Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2018) 26: 631 – 643

doi:10.3906/elk-1702-77

Consistency-based trust management in P2P networks

Yasin ŞAHİN, Ahmet Burak CAN*
Department of Computer Engineering, Hacettepe University, Ankara, Turkey

Received: 07.02.2017 • **Accepted/Published Online:** 14.11.2017 • **Final Version:** 30.03.2018

Abstract: Detecting malicious peers is a challenging task in peer-to-peer networks due to their decentralized structure and lack of central authority. Trust models can help identify malicious peers by maintaining information about peer relations and interactions. Keeping information about trust relations helps to reduce risks when providing or using services. This paper introduces two consistency concepts in trust management. Feedback consistency is used to evaluate how consistent feedback is with respect to past feedbacks. On the other side, peer consistency measures consistency of a peer's past feedbacks. These metrics help to reduce malicious interactions and increase successful downloads. Furthermore, the model offers better service quality for good peers by using consistency metrics. A file-sharing application is implemented on a simulation environment. The proposed model can effectively reduce the malicious download rate, even in 50% malicious environments, and increases successful download rates.

Key words: Peer-to-peer systems, trust models, peer consistency, feedback consistency

1. Introduction

With the rapid growth of the Internet community, server-based centralized solutions are having difficulties satisfying the increasing demands of clients on network bandwidth and hardware capabilities. Peer-to-peer (P2P) networks can provide scalable decentralized solutions by distributing network traffic and processing costs to peers. CPU- or disc-sharing networks, content-sharing platforms, file-distribution platforms, and many other systems are implemented as P2P systems to overcome problems of server-based solutions. However, malicious peers may degrade the effectiveness of P2P systems. In addition to sharing services, peers can share experiences about provided services to decrease activities of malicious peers. Thus, each peer could have the opportunity to evaluate another peer by using other peers' experiences, even without knowing about the evaluated peer. A peer may collect feedbacks of others and combine them with its own experience to calculate trust- and reputation-related metrics. Since some feedback providers might be malicious, the calculation of metrics can be challenging. A trust model should consider such cases and provide robust metrics to make trusting decisions about service providers.

We propose a consistency-based trust model to identify malicious peers by using feedback consistency and peer consistency metrics. Feedback consistency evaluates how consistent a feedback about a peer is compared to previous feedbacks. Thus, malicious feedbacks can be detected and their importance in trust calculation can be decreased. Furthermore, when a peer's malicious feedbacks are detected, its peer consistency value is decreased. In other words, peer consistency metric measures how good a peer is in providing feedbacks. This metric is used by service providers when accepting service requests. If a service requester has low peer consistency, its

*Correspondence: abc@hacettepe.edu.tr

feedback quality is considered low and then a service provider would more likely reject its requests. Furthermore, peers with larger peer consistency values obtain better quality of service. The proposed model was tested on the Peersim [1] simulation environment and produced robust results, even in extremely malicious environments with a 50% malicious peer ratio.

In the rest of the paper, Section 2 discusses the related research. Section 3 explains the computational model of the proposed approach. Section 4 defines the experimental model. Section 5 presents experimental results. Section 6 concludes the study and gives possible future work directions.

2. Related work

At the beginning of the 21st century, centralized server systems started to be replaced by distributed systems. P2P systems appeared as an approach to solve scalability issues in centralized systems. Due to the decentralized nature of P2P systems, trust management is an important part of these systems. Mathematical and statistical approaches are frequently used to model trust management problems in P2P systems. According to network type, P2P trust models are generally evaluated in two categories: distributed hash table (DHT)-based approaches and unstructured network-based approaches.

In DHT-based approaches, global trust information about a peer is stored by an archiver peer, which is selected by the DHT mechanism. Thus, the entire interaction history of a peer can be accessed from its archiver via the DHT. Aberer et al.'s model [2] was the first study in the field, introducing the reputation-based trust management concept and a DHT-based trust management algorithm. In this model, negative experiences are shared on a P-grid structure. However, this approach ignores the difference between a new and an old peer and makes the model vulnerable to whitewashing attacks. The Eigentrust model [3] calculates local and global trust values iteratively based on Eigenvector calculation over a distributed and decentralized model. The PeerTrust [4] model defined the concept of feedback credibility for the first time. PeerTrust has created a strong belief in the evaluation of similarity. Through the concepts of transaction and community context factor, modeling relationships can be customized according to semantics of transactions and community. Guo et al. [5] keep similarity measurements in a vector and propose a method to compute the vector via time effect. Liu et al. [6] propose an approach to detect malicious feedbacks and measure service quality with integrality, authenticity, and credibility metrics. FCTrust [7] uses a trust model based on feedback credibility for evaluating the trustworthiness of participants.

In trust models on unstructured networks, a peer maintains trust information about peers interacted with previously or peers in the neighborhood. Peers flood trust queries to their neighbors to learn trust information about a peer and neighbors forward queries to their neighbors and so on. However, the trust value computed from the collected data does not reflect opinions of all peers generally. The SORT [8] model manages trust relations with historical data and feedbacks of neighbors. In that study, service and recommendation contexts are defined and a service is evaluated with satisfaction, weight, and fading effect parameters. Cornelli et al. [9] proposed a model of reputation sharing, which is based on a distributed polling algorithm while maintaining the requestor's and provider's anonymity. Selcuk et al. [10] focused on preventing malicious nodes and infected content, as well as proposing a solution to safeguard the ownership and authentication of messages. Su et al. [11] propose the ServiceTrust model to measure the quality of service. The changes in local trust values are measured and *credibility* is imported from PeerTrust. Su et al. carried their work further with the ServiceTrust++ model [12] and included decay factor, similarity, threshold, controlled randomness, and jump strategy to the model.

