GossipTrust for Fast Reputation Aggregation in Peer-to-Peer Networks

Runfang Zhou, *Member IEEE,* Kai Hwang, *Fellow IEEE Computer Society,* and Min Cai, *Member IEEE*

Abstract— In *peer-to-peer* (P2P) networks, reputation aggregation and peer ranking are the most time-consuming and space-demanding operations. This paper proposes a gossip-based reputation system (*GossipTrust*) for fast aggregation of global reputation scores. It leverages a Bloom filter based scheme for efficient score ranking. GossipTrust does not require any secure hashing or fast lookup mechanism, thus is applicable to both unstructured and structured P2P networks. Randomized gossiping with effective use of power nodes enables fast aggregation and fast dissemination of global scores in O($\log_2 n$) time steps, where n is the network size. The gossip-based protocol is designed to tolerate dynamic peer joining and departure, as well as to avoid possible peer collusions. The scheme has a considerably low gossiping message overhead, i.e. O($n\log_2 n$) messages for n nodes. Bloom filters reduce the memory overhead per node to 512 KB for a 10,000-node network. We evaluate the performance of GossipTrust with both P2P file-sharing and parameter-sweeping applications. The simulation results demonstrate that GossipTrust has small aggregation time, low memory demand, and high ranking accuracy. These results suggest promising advantages of using the GossipTrust system for trusted P2P computing.

Index Terms— Bloom filter, gossip communication, peer-to-peer computing, and reputation system.

1 INTRODUCTION

The growth of *peer-to-peer* (P2P) traffic is largely attributed to the applications in P2P file-sharing [11, 21], digital content delivery [21, 24], and P2P media and Grid services [12, 15, 28, 43]. Peer anonymity and dynamic joining or departure make P2P networks quite vulnerable to attacks by selfish and malicious peers [7, 9 37]. Most P2P file-sharing networks like Gnutella [13], KaZaA, eMule, and BitTorrent [25] are built with autonomous peers with different or even conflicting self-interests [37].

In the past, several reputation systems such as EigenTrust [16], PeerTrust [35], and PowerTrust [40] were proposed to evaluate peer behavior in structured P2P networks like Chord or CAN. Reputation management in P2P networks encourages resource sharing among well-behaved peers and fighting against selfish or malicious peers [6, 8, 18].

A good on-line reputation system enables peers to evaluate among themselves and to establish trusted interactions [25]. Without reputation management, peers may hesitate to interact with unknown peers due to the concern of receiving poisoned files or being compromised by malwares [7, 19]. Furthermore, identifying trustworthy peers is needed in P2P auctions, trusted content delivery, pay-per-transaction, and P2P service discovery [15, 32].

- *R. Zhou is with the Microsoft Corp, Mountain View, CA 94043. E-mail: ruzhou@microsoft.com.*
- *K. Hwang is with Department of Electrical Engineering, University of Southern California, Los Angeles, CA. 91789. E-mail: kaihwang@usc.edu.*
- *M. Cai is with VMware Inc, Palo Alto, CA 94304. E-mail: mcai@vmware.com.*

Manuscript TKDE- 0003-01077 received Jan. 3, 2007 and R1 version revised October 10, 2007 Fimalversion submitted Feb.11, 2008. .

By making the global reputation scores public, peers are able to make informed decision about which peers to perform the transactions with. So reputation systems help peers avoid unreliable or malicious peers. Any reputation system for unstructured P2P networks has to meet three basic requirements in low aggregation overhead, per-node storage efficiency, and score ranking accuracy.

First, computation and communication overheads for global reputation aggregation have to be considerably low. Second, the amount of space for each node to store collected reputation data has to be reduced as small as possible, preferably using a constant memory space. Third, the global reputation scores must be sufficiently accurate [42].

Our goal is to achieve the scoring error within a small window [$(1-\varepsilon)v$, $(1+\varepsilon)v$], where v is the true reputation score and ε is a small error fraction. The high accuracy demands increase in aggregation time and storage overhead. The tradeoffs among these three requirements are explored to choose proper design parameters.

This paper proposes a scalable, robust, and secure reputation management system called *GossipTrust*, specifically designed for unstructured P2P networks. This system applies a gossip-based protocol to aggregate global reputation scores rapidly and accurately. The idea is to allow peers sharing weighted local trust scores with randomly-selected neighbors until reaching some global consensus on peer reputations.

Gossip-based protocols [2, 14] do not require an explicit error recovery mechanism, and thus enjoy the simplicity and low overhead, compared with using deterministic protocols [17]. We aim to strengthen the robustness of the gossip protocol under abuses by malicious peers. The system is built with a novel space-efficient storage scheme for ranking reputation scores.

The remaining of this paper is organized as follows: Section 2 reviews related work on reputation systems for both structured and unstructured P2P networks. In Section 3, we introduce the GossipTrust system architecture. We present basic GossipTrust algorithms in Section 4. Then, we analyze GossipTrust performance in Section 5. We propose a Bloom filter-based storage scheme for fast retrieval of reputation data in Section 6. We report simulated performance in Section 7. Finally, we conclude with a summary of contributions and make suggestions for further research.

