SORT: A Self-ORganizing Trust Model for Peer-to-Peer Systems

Ahmet Burak Can, *Member*, *IEEE*, and Bharat Bhargava, *Fellow*, *IEEE*

Abstract—Open nature of peer-to-peer systems exposes them to malicious activity. Building trust relationships among peers can mitigate attacks of malicious peers. This paper presents distributed algorithms that enable a peer to reason about trustworthiness of other peers based on past interactions and recommendations. Peers create their own trust network in their proximity by using local information available and do not try to learn global trust information. Two contexts of trust, service, and recommendation contexts, are defined to measure trustworthiness in providing services and giving recommendations. Interactions and recommendations are evaluated based on importance, recentness, and peer satisfaction parameters. Additionally, recommender's trustworthiness and confidence about a recommendation are considered while evaluating recommendations. Simulation experiments on a file sharing application show that the proposed model can mitigate attacks on 16 different malicious behavior models. In the experiments, good peers were able to form trust relationships in their proximity and isolate malicious peers.

Index Terms—Peer-to-peer systems, trust management, reputation, security

1 Introduction

Peer-to-peer (P2P) systems rely on collaboration of peers to accomplish tasks. Ease of performing malicious activity is a threat for security of P2P systems. Creating long-term trust relationships among peers can provide a more secure environment by reducing risk and uncertainty in future P2P interactions. However, establishing trust in an unknown entity is difficult in such a malicious environment. Furthermore, trust is a social concept and hard to measure with numerical values. Metrics are needed to represent trust in computational models. Classifying peers as either trustworthy or untrustworthy is not sufficient in most cases. Metrics should have precision so peers can be ranked according to trustworthiness. Interactions and feedbacks of peers provide information to measure trust among peers. Interactions with a peer provide certain information about the peer but feedbacks might contain deceptive information. This makes assessment of trustworthiness a challenge.

In the presence of an authority, a central server is a preferred way to store and manage trust information, e.g., eBay. The central server securely stores trust information and defines trust metrics. Since there is no central server in most P2P systems, peers organize themselves to store and manage trust information about each other [1], [2]. Management of trust information is dependent to the structure of P2P network. In distributed hash table (DHT)-based approaches, each peer becomes a trust holder by storing feedbacks about other peers [1], [3], [4]. Global trust information stored by trust holders can be accessed through DHT efficiently. In unstructured networks, each peer stores trust information about peers in its neighborhood or peers interacted in the past [2], [5], [6]. A peer sends trust queries to learn trust information of other peers. A trust query is either flooded to the network or sent to neighborhood of the query initiator. Generally, calculated trust information is not global and does not reflect opinions of all peers.

We propose a Self-ORganizing Trust model (SORT) that aims to decrease malicious activity in a P2P system by establishing trust relations among peers in their proximity. No a priori information or a trusted peer is used to leverage trust establishment. Peers do not try to collect trust information from all peers. Each peer develops its own local view of trust about the peers interacted in the past. In this way, good peers form dynamic trust groups in their proximity and can isolate malicious peers. Since peers generally tend to interact with a small set of peers [7], forming trust relations in proximity of peers helps to mitigate attacks in a P2P system.

In SORT, peers are assumed to be *strangers* to each other at the beginning. A peer becomes an *acquaintance* of another peer after providing a service, e.g., uploading a file. If a peer has no acquaintance, it chooses to trust strangers. An acquaintance is always preferred over a stranger if they are equally trustworthy. Using a service of a peer is an *interaction*, which is evaluated based on weight (importance) and recentness of the interaction, and satisfaction of the requester. An acquaintance's feedback about a peer, *recommendation*, is evaluated based on recommender's trustworthiness. It contains the recommender's own experience about the peer, information collected from the recommender's acquaintances, and the recommender's level of confidence in the recommendation. If the level of confidence is low, the recommendation has a low value in evaluation and affects less the trustworthiness of the recommender.

A peer may be a good service provider but a bad recommender or vice versa. Thus, SORT considers providing services and giving recommendations as different tasks

- *A.B. Can is with the Department of Computer Engineering, Hacettepe University, Beytepe, 06800 Ankara, Turkey. E-mail: abc@hacettepe.edu.tr.*
- *B. Bhargava is with the Department of Computer Science, Purdue University, 305 N. University Street, West Lafayette, IN 47907. E-mail: bb@cs.purdue.edu.*

Manuscript received 5 Sept. 2011; revised 6 May 2012; accepted 14 Aug. 2012; published online 22 Aug. 2012.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-2011-09-0209. Digital Object Identifier no. 10.1109/TDSC.2012.74.

and defines two contexts of trust: *service* and *recommendation contexts*. Information about past interactions and recommendations are stored in separate histories to assess competence and integrity of acquaintances in these contexts.

SORT defines three trust metrics. *Reputation* metric is calculated based on recommendations. It is important when deciding about strangers and new acquaintances. Reputation loses its importance as experience with an acquaintance increases. *Service trust* and *recommendation trust* are primary metrics to measure trustworthiness in the service and recommendation contexts, respectively. The service trust metric is used when selecting service providers. The recommendation trust metric is important when requesting recommendations. When calculating the reputation metric, recommendations are evaluated based on the recommendation trust metric.

We implemented a P2P file sharing simulation tool and conducted experiments to understand impact of SORT in mitigating attacks. Parameters related to peer capabilities (bandwidth, number of shared files), peer behavior (online/offline periods, waiting time for sessions), and resource distribution (file sizes, popularity of files) are approximated to several empirical results [8], [9], [10]. This enabled us to make more realistic observations on evolution of trust relationships. We studied 16 types of malicious peer behaviors, which perform both service and recommendation-based attacks. SORT mitigated service-based attacks in all cases. Recommendation-based attacks were contained except when malicious peers are in large numbers, e.g., 50 percent of all peers. Experiments on SORT show that good peers can defend themselves against malicious peers without having global trust information. SORT's trust metrics let a peer assess trustworthiness of other peers based on local information. Service and recommendation contexts enable better measurement of trustworthiness in providing services and giving recommendations.

Outline of the paper is as follows: Section 2 discusses the related research. Section 3 explains the computational model of SORT. Section 4 presents the simulation experiments and results. Section 5 summarizes the results and possible future work directions.

2 Related Work

Marsh [11] defines a formal trust model based on sociological foundations. An agent uses own experiences to build trust relations and does not consider information of other agents. Abdul-rahman and Hailes [12] evaluate trust in a discrete domain as an aggregation of direct experience and recommendations of other parties. They define a semantic distance measure to test accuracy of recommendations. Yu and Singh's model [13] propagates trust information through referral chains. Referrals are primary method of developing trust in others. Mui et al. [14] propose a statistical model based on trust, reputation, and reciprocity concepts. Reputation is propagated through multiple referral chains. Jøsang et al. [15] discuss that referrals based on indirect trust relations may cause incorrect trust derivation. Thus, trust topologies should be carefully evaluated before propagating trust information. Terzi et al. [16] introduce an algorithm to classify users and assign them roles based on trust relationships. Zhong [17] proposes a dynamic trust concept based on McKnight's social trust model [18]. When building trust relationships, uncertain evidences are evaluated using second-order probability and Dempster-Shaferian framework.

In e-commerce platforms, reputation systems are widely used as a method of building trust , e.g., eBay, Amazon, and Epinions. A central authority collects feedbacks of past customers, which are used by future customers in shopping decisions. Resnick et al. [19] discuss that ensuring long-lived relationships, forcing feedbacks, checking honesty of recommendations are some difficulties in reputation systems. Despotovic and Aberer [20] point out that trust-aware exchanges can increase economic activity since some exchanges may not happen without trust. Jsang et al. [21] indicate that reputation systems are vulnerable to incorrect and bogus feedback attacks. Thus feedback ratings must be based on objective criteria to be useful. Dellarocas [22] proposes controlled anonymity and cluster filtering methods as countermeasures to unfairly high/low ratings and discriminatory seller behavior attacks. Yu and Singh [23] present a weighted majority algorithm against three attacks on reputation: complementary, exaggerated positive/negative feedbacks. Guha et al. [24] use trust and distrust concepts in a discrete domain. Their results on Epinions web site's data show that distrust is helpful to measure trustworthiness accurately. Reputation systems are vulnerable to sybil attacks [25], where a malicious entity can disseminate bogus feedbacks by creating multiple fake entities. To defend against sybil attacks, Yu et al. [26] and Tran et al. [27] propose techniques based on the observation that fake entities generally have many trust relationships among each other but they rarely have relationships with real users.

Trust models on P2P systems have extra challenges comparing to e-commerce platforms. Malicious peers have more attack opportunities in P2P trust models due to lack of a central authority. Hoffman et al. [28] discuss five common attacks in P2P trust models: self-promoting, white-washing, slandering, orchestrated, and denial of service attacks. They point out that defense techniques in trust models are dependent to P2P system architecture.

On a structured P2P system, a DHT structure can provide decentralized and efficient access to trust information. In Aberer and Despotovic's trust model [1], peers report their complaints by using P-Grid [29]. A peer is assumed as trustworthy unless there are complaints about it. However, preexistence of trust among peers does not distinguish a newcomer and an untrustworthy one. Eigentrust [3] uses transitivity of trust to calculate global trust values stored on CAN [30]. Trusted peers are used to leverage building trust among regular peers and mitigate some collaborative attacks. PeerTrust [4] defines transaction and community context parameters to make trust calculation adaptive on P-Grid. While transaction context parameter addresses application dependent factors, community context parameter addresses P2P community related issues such as creating incentives to force feedbacks. Both Eigentrust and Peertrust evaluate a recommendation based on trustworthiness of the recommender. Song et al. [31] propose a fuzzy-logic-based trust model on Chord [32],

which performs similar results to Eigentrust with lower message overhead. PowerTrust [33] constructs an overlay network based on the Power law distribution of peer feedbacks. By using a random-walk strategy and utilizing power nodes, feedback aggregation speed, and global reputation accuracy are improved.