Beyond statistical approaches, some other methods are applied to trust management. Song et al. [13]

calculated the local trust value of peers and the recommendation information by using a fuzzy logic inference model. Tian et al. [14] proposed an evidence-theory–based fuzzy trust model that combined advanced fuzzy rules with D-S evidence theory. FRTrust [15] applied a fuzzy model to cluster nodes based on semantic similarities between their resources. Guo et al. [16] classified fuzzy data over the maximum tree with fuzzy clustering for large-scale P2P networks to improve performance. GenTrust [17] evaluates service and reputation contexts separately and uses peer and interaction features as input for genetic programming computation to detect malicious peers. Liu et al. [18] proposed a trust model based on machine learning and used real datasets from eBay and Allegro. They grouped features as features of a node from itself, features from other nodes, and features of a service provided from a node.

3. Trust model

In the proposed trust model, each peer provides some resources or services and stores trust information about other peers. Peers and resources are assumed to have unique ids. A peer starts interactions with others by requesting their services. In an interaction, a peer becomes a provider peer if it provides a service. Otherwise, it is a receiver peer. Trust information stored by a peer is assumed to be efficiently accessed over a DHT structure. A peer cannot delete or damage its interaction history or trust information since this information about the peer is stored by another peer (archiver).

3.1. Archiver

An archiver of a peer stores all trust information about the peer. Archiver of peer x is denoted by A_x, which stores the following trust information about x:

- $\mathcal{F}_p(x)$:Feedbacks given about x as a service provider
- $\mathcal{F}_r(x)$:Feedbacks given by x as a service receiver
- Consistency ($PC(x)$) and trust ($T(x)$) values for x
- Continuing interactions

Each feedback is stored as a tuple. Assuming $f_i(x,y) = (s_i(x,y), FC_i(x,y))$ is the tuple representing ith feedback of x (service provider) given by y (service receiver), $s_i(x,y)$ represents the satisfaction value of $f_i(x,y)$ and $FC_i(x,y)$ represents the feedback consistency value of $f_i(x,y)$. An interaction may complete successfully, may be terminated by the provider without completing the service (might be due to going offline), or may be attacked if the service provider behaves maliciously during interaction. According to these cases, the service receiver assigns the satisfaction value as follows:

$$s_i(x,y) = \begin{cases} 1, & \text{if the interaction is successful} \\ 0, & \text{if } x \text{ terminates the interaction} \\ -1, & \text{if } x \text{ is malicious during the interaction} \end{cases} \quad (1)$$

An archiver may misbehave by providing false trust information about the archived peer. In the proposed model, it is assumed that each peer has multiple archivers. Thus, such attacks can be prevented by cross-validation of results from different archivers.

3.2. Feedback consistency

When a service receiver finishes its interaction with the service provider, it sends a feedback to the archiver of the service provider. The archiver calculates a feedback consistency value. Feedback consistency measures how similar a feedback is with the past feedbacks about a peer. Most studies in the literature [11,16,19,20] use vector-based comparisons to measure the similarity between two specific peers, while we aim to compare a single feedback value with all past feedbacks given about a peer. Thus, the vector-based comparison is not appropriate for our purpose. In other words, *feedback consistency* measures the similarity between a feedback and all past feedbacks about the evaluated peer. To evaluate this metric, the number of feedbacks with the same feedback values is considered a measure of feedback consistency. Assuming x provides a service to y and, as a result, ith interaction of x happens with y, the archiver of x (which is A_x) calculates the feedback consistency as follows:

$$FC_i(x,y) = \frac{[\mathcal{F}_p(x) \cap s_i(x,y)]}{[\mathcal{F}_p(x)]}, \tag{2}$$

where $[\mathcal{F}_p(x)]$ represents the number of feedbacks in $\mathcal{F}_p(x)$ and $[\mathcal{F}_p(x) \cap s_i(x,y)]$represents the number of feedbacks that have the same satisfaction values in $\mathcal{F}_p(x)$ with the satisfaction value $s_i(x,y)$.

3.3. Peer consistency

Peer consistency measures the consistency of a peer in terms of giving true feedbacks. Considering past feedbacks of a peer y as a service receiver, feedback consistency values of all previous feedbacks in $\mathcal{F}_r(y)$ can be considered a measure of peer consistency. Thus, we calculate peer consistency for peer y as follows:

$$PC(y) = \frac{\sum\limits_{f_i(*,y) \in \mathcal{F}_r(y)} FC_i(*,y)}{[\mathcal{F}_r(y)]}, \tag{3}$$

where $f_i(*,y)$ is the ith feedback given by y about a peer and $FC_i(*,y)$is its corresponding feedback consistency value.

While feedback consistency measures a feedback's similarity with the previous feedbacks about a peer, peer consistency measures how good a peer is at providing consistent feedbacks.

