hiREP: Hierarchical Reputation Management for Peer-to-Peer Systems

Xiaomei Liu, Li Xiao
Department of Computer Science and Engineering
Michigan State University, East Lansing, MI 48824
{liuxiaom, lxiao}@cse.msu.edu

Abstract

The open feature of peer-to-peer systems invites the spread of the malfunctioning data. Additional reputation systems are constructed to guarantee the data authenticity. One challenge in these systems is how to store and spread trust value securely and efficiently. Devoid of a central control system, most of the recently proposed P2P reputation systems adopt flooding based polling mechanisms, which need to inquire into every node in the system. The polling mechanisms create heavy traffic, do not guarantee voter anonymity, and make it hard for peers to filter out the fake trust values. In this paper, we propose a reputation management system - hiREP to address these problems. A peer in hiREP system only needs to contact a small group of reputation agents to obtain the trust values. hiREP guarantees data authenticity and voter anonymity, and also makes it easier for peers to filter out fake trust values.

1 Introduction

Peer-to-peer (P2P) systems are fully distributed with no central control. Anyone can freely join and leave the system. In addition, many P2P systems require that the service requestors (and/or providers) remain anonymous. These features make it easy to inject malfunctioning data into the system and hard to trace the data source. For instance, investigations show that large amount of "polluted" data have been injected into KaZaA, one of the most popular P2P systems [1]. To prevent the spreading of malicious data, and provide high quality services, research was deployed to constructing reputation systems in P2P networks.

The typical query process with a reputation system is shown in Figure 1: a requestor sends out a query request to the whole system. Upon receiving responses, the requestor chooses a group of file provider candidates and sends the trust value request to fetch the trust values of these candidates. Based on the received trust values, the candidate chooses the provider. Two issues are important here: how to compute the trust value of each node and how to distribute and access the trust values [2]. Constructing trust value computation model has been studied extensively [3-12] and is beyond the scope of this paper. Our research focuses on the storage and spreading of the trust values in the unstructured P2P systems.

Unstructured P2P system is the prevalent model of P2P systems in actual practice. Its flooding based query system is easy to implement and robust to node failures. However, the overwhelming traffic caused by flooding is the main hurdle of the system scalability and many mechanisms had been proposed to replace the pure flooding system.

With other issues of P2P systems in head, the following requirement should be fulfilled to design reputation systems in an unstructured P2P network: 1) the trust value spreading process of the reputation system cannot base on pure flooding mechanisms that cause overwhelming web traffic. 2) voter anonymity should be guaranteed to protect voters' privacy, i.e., the real identity of voters should be hidden from other parties. 3) the authenticity of the trust value should be guaranteed, i.e. the system should be robust to the attacks that try to invalidate the reputation evaluation. Most reputation system proposed recently can fulfill the third requirement and part of the second requirement. But none of them consider the first requirement in their design.

In this paper, we propose hiREP, a hierarchical reputation system for unstructured P2P systems that can fulfill all three requirements. Peers in hiREP can be divided into three types: general peers, *reputation agents*, and *trusted reputation agents* (or, *trusted agents* for short). Any peer with a bandwidth greater than 64k can choose to function as a reputation agent, but only a qualified one can be selected by other peers as a trusted agent. All trusted agents construct a reputation agent community. A peer reports transaction results only to its trusted agents, and checks only with its trusted agents to fetch the trusted values of other peers. hiREP limits the traffic created by trust value inquiries per peer to O(C), where C is the number of the trusted agents per peer.

In order to guarantee the anonymity of voters, hiREP adopts onion routing for the communication of a peer and its trusted agents. Each peer is assigned a unique *nodeID* together with the usage of public key system to provide authenticity of the votes.

This work was supported in part by the US National Science Foundation under grants CCF-0325760, CCF-0514078, CNS-0549006, and CNS 0551464.

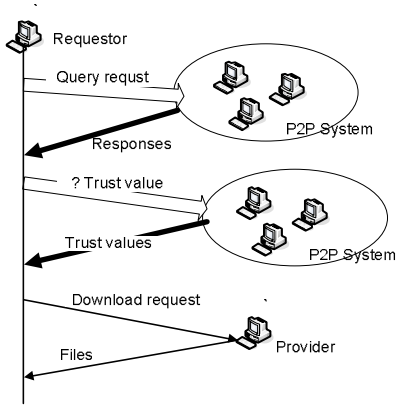


Figure 1 P2P systems with a reputation system

Our contributions in this paper are as follows:

- We introduce an efficient hierarchical structure to construct unstructured P2P reputation management systems. The challenge of constructing efficient hierarchical reputation management system in an open and anonymous system comes from how to locate the suitable trusted agents for each peer.
- Avoiding flooding based polling mechanisms in trust value distribution process, we greatly reduce the traffic overhead introduced by the system.
- By using public key hash as the nodeID of a peer, we construct a mechanism to distribute public keys in the P2P systems without third party certificate authority.
- With the adaptation of onion routing based communication in the reputation request process, we hide the real identity of voters with fairly light overhead.