2 Related Work and Our Approach

In an open and decentralized P2P system, there is no centralized authority to maintain and distribute reputation data. A P2P reputation system calculates global peer scores by aggregating peer feedbacks in a distributed fashion [20, 26, 32]. Two important tasks: *local score aggregation* and *global score dissemination* are performed in this process.

These two tasks involve the scoring, collection, storage, and distribution of peer reputation scores that are dynamically changing [40]. The local scores are evaluated independently by individual peers. Just like eBay users, each peer provides a good or bad score for those who conducted transactions with them. Of course, the malicious peers may provide fake scores to disrupt the reputation system [35].

The peer collusion problem has been studied by many researchers [19, 35, 38, 39]. However, the peer global scores are more interesting to the general public. These global scores are generated out of consensus over all relevant local scores. The global score aggregation thus becomes the core process in P2P reputation systems. The process involves massive communications and data exchanges. [16, 27, 34].

Several reputation systems employed the *distributed hash table* (DHT) to support the global reputation aggregations. The EigenTrust algorithm [16] aggregates trust information by having peers perform a distributed calculation of eigenvectors of the trust matrix. In [20], peer reputation scores are aggregated by limited local peers. Related P2P content poisoning and copyright protection problems were studied in [7, 19].

Xiong and Liu [35] presented the *PeerTrust* that computes peer reputation scores based on five contributing factors. Srivatsa [28] proposes the *TrustGuard* to improve system robustness. *TrustMe* [26] offers another approach towards anonymous trust management.

We have proposed a *PowerTrust* system [38] for DHT-based networks, which leverages the Power law distribution of peer feedbacks. The concept of *power nodes* [10] was applied in PowerTrust development. Power nodes correspond to the most reputable peers in a P2P reputation system.

Most peers prefer to conduct transactions with power nodes. It was proved that power nodes dominate the global score aggregation process and thus the reputation updating process. The peer collusion problem is thus greatly alleviated by using the power nodes dynamically. Details of both PowerTrust and GossipTrust can be found in Zhou's Ph.D. Thesis [42].

Trust management in unstructured P2P networks was pioneered by Aberer and Despotovic [1] in 2001. Another approach uses Bayesian learning [33], by which the first-hand information is exchanged frequently and second-hand information is merged with current reputation ratings. The fault tolerance issue in unstructured P2P networks was studied in [36].

The gossip protocols do not rely on specific network topologies. They support the computation of aggregate functions like weighted sum, average value, and maximum over large collection of distributed numeric values [17]. The GUARD system was proposed for autonomous resource detection and membership management [23].

Our GossipTrust system aggregates a large number of local reputation scores to compute the global peer scores. The basic idea was first presented in *IEEE IPDPS-2007* [41]. This paper is significantly extended from the conference version. The idea of using Bloom filtering for trust information management was inspired by the work by Bloom, et al [3, 4]. We apply *identity-based encryption* (IBC) to secure the GossipTrust reputation system. Related work on IBS can be found in [4, 30].

3 GossipTrust System Architecture

In this section, we formulate the reputation aggregation problem in P2P netwoks. We define recursive computations on global reputation vectors in successive aggregation cycles. Then we introduce the GossipTrust system architecture and illustrate the functional modules of GossipTrust on each node.

This architectural model paves the way to specify the gossip protocols for reputation aggregation in Section 4. To benefit our readers with a quick reference, we summarize in Table 1 the notations and basic definitions used. All time instances are discrete and node identifiers are integers. These terms will be defined with more details in subsequent sections.

Table 1. Notations and Definitions of Basic Terms

Notation	Basic Definition
s_{ij}	Local trust score of peer *j* evaluated by peer *i*
$V(t) = \{v_i(t)\}$	Global trust vector over *n* peers
$k \le g$	Gossiping step and upper bound
$t \le d$	Aggregation cycle and upper bound
$x_i(k)$	Weighted score by peer *i* at gossip step *k*
$w_i(k)$	Gossip weight applied by peer *i* at step *k*
m	Bloom filter array size (bit count)
E	RMS global aggregation error
α	Greedy factor of peer random walking
θ	Peer selection threshold
ε	Gossiping error threshold
δ	Aggregation error threshold
γ	Percentage of malicious peers in system
$x_i(k)$	Consensus factor of peer *i* at step *k*

3.1 The Reputation Aggregation Problem

In a P2P network of n nodes, each node evaluates the trustworthiness of other nodes with local trust scores after conducting a P2P transaction, such as a file download. Consider a *trust matrix* $R = (r_{ij})$, $1 \le i, j \le n$, where r_{ij} is the local score issued by node i for node j. If there is no feedback between two nodes, r_{ij} is set to be zero.

For global reputation aggregation, each node must normalize all local scores issued by itself. The *normalized local score* s_{ij} is defined as follows:

$$s_{ij} = r_{ij} / \sum_j r_{ij} \qquad (1)$$

Then we have a normalized trust matrix $S = (s_{ij})$. Note that $0 \le s_{ij} \le 1$ and each row sum $\sum_{j=1}^{n} s_{ij} = 1$ for all rows $i = 1, 2, \ldots, n$. In other words, the normalized trust matrix S is a stochastic matrix, in which all entries are fractions and all row entries add to be 1.

