social interests. We consider friendship, social contact, and community of interest social relationships to select trust feedbacks. Further we develop a novel adaptive filtering technique to determine the best way to combine direct trust and indirect trust feedbacks dynamically to minimize both convergence time and trust bias. We demonstrate the effectiveness of the proposed trust management through a service composition application in SOA-based IoT systems.**

Keywords— **Trust management; Internet of things; social networks; service composition; SOA; performance analysis.**

1 INTRODUCTION

An Internet of Things (IoT) system connects the physical world into cyberspace via radio frequency identification (RFID) tags, sensors, and mobile devices. IoT systems challenge trust management in the following aspects. First, an IoT system evolves with new nodes joining and existing nodes leaving. A trust management protocol must address this issue to allow newly joining nodes to build up trust quickly with a reasonable degree of accuracy. Second, the building blocks or entities of IoT systems are mostly human carried or human operated devices [2], so trust management must take into account social relationships among device owners in order to maximize protocol performance. Lastly, a social IoT system essentially consists of uncensored IoT devices providing a wide variety of services. Inherently, many of them (the owners) will be malicious for their own gain. A trust management protocol for IoT must be resilient to malicious attacks to survive in hostile environments.

The problem we aim to solve is design and validation of an adaptive and survivable trust management protocol for SOA-based social IoT systems [1] capable of answering the challenges discussed above. The trust management protocol must be executed autonomously by SOA-based IoT devices with little human intervention. The goals are two-fold: (a) trust bias minimization; (b) application performance optimization. This is achieved by dynamic trust management, i.e., adjusting trust protocol parameters in response to environment changes dynamically. We illustrate application performance optimization via a service composition example.

Despite the abundance of trust protocols for P2P and mobile ad hoc and sensor networks [8, 9], there is little work on trust management for IoT systems. Chen et al. [7] proposed a trust management model based on fuzzy reputation for IoT. However, their trust management model considers a specific IoT environment consisting of only wireless sensors with QoS trust metrics such as packet forwarding/delivery ratio and energy consumption, and does not take into account the social relationship which is important in social IoT systems. Bao and Chen [3, 4] proposed a trust management protocol considering both social trust and QoS trust metrics and using both direct observations and indirect recommendations to update trust. Their proposed trust management protocol considers a social IoT environment where environment conditions are dynamically changing, and interaction pattern changes. To address the scalability issue, Bao and Chen further proposed a scalable trust management protocol [5] for large-scale IoT systems by utilizing a scalable storage management strategy. Relative to prior work, we have the following contributions: (1) we utilize distributed collaborating filtering [12] to select trust feedbacks from nodes sharing similar social interests; (2) we develop a novel adaptive filtering technique to dynamically adjust trust parameter settings so as to minimize trust estimation bias; (3) we apply the proposed trust management to a trust-based service composition application in SOA-based IoT systems [1] to demonstrate application performance optimization; and (4) we validate the proposed trust management and its application through simulation based on real trace data [6].

2 SYSTEM MODEL

We consider a social SOA-based IoT environment [1, 2] where nodes are physical connected via communication networks and socially connected via users' social networks. Each node has a unique address to identify (i.e., URI). There is no centralized trusted authority. There are two types of nodes: devices and users (or owners). The user-device relationship is a one-to-multiple relationship. In our trust management, the trustor is a user and the trustee is a device (owned by another user). For each user, the trust evaluation information is computed and stored in a designated high-end device owned by the user.

We consider the following three social relationships: *friendship*, *social contact*, and *community of interest* (*CoI*). The reason is that device owners that are friends, have frequent social contacts in common locations, or share similar community of interests tend to have close social relationships and their referrals or recommendations are considered more trustable or reliable than a complete stranger [2]. These social relationships are represented by three lists: a *friend list* with current friends, a *location list* with locations frequently visited for social contact, and a *CoI list* with devices (services) directly interacted with. Each user has at least one designated high-end device (i.e., smart phone and laptop) storing these lists in the user's profile (see Figure 1). Other devices of the same user have the privilege to access the profile. By delegating the storage and computation of social networks to a high-end device for each user, many low-end devices (i.e., sensors) are able to share and utilize the same social information to maximize its performance.