The rest of the paper is organized as follows. Section 2 provides a survey of the related work. We present the design of hiREP in Section 3 and the system analysis in Section 4. Section 5 evaluates the performance of hiREP. The conclusion is presented in Section 6.

2 Background and related work

Many works about constructing effective reputation systems have been done in e-commerce society early [3, 8, 13, 14]. However, these systems are either too complicated to implement in P2P systems or rely on some central servers to provide necessary information or organize the system.

P2P systems can be divided into structured P2P systems and unstructured P2P systems [15]. For structured P2P systems, people can utilize its tightly controlled structure and the system information to distribute the trust value. Therefore, most efforts focus on constructing efficient trust value computation model.

Aberer and Depotovic [4] propose a binary trust model which utilizes only the complains for peers. Researchers in Georgia Institute of Technology show that computing peers' trust value only based on complains is not accurate enough and propose to compute trust value based on more metrics and context factors [9, 12]. They further formalize their trust computation model with five trust metrics: satisfaction feedback, number of transactions, feedback credibility, transaction context factor and general trust metric [11]. EigenTrust computes trust values based on both local experience and evaluation given by other peers, and tries to rule out the negative effects of malicious peers [5]. All of the aforementioned mechanisms utilize topology information and specific search/routing algorithm of the structured P2P systems to distribute the trust value messages in the system.

Instead of computing reputation for peers, Morselli et al. [7] propose a mechanism to protect the authenticity of cached index with the signed digests of generated node sets. However, this method doesn't provide the guarantee for the authenticity of file providers.

For unstructured P2P systems, it is hard and expensive to organize peers. Both TrustMe and P2PREP are using flooding based mechanism to transmit the trust values [2, 16]. In P2PREP, the trust value of the peers is locally computed by and stored in their transaction partners. The requestor broadcasts trust value query messages to the entire system. Upon receiving the request, all of the peers that own the trust value of the potential provider return the trust value back to the requestor.

Like hiREP, TrustMe also stores the trust value of a peer remotely. In TrustMe, a peer A that stores another peer B's trust value is called B's trust-holding agent (THA). Unlike hiREP, a peer's THAs are not chosen by itself but assigned randomly by the bootstrap server during the bootstrapping process. As the selection of THA is random, the probability of each peer to be a THA is similar. TrustMe is not a hierarchical system since trust values eventually are scattered all over the system, while in hiREP, trust values are stored in a relative small number of *trusted reputation agents*.

TrustMe deploys broadcasting twice in its trust value distribution and storage process. A requestor in TrustMe broadcasts the trust value query message to the entire system. After each transaction, a peer broadcasts transaction results to the entire system to let the THAs of its transaction partner store the results. In both P2PREP and TrustMe, public keys are used to guarantee the authenticity of the trust values, where key distribution either rely on out band communication or bootstrap servers.

Besides P2PREP and TrustMe, Gupta et al. [17] proposed a centralized reputation system on top of unstructured P2P network, which requires an extra reputation computation agent (RCA). Kinateder et al. [18, 19] use Chaum mix to provide pseudonymous communication in their proposed UniTEC systems, where the reputation request process is essentially a broadcasting based mechanism and a trusted third party is required as the certificate authority.

3 hiREP: A hierarchical reputation management mechanism

We will present here the design of hiREP in detail. hiREP focuses on the storage and distribution of trust value. It also guarantees the authenticity of the transaction results reported to the reputation agents.

3.1 Fully distributed, centralized, or hierarchical

A fully distributed mechanism looks like a natural fit for functionalities atop an unstructured P2P systems, which itself is fully distributed. In such systems, trust values of a peer are stored in its transaction partners locally. The trust requestor has to poll every node in the system to collect the trust values. This leads to a flooding based polling process.

Like the reputation system in the real world, the reputation systems for the e-commerce society are based on a centralized structure: credits of individual entity are reported to reliable centralized reputation management servers, which send out credit report to the other parties. However, a centralized system requires extra reliable servers to store the trust values, which is not practical and against the principle of unstructured P2P systems. Centralized structures are also inevitably accompanied with the problems like traffic bottleneck and single point of failure.