It is possible to encounter sparse trust matrix. After sufficient number of aggregations, the entries of a trust matrix may even become all zero. This will not post a problem, because the Power nodes predominate the aggregation process based on Power-law distribution of peer feedbacks as proven in our PowerTrust paper [40].

Let $v_i(t)$ be the *global reputation score* of node i at *aggregation cycle t*, where $i = 1, 2, \ldots, n$ and $t = 0, 1, 2, \ldots, d$ for d cycles. The global scores of all nodes form a *normalized reputation vector* with n components $V(t) = \{v_i(t)\}^T$, where $\sum_i v_i(t) = 1$.

The iterative method specified below calculates the $V(t)$ at cycle t. Let $V(0)$ be the initial reputation vector value. For all iterative cycles $t = 1, 2, \ldots, d$, we generate successive reputation vectors, recursively, by performing the following matrix-vector computations:

$$V(t+1) = S^T \times V(t) \qquad (2)$$

Initially, all nodes are equally trusted, i.e. $v_i(0) = 1/n$, where $i = 1, 2, \ldots, n$. The iterative computation in Eq.(2) continues until the average relative error between $V(d)$ and $V(d+1)$ is lower than δ for a given *aggregation error threshold* δ at the last cycle d.

We have proved in [41] that $d \le \lceil \log_b \delta \rceil$ with $b = \lambda_2/\lambda_1$, where λ_1 and λ_2 are the largest and second largest Eigenvalues of the trust matrix S. The convergence threshold δ is predefined by system designers. After d cycles, the global reputation vector converges to the eigenvector of trust matrix S.

This recursive process is motivated by Markov random walk among nodes, which is widely used in ranking web pages. Consider a random surfer hopping from nodes to nodes to search for a reputable node. At each surfing step, the surfer selects a neighbor according to current distribution of local scores. The Markov chain distribution is the converged global reputation vector.

Both EigenTrust and PowerTrust have developed scalable algorithms to calculate the global reputation vector $V(t)$ for DHT-based P2P networks. In this paper, we propose a GossipTrust system for global reputation aggregation in unstructured P2P networks.

3.2 The GossipTrust Architecture

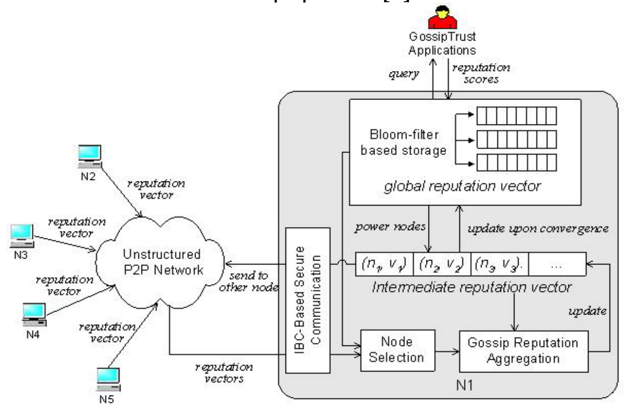
We have developed a simulated *GossipTrust* reputation system at USC Internet and Grid Computing Laboratory. Figure 1(a) shows the architecture of the GossipTrust system running on a typical node N_i, where $i = 1, 2, \ldots, n$. In this system, each node keeps a row vector of trust matrix S based on its outbound local trust scores.

In addition, each node also maintains a global reputation vector $V(t)$ at aggregation cycle t. Internally, this vector is represented by a collection of <*node_id*, *score*> pairs. At the first aggregation cycle, $V(0)$ is initialized with equal global reputation scores, i.e. $v_i(0) = 1/n$, i=1,2,…,n.

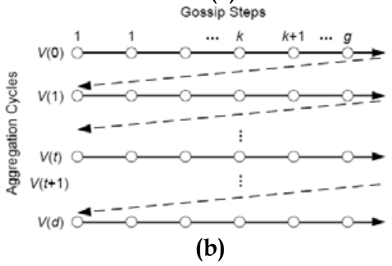
To compute the successive reputation vectors, GossipTrust uses a gossip-based protocol to perform the matrix-vector computation specified in Eq.(2). Gossiping supports light-weight communications among nodes during the aggregation process.

In GossipTrust, each aggregation cycle consists of several gossip steps as shown in Fig.1(b). In a gossip step, each node receives reputation vectors from others, selectively integrates the vectors with its current reputation vector, and then sends the updated one to a random node in the network.

This gossiping process continues until the gossiped scores converge in g steps, where g is determined by a set *gossiping error threshold* ε. After the convergence of gossip steps, GossipTrust continues the next aggregation cycle until the global reputation vectors converge in d cycles, where d is determined by the *aggregation error threshold* δ. To reduce the memory on each node, we design a novel Bloom-filter scheme for storage and retrieval of ranked global scores. A Bloom filter is a space-efficient data structure for membership queries [3].



(a)



(b)

Figure 1. GossipTrust architecture and global reputation aggregation process. (a) GossipTrust system architecture. (b) Global reputation aggregation by iterative gossiping.

4 Gossip-based Reputation Aggregation

In this section, we present the gossip protocol for global reputation aggregation. We first illustrate the Gossiping process to update the global score by an example. Then we introduce two gossip-based procedures for computing the global score of any node in Algorithm 1 and for aggregating the global reputation scores on all nodes in Algorithm 2, concurrently.