Each time when device $d1$ requests a service from device $d2$, $d1$ updates the user satisfaction experience record (in the *user satisfaction experience list* in Figure 1) towards $d2$ stored in the designated device of $d1$'s user. Similarly, $d1$ can query the trust information (in the *trust list* in Figure 1) towards $d2$ from the designated device of $d1$'s user. Please note that elements in the user interaction experience list correspond to devices in the *CoI list.*

In the context of SOA, an owner provides services via its IoT devices. An IoT device providing a service will have to compete with other IoT devices which provide a similar type of service. A malicious IoT device (because its owner is malicious) can perform the following attacks for its own gain:

Self-promoting attacks: it can promote its importance (by providing good recommendations for itself) so as to be selected as the service provider, but then can provide bad or malfunctioned service.

Bad-mouthing attacks: it can ruin the reputation of a well-behaved device (by providing bad recommendations against it) so as to decrease the chance of that good device being selected as a service provider.

Ballot stuffing attacks: it can collude with a bad device and boost the reputation of the bad device (by providing good recommendations) so as to increase the chance of that bad device being selected as a service provider.

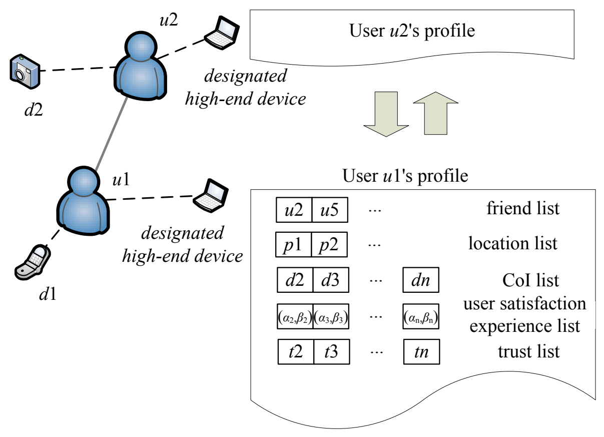

Figure 1: User Profile.

3 TRUST MANAGEMENT PROTOCOL

Our trust management protocol for IoT systems is distributed. Each user maintains its own trust assessment towards devices. For scalability, a user just keeps its trust evaluation results towards a limited set of devices of its interests. Each user stores its profile in a designated high-end device (Figure 1). The profile of user u_x includes:

(1) A "friend" list including all friends of u_x, denoted by a set $F_x = \{u_a, u_b, \dots\}$;
(2) Locations that u_x frequently visited for social contact, denoted by a set $P_x = \{p_{x,1}, p_{x,2}, \dots\}$;
(3) List of devices that u_x has directly interacted with and the corresponding user satisfaction experience values, denoted by set $D_x = \{d_i, d_j, \dots\}$ and set $B_x = \{(\alpha_{x,i}, \beta_{x,i}), (\alpha_{x,j}, \beta_{x,j}), \dots\}$, where $\alpha_{x,i}$ and $\beta_{x,i}$ are the accumulated positive and negative user satisfaction experiences of user u_x towards device d_i;
(4) Trust values of user u_x towards IoT devices, denoted by a set $T_x = \{t_{x,i}, t_{x,j}, \dots\}$.

3.1 Direct Interaction Experiences

We adopt Bayesian framework [14] as the underlying model for evaluating *direct trust* from direct user satisfaction experiences. The reason we choose Bayesian because it is well-established and because of its popularity in trust/reputation systems. In service computing, a service requester could rate a service provider after direct interaction based on nonfunctional characteristics. The nonfunctional characteristics include user-observed response time, failure probability, prices, etc. The current user satisfaction experience of user u_x toward device d_i is represented by a value, $f_{x,i}$. We consider the simple case in which the direct user satisfaction experience $f_{x,i}$ is a binary value, with 1 indicating satisfied and 0 not satisfied. Then, we can consider $f_{x,i}$ as an outcome of a Bernoulli trial with the probability of success parameter $\theta_{x,i}$ following a Beta distribution (a conjugate prior for the Bernoulli distribution), i.e., Beta($\alpha_{x,i}$, $\beta_{x,i}$). Then, the posterior $p(\theta_{x,i}|f_{x,i})$ has a Beta distribution as well, i.e., Beta($\alpha_{x,i} + f_{x,i}$, $\beta_{x,i} + 1 - f_{x,i}$). Equation 1 shows how the hyper-parameters $\alpha_{x,i}$ and $\beta_{x,i}$ are updated considering trust decay.