Adopting a hierarchical structure, hiREP tries to achieve the advantages of both the centralized and fully distributed system, and avoid their disadvantages. In the hierarchical reputation system, the trust values of nodes are accumulated and stored in a group of limited number of reputation agents. Unlike in centralized systems, reputation agents in a hierarchical system are composed of general peers instead of dedicated servers. At the same time peers only need to contact with a small group of trusted agents to obtain the trust values and evaluate these agents instead of massive number of individual nodes to filter out the malicious nodes when an attack happens.

3.2 Overview

In hiREP system, each peer selects a group of agents as its trusted agents. Any peer with a bandwidth greater than 64k can claim itself a reputation agent, though not every reputation agent can be trusted by other peers. Peers update their trusted agent lists periodically. A reputation agent computes the trust value of each node using its own trust value computation model. A peer sends trust value request messages and reports the transaction results only to its trusted agents. Peers and trusted agents form a hierarchical structure as shown in Figure 2.

Voter anonymity is preserved by adopting an onion routing based communication mechanism between peers and its trusted agents. A public key system is used to provide data authenticity in communication processes.

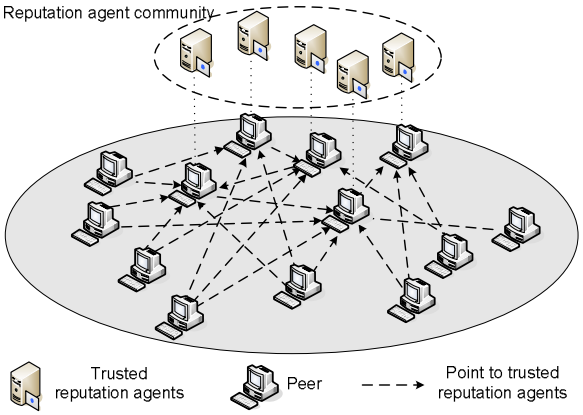


Figure 2 Hierarchical structure of hiREP

3.3 nodeID, public key system, and onion

Any peer in hiREP has two types of public key pairs: anonymity key pair (*AP*, *AR*) to help providing anonymity and signature key pair (*SP*, *SR*) to help providing message authenticity. *AR* and *SR* are the private keys; *AP* and *SP* are public keys. The nodeID is generated by the peer itself and is the hash of *SP* generated by a hash function like SHA-1. By associating *SP* with a node's nodeID, hiREP is effectively protected from Man-In-The-Middle attack: as nodeID is uniquely determined by *SP*, it is impossible for attackers to replace the public key of a particular nodeID. nodeID helps a peer to build its reputation in the system and only has relations with *SP*. Attackers cannot associate nodeID with a node's identity in real world such as its IP address.

(*AP*, *AR*) is associated with a peer's IP address and sent to other peers upon request. As shown in Figure 3, when a peer *P* picks peer *K* as its onion routing relay (*P* knows *K*'s IP address as it picks up K), it will send a routing relay request (R_o, AP_p, IP_p) directly to *K*, where R_o is the routing relay request. *K* then sends a response back to *P* with the form of $AP_p(AP_k, IP_k,$ nounce) to *P*, where nounce is used to prevent replay attacks. Upon receiving the response, *P* sends a public key verification message in the form of $AP_k(AP_P, IP_p,$ nounce) to *K*. When *K* receives message, it decrypts it and sends back a key confirmation message of the form AP_p(confirmed, $IP_k,$ nounce). If *P* cannot receive the confirmation, it knows AP_k is invalid.

After receiving the anonymity key from its onion routing relays, *P* can form its own onions which define a path to it. The onion format is:

$(((((((\text{fakeOnion})AP_p)IP_p)AP_1)IP_1)\ \ldots\ldots AP_k)IP_k, sq)\ SR_p$

AP_i is the anonymity public key of peer *i*. *sq* is the non-decrease sequence number used to indicate the age of the onion. SR_p is used to guarantee the authenticity of the onion. A node has *P*'s onion and SP_p can decrypt the onion using SP_p and sends messages to *K*. *K* then peels one more layer of onion using AR_k and sends the message to the next layer relay and so on until it reaches *P*. As the formats of the on-

ions sent to each relay are exactly the same, even the relay next to *P* does not know *P* is the receiver.

3.4 Reputation agent community formation.

In hiREP system, high performance reputation agents can be selected as trusted agents. All trusted agents form the reputation agent community.

3.4.1 Trusted agent list request. Each peer keeps a trusted agent list locally. The format for each list entry is like this: {weight, agent nodeID, $\text{Onion}_{\text{agent}}$, SP_e.}. Weight is decided by the expertise of this reputation agent. SP_e is the private key of the agent.