4.1 The Gossip Aggregation Protocol

The index k indicates the gossip step. According to Kempe, et al [17], k is upper bounded by a final step $g = O(\log_2 n)$. The time index t refers a discrete time instance of the aggregation cycle. The upper bound for t is d iterations specified in section 3.1.

Associated with each node i is a *gossip pair* $\{x_i(k), w_i(k)\}$ at step k. The $x_i(k)$ is the *gossiped global score* of node i up to step k. The $w_i(k)$ is the *gossip weight* applied by node i at step k. The ratio $x_i(k)/w_i(k)$ restores the global score of node i gossiped up to step k.

During each gossip step, each node i executes two computing threads: One thread sends a halved gossip pair $\{½\, x_i(k), ½\, w_i(k)\}$ to itself (node i) and to a randomly selected node. Another thread receives all halved pairs from other nodes and computes the updated pair $\{x_i(k+1), w_i(k+1)\}$ as follows, where the index j covers all source nodes that have sent a gossip pair to node i at step k:

$$x_i(k+1) = \textstyle\sum_j x_j(k)/2 \quad (3)$$

$$w_i(k+1) = \textstyle\sum_j w_j(k)/2 \quad (4)$$

This process continues until a *consensus score* value $x_i(k)/w_i(k)$ is reached agree on all nodes $i = 1, 2,\ldots, n$. The *global score* $v_j(t+1)$ for node i is thus generated on all n nodes at the final gossip step $k = g$.

$$v_i(g) = x_i(g)/w_i(g) \quad (5)$$

The above gossiping process is best illustrated by a small example in Fig.2. Consider 3 connected nodes of a P2P network with n nodes. Unconnected nodes are not shown in this drawing. All gossip pairs passed around between nodes are shown as edge labels. At current time t, global scores are given: $v_1(t) = 0.5$, $v_2(t) = 0.33$, and $v_3(t) = 0.17$. Given also normalized local scores $s_{12} = 0.2$, $s_{22} = 0$, and $s_{32} = 0.6$ issued by 3 nodes to evaluate node $N2$. By Eq.(2), the global score of node $N2$ is updated by:

$$\begin{aligned} v_2(t+1) &= v_1(t)\times 0.2 + v_2(t)\times 0 + v_3(t)\times 0.6 \\ &= 0.5\times 0.2 + 0.17\times 0.6 = 0.202 \quad (6) \end{aligned}$$

We use Table 2 to summarize how gossiping procedure can produce the same global score for node $N2$ in 2 steps. The end result is to reach the consensus that all nodes agree on the global score $v_2(t+1) = 0.2$ for node $N2$. Here, we focus on the gossiped calculation of the global score of node $N2$. Initially, we set the nodal weight $w_2(0) = 1$ because node $N2$ is the focal point of interest and the other two weights are set zero, $w_1(0) = w_3(0) = 0$. The initial weighted scores $x_1(0) = 0.5\times 0.2 = 0.1$, $x_2(0) = 0.33\times 0 = 0$, and $x_3(0) = 0.17\times 0.6 = 0.102$.

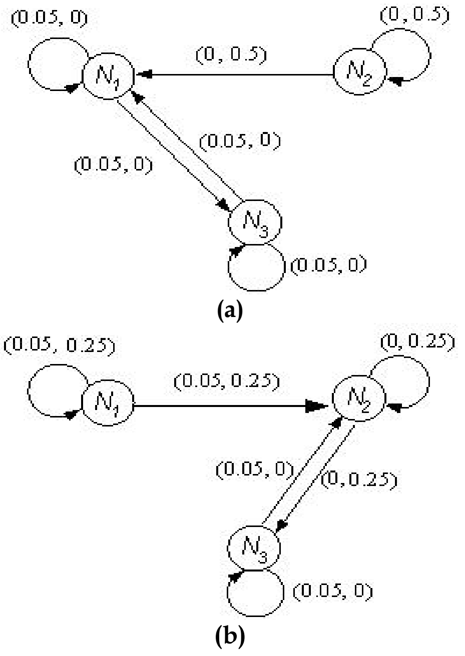


Figure 2. Gossip aggregation to generate the global score at node N2, where weighted {local score, weight} are distributed at (a) Step 1 and (b) Step 2 subsequently to reach the consensus of a global score 0.2 at node N2.

After the first gossip step in Fig.2(a), *N1* sends a gossip pair $\{0.5x_1(0), 0.5w_1(0)\} = (0.05, 0)$ to itself and to a randomly-selected node say *N3*. Similarly, node *N2* sends a gossip pair (0, 0.5) to itself and to a random node *N1.* Node *N3* sends a gossip pair (0.05, 0) to itself *N3* and the node *N1* pointed by the arrowed edge labels in Fig.2(a).

Node *N1* then adds up all received weighted scores, i.e. $x_1(1) = 0.05 + 0 + 0.05 = 0.1$ using Eq.(3) and updates the weight $w_1(1) = 0 + 0.5 + 0 = 0.5$ using Eq.(4). After first gossip step, *N1* has the pair $\{x_1(1), w_1(1)\} = (0.1, 0.5)$ as in the top row of Table 2. The restored score $x_1(1)/w_1(1) = 0.2$ is now available on *N1.* Similarly, nodes *N2* and *N3* go through the same process to produce the gossiped scores $x_2(1)/w_2(1) = 0$ and $x_3(1)/w_3(1) = \infty$ (an infinity).