Solutions on a structured network rely on a DHT structure to store trust information. Each peer becomes a trust holder of another peer, which is assumed to provide authentic global trust information. However, a trust holder might be malicious and provide inauthentic information. In SORT, instead of considering a particular trust holder's feedback as authentic, public opinion from all acquaintances is considered as a more credible information. Instead of considering global trust information, local trust information is enough to make decisions as peers develop their own trust networks.

On unstructured P2P systems, trust queries are generally flooded to the whole network. Cornelli et al. [34] flood trust queries in Gnutella network [9]. A detailed computational trust model is not defined. Peers make decisions based on collected feedbacks to mitigate inauthentic file downloads. Selcuk et al. [5] present a vector-based trust metric relying on both interactions and recommendations. A reputation query is sent to neighbors if there are enough neighbors. Otherwise, the query is flooded to network. Although five types of malicious peers are studied, recommendation-based attacks are not considered in the experiments. Yu et al. [35] store a history of interactions and consider ratings and recentness of interactions when evaluating trust. Number of interactions with a peer is a measure of confidence about the peer. GossipTrust [6] defines a randomized gossiping [36] protocol for efficient aggregation of trust values. A query is randomly forwarded to some neighbors instead of all neighbors. Comparing to flooding approach, gossiping reduces reputation query traffic. In SORT, peers send reputation queries only to peers interacted in the past, which reduces network traffic comparing to flooding-based approaches. Furthermore, each peer expands its trust network with time and can obtain more credible recommendations from acquaintances.

Some trust models use signed credentials to store trust information. Ooi et al. [37] propose that each peer stores its own reputation using signed certificates. When a peer needs to know about a stranger, it requests certificates of the stranger. NICE [38] uses signed cookies as a proof of good behavior. Peers dynamically form trust groups to protect each other. Peers in the same group have a higher trust in each other. Trust-based pricing and trading policies help to protect integrity of groups. Using signed credentials eliminates the need for reputation queries but ensuring validity of trust information in credentials is a problem. If a peer misbehaves after collecting good credentials, it is hard to revoke credentials without using a central authority. Furthermore, a public-key infrastructure is generally needed.

How to evaluate interactions and how to define trust metrics are important problems in trust models. Wang and Vassileva [39] propose a Bayesian network model which uses different aspects of interactions on a P2P file sharing application. Victor et al. [40] define trust and distrust metrics. A nonzero distrust value lets an agent to distinguish an untrusted user from a new user. A lattice structure with trust and knowledge axis is used to model various trusting conditions. Swamynathan et al. [41] decouple trust metrics on service and recommendation contexts to assess trustworthiness better. Creating contexts of trust can be helpful to address issues in various domains. Gupta et al. [42] use reputation as a currency. A central agent issues money to peers in return for their services to others. This money can be used to get better quality of service. Bhargava et al. [43] discusses trading privacy to gain more trust in pervasive systems. In another interesting study, Virendra et al. [44] use trust concept in mobile ad-hoc networks to establish keys among nodes and group nodes into domains. Trustworthiness is measured according to lost and misrouted packets. Trust establishment phases are defined for starting up new nodes, maintaining trust of old peers, and reestablishing trust in malicious nodes.

In SORT, to evaluate interactions and recommendations better, importance, recentness, and peer satisfaction parameters are considered. Recommender's trustworthiness and confidence about recommendation are considered when evaluating recommendations. Additionally, service and recommendation contexts are separated. This enabled us to measure trustworthiness in a wide variety of attack scenarios. Most trust models do not consider how interactions are rated and assume that a rating mechanism exists. In this study, we suggest an interaction rating mechanism on a file sharing application and consider many real-life parameters to make simulations more realistic.

3 The Computational Model of SORT

We make the following assumptions. Peers are equal in computational power and responsibility. There are no privileged, centralized, or trusted peers to manage trust relationships. Peers occasionally leave and join the network. A peer provides services and uses services of others. For simplicity of discussion, one type of interaction is considered in the service context, i.e., file download.

3.1 Preliminary Notations

p_i denotes the ith peer. When p_i uses a service of another peer, it is an *interaction* for p_i. Interactions are unidirectional. For example, if p_i downloads a file from p_j, it is an interaction for p_i and no information is stored on p_j.

If p_i had at least one interaction with p_j, p_j is an *acquaintance* of p_i. Otherwise, p_j is a *stranger* to p_i. A_i denotes p_i's set of acquaintances. A peer stores a separate history of interactions for each acquaintance. SH_{ij} denotes p_i's *service history* with p_j where sh_{ij} denotes the current size of the history. sh_{max} denotes the upper bound for service history size. Since new interactions are appended to the history, SH_{ij} is a time ordered list.

Parameters of an interaction. After finishing an interaction, p_i evaluates quality of service and assigns a satisfaction value for the interaction. Let $0 \leq s_{ij}^k \leq 1$ denote p_i's satisfaction about kth interaction with p_j. If an interaction is not completed, $s_{ij}^k = 0$. An interaction's importance is measured with a *weight* value. Let $0 \leq w_{ij}^k \leq 1$ denote the weight of kth interaction of p_i with p_j.

Semantics to calculate s_{ij}^k and w_{ij}^k values depend on the application. In a file sharing application, authenticity of a

TABLE 1
Notations on the Trust Metrics

Notation	Description
s_{ij}^k	p_i's satisfaction about k^{th} interaction with p_j
w_{ij}^k	weight of p_i's k^{th} interaction with p_j
f_{ij}^k	fading effect of p_i's k^{th} interaction with p_j
r_{ij}	p_i's reputation value about p_j
st_{ij}	p_i's service trust value about p_j
rt_{ik}	p_i's recommendation trust about p_k
sh_{ij}	size of p_i's service history with p_j

file, average download speed, average delay, retransmission rate of packets and online/offline periods of the service provider might be some parameters to calculate s_{ij}^k. Size and popularity of a file might be some parameters to calculate w_{ij}^k. In Section 4, we suggest equations to calculate these values in a file sharing application.

Importance of an old interaction should decrease as new interactions happen. The fading effect parameter addresses this issue and forces peers to stay consistent in future interactions. Old interactions lose their importance so a peer cannot easily misbehave by relying on its good history. Let $0 \leq f_{ij}^k \leq 1$ denote the fading effect of kth interaction of p_i with p_j. It is calculated as follows:

$$f_{ij}^k = \frac{k}{sh_{ij}}, \quad 1 \leq k \leq sh_{ij}. \tag{1}$$

After adding (or deleting) an interaction to SH_{ij}, p_i recalculates f_{ij}^k values. The fading effect can be defined as a function of time but it has to be recalculated whenever its value is needed. Furthermore, interactions continually lose value with time causing all peers to lose reputation even though no bad interaction happens.

Let $SH_{ij} = \{\tau_{ij}^1, \tau_{ij}^2, \ldots, \tau_{ij}^{sh_{ij}}\}$, where $\tau_{ij}^k = (s_{ij}^k, w_{ij}^k)$ is a tuple representing the information about kth interaction. When adding a new interaction, τ_{ij}^1 is deleted if $sh_{ij} = sh_{max}$. An interaction is deleted from the history after an expiration period, which should be determined according to sh_{max} and happening rate of interactions.

Trust metrics. Let $0 \leq r_{ij} \leq 1$ denote p_i's reputation value about p_j. Similarly, $0 \leq st_{ij}, rt_{ij} \leq 1$ denote p_i's service and recommendation trust values about p_j, respectively. When p_j is a stranger to p_i, we define that $r_{ij} = st_{ij} = rt_{ij} = 0$. This is a protection against pseudonym changing of malicious peers [45] since such peers lose reputation and cannot gain advantage by appearing with new identities. The following sections explain calculation of these metrics. For easy reading of these sections, Table 1 lists some notations related to trust metrics.

3.2 Service Trust Metric (st_{ij})

When evaluating an acquaintance's trustworthiness in the service context, a peer first calculates competence and integrity belief values using the information in its service history. *Competence belief* represents how well an acquaintance satisfied the needs of past interactions [18], [17], [46]. Let cb_{ij} denote the competence belief of p_i about p_j in the service context. Average behavior in the past interactions is a measure of the competence belief. When evaluating competence, interactions should be considered in proportion to their weight and recentness. Then, p_i calculates cb_{ij} as follows:

$$cb_{ij} = \frac{1}{\beta_{cb}} \sum_{k=1}^{sh_{ij}} \left(s_{ij}^k \cdot w_{ij}^k \cdot f_{ij}^k\right). \tag{2}$$

$\beta_{cb} = \sum_{k=1}^{sh_{ij}} (w_{ij}^k \cdot f_{ij}^k)$ is the normalization coefficient. If p_j completes all interactions perfectly ($s_{ij}^k = 1$ for all k), the coefficient β_{cb} ensures that $cb_{ij} = 1$. Since $0 \leq s_{ij}^k, w_{ij}^k, f_{ij}^k \leq 1$ by definition, cb_{ij} always take a value between 0 and 1.