When a peer first joins the system or it wants to collect some good reputation agents with other peers' recommendations, it sends out a trusted agent list request with the format of {R_{al}, token, TTL}. R_{al} is the agent list request. The amount of agent list request messages is limited by both TTL value and token number. We recommend a default TTL value as 7 to be consistent with the query TTL value in Gnutella [20]. The amount of tokens equals the number of trusted agent lists that a peer wants to collect. A token was used up only when a node returns its trusted agent list to the requestor. The node can return its own nodeID if it has no trusted agent list. An example of the agent list request process is illustrated in Figure 4. Requestor *R* plans to collect six reputation lists from other peers. Therefore, *R* distributes agent request messages to its neighbors with 6 tokens in Figure 4(a). In Figure 4(b), *A* and *B* return reputation lists to *R* and use up one token respectively. As *C* doesn't have any reputation list, it forwards the message with untouched tokens to *F*. As *F* receives two tokens from *B* and *C* respectively, it uses one by itself and sends the message with the left tokens to *I*. *I* sends a list back to *R*, uses up the last token, and ends the message forwarding.

3.4.2 Agent rank and selection. Upon receiving the lists, the requestor ranks each reputation agent in different reputation lists according to their weight: assume the requestor wants to collect *n* reputation agents. For a reputation agent of the greatest weight, it is ranked as value *n*; the one of the second greatest weight is ranked as value *n-1* and so on. If there are more agents in a received agent list than what a requestor needs, say *m*, all the agents ranked less

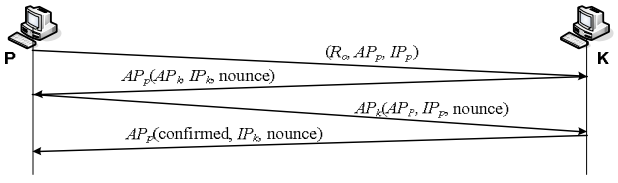


Figure 3 Fetch the anonymity key of a routing relay

than *n-m* will be assigned a rank value 0. For the same agent who gets different rank values from different agent lists, the highest rank value will be its final rank. The requestor then selects its trusted agents according to their ranks. If several agents have the same rank, requestor picks up its trusted agents from them randomly.

3.4.3 Trusted agent list maintenance. After selecting its trusted agents, a peer will assign an initial expertise value of 1 to each agent and keep updating the expertise values after every transaction. Assume accuracy of agent *E* in current transaction is A_c, and its cumulative accuracy of previous transactions A_p. The accuracy of agent *E* is $\alpha A_c + (1-\alpha)A_p$; $\alpha \in (0,1)$. Current accuracy A_c is either 0 or 1. It is 1 only when the evaluation given by this agent node is consistent with the transaction result.

If an agent is offline and its accuracy value is positive, it will be moved to the backup agent cache. Otherwise, it will be removed from the trusted agent list. Backup agent cache is updated following the most recently first principle. When the amount of agents in its agent list is smaller than some threshold, say 50, the peer first probes all back up agents. If the result is not satisfying, the peer will send out an agent request message to find other qualified agents.

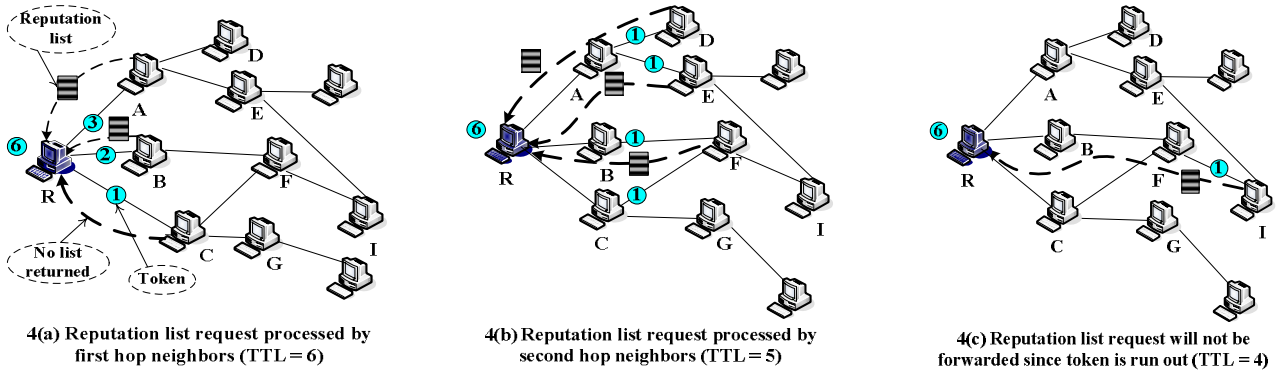
3.5 Trust value distribution

Trust value distribution should guarantee both anonymity of voters and the authenticity of trust values. The voters here refer to two parties: trusted agents that send trust values to the trust value requestors, and peers that send their transaction results to their trusted agents.