Table 2 Two Gossip Steps to Aggregate Weighted Local Scores to generate the Global Score at Node N$_2$ in Fig.2

Node *N*1	**Score $x_1(k)$**	**Weight $w_1(k)$**	**Restored Score $x_1(k)/w_1(k)$**
Step 1	0.1	0.5	0.2
Step 2	0.05	0.25	0.2 (Consensus)
Node *N*2	**$x_2(k)$**	**$w_2(k)$**	**$x_2(k)/w_2(k)$**
Step 1	0	0.5	0
Step 2	0.1	0.5	**0.2 (Consensus)**
Node *N*3	**$x_3(k)$**	**$w_3(k)$**	**$x_3(k)/w_3(k)$**
Step 1	0.1	0	∞ (infinity)
Step 2	0.05	0.25	0.2 (Consensus)

Figure 2(b) illustrates the halved score sharing by the same gossip process in step 2. After step 2, we have reached the consensus that $x_1(2)/w_1(2) = x_2(2)/w_2(2) = x_3(2)/w_3(2) = 0.2$. Thus, we accept the agreed global score for node $N2$ as $v_2(t+1) = x_1/w_1 = x_2/w_2 = x_3/w_3 = 0.2$. This gossip result matches with the theoretical value calculated in Eq.(6). That proves the gossiping accuracy.

The 3 nodes agreed on the gossiped 0.2 global score for node *N2* at the end of the process. If we gossip another step, no more changes in the consensus value. Note that gossiping is taking place on all nodes simultaneously containing massive parallelism in the process, if millions of nodes are involved. Gossiping are done independently and concurrently at all nodes within each gossiping cycle.

4.2 Distributed Aggregation Algorithms

The iterative method in Eq.(2) specifies global reputation aggregation in each cycle. Mathematically, we need to compute the weighted sum of all local scores sij for each peer $j = 1, 2, \dots, n$ in Algorithm 1, where the normalized global scores {vi(t-1)} are the weights applied.

The numerical computation in Eq.(6) for the 3-node network example in Fig.2 is generalized for a general P2P network having n nodes indexed by $j = 1, 2, \dots, n$.

$$v_j(t) = \sum_{i=1}^{n} v_i(t-1) \times s_{ij} \qquad (7)$$

The gossip protocol exemplified in Section 4.1 is now generalized by the following Algorithm 1 for an *n*-node P2P network. The procedure shows how to perform the gossiping operation in *g* steps, when the gossiped scores agree on all nodes. Algorithm 1 is executed on all *n* nodes, concurrently.

The gossip process converges to consent global scores. In GossipTrust, every node has a unique identifier and keeps a global reputation vector. The global-scale gossip process is specified in Algorithm 2. During each aggregation cycle *t*, each gossip pair $\{x_i, w_i\}$ is represented by a triplet (x_{id}, id, w_{id}), where *id* is a node identifier, x_{id} is the weighted score in Eq.(3) and w_{id} is the weight in Eq.(4). We aggregate all triplets, concurrently.

Algorithm 1: Gossip Protocol to Compute The Global Peer Scores

```
1:  INPUT: local score $s_{ij}$, global score $v_i(t-1)$
        where $i = 1,2,\dots,n$ and gossip threshold $\varepsilon$
2:  OUTPUT: Global reputation score
        $v_j(t)$ of node $j$ at time $t$
3:  forall $i = 1, 2, \dots .n$ do
4:      $x_i \leftarrow s_{ij} \times v_i(t-1)$
            // initialize weighted score $x_i$ //
5:      if ($i == j$), set $w_i \leftarrow 1$, else $w_i = 0$
            //Initialize consensus factor $w_i$ //
6       $k \leftarrow 0$   // initialize gossip step k //
7:      repeat
8:          $u \leftarrow x_i/w_i$   // save previous score //
9:          let $\{(x_r, w_r)\}$ be all gossip pairs
                sent to $i$ in previous step
10:         $x_i \leftarrow \sum_r x_r$, $w_i \leftarrow \sum_r w_r$
            // Update the $x_i$ and weight $w_i$ //
11:         Choose a random node $q$
12:         send pair ($\frac{1}{2} x_i$, $\frac{1}{2} w_i$) to node $q$
                and node $i$ itself
13:         $k \leftarrow k+1$   // Next gossip step //
14:     until $|x_i/w_i - u| \le \varepsilon$
            // Check with gossip threshold //
15: endforall
16: output $v_j(t) \leftarrow x_i/w_i$
```

During each gossip step, every node *i* sends its reputation vector to a randomly chosen node, which can be a neighbor node or any node in the reputation vector. Upon receiving the global reputation vectors from others, node i updates x_{id} and w_{id} for every triplet in reputation vector. At the aggregation end, the process reaches a consensus with the same score x_{id}/w_{id} on all nodes.