$$\alpha_{x,i} = e^{-\varphi\Delta t} \cdot \alpha_{x,i}^{(old)} + f_{x,i}$$
$$\beta_{x,i} = e^{-\varphi\Delta t} \cdot \beta_{x,i}^{(old)} + 1 - f_{x,i} \tag{1}$$

In Equation 1, $f_{x,i}$ contributes to positive observations and $1 - f_{x,i}$ contributes to negative observations. When updating $\alpha_{x,i}$ and $\beta_{x,i}$, we consider an exponential decay, $e^{-\varphi\Delta t}$, on $\alpha_{x,i}^{(old)}$ and $\beta_{x,i}^{(old)}$, where φ is the decay factor which is normally is a small number to model small trust decay over time, and Δt is the trust update interval.

The direct trust of user u_x to device d_i, $t_{x,i}^d$, is calculated as the expected value of $\theta_{x,i}$, i.e.,

$$t_{x,i}^d = E[\theta_{x,i}] = \frac{\alpha_{x,i}}{\alpha_{x,i} + \beta_{x,i}} \tag{2}$$

In the literature, $\alpha_{x,i}$ and $\beta_{x,i}$ are initially set to 0 or 1 since no prior knowledge available. In this paper, we consider the social relationships (if available) between u_x and the user of d_i (say u_y) as the prior knowledge and set initial values of $\alpha_{x,i}$ and $\beta_{x,i}$ to $sim(u_x, u_y)$ and $1 - sim(u_x, u_y)$, respectively, where $sim(u_x, u_y)$ is the similarity between u_x and u_y, characterizing their social connections. This is discussed in Section 3.2 below.

3.2 Recommendations

When the devices of two users have direct interactions, they can exchange their profiles and provide trust recommendations. In addition, a device can also aggressively request trust recommendations from another device belonging to a friend if necessary. To preserve privacy, one can use a hash function (with session key) to prevent the identities of uncommon friends/devices from being revealed. Our protocol design is that a node will first measure its "social similarity" with a recommender in friendship, social contact (representing physical proximity) and CoI (representing knowledge on the subject matter) and then decide if the recommendation is trustable. The three social similarity measures are estimated dynamically as follows:

- **Friendship Similarity** (sim_f): The friendship similarity is a powerful social relationship (intimacy) for screening recommendations. After two users u_x and u_y exchange their friend lists, F_x and F_y, they could compute two binary vectors, $\overrightarrow{VF_x}$ and $\overrightarrow{VF_y}$, each with size $|F_x \cup F_y|$. An element in $\overrightarrow{VF_x}$ (or $\overrightarrow{VF_y}$) will be 1 if the corresponding user is in F_x (or F_y), otherwise 0. Let $\|\vec{A}\|$ be the norm of vector $\vec{A}$ and $|B|$ be the cardinality of set B. Then, we could use the "cosine similarity" of $\overrightarrow{VF_x}$ and $\overrightarrow{VF_y}$ (giving the cosine of the angle between them) to compute sim_f as follows:

$$sim_f(u_x, u_y) = \frac{\overrightarrow{VF_x} \cdot \overrightarrow{VF_y}}{\|\overrightarrow{VF_x}\|\|\overrightarrow{VF_y}\|} = \frac{|F_x \cap F_y|}{\sqrt{|F_x| \cdot |F_y|}}$$