3.4. Calculating trust value

The archiver of a peer calculates a trust value for the peer by evaluating the feedbacks given about the peer as a service provider. When evaluating a feedback, feedback consistency and consistency of the feedback provider are considered. The archiver of a peer performs a trust calculation after receiving a new feedback about the peer. Assuming peer x provides its ith service to peer y and y sends its feedback $f_i(x,y)$ to A_x, the trust value of x is calculated by A_x as follows:

$$T_i(x) = \propto E_i(x,y) + (1 - \propto) T_{i-1}(x) \tag{4}$$

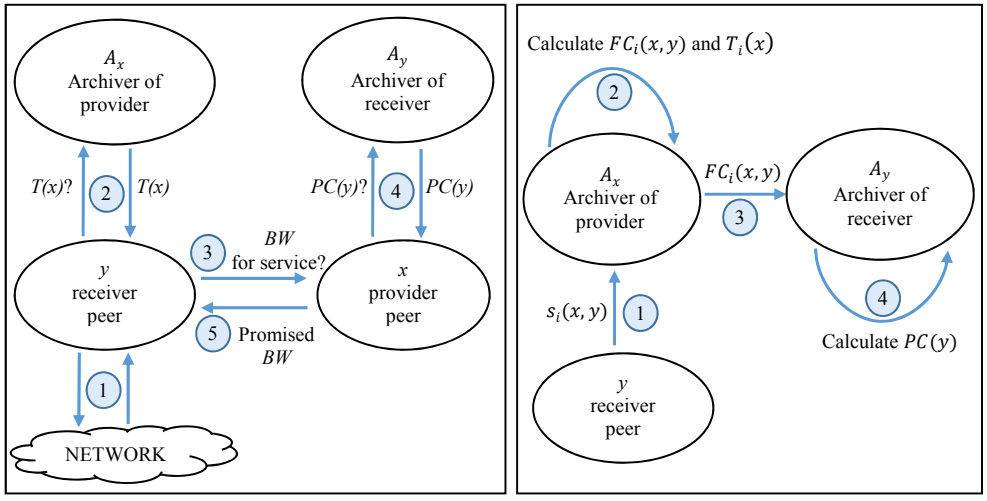
$$E_i(x,y) = s_i(x,y)\, FC_i(x,y) PC(y), \tag{5}$$

where $T_i(x)$ is the trust value of peer x after ith interaction, $E_i(x,y)$ is evaluation of ith feedback about peer x, and $0 < \propto < 1$ is a constant value to determine the effect of the last feedback on the trust value.

When calculating $E_i(x,y)$, consistency of the feedback ($FC_i(x,y)$) and consistency of the feedback provider (i.e. consistency of the service receiver $-PC(y)$) are considered. In this way, a feedback has more effect on the trust value if it is consistent with the previous feedbacks and its provider is consistent. To be able to bootstrap the network and give peers a chance to start interactions, each peer x is assigned to an initial trust value $T_0(x) = 0.2$ in our model. Furthermore, $\propto = 0.2$ to balance the effects of feedback history and the new feedback. These values were selected after performing extensive experiments.

3.5. Starting an interaction

Figure 1a shows how an interaction is started in our model. As the first step of starting an interaction (i.e. file download), the service receiver y queries the network to learn possible resource providers (Step 1). As a result of this query, a list of service providers and their archivers are returned by the network. In this study, it is assumed that all resource providers in the network can be learned with a single query. However, some network infrastructures may return only a group of providers, which does not affect our calculations. Then the service receiver y queries all archivers of service providers returned in Step 1. For ease of explanation, only the service provider x and its archiver A_x are shown in Figure 1a (Step 2). A_x returns x's current trust value $T(x)$. If this value is larger than a threshold value, y decides to send a request to x asking how much bandwidth it can allocate (Step 3). When selecting service providers, the trust threshold value is set to 0.8 at first. If there is no service provider having a larger trust value than the threshold, or the request of x is rejected by all service providers, the threshold value is decreased to 0.6, 0.4, and 0.2 until a service provider is found and accepts providing the service to x. In this way, x increases its chance of finding a service provider. If the threshold value reaches 0, the search is stopped and the service request is canceled. However, the search can be stopped at a higher threshold value if more trustworthy interactions are desired.



a. Starting an interaction

b. Finalizing an interaction

Figure 1. Lifecycle of an interaction.

When y requests the resource, the service provider x queries y's consistency value ($PC(y)$) from its archiver A_y (Step 4). If its consistency is higher than a threshold, x determines the amount of bandwidth/resource to promise and returns its bandwidth promise to y (Step 5). After performing extensive evaluations, we set the threshold value for $PC(y)$ as 0.5 in our model. If $PC(y) > 0.5$, x promises bandwidth as

a service provider to y. When calculating bandwidth promise, x considers its ongoing services and uses the following equations:

$$pBW_y = \frac{PC(y)}{tPC_x} \times tBW_x \tag{6}$$

$$tPC_x = \sum_{i \in R_x} PC(i), \tag{7}$$

where tBW_x is x's total bandwidth, pBW_y is the promised bandwidth for y, and tPC_x is the sum of consistency values of receiver peers that are currently receiving service from x, which are denoted by R_x. In other words, x shares its total bandwidth fairly among its service requesters based on their peer consistency values.

For ease of explanation, y requests service from only x in Figure 1a. However, in a general application, y may collect bandwidth promises from several service providers and select the peer who promises the greatest bandwidth.

3.6. Finalizing an interaction

Figure 1b shows how an interaction is finalized. When an interaction is completed or terminated, the receiver peer y sends its satisfaction value ($s_i(x,y)$) about the interaction to the archiver(s) of the provider peer x (Step 1). A_x calculates and stores feedback consistency ($FC_i(x,y)$) and trust ($T_i(x)$) values (Step 3). Then A_x sends $FC_i(x,y)$ value to the service receiver's archiver A_y (Step 3). Finally, A_y recalculates and stores y's peer consistency value, $PC(y)$ (Step 4). If y provides misleading feedbacks, $FC_i(x,y)$ will be low, which will decrease $PC(y)$ as well.