A peer can be competent but may present erratic behavior. Consistency is as important as competence. Level of confidence in predictability of future interactions is called *integrity belief* [18], [17], [46]. Let ib_{ij} denote the integrity belief of p_i about p_j in the service context. Deviation from average behavior (cb_{ij}) is a measure of the integrity belief. Therefore, ib_{ij} is calculated as an approximation to the standard deviation of interaction parameters

$$ib_{ij} = \sqrt{\frac{1}{sh_{ij}} \sum_{k=1}^{sh_{ij}} \left(s_{ij}^k \cdot w_{ij}^\mu \cdot f_{ij}^\mu - cb_{ij}\right)^2}. \tag{3}$$

A small value of ib_{ij} means more predictable behavior of p_j in the future interactions. w_{ij}^μ and f_{ij}^μ are the mean of w_{ij}^k and f_{ij}^k values in SH_{ij}, respectively. Since the weight and fading effect parameters are independent from the satisfaction parameter and we are interested in the average behavior (satisfaction), these parameters are eliminated in calculation by using w_{ij}^μ and f_{ij}^μ values for all interactions. We can approximate f_{ij}^μ as follows:

$$f_{ij}^\mu = \frac{1}{sh_{ij}} \sum_{k=1}^{sh_{ij}} f_{ij}^k = \frac{sh_{ij}+1}{2sh_{ij}} \approx \frac{1}{2}. \tag{4}$$

Based on the past interactions with p_j, p_i has an expectation about future interactions. p_i wants to maintain a level of satisfaction according to this expectation. If the satisfaction parameter is assumed to follow a normal distribution, cb_{ij} and ib_{ij} can be considered as approximations of mean (μ) and standard deviation (σ) of the satisfaction parameter, respectively. According to the cumulative distribution function of normal distribution, an interaction's satisfaction is less than cb_{ij} with a $\Phi(0) = 0.5$ probability. If p_i sets $st_{ij} = cb_{ij}$, half of the future interactions will likely to have a satisfaction value less than cb_{ij}. Thus, $st_{ij} = cb_{ij}$ is an overestimate for p_j's trustworthiness. A lower estimate makes p_i more confident about p_j since there will be less future interactions that have a lower satisfaction value than st_{ij} value. p_i may calculate st_{ij} as follows:

$$st_{ij} = cb_{ij} - ib_{ij}/2. \tag{5}$$

In this case, a future interaction's satisfaction is less than st_{ij} with $\Phi(-0.5) = 0.3185$ probability. Adding ib_{ij} into calculation forces p_j to behave more consistently since erratic behavior increases ib_{ij} value. Selection of $\Phi(-0.5)$ comes from our experimental results. In real life, the satisfaction parameter may follow a different distribution. Each peer can use statistical analysis to determine a more precise distribution based on its past interactions and change (5) accordingly. This analysis can be extended for each acquaintance

so a peer can determine a specialized distribution for each acquaintance and customize its trust calculation according to its acquaintances.

Equation (5) is not complete since reputation of p_j is not considered. Reputation is especially important in the early phases of a trust relationship. When there are no (or few) interactions with an acquaintance, a peer needs to rely on the reputation metric. As more interactions happen, the competence and integrity belief values gain more importance. Therefore, p_i calculates st_{ij} as follows:

$$st_{ij} = \frac{sh_{ij}}{sh_{max}}(cb_{ij} - ib_{ij}/2) + \left(1 - \frac{sh_{ij}}{sh_{max}}\right) r_{ij}. \quad (6)$$

Equation (6) balances effects of interactions and the reputation value on st_{ij} value. When p_j is a stranger to p_i, $sh_{ij} = 0$, and $st_{ij} = r_{ij}$. As more interactions happen with p_j, sh_{ij} gets larger and r_{ij} becomes less important. When $sh_{ij} = sh_{max}$, r_{ij} has no effect on st_{ij}.

3.3 Reputation Metric (r_{ij})

The reputation metric measures a stranger's trustworthiness based on recommendations. In the following two sections, we assume that p_j is a stranger to p_i and p_k is an acquaintance of p_i. If p_i wants to calculate r_{ij} value, it starts a reputation query to collect recommendations from its acquaintances.

Algorithm 1 shows how p_i selects trustworthy acquaintances and requests their recommendations. Let η_{max} denote the maximum number of recommendations that can be collected in a reputation query and $|S|$ denote the size of a set S. In the algorithm, p_i sets a high threshold for recommendation trust values and requests recommendations from highly trusted acquaintances first. Then, it decreases the threshold and repeats the same operations. To prevent excessive network traffic, the algorithm stops when η_{max} recommendations are collected or the threshold drops under $(\mu_{rt} - \sigma_{rt})$ value.

Algorithm 1. GETRECOMMENDATIONS(p_j)

1: $\mu_{rt} \Leftarrow \frac{1}{|A_i|}\sum_{p_k \in A_i} rt_{ik}$
2: $\sigma_{rt} \Leftarrow \frac{1}{|A_i|}\sqrt{\sum_{p_k \in A_i}(rt_{ik} - \mu_{rt})^2}$
3: $th_{high} \Leftarrow 1$
4: $th_{low} \Leftarrow \mu_{rt} + \sigma_{rt}$
5: $rset \Leftarrow \emptyset$
6: **while** $\mu_{rt} - \sigma_{rt} \leq th_{low}$ and $|rset| < \eta_{max}$ **do**
7: **for all** $p_k \in A_i$ **do**
8: **if** $th_{low} \leq rt_{ik} \leq th_{high}$ **then**
9: $rec \Leftarrow$ RequestRecommendation(p_k,p_j)
10: $rset \Leftarrow rset \cup \{rec\}$
11: **end if**
12: **end for**
13: $th_{high} \Leftarrow th_{low}$
14: $th_{low} \Leftarrow th_{low} - \sigma_{rt}/2$
15: **end while**
16: return $rset$

Let $T_i = \{p_1, p_2, \ldots p_{t_i}\}$ be the set of trustworthy peers selected by Algorithm 1 and t_i be the number of peers in this set. If $p_k \in A_i$ had at least one interaction with p_j, it replies the following information as a recommendation:

- cb_{kj}, ib_{kj}. These values are a summary of p_k's interaction history with p_j.
- sh_{kj}. The history size with p_j is a measure of p_k's confidence in cb_{kj} and ib_{kj} values. If p_k had many interactions, cb_{kj} and ib_{kj} values are more credible.
- r_{kj}. If p_k had any interaction with p_j, it should already have calculated r_{kj} value, which is a summary of recommendations of p_k's acquaintances. p_i can make a better approximation to global reputation of p_j by aggregating r_{kj} values.
- η_{kj}. η_{kj} denotes the number of p_k's acquaintances which provided a recommendation during the calculation of r_{kj}. This value is a measure of p_k's confidence in r_{kj} value. If η_{kj} value is close to η_{max}, r_{kj} value is credible since more peers agree on r_{kj} value.

Including sh_{kj} and η_{kj} values in the recommendation protects p_k's credibility in the view of p_i. If p_k's knowledge about p_j is insufficient, p_i figures this out by examining sh_{kj} and η_{kj} values. Thus, p_i does not judge p_k harshly if cb_{kj}, ib_{kj}, r_{kj} values are inaccurate comparing to recommendations of other peers.

A recommendation is evaluated according to recommendation trust value of the recommender. In particular, p_i evaluates p_k's recommendation based on rt_{ik} value. The calculation of rt_{ik} value is explained in Section 3.4. If p_i has never received a recommendation from p_k, we set $rt_{ik} = r_{ik}$.

After collecting all recommendations, p_i calculates an estimation about reputation of p_j by aggregating r_{kj} values in all recommendations. Let er_{ij} denote p_i's estimation about reputation of p_j. In this calculation, r_{kj} values are considered with respect to η_{kj} as shown:

$$er_{ij} = \frac{1}{\beta_{er}} \sum_{p_k \in T_i} \left(rt_{ik} \cdot \eta_{kj} \cdot r_{kj}\right). \quad (7)$$

If p_k is trustworthy and collected many recommendations when calculating r_{kj} value, rt_{ik} and η_{kj} values will be larger and r_{kj} will affect the result more. $\beta_{er} = \sum_{p_k \in T_i}(rt_{ik} \cdot \eta_{kj})$ is the normalization coefficient.

Then, p_i calculates estimations about competence and integrity beliefs of p_j denoted by ecb_{ij} and eib_{ij}, respectively. These values are calculated using cb_{kj} and ib_{kj} values in all recommendations. Additionally, cb_{kj} and ib_{kj} are evaluated based on sh_{kj} values. Equations (8) and (9) show calculation of ecb_{ij} and eib_{ij}, where $\beta_{ecb} = \sum_{p_k \in T_i}(rt_{ik} \cdot sh_{kj})$ is the normalization coefficient.

$$ecb_{ij} = \frac{1}{\beta_{ecb}} \sum_{p_k \in T_i} \left(rt_{ik} \cdot sh_{kj} \cdot cb_{kj}\right), \quad (8)$$

$$eib_{ij} = \frac{1}{\beta_{ecb}} \sum_{p_k \in T_i} \left(rt_{ik} \cdot sh_{kj} \cdot ib_{kj}\right). \quad (9)$$

While er_{ij} represents the information collected from acquaintances of p_i's acquaintances, ecb_{ij} and eib_{ij} represent own experiences of p_i's acquaintances with p_j.

p_i calculates the average of history sizes in all recommendations, $\mu_{sh} = \sum_{p_k \in T_i}(sh_{kj})/t_i$ value. If μ_{sh} is close to sh_{max} value, p_i's acquaintances had a high level of own experience with p_j and ecb_{ij}, eib_{ij} values should be given more importance than er_{ij} value. Otherwise, er_{ij} value is more

important. With these observations, r_{ij} is calculated as follows:

$$r_{ij} = \frac{\lfloor \mu_{sh} \rfloor}{sh_{max}} (ecb_{ij} - eib_{ij}/2) + \left(1 - \frac{\lfloor \mu_{sh} \rfloor}{sh_{max}}\right) er_{ij}. \quad (10)$$

In (10), if μ_{sh} is close to sh_{max}, own experiences of acquaintances (ecb_{ij} and eib_{ij} values) will be more important than estimation to the reputation (er_{ij}).

3.4 Recommendation Trust Metric (rt_{ik})

After calculating r_{ij} value, p_i updates recommendation trust values of recommenders based on accuracy of their recommendations. This section explains how p_i updates rt_{ik} according to p_k's recommendation.

Similar to interactions, three parameters are calculated about recommendations. Let $0 \leq rs^z_{ik}, rw^z_{ik}, rf^z_{ik} \leq 1$ denote the satisfaction, weight, and fading effect of p_i's zth recommendation from p_k, respectively. Let $\gamma^z_{ik} = (rs^z_{ik}, rw^z_{ik})$ be a tuple representing the information about zth recommendation and $RH_{ik} = \{\gamma^1_{ik}, \gamma^2_{ik} \ldots \gamma^{rh_{ik}}_{ik}\}$ be the recommendation history of p_i with p_k. Then, rh_{ik} is the size of RH_{ik} and rh_{max} denotes the upper bound of recommendation history size.