- **Social contact Similarity** (sim_l): The social contact similarity presents closeness and is an indication if two nodes have the same physical contacts and thus the same sentiment towards devices which provide the same service. The operational area could be partitioned into sub-grids. User u_x records the IDs of sub-grids it has visited in its *location list* P_x for social contact. After two users u_x and u_y exchange their location lists, P_x and P_y, they could compute sim_l in the same way of computing sim_f as follows:

$$sim_l(u_x, u_y) = \frac{|P_x \cap P_y|}{\sqrt{|P_x| \cdot |P_y|}}$$

- **Community of Interest Similarity** (sim_c): Two users in the same COI share similar social interests and most likely have common knowledge and standard toward a service provided by the same device. Also very likely two users who have used services provided by the same IoT device can form a CoI (or are in the same CoI). After two users u_x and u_y exchange their device lists, D_x and D_y, they could compute sim_c in the same way of computing sim_f as follows:

$$sim_c(u_x, u_y) = \frac{|D_x \cap D_y|}{\sqrt{|D_x| \cdot |D_y|}}$$

The social similarity between two users can be a weighted combination of all social similarity metrics, i.e., friendship, social contact, and community of interest, considered in this paper:

$$sim(u_x, u_y) = \sum_{v \in \{f,l,c\}} w_v \cdot sim_v(u_x, u_y) \tag{3}$$

where $w_f + w_l + w_c = 1$ and $0 \le w_f, w_l, w_c \le 1$. Each user can send trust recommendations request to its friends periodically (Δt interval) or before requesting a service. Upon receiving recommendations, user u_x selects top-k recommendations from k users with the highest similarity values with u_x and calculates the indirect trust ($t_{x,i}^r$) towards device d_i as follows:

$$t_{x,i}^r = \frac{\sum_{u_y \in U} sim(u_x, u_y) \cdot t_{y,i}^d}{\sum_{u_y \in U} sim(u_x, u_y)} \tag{4}$$

Here, U is a set of up to k users whose $sim(u_x, u_y)$ values are the highest, and $t_{y,i}^d$ is the direct trust of user u_y toward device d_i serving as u_y's recommendation toward d_i provided to u_x. Here we note that if u_y is malicious then it can provide $t_{y,i}^d$=0 against a good device for bad-mouthing attacks, and $t_{y,i}^d$=1 for a bad node for ballot stuffing attacks.

3.3 Adaptive Control of the Weight Parameter

The trust value of user u_x toward d_i is denoted as $t_{x,i}$ and is obtained by combining direct trust and indirect recommendations (if available) as follows,

$$t_{x,i} = \mu \cdot t_{x,i}^d + (1 - \mu) \cdot t_{x,i}^r \tag{5}$$

Here, μ is a weight parameter $(0 \leq \mu \leq 1)$ to weigh the importance of direct trust relative to indirect trust feedback. The selection of μ is critical to trust evaluation. A contribution of the paper is that we propose a method based on adaptive filtering [12] to adjust μ dynamically in order to improve trust evaluation performance. The basic design principle is that a successful trust management protocol should provide high trust toward devices who have more positive user satisfaction experiences and, conversely, low trust toward those with more negative user satisfaction experiences. Specifically, the current trust evaluation (i.e., $t_{x,i}(\mu)$ as a function of μ) should be as close to the average user satisfaction experiences observed over that last trust update window Δt. Therefore, we formulate the selection of μ as an optimization problem as follows:

$$\begin{aligned}&\underline{\text{Find}}\text{: } \mu, 0 \leq \mu \leq 1\\ &\underline{\text{Minimize}}\text{: } \mathrm{MSE}(\mu) = \textstyle\sum_i \left(t_{x,i}(\mu) - \overline{f_{x,\iota}^{(new)}}\right)^2\end{aligned} \tag{6}$$

Here, $t_{x,i}(\mu)$ is obtained from Equation 5 using past direct user satisfaction experiences and indirect trust feedback, and $\overline{f_{x,\iota}^{(new)}}$ is the most recent direct user satisfaction experiences observed by user u_x within the last trust update interval Δt. The minimization objective can be achieved by minimizing the mean square error (MSE) of trust evaluations against actual user satisfaction experiences towards all applicable devices, such that the trust value could be a good indicator or predictor for quality of service (with direct user satisfaction experiences considered as ground truth). After user u_x obtains new user satisfaction experiences over Δt, it can compute the average user satisfaction experience value $\overline{f_{x,\iota}^{(new)}}$ and update μ by minimizing MSE in Equation 6. The optimization problem in Equation 6 can be solved by plugging $t_{x,i}(\mu)$ in Equation 5 into Equation 6 and minimizing $\mathrm{MSE}(\mu)$ as follows:

$$\mathrm{MSE}(\mu) = \sum_i \left(\mu \cdot t_{x,i}^d + (1-\mu) \cdot t_{x,i}^r - \overline{f_{x,\iota}^{(new)}}\right)^2 \tag{7}$$