4. Experiment

To evaluate the proposed model, we implemented a simulation model based on the Peersim environment [1]. Peersim has cycle-based and event-based simulation capabilities. In this study, we designed a cycle-based environment to model a P2P file download application. At the start of each cycle, peers may start new interactions (i.e. file download), finish a completed interaction, or advance a continuing interaction. Each simulation configuration is run five times for 1000 cycles. The presented statistical results are the average of five runs. As stated in Section 3.4, $T_0(x) = \propto = 0.2$ in Eq (3). Due to space limitations, we do not present the experimental results that led to selecting these values. The most important statistics collected during experiments are given in the Table. Among these statistics, *startedServiceCount* is collected at the beginning of services but other statistics are collected after finishing services.

Table. Statistics collected in the simulation experiments.

Statistic	Description
startedServices	Number of services started in a cycle.
succeededServices	Number of good services finished in a cycle.
maliciousServices	Number of malicious services finished in a cycle.
terminatedServices	Number of services terminated in a cycle.
maliciousFeedbacks	Number of malicious feedbacks for finished services in a cycle.

4.1. Attacker model

In the experiments, we studied four types of attackers. A malicious peer's behavior is determined according to collaborating strategy and attacking frequency. Attackers may behave as either individuals or collaborators, according to collaboration strategy. When attacking frequency is considered, attackers may behave as either naïve or hypocritical.

4.1.1. Individual malicious peers

Individual malicious peers attack individually and do not collaborate with others. There are two types of them:

- Naïve individual malicious peers: A naïve individual malicious peer always serves infected resources (files) and gives misleading feedbacks.
- Hypocritical individual malicious peers: This type of attacker serves infected resources and gives misleading feedbacks to its victims with a given probability.

4.1.2. Collaborative malicious peers

Collaborative malicious peers behave as a collaboration group and work as a group to manipulate the system. They always serve good files and true feedbacks to each other, but might serve infected files and give malicious feedbacks to others. When submitting a malicious feedback, collaborators give a high recommendation if the feedback is about a collaborator; otherwise they give a bad recommendation. Thus, they try to increase trust values of each other in the system. We define two types of collaborators:

- Naïve collaborative malicious peers: A naïve collaborative malicious peer always provides infected files and gives misleading feedbacks to others.
- Hypocritical collaborative malicious peers: This type of attacker performs the same attacks as the naïve collaborative malicious peer but within an attack probability.

In the simulation experiments, the attack probability of hypocritical malicious peers is selected as 20%. This means that the hypocritical attacker serves infected resources and gives misleading feedbacks at a probability of 0.2.

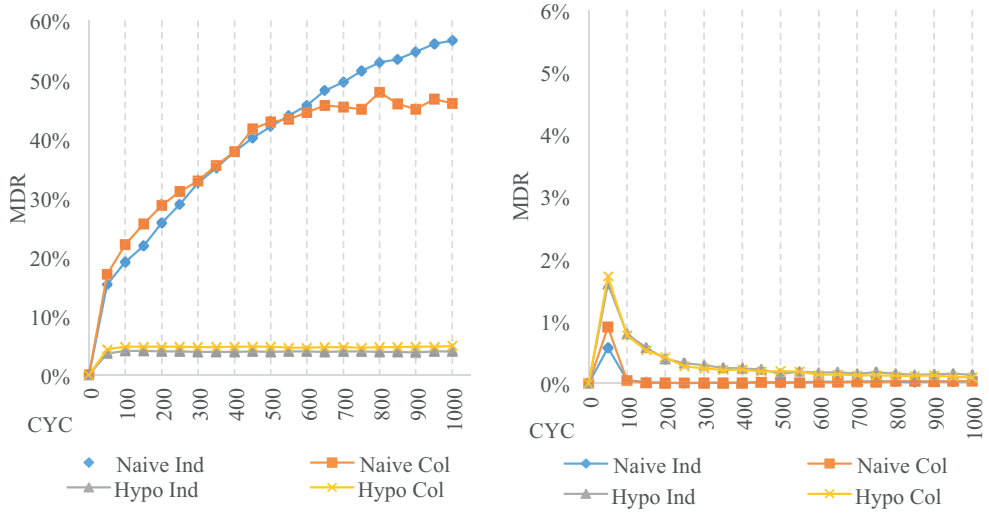
5. Results

In this section, we present the experimental results of the proposed model. In the experiments, malicious download rates, malicious feedback rates, the bootstrapping phase of the network, and an extreme scenario with a 50% malicious network are analyzed. Lastly, a comparison is done with the Eigentrust model [3]. In all simulation experiments except the Eigentrust comparison, the peer population consists of 10,000 peers. In the first three experiments, attackers occupy 20% of the network.

5.1. Malicious download rate

As a first experiment, we evaluate the effect of the proposed model on decreasing malicious download rate, which is measured as the ratio of malicious downloads to all downloads. Each simulation is run by using the trust model and then without any trust model, and then results are compared to understand how much the model decreases the malicious download rate. The run without any trust evaluation is the base case to understand how much attacks can affect system performance. Figure 2a shows the results of the simulation without a trust

evaluation. When a trust model is not used, the effect of naïve attackers increases continuously. As the naïve attackers collect more files, they receive more download requests and their attack capability increases. After 800 cycles, more than half of the downloads are flagged as malicious in the naïve attacker simulation. In hypocritical malicious peers, malicious download rate is stable at about 4%–5%, as in Figure 2a. Since they occupy 20% of the whole peer population and their attack probability is 0.2, their effect in all downloads is limited to 4%–5%.



a. Without trust model

b. With trust model

Figure 2. Malicious download rate (MDR) versus cycles (CYC).