To calculate the satisfaction parameter, p_i compares r_{kj}, cb_{kj}, ib_{kj} values with $er_{ij}, ecb_{ij}, eib_{ij}$ values, respectively. If these values are close, p_k's recommendation is good and a high value should be assigned to the satisfaction parameter. Then, the calculation of rs^z_{ik} is as follows:

$$rs^z_{ik} = \left(\left(1 - \frac{|r_{kj} - er_{ij}|}{er_{ij}}\right) + \left(1 - \frac{|cb_{kj} - ecb_{ij}|}{ecb_{ij}}\right) + \left(1 - \frac{|ib_{kj} - eib_{ij}|}{eib_{ij}}\right)\right) \Big/ 3. \quad (11)$$

From (7)-(9), we know that p_k's recommendation affects r_{ij} in proportion to sh_{kj} and η_{kj} values. The effect of these values on r_{ij} is also proportional to $\lfloor \mu_{sh} \rfloor$ due to (10). Thus, the weight of a recommendation should be calculated with respect to $sh_{kj}, \eta_{kj}, \lfloor \mu_{sh} \rfloor$ values. p_i calculates rw^z_{ik} as follows:

$$rw^z_{ik} = \frac{\lfloor \mu_{sh} \rfloor}{sh_{max}} \frac{sh_{kj}}{sh_{max}} + \left(1 - \frac{\lfloor \mu_{sh} \rfloor}{sh_{max}}\right) \frac{\eta_{kj}}{\eta_{max}}. \quad (12)$$

If sh_{kj} and η_{kj} values are large, rw^z_{ik} will have a value close to 1. This is the desired result based on our observations in (7)-(9). $\lfloor \mu_{sh} \rfloor$ value balances the effects of sh_{kj} and η_{kj} values on rw^z_{ik} value. If p_i's acquaintances had many interactions with p_j, the value of $\lfloor \mu_{sh} \rfloor$ will be large and sh_{kj} will have more effect on rw^z_{ik}. When $\lfloor \mu_{sh} \rfloor$ is small, η_{kj} is more important.

Let rcb_{ik} and rib_{ik} denote p_i's competence and integrity beliefs about p_k in the recommendation context, respectively. p_i calculates rt_{ik} in a similar way to st_{ik}

$$rcb_{ik} = \frac{1}{\beta_{rcb}} \sum_{z=1}^{rh_{ik}} \left(rs^z_{ik} \cdot rw^z_{ik} \cdot rf^z_{ik}\right), \quad (13)$$

$$rib_{ik} = \sqrt{\frac{1}{rh_{ik}} \sum_{z=1}^{rh_{ik}} \left(rs^z_{ik} \cdot rw^{\mu}_{ik} \cdot rf^{\mu}_{ik} - rcb_{ik}\right)^2}, \quad (14)$$

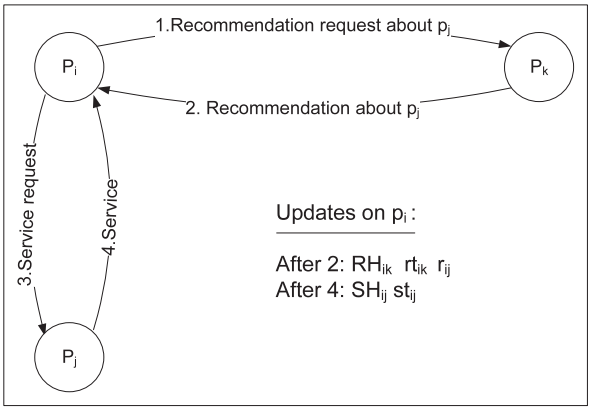


Fig. 1. Operations when receiving a recommendation and having an interaction.

$$rt_{ik} = \frac{rh_{ik}}{rh_{max}} (rcb_{ik} - rib_{ik}/2) + \frac{rh_{max} - rh_{ik}}{rh_{max}} r_{ik}. \quad (15)$$

$\beta_{rcb} = \sum_{z=1}^{rh_{ik}} (rw^z_{ik} \cdot rf^z_{ik})$ is the normalization coefficient. rw^{μ}_{ij} and rf^{μ}_{ij} are the mean of rw^k_{ij} and rf^k_{ij} values in RH_{ij}, respectively. If p_i had no recommendation from p_k, we set $rt_{ik} = r_{ik}$ according to (15).

Fig. 1 explains all scenario briefly. Assume that p_i wants to get a particular service. p_j is a stranger to p_i and a probable service provider. To learn p_j's reputation, p_i requests recommendations from its acquaintances. Assume that p_k sends back a recommendation to p_i. After collecting all recommendations, p_i calculates r_{ij}. Then, p_i evaluates p_k's recommendation, stores results in RH_{ik}, and updates rt_{ik}. Assuming p_j is trustworthy enough, p_i gets the service from p_j. Then, p_i evaluates this interaction and stores the results in SH_{ij}, and updates st_{ij}.

3.5 Selecting Service Providers

When p_i searches for a particular service, it gets a list of service providers. Considering a file sharing application, p_i may download a file from either one or multiple uploaders. With multiple uploaders, checking integrity is a problem since any file part downloaded from an uploader might be inauthentic. Some complex methods utilizing Merkel hashes, secure hashes, and cryptography [47], [48] can be used to do online integrity checking with multiple uploaders. Since this issue is beyond the scope of this paper, the next sections assume one uploader scenario.

Service provider selection is done based on service trust metric, service history size, competence belief, and integrity belief values. When p_i wants to download a file, it selects an uploader with the highest service trust value. If service trust values are equal, the peer with a larger service history size (sh) is selected to prioritize the one with more direct experience. If these values are equal, the one with a larger $cb - ib/2$ value is chosen. If $cb - ib/2$ values are equal, the one with larger competence belief value is selected. If these values are equal, upload bandwidths are compared. If the tie cannot be broken, one of the equal peers is randomly selected.

p_i might select a stranger due to its high reputation. For example, if p_m is a stranger, p_i sets $st_{im} = r_{im}$ according to (6). If p_m is more trustworthy than all acquaintances, p_i selects p_m as the service provider.

Selecting best service providers may overload some peers while others are idle. A selection algorithm should consider load balancing to utilize all resources of peers. In our simulations, a peer's number of simultaneous download/upload operations are limited to a maximum. If a peer reaches its maximum number of uploads, it rejects incoming requests so the requester can get services from others. This simple load balancing mechanism does not consider the whole system state to balance loads. Thus, this issue needs more investigation in a future work.

4 Experiments and Analysis

A file sharing simulation program is implemented in Java to observe results of using SORT in a P2P environment. Some questions studied in the experiments are as follows: how SORT handles attacks, how much attacks can be mitigated, how much recommendations are (not) helpful in correctly identifying malicious peers, and what type of attackers are the most harmful.

4.1 Method

The simulation runs as *cycles*. Each cycle represents a period of time. Downloading a file is an interaction. A peer sharing files is called an *uploader*. A peer downloading a file is called a *downloader*. The set of peers who downloaded a file from a peer are called *downloaders* of the peer. An ongoing download/upload operation is called a *session*. Simulation parameters are generated based on results of several empirical studies [8], [9], [10] to make observations realistic. More details can be found in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TDSC.2012.74 and in [49].

A file search request reaches up to 40 percent of the network and returns online uploaders only. A file is downloaded from one uploader to simplify integrity checking. All peers are assumed to have antivirus software so they can detect infected files. Four different cases are studied to understand effects of trust calculation methods under attack conditions:

- *No trust.* Trust information is not used for uploader selection. An uploader is selected according to its bandwidth. This method is the base case to understand if trust is helpful to mitigate attacks.
- *No reputation query.* An uploader is selected based on trust information but peers do not request recommendations from other peers. Trust calculation is done based on SORT equations but reputation (r) value is always zero for a peer. This method will help us to assess if recommendations are helpful.
- *SORT.* All SORT equations are used as in Section 3.
- *Flood reputation query.* SORT equations are used but a reputation query is flooded to the whole network. This method will help us to understand if getting more recommendations is helpful to mitigate attacks. A peer may request a recommendation from strangers. In this case, a peer assigns a recommendation trust value to the stranger as $rt_{stranger} = \mu_{rt} - \sigma_{rt}$, where μ_{rt} and σ_{rt} are the mean and standard deviation of recommendation trust values of all acquaintances. If a peer does not have any acquaintances, $rt_{stranger} = 0.1$.

Before starting a session, a downloader makes a bandwidth agrement with the uploader, which declares the amount of bandwidth it can devote. Parameters of each finished session are recorded as an interaction. The satisfaction parameter is calculated based on following variables:

- *Bandwidth.* The ratio of average bandwidth (AveBw) and agreed bandwidth (AgrBw) is a measure of reliability of an uploader in terms of bandwidth.
- *Online period.* The ratio of online (OnP) and offline (OffP) periods represents availability of an uploader.

p_i calculates the satisfaction of kth interaction with p_j as follows:

$$s_{ij}^k = \begin{cases} \left(\frac{AveBw}{AgrBw} + \frac{OnP}{OnP + OffP}\right) \Big/ 2 & \text{if } AveBw < AgrBw, \\ \left(1 + \frac{OnP}{OnP + OffP}\right) \Big/ 2 & \text{otherwise.} \end{cases} \tag{16}$$

Calculation of the satisfaction parameter may include more variables such as, delay, jitter, and retransmission of dropped packets [39], [44]. These variables are not included in the equation since they are not simulated.

The weight of an interaction is calculated based on two variables:

- *File size.* A large file is more important than a small one due to bandwidth consumption. However, importance is not directly related to the file size. We assume that files over 100 MB have the same importance.
- *Popularity.* Popular files are more important than unpopular ones. We assume that number of uploaders of a file is an indication of its popularity.