Node *i* checks the difference between the current global reputation vector and the one from aggregation cycle *t*-1. If the difference is larger than a pre-defined aggregation error threshold δ, node *i* will enter the next aggregation cycle *t*+1. Otherwise, the global reputation vector has converged, node *i* will replace the triplet $<x_j, j, w_j>$ with the pair $<v_j, j>$, where $v_j = \beta_j = x_j / w_j$ is the converged global score of node *j*.

Algorithm 2: Aggregation to Update All Peer Scores Concurrently

```
1:  INPUT: local trust matrix $S=(s_{ij})$ and
        tolerable aggregation error $\delta$
2:  OUTPUT: Converged global vector $V(t)$
3:  forall $i = 1, 2, \dots .n$ do
      $t \leftarrow 0$  //Initialize aggregation cycle//
4   repeat
5:  forall local scores $s_{ij}$,
        initialize scores $x_i$, $w_i$, and $\exists_i$
6:        if ($t == 0$) then $x_j \leftarrow s_{ij} / n$
7:        else $x_j \leftarrow s_{ij} \times v_i(t-1)$
8:            if ($j == i$) then $w_j \leftarrow 1$
9:            else $w_j \leftarrow 0$
10:           Add $<x_j, j, w_j>$ to vector $V(t)$
11:       endforall
12:       repeat
13:        forall $j$ in $V_r$ do
14:           $x_j \leftarrow \sum_r x_j$ , $w_j \leftarrow \sum_r w_j$ ,
15:           update $<x_j, j, w_j>$ in $V(t)$
16:        endforall
17:     choose another node $q$ randomly//
18:     send ½ $V(t)$ to nodes $q$ and $i$ itself
19:         until all $\{\exists_i = x_j / w_j\}$ equalized
                and converged to $v_i(t)$.
20:         forall $j$=1, 2, …, $n$ do
21:           $v_j \leftarrow x_j / w_j$ and replace every
              $<x_j, j, w_j>$ with updated $<v_j, j>$
22:         endforall
23:         $t \leftarrow t+1$ // Next convergence cycle//
24:    until $|V(t) - V(t-1)| < \delta$
         //Test with convergence threshold//
25: endforall
26: output Updated global reputation vector
        $V(t) = \{v_1(t), v_2(t), \dots, v_n(t)\}$
```

5 Analysis of Gossip Algorithms

The errors incurred with gossip communication are accumulative over the entire process. We prove below that the gossip error is upper bounded. We prove the convergence of the GossipTrust algorithm. We then discuss how to handle the churn problem of dynamic peer joining and leave in a P2P network.

5.1 Error Analysis of Gossiped Scores

Let $V(t) = +v_1(t), v_2(t), \dots, v_n(t) ,^T$, be the reputation vector of all global scores, computed by Eq.(2) at the *t*-th aggregation cycle. Let $U(t) = +u_1(t), u_2(t), \dots, u_n(t),^T$ be the gossiped reputation vector through the process specified in Algorithm 2. The $v_i(t)$ and $u_i(t)$ are the actual and estimated global scores of node i at aggregation cycle t. Let $\varepsilon_i(d) = [u_i(d) - v_i(d)] / v_i(d)$ be the *relative gossip aggregation error* of node i score after d cycles.

Theorem 1:

The gossip aggregation error $\varepsilon_i(d)$ on the global score $v_i(d)$ is upper bounded by:

$$\varepsilon_i(d) = [u_i(d) - v_i(d)]/v_i(d) \le O(\varepsilon \log_2 n) + O(\varepsilon^2) \quad (8)$$

where ε is a preset gossip error threshold and d denotes the final convergence cycle, which is upper bounded by $O(\log_2 n)$.

Proof: After the first gossip step, we have

$$V(1) = \begin{bmatrix} v_1(1) \\ v_2(1) \\ \vdots \\ v_n(1) \end{bmatrix} = S^{\mathrm{T}} \times V(0) = \begin{bmatrix} s_{11} & s_{21} & \cdots & s_{n1} \\ s_{12} & s_{22} & \cdots & s_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ s_{1n} & s_{2n} & \cdots & s_{nn} \end{bmatrix} \begin{bmatrix} v_1(0) \\ v_2(0) \\ \vdots \\ v_n(0) \end{bmatrix}$$

and
$$U(1) = \begin{bmatrix} v_1(1) \pm \varepsilon_1(1)v_1(1) \\ v_2(1) \pm \varepsilon_2(1)v_2(1) \\ \vdots \\ v_n(1) \pm \varepsilon_n(1)v_n(1) \end{bmatrix}.$$

As every $\varepsilon_i(1) \le \varepsilon$, we have $(1-\varepsilon)v_i(1) \le u_i(1) \le (1+\varepsilon)v_i(1)$.