Figure 2b presents the results when the trust model is used. In the early stages of the experiment, the trust model detects naïve malicious peers and malicious download rate decreases dramatically to 0.5% after 100 cycles. Considering cycles 900–1000, the successful download rate increases from 45% to 99.5% for naïve attackers compared to the no-trust case. Since the attack rate is low in hypocritical attackers, the increase in successful download rate is low for hypocritical attackers. The trust model increases the successful download rate from 96% to 99.5%.

As a result of these experiments, we can conclude that the model identifies naïve and hypocritical attackers after 50 cycles and decreases the malicious download rate.

5.2. Analyzing bootstrap of model

In order to observe effectiveness of the model in the bootstrapping phase of the network, we analyzed the first 50 cycles of the experiment. As seen in Figure 3a, at the beginning, malicious peers can affect the system easily without the trust model. Malicious download rates ramp up in the first 5 cycles, and then increase slowly in naïve attackers. Figure 3b shows the bootstrapping phase of the network with the trust model. Similar to Figure 3a, the malicious download rate peaks in the first 5 cycles, but then the rate starts to fall with the help of the trust model. After 20 cycles, the malicious download rate drops to below 0.2% and 2% for naïve and hypocritical attackers, respectively. As more trust information is gathered by archivers, good and malicious peers can be identified better in the later cycles. The model can easily exclude malicious peers, since all trust information can be accessed by all peers using the DHT-based network. However, it makes the trust model vulnerable to malicious feedbacks in the early cycles. After 5 cycles, the feedback consistency concept works properly and malicious download rates drop.

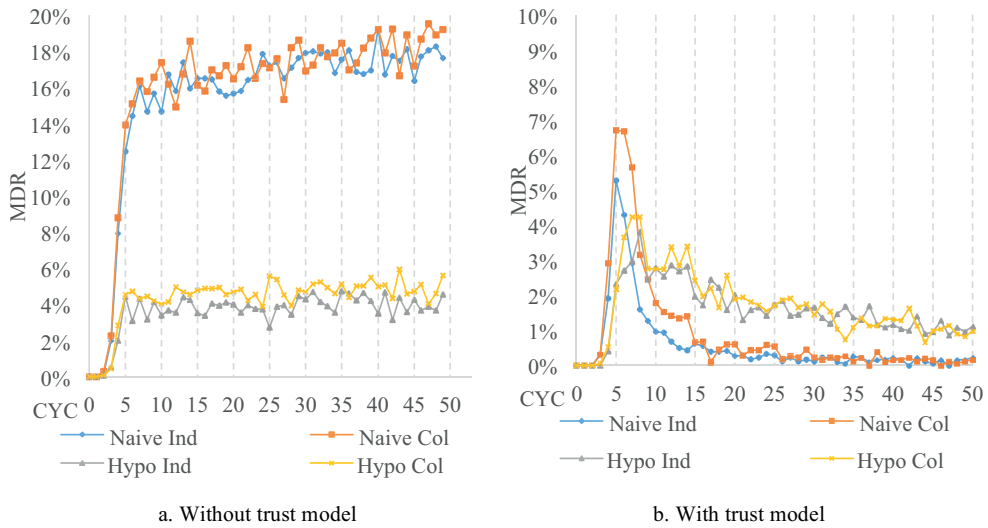


Figure 3. Malicious download rate (MDR) versus cycles (CYC) in the bootstrapping phase.

5.3. Malicious feedback rate

To further analyze the trust model, we analyzed the feedback and peer consistency concepts and performed experiments on these concepts. Figure 4a shows malicious feedback rates over time when consistency is off within the trust model. Malicious feedback rates take variable values under 30% for naïve attackers. Hypocritical attackers' malicious feedback rate stays stable at around 5%.

Figure 4b shows malicious feedback rates when consistency is activated. Naïve attackers are detected and isolated from the system in the early cycles (the lines of naïve individuals and collaborators overlap). Malicious feedback rate of hypocritical attackers stays stable over time, but their rate is decreased by nearly 20%–40% compared to Figure 4a. Although malicious feedbacks of hypocritical attackers are not eliminated substantially, the successful download rate increases from 95.6% to 99.5%, as discussed in Section 5.1. Thus, their effect on the whole system remains negligible.