Let $Uploader_{max}$ be the number of uploaders of the most popular file. $size$ and $\#Uploaders$ denote the file size and the number of uploaders, respectively. p_i calculates the weight parameter of kth interaction with p_j as follows:

$$w_{ij}^k = \begin{cases} \left(\frac{size}{100MB} + \frac{\#Uploaders}{Uploader_{max}}\right) \Big/ 2 & \text{if } size < 100MB, \\ \left(1 + \frac{\#Uploaders}{Uploader_{max}}\right) \Big/ 2 & \text{otherwise.} \end{cases} \tag{17}$$

Sometimes rarely shared files might be important but this case is discarded for simplicity.

4.2 Attacker Model

Attackers can perform service-based and recommendation-based attacks. Uploading a virus infected or an inauthentic file is a *service-based attack*. Giving a misleading recommendation intentionally is a *recommendation-based attack*. There are two types of misleading recommendations [22]:

- *Unfairly high.* Giving a positively-biased trust value about a peer where $r = cb = ib = 1$ and $sh = sh_{max}$.
- *Unfairly low.* Giving a negatively-biased trust value about a peer where $r = cb = ib = 0$ and $sh = sh_{max}$.

Setting $sh = sh_{max}$ maximizes the effect of a misleading recommendation. A *fair recommendation* is the recommender's unbiased trust information about a peer. A service-based attack can be detected immediately since a virus infected or an inauthentic file can be recognized after the download. However, it is hard for a peer to determine a recommendation-based attack if the peer's own experience conflicts with a recommendation. Since a recommender might be cheated by attackers, there is no evidence to prove that a recommendation is intentionally given as misleading.

A *good peer* uploads authentic files and gives fair recommendations. A *malicious peer (attacker)* performs both service and recommendation-based attacks. Four different attack behaviors are studied for malicious peers: naive, discriminatory, hypocritical, and oscillatory behaviors. A *nonmalicious network* consists of only good peers. A *malicious network* contains both good and malicious peers. If malicious peers do not know about each other and perform attacks independently, they are called as *individual attackers*. Individual attackers may attack each other. For individual attackers, attack behaviors are as follows:

1. *Naive.* The attacker always uploads infected/inauthentic files and gives unfairly low recommendations about others [22].
2. *Discriminatory.* The attacker selects a group of victims and always uploads infected/inauthentic files to them [22], [5]. It gives unfairly low recommendations about victims. For other peers, it behaves as a good peer.
3. *Hypocritical.* The attacker uploads infected/iauthentic files and gives unfairly low recommendations with x percent probability [3], [5]. In the other times, it behaves as a good peer.
4. *Oscillatory.* The attacker builds a high reputation by being good for a long time period. Then, it behaves as a naive attacker for a short period of time. After the malicious period, it becomes a good peer again.

If malicious peers know each other and coordinate when launching attacks, they are called as *collaborators*. Collaborators always upload authentic files and provide fair recommendations to each other. Collaborators give unfairly high recommendations about each other when recommendations are requested by good peers (to promote reputation of collaborators). All collaborators behave same for these situations. In the other situations, they behave according to the attack behavior described below.

1. *Naive.* Collaborators always upload infected/inauthentic files to good peers and give unfairly low recommendations about them.
2. *Discriminatory.* Collaborators select a group of peers as *victims*. They upload infected/inauthentic files to the victims and give unfairly low recommendations about them. They upload authentic files to *nonvictim peers*. They also give fair recommendations about nonvictim peers.
3. *Hypocritical.* Collaborators upload infected or inauthentic files to good peers or give unfairly low recommendations about them with x percent probability. In the other cases, they behave as good peers.

TABLE 2
Percentage of Service-Based Attacks Prevented for Individual Attackers

		NoRQ	SORT	FloodRQ
10% malicious	Naive	65.3	73.4	73.5
	Discriminatory	72.2	78.9	79.9
	Hypocritical	36.4	61.1	65.2
	Oscillatory	34.3	63.7	69.6
50% malicious	Naive	63.5	65.3	64.7
	Discriminatory	68.7	72.1	72.7
	Hypocritical	27.5	43.0	47.5
	Oscillatory	16.3	37.0	44.0

4. *Oscillatory.* Collaborators behave as good peers for a long time period. Then, they do service-based and recommendation-based attacks in the naive collaborator mode for a short period of time. Good/malicious periods are synchronized among all collaborators.

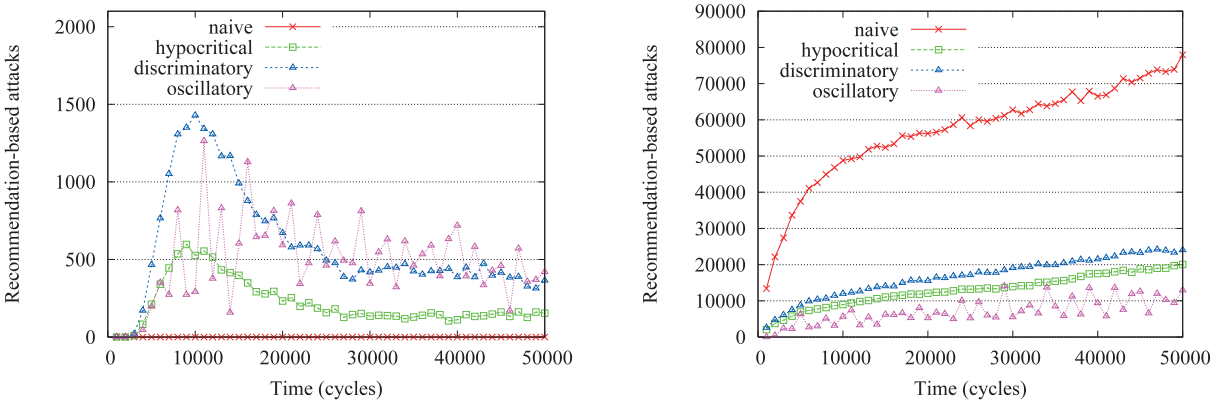
Another type of attackers are *pseudospoofers*, who change their pseudonym to escape from being identified [45], [5]. In addition to above attacker behaviors, we studied individual attackers and collaborators in the pseudonym changing mode. In the experiments, an individual attacker change its pseudonym periodically. Collaborators are assumed to use a special protocol to achieve synchronization during pseudonym change so they can keep coordination.

4.3 Analysis on Individual Attackers

This section explains the results of experiments on individual attackers. For each type of individual attacker, two separate network topologies are created: one with 10 percent malicious and one with 50 percent malicious. Each network topology is tested with four trust calculation methods. In the experiments, a hypocritical attacker behaves malicious in 20 percent of all interactions. A discriminatory attacker selects 10 percent of all peers as victims. An oscillatory attacker behaves good for 1,000 cycles and malicious for 100 cycles.

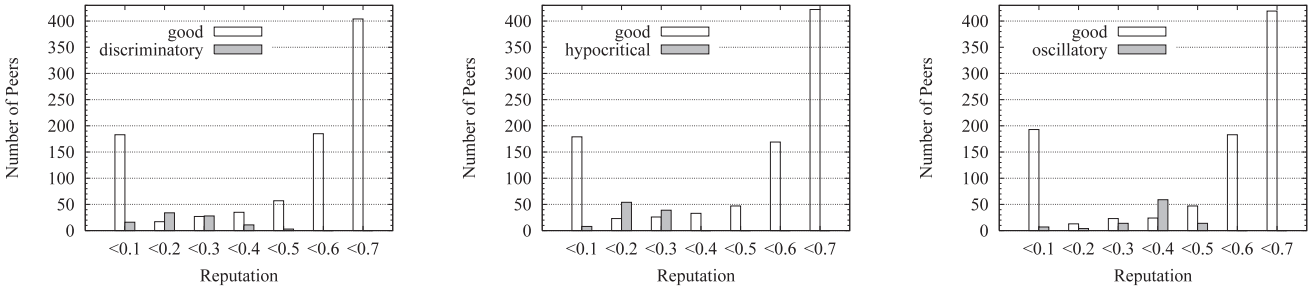
Service-based attacks. Table 2 shows the percentage of service-based attacks prevented by each trust calculation method. When a malicious peer uploads an infected/inauthentic file, it is recorded as a service-based attack. Number of attacks in No Trust method is considered as the base case to understand how many attacks can happen without using trust information. Then, number of attacks observed for each trust calculation method is compared with the base case to determine the percentage of attacks prevented. In the table, NoRQ and FloodRQ denote "No reputation query" and "Flood reputation query" methods, respectively.

In a 10 percent malicious network, all methods can prevent more than 60 percent of attacks of naive attackers. NoRQ method's performance is close to other methods since a good peer identifies a naive attacker after having the first interaction. Thus, recommendations are not very helpful in the naive case. For discriminatory attackers, the situation is similar since their naive attacks easily reveal their identity to victims. For the hypocritical and oscillatory attackers, a good peer may not identify an attacker in the first interaction. Therefore, recommendations in SORT and



(a) SORT (10% malicious) (b) FloodRQ (10% malicious)

Fig. 2. Recommendation-based attacks of individual attackers with respect to the time.



(a) Discriminatory attackers (10% malicious) (b) Hypocritical attackers (10% malicious) (c) Oscillatory attackers (10% malicious)

Fig. 3. Reputation values in individual attacker experiments (with SORT).

FloodRQ methods can be helpful to identify some attackers before an attack happens.

In a 50 percent malicious network, the prevention ratio can be maintained over 60 percent for naive and discriminatory behaviors since attackers are identified quickly. NoRQ method can perform close to SORT and FloodRQ methods. In hypocritical and oscillatory behaviors, SORT can prevent nearly 40 percent of all attacks, which is still good considering the extreme number of attackers. Although attack prevention ratio is higher in the naive behavior, number of attacks is 4-8 times higher than other attacker types. Thus, naive attackers can be considered more successful than other type of individual attackers.

In SORT, a peer interacts less with strangers as its set of acquaintances grows. Therefore, rate of service-based attacks decreases with time. In all cases, SORT's prevention ratio for service-based attacks is close to FloodRQ method. However, FloodRQ method causes 7-10 times more recommendation traffic than SORT. The difference in misleading recommendations is much higher as explained below. Thus, SORT has a better performance tradeoff than FloodRQ method.