The minimum value of $\mathrm{MSE}(\mu)$ is obtained at the point where the derivative is zero, i.e., $\mathrm{MSE}'(\tilde{\mu}) = 0$. Thus, $\tilde{\mu}$ 1s obtained as follows,

$$\tilde{\mu} = \frac{\sum_i \left(\overline{f_{x,\iota}^{(new)}} - t_{x,i}^r\right)\left(t_{x,i}^d - t_{x,i}^r\right)}{\sum_i \left(t_{x,i}^d - t_{x,i}^r\right)^2} \tag{8}$$

The optimal value of μ (i.e., $\hat{\mu}$) should be in the range of [0, 1] because of it is a weight parameter, therefore,

$$\hat{\mu} = \begin{cases} 0 & \tilde{\mu} < 0 \\ \tilde{\mu} & 0 \leq \tilde{\mu} \leq 1 \\ 1 & \tilde{\mu} > 1 \end{cases} \tag{9}$$

Each user maintains its own optimal value of μ (i.e., $\hat{\mu}$) and updates it dynamically in very time interval Δt. This adaptive design is applicable to other trust parameters (i.e., λ and (w_f, w_l, w_c)) as well. However, introducing these trust parameters in Equation 6 leads to a more complex optimization problem and may not be feasible for IoT devices with limited resources.

Table 1: Parameter List and Default Values Used.

parameter	value	parameter	value	parameter	value
N_T	400	$m \times m$	16×16	T	200hrs
N	40	P_M	20%	φ	0.001
Δt	2 hrs	σ_c	0.01	λ	1/day

4 TRUST PROTOCOL PERFORMANCE

In this section, we report ns3 simulation results obtained as a result of executing our proposed autonomous trust management protocol by IoT devices. Table 1 lists the default parameter values. We consider an IoT environment with N_T = 400 heterogeneous smart objects/devices. These IoT devices are randomly assigned to N = 40 users. Users are connected in a social network represented by a friendship matrix [13]. We consider these users moving according to the SWIM mobility model [11] modeling human social behaviors in an $m \times m$=16×16 operational region for the purpose of assessing the social contact similarity metric between any pair of users. Direct trust of node i toward node j is assessed upon completion of a service request from node i to node j. Each node requests services from a selected device with a time interval following an exponential distribution with parameter λ, with 1/day being the default unless otherwise specified. The trust update interval Δt is 2 hours at which time if there is no direct trust update due to service request and completion, direct trust will be decayed according to Equation 1. Indirect trust is always updated in every Δt interval according to Equation 4.

The user satisfaction levels of service invocations are generated based on a real dataset [6] and are used as "ground truth" based on which the accuracy of our trust protocol is assessed. As the direct trust of user u_x toward device/service provider d_i (i.e., $t_{x,i}^d$) is calculated based on "ground truth" interaction experiences per Equation 1, $t_{x,i}^d$ essentially is equal to ground truth. However, we account for the presence of noise in the IoT environment (i.e., error of assessing user satisfaction level received) by considering a standard deviation parameter σ_c (set to 1% as default) to reflect the deviation of the actual user satisfaction level as recorded in the database from the direct trust evaluation outcome in terms of $t_{x,i}^d$. Initially, $t_{x,i}$ is set to 0.5 (ignorance) by user u_x for all i's. Then, trust is updated dynamically as nodes encounter each other, as services are requested and rendered, and as trust feedbacks are acquired. We consider $w_f = w_l = w_c = 1/3$ (in Equation 3) for the three social relationships considered for the calculation of social similarity and indirect trust $t_{x,i}^r$.