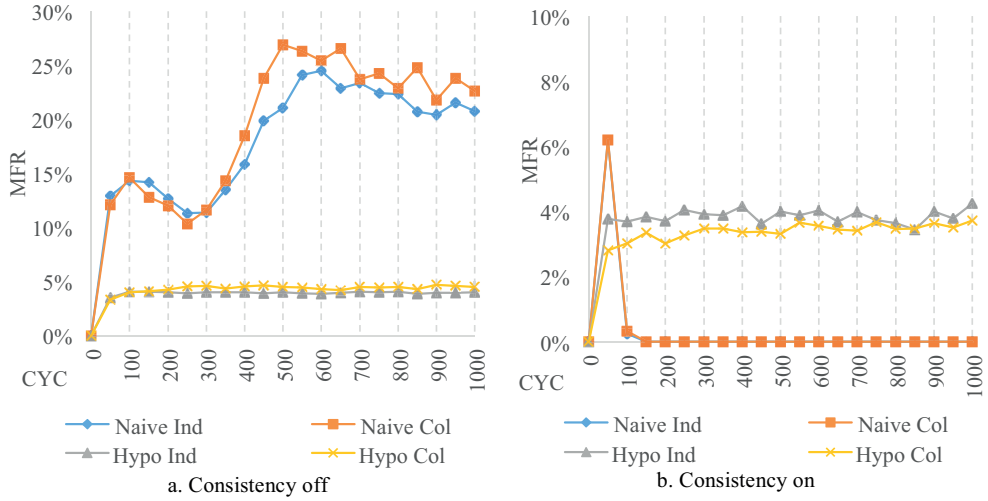
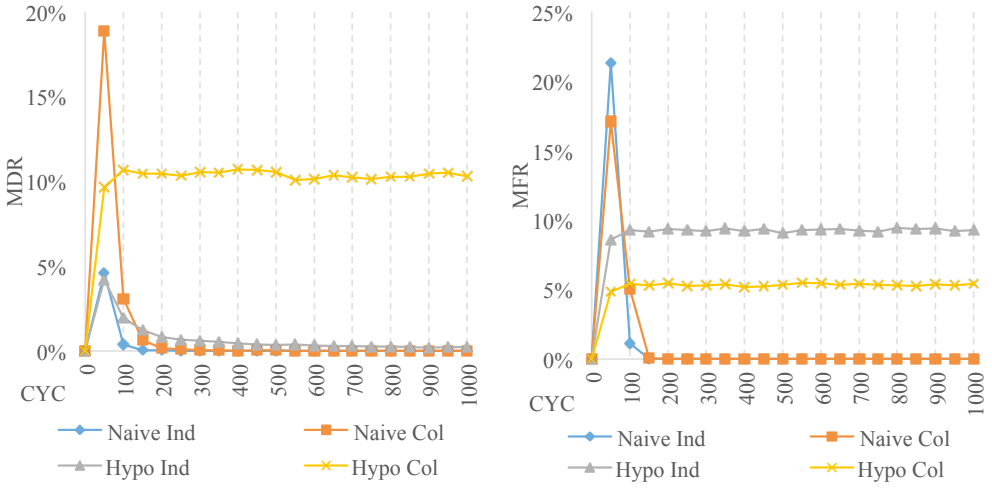


Figure 4. Malicious feedback rate (MFR) versus cycles (CYC).

5.4. Extremely malicious environments

A trust model should robustly mitigate attacks, even if the ratio of malicious peers is excessive in the network. In this experiment, 50% of the network is configured as malicious to analyze the trust model's performance in an extremely malicious environment. Figure 5a shows the malicious download rates observed in this experiment. Hypocritical collaborative attackers can deceive the trust model and convince good peers to download and then continue malicious uploads. Although the malicious download rate reaches 20% for naïve collaborators in the first 50 cycles, it quickly falls below 1% after 200 cycles. Naïve and hypocritical individuals can be detected earlier than collaborators.



a. Malicious download

b. Malicious feedback rate

Figure 5. Malicious download (MDR) and feedback (MFR) rates for the extreme scenario.

When we analyze the malicious feedback rates in Figure 5b, hypocritical attackers can disseminate more malicious feedbacks than naïve attackers. Since hypocritical individuals occupy 50% of all the population and their attack probability is 0.2%, we observe a steady 10% malicious feedback rate. Hypocritical collaborators provide malicious feedbacks only when interacting with good peers (50% of all peers). Thus, their malicious feedback rate remains around 5%. On the other hand, our trust model works well for naïve attackers and effectively prevents their malicious feedbacks after 100 cycles.

5.5. Comparison with Eigentrust algorithm

In this section, we compare our model with the best-known trust management algorithm, Eigentrust [3]. We implemented Eigentrust on the Peersim environment. As in our model, cycle-based simulation is performed to make a fair comparison. File distribution and download attempt ratios are the same for both models. Malicious peer counts, attack rates of hypocritical attackers, and collaborative activity of peers are also applied in the same way for both models. We evaluate the models based on two different parameters: malicious download rates and started download rates. Since reducing malicious feedbacks is not Eigentrust's primary goal, malicious feedback rates of the two models are not compared. In this section, all experiments are run with 1000 peers for 1000 cycles. In Eigentrust simulations, 30 pretrusted peers exist in the network.

All attacker models are applied in both models separately and their performance is observed. Our model performs better than the Eigentrust algorithm in all scenarios. Figure 6 shows malicious download rates for

different attacker models. In all cases, malicious download rates decrease close to 0% with our model. The Eigentrust algorithm also performs well, but it rarely falls under 2% malicious download rate. An important feature of a trust model is that the model should not decrease interactions and peers should be able to start interactions with reasonable trust level. Figure 7 shows the ratio of started interactions to all download attempts. While our model allows interactions to start in 70%–90% of all interactions, Eigentrust allows less than 40%. This shows that our model enables more interactions to start while keeping malicious download rates at a low level. In naïve attackers, since 20% of the peers are malicious and our model generally does not allow malicious peers to receive services, started download ratio remains around 80%. The same situation is observed with hypocritical attackers proportional to their attack rate. With the hypocritical attackers, about 95% of download intention is allowed in our model, while Eigentrust allows at most 40% of downloads. Thus, we can conclude that our model provides security by blocking download attempts of malicious peers, not good peers.