Recommendation-based attacks. In the simulations, when a malicious peer gives a misleading recommendation, it is recorded as a recommendation-based attack. Fig. 2 shows the rate of recommendation-based attacks in the 10 percent malicious network. When SORT is used, peers form their own trust network with time and do not request recommendations from untrustworthy peers. Therefore, SORT can effectively mitigate recommendation-based attacks with time. In FloodRQ method, peers collect more recommendations from both acquaintances and strangers. Therefore, attackers find opportunity to disseminate more misleading recommendations as strangers. In FloodRQ method, number of recommendation-based attacks are roughly 10 to 30 times more than SORT in discriminatory, hypocritical, and oscillatory behaviors. Naive attackers cannot disseminate misleading recommendations with SORT since they are identified after the first interaction. In FloodRQ method, if a peer is not interacted with a naive attacker before, it can request recommendations from the attacker as a stranger. Therefore, naive attackers can disseminate more misleading recommendations than other attacker types in FloodRQ method. This observation shows that instead of considering public opinion, collecting recommendations from acquaintances provides more reliable information.

In 50 percent malicious network, recommendation trust values of attackers are close to good peers so attackers can disseminate more misleading recommendations. However, SORT still mitigates misleading recommendations 5-10 times more than FloodRQ method.

Distribution of trust metrics. Peers with higher capabilities (network bandwidth, online period, and number of shared files) can finish more interactions successfully. Thus, they generally have better reputation and service trust values. Recommendation trust values are not directly related to peer capabilities since giving a recommendation does not require high capabilities.

Fig. 3 shows average reputation values of peers at the end of simulation in 10 percent network with SORT. In our experiments, some peers had 500-600 downloaders. Naive attackers are not shown in the figure. They have zero reputation values since they attack in all interactions. Discriminatory attackers only attack to victims so they can build up some reputation among nonvictim peers. Oscillatory

TABLE 3
Percentage of Service-Based Attacks Prevented for Individual Pseudospoofers

		NoRQ	SORT	FloodRQ
10% malicious	Naive	27.4	54.9	52.7
	Discriminatory	29.0	63.9	64.9
	Hypocritical	27.2	62.6	64.8
	Oscillatory	15.8	58.1	60.1
50% malicious	Naive	20.6	31.8	27.5
	Discriminatory	38.3	49.5	50.1
	Hypocritical	33.8	47.0	48.3
	Oscillatory	36.9	51.2	55.0

TABLE 4
Percentage of Service-Based Attacks Prevented for Collaborators

		NoRQ	SORT	FloodRQ
10% malicious	Naive	66.7	73.1	73.0
	Discriminatory	76.5	79.0	80.1
	Hypocritical	40.7	58.3	63.4
	Oscillatory	27.4	42.6	53.2
50% malicious	Naive	58.8	60.3	58.9
	Discriminatory	65.1	65.6	65.4
	Hypocritical	27.3	25.4	25.1
	Oscillatory	16.0	14.9	14.8

attackers gain the highest reputation since their attack period is 10 percent of their good period. Hypocritical attackers have lower reputation value than oscillatory ones since they attack in 20 percent of all interactions. In 50 percent malicious network, average reputation values of good peers drops due to large number of misleading recommendations. However, attackers have lower reputation values than good peers in all experiments.

Generally, a peer performs few interactions with an acquaintance but may request many recommendations from the acquaintance before each download operation. Thus, reputation values have a strong correlation with service trust values but not with recommendation trust values. Giving many recommendations increases the chance of giving inaccurate information, especially in hypocritical and oscillatory attackers. Furthermore, having a large set of downloaders might have a negative effect on recommendation trust values. A peer with many downloaders gets more recommendation requests and its average recommendation trust value drops. Naive attackers have zero recommendation trust values since their reputation is zero. In the other behaviors, attackers have less recommendation trust values than good peers so recommendations of good peers are more credible.

4.4 Analysis on Individual Pseudospoofers

This section explains the results of experiments on individual pseudospoofers. Pseudospoofers change their pseudonyms after every 1,000 cycles. For other parameters, attackers behave as described in Section 4.3.

Service-based attacks. Table 3 shows the attack prevention ratio for individual pseudospoofers. The values obtained by comparing the base case with each trust calculation method. After every pseudonym change, attackers become strangers to others. This behavior has two effects: 1) Pseudospoofers clear their bad history. Hence a good peer may interact with them when it cannot find more reliable uploaders, which increases attacks. 2) Pseudospoofers become more isolated from good peers. They lose their ability to attract good peers with time, which decreases attacks.

In all experiments, NoRQ method performs 20-40 percent worse than other trust calculation methods. Since no recommendation is collected in NoRQ method, chance of selecting an attacker again is higher after every pseudonym change. Therefore, SORT and FloodRQ methods have better results in the experiments. Recommendations increase the chance of finding good peers among strangers.

When SORT or FloodRQ methods are used, the prevention ratio of naive pseudospoofers is less than other type of attackers. This is a result of the first effect. Naive pseudospoofers attack in all interactions and clean their bad history after every pseudonym change. Thus, naive pseudospoofers can perform 40 percent more attacks than naive individual attackers. In the other types of pseudospoofers, the second effect gains importance and attackers become more isolated. Therefore, attack prevention ratio does not decrease as much as naive pseudospoofers, even increases in some cases.

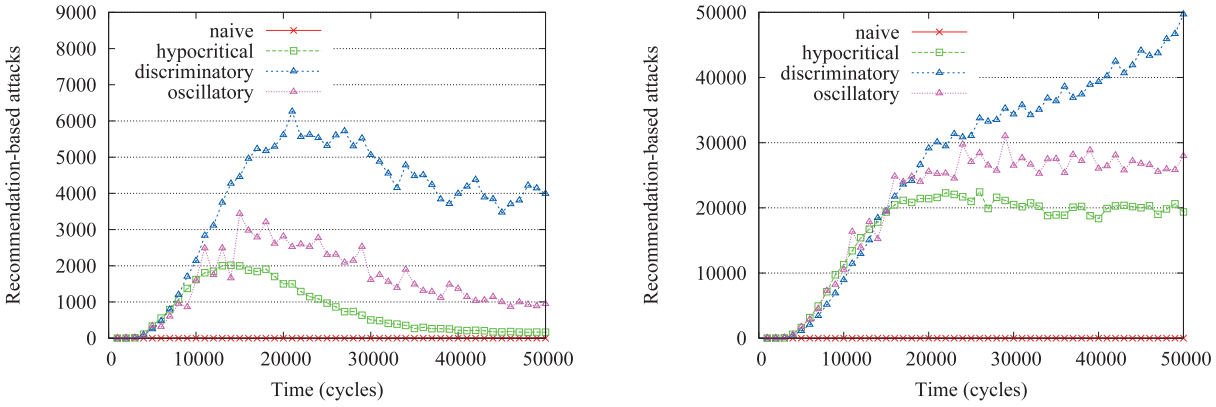
Recommendation-based attacks. Comparing to non-pseudonym changing individual attackers, recommendation-based attacks decrease due to the second effect. Attackers become more isolated from good peers after every pseudonym change and get less recommendation requests. Therefore, their recommendation-based attacks sharply decrease after every pseudonym change in a 10 percent malicious network. This situation is slightly different in a 50 percent malicious network. The good peers need to interact with more strangers since they can hardly find each other. Hence attackers can distribute more misleading recommendations. However, recommendation-based attack rates of individual pseudospoofers are still 70-80 times less than individual attackers explained in Section 4.3.

Distribution of trust metrics. Since pseudospoofers clear their old history in every 1,000 cycles, they cannot gain a high reputation among good peers. This situation is same for service trust and recommendation trust metrics. Average reputation, service trust, and recommendation trust values of pseudospoofers remain under 0.1 value in most simulations.

4.5 Analysis on Collaborators

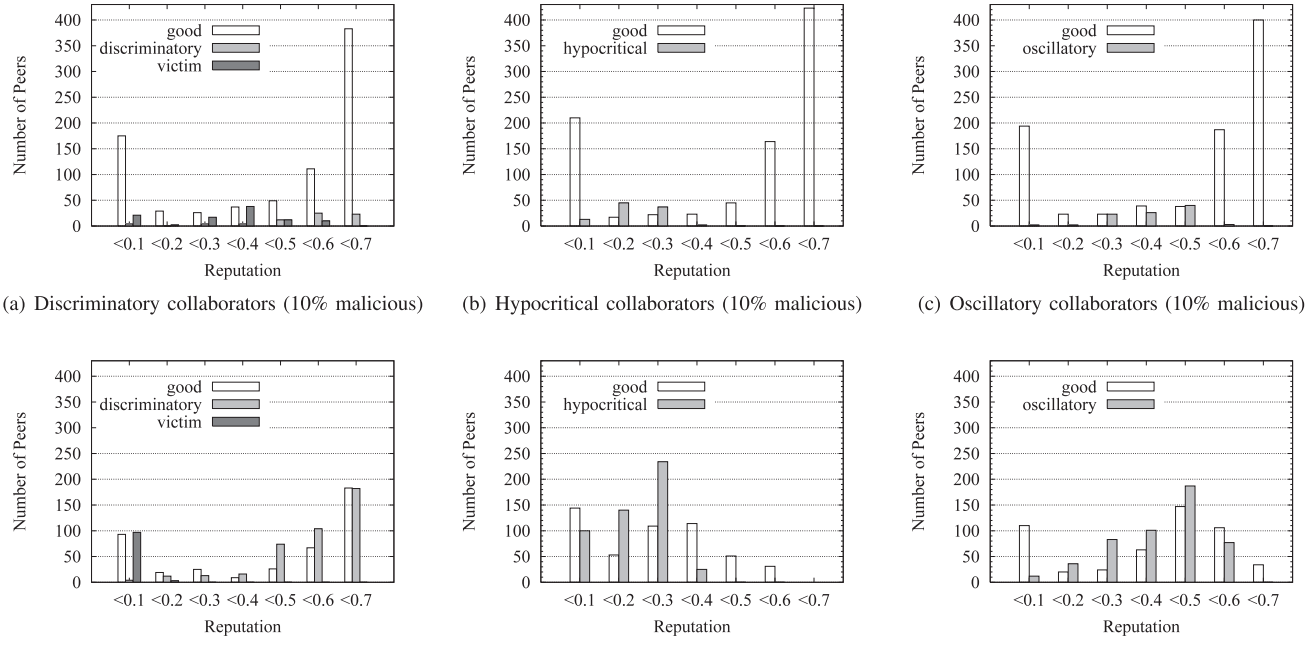
Collaboration of attackers generally makes attack prevention harder. This section presents experimental results on collaborators. Collaborators form teams of size 50 and launch attacks as teams. We first tried teams of size 10 but this was not enough to benefit from collaboration and results were close to individual attackers. Hence team size is increased to observe effects of collaboration better. Attack probability of hypocritical collaborators is set to 0.2. Oscillatory collaborators behave good for 1,000 cycles and malicious for 100 cycles. Discriminatory collaborators attack to the same group of victims, which are 10 percent of all peers. In other words, different teams are attacking the same victims and stay honest with others.