In hiREP, a trusted reputation agent keeps a public key list to store the public signature keys. The format of the list is: {nodeID_1, SP_1; nodeID_2, $SP2$; …nodeID_n, SP_n}, where SP_i is the public signature key of the node that chooses this agent as its trusted agent.

3.5.1 Trust value request. When a peer *P* wants to get the trust value of a particular node from its trusted agent *E*, *P* sends out the trust value request message using an onion of *E* stored in its trusted agent list. The format of trust value request is {$SP_e(R)$, SP_p, $Onion_p$}. SP_e is the public key of *E*. SP_p is the public key of *P*. $Onion_p$ is the Onion issued by *P*. *R* is the request message with the format {request, nonce}.

3.5.2 Trust value response. After receiving the trust value request message, *E* computes the nodeID of *P* using the pre-known hash function. *E* will add the nodeID and public key of *P* to its public key list if *P*'s nodeID is not in the list. *E* then sends back to *P* a trust value response message using $Onion_p$. The format of the message is {$SP_p(T)$, SP_e, $Onion_e$}. SP_p and SP_e are the same as in request message. $Onion_e$ is a fresh Onion issued by *E*. *T* is the response



4(a) Reputation list request processed by first hop neighbors (TTL = 6)

4(b) Reputation list request processed by second hop neighbors (TTL = 5)

4(c) Reputation list request will not be forwarded since token is run out (TTL = 4)

Figure 4 Trusted agent list request process

message with the format {trust value, nonce}, where nonce is the same one included in request message *R*.

3.5.3 Transaction result report. After a transaction, *P* will report the transaction results with the format of (SR_p(result, *nounce*), nodeID$_p$) to *E* using $Onion_e$. *E* then locates SP_p in its public key list using nodeID$_p$ and tries to decrypt the signed transaction result. If the result cannot be decrypted, the message will be dropped.

Voter anonymity and data authenticity are provided in all three processes. With the adaptation of onions, the real identity of the sender and receiver is hidden from each other and third parties. This provides protection for voters' privacy. The authenticity of both trust values and transaction reports is also ensured by the private key signature of senders.

Until now, we assume the public keys can not be cracked. This assumption can be loosed by allowing peers to update their public key pair periodically. New public keys signed by current private key can be sent out using the most recently received onions. It is also easy for a peer who receives the update message to map and replace an old nodeID to a new nodeID.

3.6 Transaction in a P2P system with hiREP

The transaction process here includes the query process, transaction (file downloading), and transaction reporting. The basic query process in a P2P system with hiREP is similar as the typical query process in other P2P reputation systems discussed in Section 1, except that trust value request will not be broadcast to whole system but requestor's trusted agents. After receiving the trust values, the requestor computes the final estimated trust value of the potential file providers and selects the one with the highest estimated trust value to download the file. After downloading, the requestor updates the expertise values of its trusted agents and sends its transaction results to all of its trusted agents as discussed in Section 3.5.

4 Analysis of hiREP

We here present the analysis of the traffic overhead and robustness of hiREP.

4.1 Traffic overhead

As the reputation list initialization is executed only once for each peer and all the other messages created in the hiREP system are either sent out only as necessary or piggied back in other messages, the main traffic overhead of hiREP comes from trust value distribution.

Assume each peer has c trusted agents in average. Each agent's onion has o_i relays and each transaction result reporter's onion has o_j relays. The messages created for trust value distribution in one transaction will be $2c(o_i + o_j)$. Considering that o_i and o_j are generally less than 10, the messages for the trust value distribution of one transaction are in the order of O(c).

4.2 Robustness against attacks

4.2.1 Manipulate trusted agents. Malicious nodes try to hinder peers in selecting proper trusted agents by giving multiple bad recommendations to reputation agents with high performance or multiple good recommendations to reputation agents with poor performance. The former case is discouraged by the system: as an agent is always ranked according to the greatest weight it received, the bad recommendation given by attackers will be ignored. As for the latter case, multiple high recommendations for an agent have the same effect as one single high recommendation.