6. Conclusion

A consistency-based trust model is proposed to identify malicious peers in P2P networks. The feedback and peer consistency concepts of the trust model are robust methods for detecting malicious peers. The trust model is tested in the Peersim simulation environment for a file simulation application. Networks of 10,000 peers are

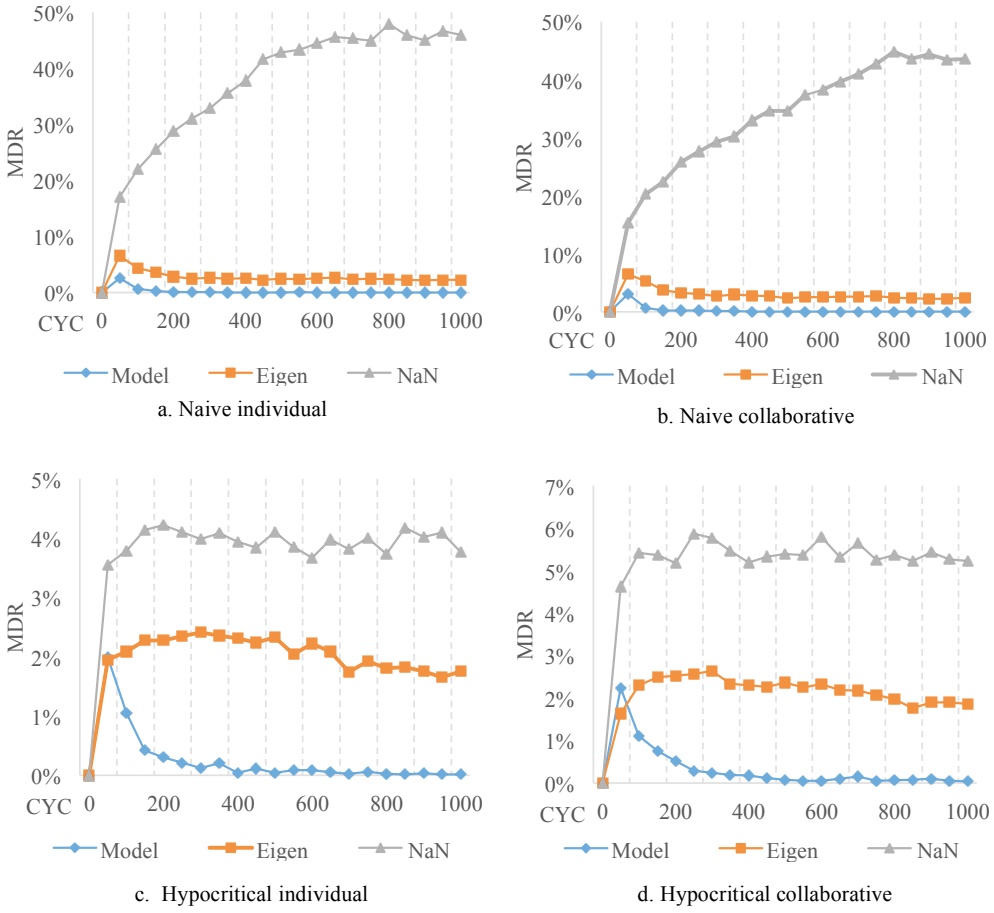


Figure 6. Comparison of malicious download rates (MDR) without a trust model (NaN) and with both models (Eigentrust and consistency-based trust models).

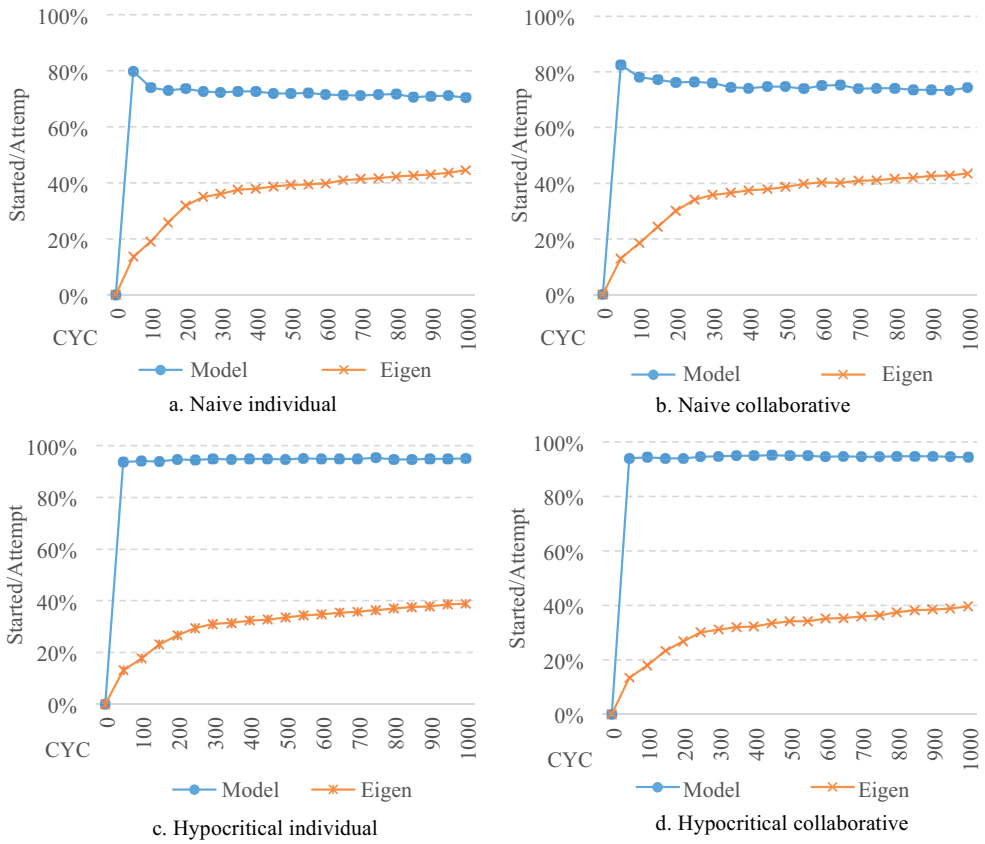


Figure 7. Comparison of started download rates over all download attempts.

modeled for a better simulation of large networks. Individual and collaborative attackers are simulated in naïve and hypocritical behaviors. Malicious download and feedback rates for four types of attackers are observed in the experiments. The simulation experiments show that the trust model decreases malicious download rates in all cases. Even with a 50% malicious network, the model mitigates the malicious download rate. In the worst case, the malicious download rate stays around 10% for hypocritical collaborators in a 50% malicious network, which is a promising result for such an extreme environment. Comparison with the Eigentrust algorithm [3] showed that the model decreases malicious download rates more than Eigentrust does, while allowing more downloads than Eigentrust does.