Service-based attacks. Table 4 shows the percentage of attacks prevented by each method. Attacks of naive collaborators can be prevented by 60 percent or more. Naive collaborators are identified by good peers after the first interaction so they are not asked for recommendations.



(a) SORT (10% malicious) (b) SORT (50% malicious)

Fig. 4. Recommendation-based attacks of collaborators with respect to the time.



(a) Discriminatory collaborators (10% malicious) (b) Hypocritical collaborators (10% malicious) (c) Oscillatory collaborators (10% malicious)

(d) Discriminatory collaborators (50% malicious) (e) Hypocritical collaborators (50% malicious) (f) Oscillatory collaborators (50% malicious)

Fig. 5. Reputation values in collaborative attack experiments (with SORT).

Thus, they cannot praise each other with unfairly high recommendations and cannot take advantage of collaboration. Discriminatory collaborators naively attack to victims so they are quickly identified by the victims. Their collaboration does not help to launch more attacks than individual discriminatory attackers. Hypocritical and oscillatory collaborators can take advantage of collaboration. They attract more good peers than individual attackers by praising each other. They are not quickly identified since they perform attacks occasionally. Especially in a 50 percent malicious network, SORT performs worst than NoRQ method for hypocritical and oscillatory behaviors. In such an extreme environment, misleading recommendations of collaborators cause a pollution in the recommendation pool and affect decisions of peers negatively. In such extremely malicious environments, some trusted peers might help good peers for finding each other.

Recommendation-based attacks. Fig. 4 shows recommendation-based attack rate with SORT. In a 10 percent malicious network, attacks can be contained. However, collaboration enables to disseminate more misleading recommendations than individual attack scenarios.

In a 50 percent malicious network, attacks of hypocritical and oscillatory collaborators can be contained on a level but cannot be decreased to an acceptable level. They can continue to disseminate misleading recommendations due to their large number. Discriminatory collaborators can disseminate more misleading recommendations than others since they are trusted by 90 percent of all peers. Discriminatory collaborators constitute 50 percent of all peer population while victims are 10 percent of all population. Therefore, they can deceive other good peers with their misleading recommendations. Although they can continue to propagate misleading recommendations about victims, they cannot launch more service-based attacks since they are identified by victims.

Distribution of trust metrics. Fig. 5 shows reputation values of collaborators. Naive collaborators are not shown in the figure since they are quickly identified by good peers and have zero reputation values.

TABLE 5
Percentage of Service-Based Attacks Prevented for Collaborating Pseudospoofers

		NoRQ	SORT	FloodRQ
10% malicious	Naive	28.0	58.3	53.0
	Discriminatory	28.9	65.5	60.2
	Hypocritical	25.2	62.7	58.4
	Oscillatory	32.2	67.0	63.0
50% malicious	Naive	19.7	31.0	20.2
	Discriminatory	35.3	47.0	44.0
	Hypocritical	32.7	44.8	35.7
	Oscillatory	35.6	50.8	45.4

In a 10 percent malicious network, misleading recommendations of discriminatory collaborators slightly decrease the reputation of victims as shown in Fig. 5a. This is not observed in the individual discriminatory behavior since each attacker selects a different set of victims and misleading recommendations are evenly distributed among all peers. Discriminatory collaborators attack to only 10 percent of peers so impact of misleading recommendations is concentrated on a small group. Since they do not attack nonvictim peers and give unfairly high recommendations about each other, they can maintain a high reputation. This situation is more clear with the 50 percent malicious network as shown in Fig. 5d. Victims has a very low reputation since 50 percent of all peers give misleading recommendations about them. Reputation of nonvictim good peers does not change much since they are not attacked.

As shown in Fig. 5b, good peers can maintain higher reputation than hypocritical collaborators in the 10 percent malicious network. In 50 percent malicious network setup, collaborators gain higher reputation values and decrease reputation of good peers as shown in Fig. 5e. However, they still have lower repuation values than good peers. Oscillatory collaborators have a similar distribution to hypocritical collaborators. They have a higher reputation average since their attack frequency is less than hypocritical collaborators.

Distributions of service trust and recommendation trust values are similar to reputation values. Thus, we will not discuss distributions of these metrics further.

4.6 Analysis on Collaborating Pdesudospoofers

This section presents the results of experiments on collaborating pseudospoofers. Collaborating pseudospoofers are assumed to change their pseudonyms in every 1,000 cycles using an external synchronization method. For the other parameters, they behave like collaborators.

Service-based attacks. Table 5 shows the percentage of attacks prevented by each trust calculation method. The results verify our observations in Section 4.4. In naive and discriminatory behaviors, changing pseudonym causes the first effect: collaborators clear bad history and get more attack opportunities. Therefore, attack prevention ratio drops for these collaborators. In hypocritical and oscillatory behaviors, the second effect becomes more important. After every pseudonym change, collaborators become more isolated from good peers and lose their attack opportunities. Therefore, attack rate slightly drops and attack prevention ratio increases.

SORT's performance is the best in all test cases. SORT enables peers to establish stronger trust relationships than NoRQ and FloodRQ methods. In NoRQ, a good peer cannot learn experience of others through recommendations and pseudonym changing lets attackers to launch more attacks. In FloodRQ, collecting recommendations of strangers enables collaborators to disseminate more misleading recommendations. Since SORT collects recommendations only from acquaintances, reputation queries return more reliable information than FloodRQ method.

Recommendation-based attacks. As in individual pseudospoofers, collaborating pseudospoofers are isolated more from good peers after every pseudonym change. They get less recommendation requests and thus they can do nearly zero recommendation-based attacks in 10 percent malicious network. In 50 percent malicious network, collaborating pseudospoofers can distribute more misleading recommendations since good peers need to interact with more strangers to find each other. However, these misleading recommendations are still in a negligible level.

Trust metrics. Like individual pseudospoofers, collaborating pseudospoofers cannot gain high reputation, service trust, or recommendation trust values since they lose reputation after every pseudonym change. Due to their low recommendation trust values, collaborators are not asked for recommendations by good peers. Therefore, they can distribute small number of misleading recommendations.

5 Conclusion

A trust model for P2P networks is presented, in which a peer can develop a trust network in its proximity. A peer can isolate malicious peers around itself as it develops trust relationships with good peers. Two context of trust, service and recommendation contexts, are defined to measure capabilities of peers in providing services and giving recommendations. Interactions and recommendations are considered with satisfaction, weight, and fading effect parameters. A recommendation contains the recommender's own experience, information from its acquaintances, and level of confidence in the recommendation. These parameters provided us a better assessment of trustworthiness.

Individual, collaborative, and pseudonym changing attackers are studied in the experiments. Damage of collaboration and pseudospoofing is dependent to attack behavior. Although recommendations are important in hypocritical and oscillatory attackers, pseudospoofers, and collaborators, they are less useful in naive and discriminatory attackers. SORT mitigated both service and recommendation-based attacks in most experiments. However, in extremely malicious environments such as a 50 percent malicious network, collaborators can continue to disseminate large amount of misleading recommendations. Another issue about SORT is maintaining trust all over the network. If a peer changes its point of attachment to the network, it might lose a part of its trust network. These issues might be studied as a future work to extend the trust model.

Using trust information does not solve all security problems in P2P systems but can enhance security and effectiveness of systems. If interactions are modeled correctly, SORT can be adapted to various P2P applications, e.g., CPU sharing, storage networks, and P2P gaming. Defining application specific context of trust and related metrics can help to assess trustworthiness in various tasks.

REFERENCES

[1] K. Aberer and Z. Despotovic, "Managing Trust in a Peer-2-Peer Information System," *Proc. 10th Int'l Conf. Information and Knowledge Management (CIKM),* 2001.

[2] F. Cornelli, E. Damiani, S.D.C. di Vimercati, S. Paraboschi, and P. Samarati, "Choosing Reputable Servents in a P2P Network," *Proc. 11th World Wide Web Conf. (WWW),* 2002.

[3] S. Kamvar, M. Schlosser, and H. Garcia-Molina, "The (Eigentrust) Algorithm for Reputation Management in P2P Networks," *Proc. 12th World Wide Web Conf. (WWW),* 2003.

[4] L. Xiong and L. Liu, "Peertrust: Supporting Reputation-Based Trust for Peer-to-Peer Ecommerce Communities," *IEEE Trans. Knowledge and Data Eng.,* vol. 16, no. 7, pp. 843-857, July 2004.

[5] A.A. Selcuk, E. Uzun, and M.R. Pariente, "A Reputation-Based Trust Management System for P2P Networks," *Proc. IEEE/ACM Fourth Int'l Symp. Cluster Computing and the Grid (CCGRID),* 2004.

[6] R. Zhou, K. Hwang, and M. Cai, "Gossiptrust for Fast Reputation Aggregation in Peer-to-Peer Networks," *IEEE Trans. Knowledge and Data Eng.,* vol. 20, no. 9, pp. 1282-1295, Sept. 2008.