After convergence cycle 2, we obtain

$$S^T U(1) = \begin{bmatrix} s_{11} & s_{21} & \cdots & s_{n1} \\ s_{12} & s_{22} & \cdots & s_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ s_{1n} & s_{2n} & \cdots & s_{nn} \end{bmatrix} \begin{bmatrix} v_1(1) \pm \varepsilon_1(1)v_1(1) \\ v_2(1) \pm \varepsilon_2(1)v_2(1) \\ \vdots \\ v_n(1) \pm \varepsilon_n(1)v_n(1) \end{bmatrix}$$

$$s_{1i}v_1(1) + s_{2i}v_2(1) + \dots + s_{ni}v_n(1) = v_i(2)$$

$$-\varepsilon v_i(2) \le s_{1i}\varepsilon_1(1)v_1(1) + s_{2i}\varepsilon_2(1)v_2(1) + \dots + s_{ni}\varepsilon_n(1)v_n(1) \le \varepsilon v_i(2)$$

$$\begin{aligned}(1-\varepsilon)^2 v_i(2) &= (v_i(2) - \varepsilon v_i(2)) - \varepsilon(v_i(2) - \varepsilon v_i(2)) \\ &\le u_i(2) \le (v_i(2) + \varepsilon v_i(2)) + \varepsilon(v_i(2) + \varepsilon v_i(2)) \\ &= (1+\varepsilon)^2 v_i(2)\end{aligned}$$

$$(1-\varepsilon)^2 v_i(2) \le u_i(2) \le (1+\varepsilon)^2 v_i(2)$$

Hence, after d aggregation cycles, we observe the following bounds $(1-\varepsilon)^d v_i(d) \le u_i(d) \le (1+\varepsilon)^d v_i(d)$. Since the gossip threshold ε is set very small, e.g. $\varepsilon < 10^{-4}$, the relative gossip aggregation error is bounded by $\varepsilon_i(d) = [u_i(d) - v_i(d)]/v_i(d) \le (1+\varepsilon)^d - 1 = d\varepsilon + O(\varepsilon^2) = O(\varepsilon \log_2 n) + O(\varepsilon^2)$.

Q.E.D.

5.2 Convergence of Global Aggregation

To prove the convergence of global reputation aggregation after d cycles, we have to compare the between the gossiped reputation vector $U(d+1)$ at $d+1$ cycle with $U(d)$ at cycle d based on Eq.(2), This trigger us to define an *average convergence error* $\bar{\delta}$ of the gossiped reputation vector $U(d) = +u_1(d), u_2(d), \dots, u_n(d)$, at the d-th cycle as follows:

$$\bar{\delta} = \sum_{i=1}^{n} (|u_i(d+1) - u_i(d)| / u_i(d)) / n \quad (9)$$

We find the following error bound on this average aggregation error, which is averaged over the relative errors in n elements of the gossiped vector $U(d+1)$ compared with those in vector $U(d)$. Since both δ *and* ε are very small fractions, the following result proves that the gossip aggregation process is guaranteed to converge with a tolerable error threshold.

Theorem 2:

The global reputation aggregation process specified by in Algorithm 2 converges in d cycles within the following average aggregation error:

$$\bar{\delta} < \delta + 3d\varepsilon = \delta + 3\varepsilon O(\log_2 n) \quad (10)$$

where δ and ε are the given aggregation and gossiping error thresholds preset at the algorithm design time.

Proof: At the $d+1$ cycle, according to Eq. (2), we calculate the bound: $\sum_{i=1}^{n}(|v_i(d+1) - v_i(d)| / v_i(d))/n < \delta$. By Theorem 1, we obtain two inequities:

$(1 - d\varepsilon)v_i(d) \le u_i(d) \le (1 + d\varepsilon)v_i(d)$ and

$[1 - (d+1)\varepsilon]\, v_i(d+1) \le u_i(d+1) \le (1 + (d+1)\varepsilon)v_i(d+1)$.

So we have $\bar{\delta} = \sum_{i=1}^{n}(|u_i(d+1) - u_i(d)| / u_i(d))/n$

$< \delta + \sum_{i=1}^{n}(((d+1)\varepsilon v_i(d+1) - (-d\varepsilon)v_i(d))/v_i(d))/n$

$< \delta + 2d\varepsilon + 2d\varepsilon\delta < \delta + 3d\varepsilon.$

Therefore, Algorithm 2 converges in d cycles with an average error $\bar{\delta} < \delta + 3d\varepsilon$. ***Q.E.D.***

6 Bloom Filters for Reputation Ranking

To store global scores requires $4n$ bytes, when n is the peer count and 4 bytes is assumed to store one score. Bloom filters [3] have been suggested in many network applications including P2P overlay networks [5]. The Bloom filters are used to keep the hashed identifiers of all peers in different ranks of Bloom filters. When the converged global reputation vector is updated, their hashed identifiers are properly stored in the Bloom filters.

In practice, peers with higher reputation scores are preferred by clients. Using Bloom filters, we store relative peer reputation ranks instead of their numerical scores. We explore the space-efficiency of Bloom filters [3] to store and retrieve the ranks of peers on each node. Our goal is to use limited memory to cope with scalable growth of reputation database at each node.

6.1 Bloom Filters for Reputation Retrieval

Consider in Fig.3 a P2P network with $n = 6$ nodes, labeled as {0, 1, ..., 5}. Their current global scores are given: $v_0 = 0.05$, $v_1 = 0.2$, $v_2 = 0.3$, $v_3 = 0.1$, $v_4 = 0.3$, $v_5 = 0.05$. Figure 3 illustrates how to use several Bloom filters to store the global scores. Each *Bloom filter* requires m bits to hold multiple hashed encodings into the same class.