Although the malicious feedbacks of naïve attackers are decreased, hypocritical attackers can convince good peers and continue malicious feedbacks. This shows us an improvement opportunity for future studies. In future work, we plan to focus on hypocritical behavior in extremely malicious environments and mitigate malicious feedbacks and downloads with the help of new metrics and concepts. As another research direction, models based on machine learning can be studied to overcome these problems.

References

[1] Montresor A, Jelasity M. PeerSim: A scalable P2P simulator. In: IEEE International Conference on Peer-to-Peer Computing; 9–11 September 2009; Seattle, WA, USA. New York, NY, USA: IEEE. pp. 99-100.

[2] Aberer K, Despotovic Z. Managing Trust in a Peer-2-Peer Information System. In: Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM), 5–10 October 2001; Atlanta, GA, USA. New York, NY, USA: ACM. pp. 310-317.

[3] Kamva SD, Schlosse MT, Molina H. G. The Eigentrust Algorithm for Reputation Management in P2P Networks. In: 12th World Wide Web Conference; 20–24 May 2003; Budapest, Hungary. New York, NY, USA: ACM. pp. 640-651.

[4] Li X, Liu L. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. IEEE T Knowl Data En 2004; 16: 843-857.

[5] Guo L, Yang S, Wang J, Zhou J. Trust model based on similarity measure of vectors in P2P networks. In: International Conference on Grid and Cooperative Computing; 30 November–3 December 2005; Beijing, China. Berlin, Germany: Springer. pp. 836-847.

[6] Liu YM, Yang SM, Guo LT. A distributed trust-based reputation model in P2P System. In: Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing; 30 July–1 August 2007; Qingdao, China. Basel, Switzerland: Springer International. pp. 294-299.

[7] Hu J, Wu Q, Zhou B. FCTrust: a robust and efficient feedback credibility-based distributed P2P trust model. In: IEEE Young Computer Scientists; 18–21 November 2008; Hunan, China. New York, NY, USA: IEEE. pp. 1963-1968.

[8] Can AB, Bhargava B. Sort: A self-organizing trust model for peer-to-peer systems. IEEE T Depend Secure 2013; 10: 14-27.

[9] Cornelli F, Damiani E, di Vimercati SDC, Paraboschi S, Samarati P. Choosing Reputable Servents in a P2P Network. In: 11th World Wide Web Conf. (WWW); 7–11 May 2002; Honolulu, HI, USA. New York, NY, USA: ACM. pp. 376-386.

[10] Selcuk AA, Uzun E, Pari MR. A reputation-based trust management system for P2P networks. In: IEEE Cluster Computing and the Grid; 19–22 April 2004; Chicago, IL, USA. New York, NY, USA: IEEE. pp. 251-258.

[11] Su Z, Liu L, Li M, Fan X, Zhou Y. ServiceTrust: trust management in service provision networks. In: 2013 IEEE International Conference on Services Computing; 28 June–3 July 2013; Santa Clara, CA, USA. New York, NY, USA: IEEE. pp. 272-279.

[12] Su Z, Liu L, Li M, Fan X, Zhou Y. Reliable and resilient trust management in distributed service provision networks. ACM Trans Web 2015; 9: 1-37.

[13] Song S, Hwang K, Zhou R, Kwok YK. Trusted P2P transactions with fuzzy reputation aggregation. IEEE Internet Comput 2005; 9: 24-34.

[14] Tian C, Yang B. A DS evidence theory based fuzzy trust model in file-sharing P2P networks. Peer Peer Netw Appl 2014; 7: 332-345.

[15] Saeed J, Shojafar M, Shariatmadari S, Ahrabi SS. FR trust: a fuzzy reputation–based model for trust management in semantic P2P grids. Int J Grid Util Comp 2014; 6: 57-66.

[16] Guo L, Luo Y, Zhou Z, Ji M. A recommendation trust method based on fuzzy clustering in P2P networks. Journal of Software 2013; 8: 357-360.

[17] Tahta UE, Sen S, Can AB. GenTrust: A genetic trust management model for peer-to-peer systems. Appl Soft Comput 2015; 34: 693-704.

[18] Liu X, Tiredan G, Datta A. A generic trust framework for large-scale open systems using machine learning. Comput Intell 2014; 30: 700-721.

[19] Das A, Islam MM. SecuredTrust: A dynamic trust computation model for secured communication in multiagent systems. IEEE T Depend Secure; 2012; 9: 261-274.

[20] Xei Z, Geng Y, Bi J. STTM: Similarity transitivity chain based trust model in P2P environment. In: 2010 IEEE International Conference on Communications; 23–27 May 2010; Cape Town, South Africa. New York, NY: IEEE. pp. 1-5.

[21] Wang G, Musau F, Guo S, Abdullahi MB. Neighbor similarity trust against Sybil attack in P2P E-commerce. IEEE T Parall Distr 2015; 26: 824-833.

[22] Zhou R, Hwang K, Cai M. GossipTrust for fast reputation aggregation in peer-to-peer networks. IEEE T Knowl Data En 2008; 20: 1282-1295.

[23] Wen T, Zhong C. Research of subjective trust management model based on the fuzzy set theory. Journal of Software 2003; 8: 1401-1408.