We test the resiliency of our trust protocol against malicious node behavior (i.e., performing self-promotion, bad-mouthing and ballot-stuffing attacks) by randomly selecting a percentage P_M out of all as dishonest malicious nodes with P_M=20% as the default. A normal or good node follows the execution of our trust management protocol faithfully, while a malicious node provides false trust feedback by means of ballot stuffing, bad-mouthing, and self-promoting attacks to

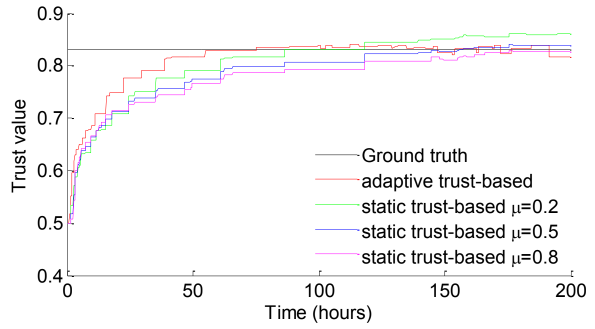

Figure 2: Convergence Behavior.

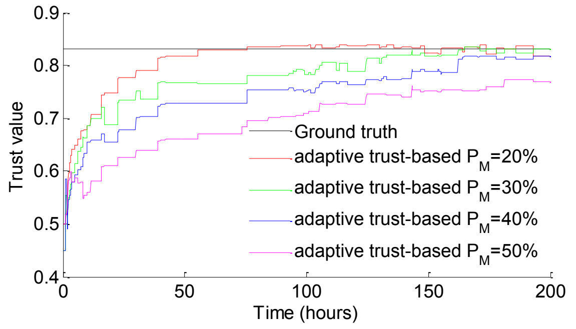

Figure 3: Resiliency against Increasing Malicious Node Population.

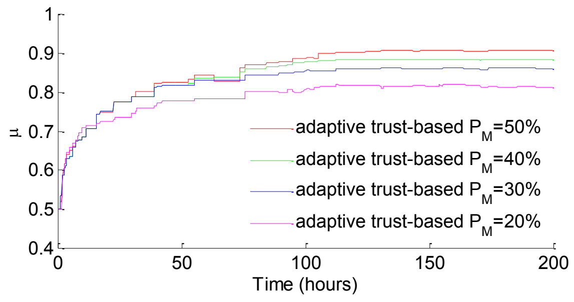

Figure 4: Adjustment of μ against Increasing Malicious Node Population.

gain advantage.

Our simulation results have two parts. First, we demonstrate the trust convergence behavior of our IoT trust protocol design. Second, we show trust bias is effectively minimized after convergence by applying our adaptive control design.

4.1 Trust Convergence Behavior

We examine the trust convergence behavior of our trust protocol design. We compare static control (i.e., μ is fixed at a constant) vs. adaptive control (i.e., μ is changed dynamically based on Equation 9). Figure 2 shows trust evaluation results for a trustor node toward a "good" trustee node randomly picked. We see that trust convergence behavior is observed for either fixed or adaptive control. There is a tradeoff between convergence time vs. trust bias. With static control, when a higher μ value is used, the trust convergence time is longer, but the trust bias is smaller, i.e., the trust value is closer to ground truth after convergence. With adaptive control, on the other hand, the trustor node is able to adjust μ dynamically to minimize both the convergence time and the trust bias after convergence.

4.2 Resiliency against Malicious Attacks

Figure 2 is for the case in which the percentage of malicious nodes P_M = 20%. We conduct experiments to test the residency of our trust protocol against increasing malicious node population. The results are shown In Figure 3. We see that as the population of malicious nodes increases, both the convergence time and trust bias increase. However, the system is found to be resilient to malicious attacks for P_M as high as 40%, with proper convergence and accuracy behaviors exhibited. In general we observe that the trust bias is minimum, e.g., < 5% when $P_M \leq$ 40% and the trust bias becomes more significant, e.g., > 10% when $P_M \geq$ 50%. This demonstrates the resiliency property of our trust protocol against malicious attacks.