The system can not completely prevent poor agents from getting high ranks, but attackers cannot render requestors to assign a poor agent a weight greater than all other agents. The point here is to guarantee good agents have chances to be selected and the requestor's reputation list is not overwhelmed with poor performance agents. In an extreme

Table 1 Simulation parameters

Name	Default	Description	Name	Default	Description
Network Size	2000	Number of peers in the network	Trusted agents of a peer	60	Amounts of trusted agents on a peers trusted agent list
neighbors per node	3	Average Number of neighbors each peer	Poor performance agents	10%	Agents which can not made proper reputation of peers
Good rating	0.6 - 1	Scope of good reputation rating	TTL	4	TTL limit used in pure voting flooding process
Bad rating	0 – 0.4	Scope of bad reputation rating	Token number	10	Initial number of tokens for obtaining reputation agent lists
Relies on average in an onion	7	Agencies a peer includes in its onion			

case, the trusted agent selection process will reduce to a random selection between the "real" good reputation agents recommended by sane peers and "fake" good reputation agents recommended by attackers. Poor performance reputation agents are then filtered out in the reputation list maintenance process.

4.2.2 Manipulate peer identities. In identity spoofing attacks, attackers send out trust values or transaction results using the identities of other nodes. This is not possible in hiREP. All trust values and transaction results are signed by the private keys which are associated with senders' unique nodeID. It is impossible for attackers to get the private key of the other peers.

In sybil attacks, attackers use multiple identities in a distributed system [21]. This is not avoidable unless the system has some centralized control server to strictly control the identity a node can have [21]. However, if we consider each identity as a particular node, hiREP can reduce the damage of the sybil attack by filtering out poor performance reputation agents based on its own experience.

4.2.3 Manipulate the reputation evaluation. Attackers try to invalidate the trust value evaluation by making good evaluations for "poor" peers and bad evaluations for "good" peers. hiREP guarantees the authenticity of the transaction reports sent to the reputation agents. With the authentic transaction reports, reputation agents can decide the trust value of the peer using the next level computation model. As a trusted reputation agent receives more information for trust computation than a peer based on local experience, it is expected to compute trust values more accurately.

4.2.4 DoS/DDoS attack. DoS/DDoS attacks may be initialized to disable the service of high performance reputation agents. To issue such an attack, the attacker has to first distinguish the high performance agents. The cost of distinguish such agents is not trivial. As traffic is spread among randomly chosen onion relays and reputation agents, it is hard to identify the high performance reputation agents by analyzing the traffic flow or the content of trust value request/response packet. The attackers have to go through all the process like a normal peer to figure out the high performance reputation agents. In addition, as the size of the reputation community is large, the peers that lose some of its trusted agents can easily replace them by other high performance reputation agents.

5 Performance evaluation

In this section we evaluate hiREP with a series of simulations. We first present the performance metrics and simulation settings. Then, we present the simulation results.

5.1 Performance metrics

Two major performance metrics are used in our simulation: *traffic cost* and *trust evaluation accuracy*. Traffic cost is used to indicate the network resources consumed in message delivering process, which is more critical than the local machine resources due to the rapid development of PCs. We use messages induced in the trust query process to represent the traffic costs, ignoring the individual bandwidth and the length of links.

Trust evaluation accuracy is an important metric that is used to evaluate the effectiveness of a reputation system. We expect our system be able to achieve at least the same level of trust evaluation accuracy as that in a pure voting system. Here we use the mean square error (MSE) between the estimated trust value and the true trust value of peers to indicate the evaluation accuracy of the system.

5.2 Simulation setup

We have generated a P2P network with power law topology using BRITE [22]. Each node is randomly assigned as trusted (trusted value 1) and untrusted (trusted value 0). The reputation agents (nodes with bandwidth larger than 64K) are divided into good agents and bad agents according to their capability to make trust value evaluation. A good agent gives trust values ranging from 0.6 to 1 to trustable peers, and trust values ranging from 0 to 0.4 to untrustable peers. A poor agent makes the inconsistent evaluation: 0.6 to 1 for untrustable peers and 0 to 0.4 for trustable peers.

The default values of simulation parameters are listed in Table 1. They may be varied in the experiments.

As in paper [2], we have compared hiREP with a pure voting system (called polling system in paper [2]). The trust making process is started with randomly selecting a peer as a potential service provider. Each node in the pure voting system computes a trust value and the overall estimated trust value is based on all of them. The flooding process is simulated by deploying a Breadth First Search based search operation.

For hiREP, only the trusted agents of the peers involved in a transaction compute trust values. After a transaction, these peers update the expertise values of and report the transaction results to the trusted agents. The above processes are simulated by re-computing the trust values (by trusted agents) and trusted agent expertise values (by these peers).

5.3 Simulation results

The traffic costs of hiREP and pure voting process are explored in Figure 5, where curves of voting-*n* represent messages incurred by pure voting mechanism in a network with average node degree of *n*. As messages incurred by hiREP are only decided by the number of trusted agents per node, the messages incurred by hiREP are the same in networks of different node degree. We can see that, even in a network with average node degree of 2, the number of messages produced in hiREP system is less than ½ of that produced in pure voting system. Due to the network size limit in the simulation, we set the TTL value of trust value request message in a pure voting system to be 4. In the real system, TTL value is generally set to be 7, which suggests more messages will be sent out. We can also observe that more messages in a pure voting system are sent out in a density network than in a sparse network.