[7] J. Kleinberg, "The Small-World Phenomenon: An Algorithmic Perspective," *Proc. 32nd ACM Symp. Theory of Computing,* 2000.

[8] S. Saroiu, P. Gummadi, and S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proc. Multimedia Computing and Networking,* 2002.

[9] M. Ripeanu, I. Foster, and A. Iamnitchi, "Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design," *IEEE Internet Computing,* vol. 6, no. 1, pp. 50-57, Jan. 2002.

[10] S. Saroiu, K. Gummadi, R. Dunn, S.D. Gribble, and H.M. Levy, "An Analysis of Internet Content Delivery Systems," *Proc. Fifth USENIX Symp. Operating Systems Design and Implementation (OSDI),* 2002.

[11] S. Marsh, "Formalising Trust as a Computational Concept," PhD thesis, Dept. of Math. and Computer Science, Univ. of Stirling, 1994.

[12] A. Abdul-Rahman and S. Hailes, "Supporting Trust in Virtual Communities," *Proc. 33rd Hawaii Int'l Conf. System Sciences (HICSS),* 2000.

[13] B. Yu and M. Singh, "A Social Mechanism of Reputation Management in Electronic Communities," *Proc. Cooperative Information Agents (CIA),* 2000.

[14] L. Mui, M. Mohtashemi, and A. Halberstadt, "A Computational Model of Trust and Reputation for E-Businesses," *Proc. 35th Hawaii Int'l Conf. System Sciences (HICSS),* 2002.

[15] A. Jøsang, E. Gray, and M. Kinateder, "Analysing Topologies of Transitive Trust," *Proc. First Int'l Workshop Formal Aspects in Security and Trust (FAST),* 2003.

[16] E. Terzi, Y. Zhong, B. Bhargava, Pankaj, and S. Madria, "An Algorithm for Building User-Role Profiles in a Trust Environment," *Proc. Fourth Int'l Conf. Data Warehousing and Knowledge Discovery (DaWaK),* vol. 2454, 2002.

[17] Y. Zhong, "Formalization of Dynamic Trust and Uncertain Evidence for User Authorization," PhD thesis, Dept. of Computer Science, Purdue Univ., 2004.

[18] D.H. McKnight, "Conceptualizing Trust: A Typology and E-Commerce Customer Relationships Model," *Proc. 34th Ann. Hawaii Int'l Conf. System Sciences (HICSS),* 2001.

[19] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman, "Reputation Systems," *Comm. ACM,* vol. 43, no. 12, pp. 45-48, 2000.

[20] Z. Despotovic and K. Aberer, "Trust-Aware Delivery of Composite Goods," *Proc. First Int'l Conf. Agents and Peer-to-Peer Computing,* 2002.

[21] A. Jøsang, R. Ismail, and C. Boyd, "A Survey of Trust and Reputation Systems for Online Service Provision," *Decision Support Systems,* vol. 43, no. 2, pp. 618-644, 2007.

[22] C. Dellarocas, "Immunizing Online Reputation Reporting Systems Against Unfair Ratings and Discriminatory Behavior," *Proc. Second ACM Conf. Electronic Commerce (EC),* 2000.

[23] B. Yu and M.P. Singh, "Detecting Deception in Reputation Management," *Proc. Second Int'l Joint Conf. Autonomous Agents and Multiagent Systems,* 2003.

[24] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of Trust and Distrust," *Proc. 13th Int'l Conf. World Wide Web (WWW),* 2004.

[25] J. Douceur, "The Sybil Attack," *Proc. First Int'l Workshop Peer-to-Peer Systems (IPTPS),* 2002.

[26] H. Yu, M. Kaminsky, P.B. Gibbons, and A. Flaxman, "Sybilguard: Defending against Sybil Attacks via Social Networks," *ACM SIGCOMM Computer Comm. Rev.,* vol. 36, no. 4, pp. 267-278, 2006.

[27] N. Tran, B. Min, J. Li, and L. Subramanian, "Sybil-Resilient Online Content Voting," *Proc. Sixth USENIX Symp. Networked Systems Design and Implementation (NSDI),* 2009.

[28] K. Hoffman, D. Zage, and C. Nita-Rotaru, "A Survey of Attack and Defense Techniques for Reputation Systems," *ACM Computing Surveys,* vol. 42, no. 1, pp. 1:1-1:31, 2009.

[29] K. Aberer, A. Datta, and M. Hauswirth, "P-Grid: Dynamics of Self-Organization Processes in Structured P2P Systems," Peer-to-Peer Systems and Applications, vol. 3845, 2005.

[30] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," *ACM SIGCOMM Computer Comm. Rev.,* vol. 31, no. 4, pp. 161-172, 2001.

[31] S. Song, K. Hwang, R. Zhou, and Y.-K. Kwok, "Trusted P2P Transactions with Fuzzy Reputation Aggregation," *IEEE Internet Computing,* vol. 9, no. 6, pp. 24-34, Nov.-Dec. 2005.

[32] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *ACM SIGCOMM Computer Comm. Rev.,* vol. 31, no. 4, pp. 149-160, 2001.

[33] R. Zhou and K. Hwang, "Powertrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing," *IEEE Trans. Parallel and Distributed Systems,* vol. 18, no. 4, pp. 460-473, Apr. 2007.

[34] F. Cornelli, E. Damiani, S.D.C. di Vimercati, S. Paraboschi, and P. Samarati, "Implementing a Reputation-Aware Gnutella Servent," *Proc. Networking 2002 Workshops Web Eng. and Peer-to-Peer Computing,* 2002.

[35] B. Yu, M.P. Singh, and K. Sycara, "Developing Trust in Large-Scale Peer-to-Peer Systems," *Proc. IEEE First Symp. Multi-Agent Security and Survivability,* 2004.

[36] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized Gossip Algorithms," *IEEE/ACM Trans. Networking,* vol. 52, no. 6, pp. 2508-2530, June 2006.

[37] B. Ooi, C. Liau, and K. Tan, "Managing Trust in Peer-to-Peer Systems Using Reputation-Based Techniques," *Proc. Fourth Int'l Conf. Web Age Information Management,* 2003.

[38] R. Sherwood, S. Lee, and B. Bhattacharjee, "Cooperative Peer Groups in Nice," *Computer Networks,* vol. 50, no. 4, pp. 523-544, 2006.

[39] Y. Wang and J. Vassileva, "Bayesian Network Trust Model in Peer-to-Peer Networks," *Proc. Second Workshop Agents and Peer-to-Peer Computing at the Autonomous Agents and Multi Agent Systems Conf. (AAMAS),* 2003.

[40] P. Victor, C. Cornelis, M. De Cock, and P. Pinheiro da Silva, "Gradual Trust and Distrust in Recommender Systems," *Fuzzy Sets Systems,* vol. 160, no. 10, pp. 1367-1382, 2009.

[41] G. Swamynathan, B.Y. Zhao, and K.C. Almeroth, "Decoupling Service and Feedback Trust in a Peer-to-Peer Reputation System," *Proc. Int'l Conf. Parallel and Distributed Processing and Applications (ISPA),* 2005.

[42] M. Gupta, P. Judge, and M. Ammar, "A Reputation System for Peer-to-Peer Networks," *Proc. 13th Int'l Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDAV),* 2003.

[43] S. Staab, B. Bhargava, L. Lilien, A. Rosenthal, M. Winslett, M. Sloman, T. Dillon, E. Chang, F.K. Hussain, W. Nejdl, D. Olmedilla, and V. Kashyap, "The Pudding of Trust," *IEEE Intelligent Systems,* vol. 19, no. 5, pp. 74-88, 2004.

[44] M. Virendra, M. Jadliwala, M. Chandrasekaran, and S. Upadhyaya, "Quantifying Trust in Mobile Ad-Hoc Networks," *Proc. IEEE Int'l Conf. Integration of Knowledge Intensive Multi-Agent Systems (KIMAS),* 2005.

[45] E.J. Friedman and P. Resnick, "The Social Cost of Cheap Pseudonyms," *J. Economics and Management Strategy,* vol. 10, no. 2, pp. 173-199, 2001.

[46] S. Xiao and I. Benbasat, "The Formation of Trust and Distrust in Recommendation Agents in Repeated Interactions: A Process-Tracing Analysis," *Proc. Fifth ACM Conf. Electronic Commerce (EC),* 2003.

[47] A. Habib, D. Xu, M. Atallah, B. Bhargava, and J. Chuang, "A Tree-Based Forward Digest Protocol to Verify Data Integrity in Distributed Media Streaming," *IEEE Trans. Knowledge and Data Eng.,* vol. 17, no. 7, pp. 1010-1014, July 2005.

[48] G. Caronni and M. Waldvogel, "Establishing Trust in Distributed Storage Providers," *Proc. IEEE Third Conf. Peer-to-Peer Computing (P2P),* 2003.
[49] A.B. Can, "Trust and Anonymity in Peer-to-Peer Systems," PhD thesis, Dept. of Computer Science, Purdue Univ., 2007.

Ahmet Burak Can received the BS and MS degrees in computer science and engineering from Hacettepe University and the PhD degree in computer science from Purdue University, West Lafayette. Currently, he is affiliated with the Department of Computer Science and Engineering at Hacettepe University, Turkey. His main research areas include computer networks, distributed systems, and network security. His current research activities focus on trust and reputation management, anonymity protection, and incentive mechanisms in peer-to-peer systems. He is a member of the IEEE.

Bharat Bhargava is a professor of the Department of Computer Science at Purdue University. He is conducting research in security and privacy issues in distributed systems. He serves on seven editorial boards of international journals. He also serves the IEEE Computer Society on technical achievement award and fellow committees. He is the founder of the IEEE Symposium on Reliable and Distributed Systems, IEEE conference in the Digital Library, and the ACM Conference on Information and Knowledge Management. He has been awarded the charter Gold Core Member distinction by the IEEE Computer Society for his distinguished service. He is a fellow of the IEEE and IETE.