Corresponding, Figure 4 shows how our trust-based adaptive control protocol adjusts μ in Equation 5 in response to increasing malicious node population. The observation is that as the malicious node population increases, the system will have to rely more on direct trust by increasing μ to mitigate the effect of bad-mouthing and ballot-stuffing attacks by malicious nodes. Figure 4 shows that when P_M = 20%, the optimal converged μ value is 0.76 while when P_M = 40%, the optimal converged μ value is 0.87. The system cannot rely on direct trust 100% because malicious nodes can also perform self-promoting attacks and there is an error of assessing direct trust due to noise in the environment. Figure 4 demonstrates that our adaptive control mechanism is effective to converge μ to its optimal value under which trust bias is minimized.

5 TRUST-BASED SERVICE COMPOSITION

In this section, we apply our trust management to a trust-based service composition application in SOA-based IoT systems. We consider a travel planning service composition application (not shown here due to space limitation) for which a workflow describes the data flow and logic of the composite service. There are 9 atomic services connected by three types of workflow structures in this example, namely, *sequential*, *parallel* (AND), and *selection* (OR). Each service would have multiple service provider candidates.

In *trust-based service composition*, the service requester calculates the overall trustworthiness using its trust toward service providers, as well as the overall cost for each candidate configuration, and selects the configuration with the highest trustworthiness value among those with the overall cost under the budget limit such that user satisfaction toward the travel plan is the best. We use the average of the "true" user satisfaction levels (in the real dataset [6]) of the service providers selected as the *utility* scores to evaluate the performance of service composition. We compare the performance of our *trust-based service composition* protocol with two baseline approaches:

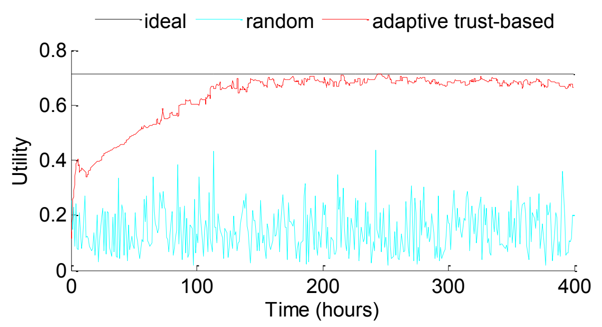

Figure 5: Utility of Trust-based Service Composition vs. Ideal and Random Service Composition.

1. *Ideal service composition* which returns the maximum achievable utility score derived from global knowledge.
2. *Random service composition* which randomly selects service providers for service composition without regard to trust.

Figure 5 shows the simulation results. We can see that *trust-based service composition* with adaptive control significantly outperforms *random service composition* and approaches the performance of *ideal service composition*. We attribute the superiority to our protocol's adaptability to adjust the best trust parameter (μ) dynamically to minimize trust bias, and, consequently, maximize the performance of the service composition application.

6 Conclusion

In this paper, we designed and analyzed an adaptive and survivable trust management protocol for user-centric IoT systems. A user performs trust evaluation based on its past direct user satisfaction experiences and trust feedbacks from other users sharing similar social interests. We considered three social relationships, i.e., friendship, social contact, and community of interest, for measuring social similarity and filtering trust feedbacks based on social similarity. We developed an adaptive filtering technique through which the best way to combine direct trust and indirect trust feedback can be determined dynamically, allowing each node to adaptively select its best trust parameter to minimize convergence time and trust bias.

To demonstrate the applicability, we applied our trust management protocol to a service composition application in SOA-based IoT systems. Our results demonstrated that with our adaptive trust protocol design, the application is able to approach the ideal performance upon convergence and can significantly outperform the counterpart non-trust-based random selection protocol.

In the future, we plan to consider more sophisticated attack behaviors including opportunistic, random and insidious attacks [10] utilizing stochastic process modeling techniques [15-18] to further test the resiliency property of our trust protocol design. We also plan to extend adaptive control to other trust parameters such as φ (the trust decay factor) and (w_f, w_l, w_c) to further improve protocol performance.

Acknowledgment

This material is based upon work supported in part by the U. S. Army Research Laboratory and the U. S. Army Research Office under contract number W911NF-12-1-0445.