Trust accuracy of the hiREP systems and that of the pure voting system are compared in Figure 6, with an assumption of 10% malicious nodes in the system. The curves of hiREP-*n* represent MSE of hiREP systems that adopt

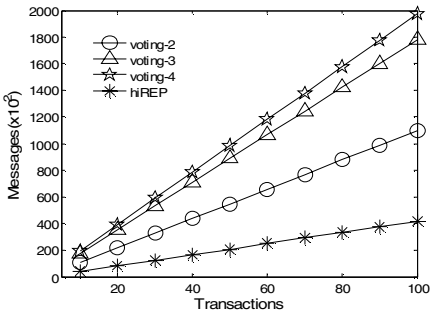


Figure 5 Trust query traffic cost of hiREP vs. pure voting process

different trusted agent threshold: hiREP-4 represents a system where a peer removes a trusted agent from its agent list if the expertise value of this agent is less than 0.4 and so on. We can see that the accuracy of hiREP is at least as good as that in pure voting. After a training process (about 100 transactions), hiREP reports trust value with much higher accuracy. MSE of trust accuracy of hiREP continues to reduce after 300 transactions, where the trust value accuracy is 90% and the system starts being stable. We can observe that a higher threshold results in a shorter convergence time.

We have investigated the effect of malicious nodes which give wrong evaluation intentionally. Figure 7 shows that trust value evaluation accuracy in pure voting system decreases much faster than that of the hiREP system. This may be because in pure voting system the trust value provided by each node is treated equally while in hiREP system, only the trust values provided by the agents of high expertise are accepted by other nodes. Therefore, malicious nodes lose their rights to express opinions due to their bad evaluation history. From Figure 7, we can observe that evaluation of pure voting may be more accurate when there are very few malicious nodes in the entire system. With the increase of the number of the malicious nodes, the performance of hiREP in trust accuracy will overwhelm that of the pure voting system. In an extreme case that 90% of reputation agents are poor performed, MSE of trust evaluation accuracy in hiREP is still under 25%.

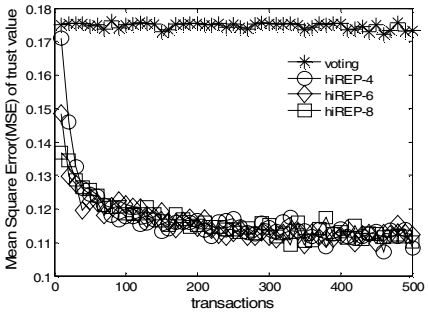


Figure 6 Trust accuracy vs. transactions

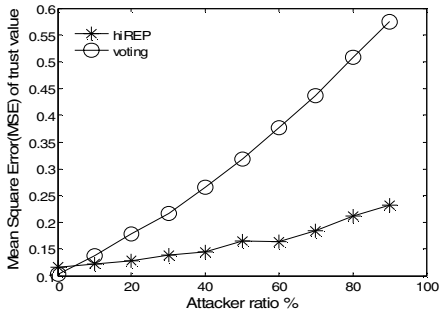


Figure 7 Trust accuracy vs. malicious nodes

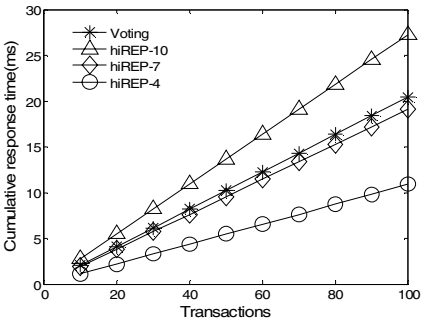


Figure 8 Cumulative response time of hiREP system

Besides the traffic cost and trust evaluation accuracy, we have investigated the response time of the trust value request process, which is defined as the time from a peer sends out request till it obtains the trust value. Figure 8 shows the cumulative response time of the pure voting system and the hiREP system. hiREP-*n* refers to the hiREP system in which there are *n* relays in an onion. We can observe that the decreases of the relay number result in the decreases of the response time of hiREP system. The average response time of hiREP is lower than that of the pure voting system.

6 Conclusions and future work

Reputation systems are developed in P2P systems to prevent malicious nodes from spreading the bogus data. Nevertheless, most of the reputation systems proposed for the unstructured P2P systems adopt the flooding based mechanism to distribute trust values, which creates heavy traffic overhead. Most reputation systems in unstructured P2P systems can not provide vote anonymity either.