References

[1] D. Guinard, et al., "Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services," *IEEE Transactions on Services Computing,* vol. 3, no. 3, pp. 223-235, July-September, 2010.

[2] L. Atzori, A. Iera, and G. Morabito, "SIoT: Giving a Social Structure to the Internet of Things," *IEEE Communication Letters,* vol. 15, no. 11, pp. 1193-1195, Nov., 2011.

[3] F. Bao, and I. R. Chen, "Dynamic Trust Management for Internet of Things Applications," *2012 Inter. Workshop on Self-Aware Internet of Things*, San Jose, California, USA, 2012.

[4] F. Bao, and I. R. Chen, "Trust Management for the Internet of Things and Its Application to Service Composition," *IEEE WoWMoM 2012 Workshop on the Internet of Things: Smart Objects and Services*, San Francisco, CA, USA, 2012.

[5] F. Bao, I. R. Chen, and J. Guo, "Scalable, Adaptive and Survivable Trust Management for Community of Interest Based Internet of Things Systems," *11th International Symposium on Autonomous Decentralized System,* Mexico City, Mexico, 2013.

[6] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating QoS of Real-World Web Services," *IEEE Transactions on Services Computing*, Nov. 2012.

[7] D. Chen, G. Chang, D. Sun, J. Li, J. Jia, and X. Wang, "TRM-IoT: A Trust Management Model Based on Fuzzy Reputation for Internet of Things," *Computer Science and Information Systems,* vol. 8, no. 4, pp. 1207-1228, Oct., 2011.

[8] J. H. Cho, et al., "Modeling and Analysis of Trust Management for Cognitive Mission-Driven Group Communication Systems in Mobile Ad Hoc Networks," *Inter. Conf. Computational Science and Engineering*, Vancouver, Canada, 2009, pp. 641-650.

[9] J. H. Cho, et al., "Modeling and analysis of trust management with trust chain optimization in mobile ad hoc networks" *Network and Computer Applications,* vol. 35, no. 3, 2012, pp. 1001-1012.

[10] R. Mitchell and I. R. Chen, "Effect of Intrusion Detection and Response on Reliability of Cyber Physical Systems," *IEEE Transactions on Reliability*, vol. 62, no. 1, pp. 199-210, 2013.

[11] S. Kosta, A. Mei, and J. Stefa, "Small World in Motion (SWIM): Modeling Communities in Ad-Hoc Mobile Networking," *7th IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, Boston, MA, USA, 2010.

[12] Z. Huang, D. Zeng, H. Chen, "A Comparison of Collaborative-Filtering Recommendation Algorithms for E-commerce," *IEEE Intelligent Systems,* vol. 22, no. 5, pp. 68-78, 2007.

[13] Q. Li, S. Zhu, and G. Cao, "Routing in Socially Selfish Delay Tolerant Networks," *IEEE Conference on Computer Communications*, San Diego, CA, 2010, pp. 1-9.

[14] A. Jøsang, R. Ismail, "The Beta Reputation System," *Electronic Commerce Conference*, Bled, Slovenia, 2002, pp. 1-14.

[15] I.R. Chen, and D.C. Wang, "Analysis of Replicated Data with Repair Dependency," *The Computer Journal*, vol. 39, no. 9, 1996, pp. 767-779.

[16] I.R. Chen, and D.C. Wang, "Analyzing Dynamic Voting using Petri Nets," *15th IEEE Symposium on Reliable Distributed Systems*, Niagara Falls, Canada, 1996, pp. 44-53.

[17] Y. Li and I.R. Chen, "Design and performance analysis of mobility management schemes based on pointer forwarding for wireless mesh networks," *IEEE Transactions on Mobile Computing,* vol. 10, no. 3, 2011, pp. 349-361.

[18] I. R. Chen, A. P. Speer, and M. Eltoweissy, "Adaptive Fault-Tolerant QoS Control Algorithms for Maximizing System Lifetime of Query-Based Wireless Sensor Networks," *IEEE Trans. on Dependable and Secure Computing*, vol. 8, no. 2, 2011, pp. 161-176.