In this paper, we have proposed hiREP, a hierarchical reputation management system to guarantee both efficiency and voter anonymity. The unique features of hiREP include: 1) a trust value distribution mechanism where the trusts value requestor communicates only with a limited number of trusted agents instead of polling every node in the system; 2) an onion based communication mechanism between peers and their trusted agents to guarantee the voter anonymity; and 3) a public key system which does not rely on third party certificate authority for key distribution to guarantee data authenticity in communication.

Our simulation shows that hiREP creates much less traffic cost and achieves better trust evaluation results than a flooding based polling system.

Future work for the hiREP includes developing a hiREP prototype based on current version of Gnutella open source code in PlanetLab [23], an open and shared test platform for developing planetary network services.

7 Reference

[1] J. Liang, R. Kumar, Y. Xi, and K. W. Ross, "Pollution in P2P File Sharing Systems," *in the proceedings of IEEE INFOCOM*, 2005.

[2] A. Singh and L. Liu, "TrustMe: Anonymous Management of Trust Relationships in Decentralized P2P Systems," *in the proceedings of P2P'03*, 2003.

[3] A. Abdul-Rahman and S. Hailes, "Supporting Trust in Virtual Communities," *in the proceedings of HICSS33*, 2000.

[4] K. Aberer and Z. Despotovic, "Managing Trust in a Peer-to-Peer Information System," *in the proceedings of CIKM*, 2001.

[5] S. D. Kamvar, M.Schlosser, and H.Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks," *in the proceedings of WWW*, 2003.

[6] S. Marti and H.Garcia-Molina, "Limited Reputation Sharing in P2P Systems," *in the proceedings of ACM Conference on Electronic Commerce (EC'04)*, 2004.

[7] R. Morselli, B. Bhattacharjee, J. Katz, and P. Keleher, "Trust-Preserving Set Operations," *in the proceedings of INFOCOM*, 2004.

[8] S. Marsh, "Formalising Trust as a Computational Concept", Ph.D., Thesis, Department of Mathematics and Computer Science, University of Stirling, 1994.

[9] M. Srivatsa, L. Xiong, and L. Liu, "TrustGuard: Countering Vulnerabilities in Reputation Management For Decentralized Overlay Networks," *in the proceedings of WWW*, 2005.

[10] Y. Wang and J. Vassileva, "Trust and Reputation Model in Peer-to-Peer Networks " *in the proceedings of Third International Conference on Peer-to-Peer Computing (P2P'03)*, 2003.

[11] L. Xiong and L. Liu, "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Communities," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, 2004.

[12] L. Xiong and L. Liu, "A Reputation-based Trust Model for Peer-to-Peer eCommerce Communities," *in the proceedings of IEEE International Conference on Electronic Commerce*, 2003.

[13] C. Dellarocas and P. Resnick, "Online Reputation Mechanisms - A Roadmap for Future Research," *in the proceedings of First Interdisciplinary Symposium on Online Reputation Mechanism*, 2003.

[14] B. Yu and M. P. Singh, "A Social Mechanism of Reputation Management in Electronic Communities," *in the proceedings of the 4th International Workshop on Cooperative Information Agents*, 2000.

[15] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks," *in the proceedings of ACM International Conference on Supercomputing*, 2002.

[16] F. Cornelli, E. Damiani, S. D. C. Vimercati, S. Paraboschi, and P. Samarati, "Choosing Reputable Servents in a P2P Network," *in the proceedings of World Wide Web Conference*, 2002.

[17] M. Gupta, P. Judge, and M. Ammar, "A Reputation System for Peer-to-Peer Networks," *in the proceedings of NOSSDAV*, 2003.

[18] M. Kinateder and S. Pearson, "A Privacy-Enhanced Peer-to-Peer Reputation System," *in the proceedings of the 4th International Conference on Electronic Commerce and Web Technologies*, 2002.

[19] M. Kinateder, R. Terdic, and K. Rothermel, "Strong Pseudonymous Communication for Peer-to-Peer Reputation Systems," *in the proceedings of SAC*, 2005.

[20] "The Gnutella protocol specification 0.6", *http://rfc-gnutella.sourceforge.net*.

[21] J. R. Douceur, "The Sybil Attack," *in the proceedings of IPTPS*, 2002.

[22] "BRITE", *http://www.cs.bu.edu/brite/*.

[23] L. Peterson, D. Culler, T. Anderson, and T. Roscoe, "A Blueprint for Introducing Disruptive Technology into the Internet," *in the proceedings of HOTNETS*, 2002.