The example divides six peers into two ranks: Nodes (0, 1, 3, and 5) with scores below 0.25 in Rank 1 and those above 0.25 (nodes 2 and 4) in Rank 2. The *score boundaries* are marked by $b_0 = 0$, $b_1 = 0.25$, and $b_2 = 1$. To hold 6 peer identifiers, we choose $m = 8$ bits per array. We apply two hash functions: $h_1(x) = x$ Mod (8) and $h_2(x) = x+2$.

Initially, every rank has one bloom filter, and all bits in the filter array are set to 0. For example, to put node 3 to rank 1, two hash functions are applied, $h_1(3) = 3$ and $h_2(3) = 5$, to set 1 in bit 3 and bit 5 in the top filter in Fig.3(a). All other peer identifiers are encoded similarly.

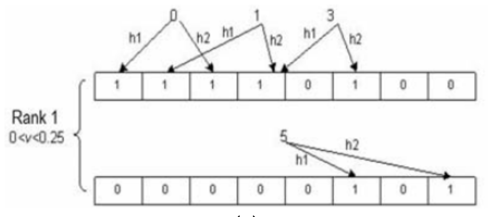
It is fine to have two peers (1 and 3) mapping into the same bit position (bit 3). Each Bloom filter can accommodate at most n/g (6/2=3) nodes (0, 1, 3). Thus we add another Bloom filter to encode node 5 in Rank 1. In total, Rank 1 requires 2 Bloom filters in Fig.3(a). Similarly, nodes 2 and 4 are encoded into one Bloom filter in Rank 2 in Fig. 3(b).

Consider a download from node 0 to node 4. The user applies hash functions $h_1(0) = 0$ and $h_2(0) = 2$ and finds both bits 0 and 2 are 1, so node 0 belongs to Rank 1. Since $h_1(4) = 4$ and $h_2(4) = 6$, in the first bloom filter in Rank 1, bit 6 is 0 and bit 4 is 1, so node 4 is not in this filter. The user then checks the second Bloom filter and finds bit 4 and bit 6 are all 0. It implies both Bloom filters do not have node 4. Then the user checks Rank 2 using the same method and find 1 in bits 4 and 6, indicating the membership of node 4 in Rank 2.

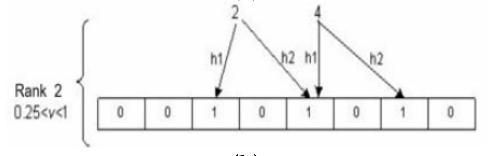
Finally, the user downloads from node 4 that has higher reputation. In this example, three Bloom filters require 3×8 = 24 bits, while 6×4×8 = 192 bits are required in using conventional memory to classify and encode peer identifiers into reputation ranks. Therefore, the storage is reduced by 8 times in this example.

Figure 4 presents the reputation database architecture for a general P2P network of n nodes. There are c reputation ranks with at most Bloom filters per rank, where n_i is the number of nodes in rank i. The rank classes are separated by score boundaries $\{b_i\}$, where $0 \le i \le c$ and $0 = b_0 < b_1 < \ldots < b_c = 1$. Rank i comprises encoded peer identifiers whose reputation scores falling in the range (b_i-1, b_i). A Bloom filer is in active state, if it is not encoded with n/c nodes.

The time needed to save a node to a proper ranking category is $O(\varphi+c)$, where φ is the number of hash functions applied and c counts the reputation classes. To identify the class for a node j, the system first checks whether all hashed bit positions $h_i(id_j)$ $(1 \le i \le \varphi)$ are set to 1 in all Bloom filters.



(a)



(b)

Figure 3 Bloom Filter storage for a 6-node P2P network. (a) Rank 1 encoded in 2 filters. (b) Rank 2 with 1 filter

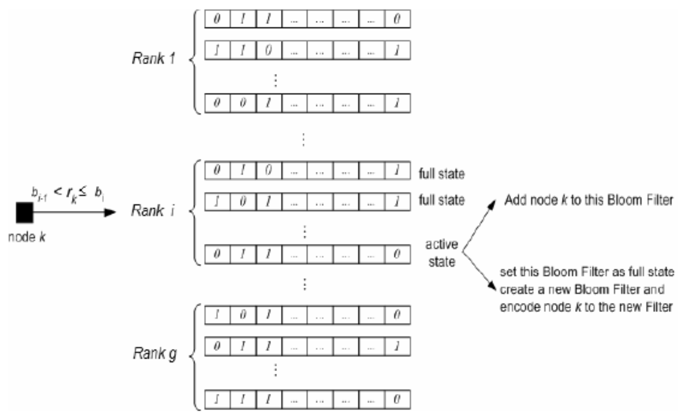

Rank 1
Rank i
Rank g
$b_{i-1} < r_k \le b_i$
node k
full state
full state
active state
Add node k to this Bloom Filter
set this Bloom Filter as full state
create a new Bloom Filter and encode node k to the new Filter

Figure 4 Bloom filter storage scheme for efficient retrieval of ranked peer reputations.

The system returns with the matching rank number. To identify the rank for a node j, the system first checks whether all hashed bit positions $h_i(id_j)$ $(1 \le i \le \varphi)$ are set to 1 in all Bloom filters. The system returns with the matching rank number. The notation f_i refers the number of Bloom filters in each rank class. In total, there are $F = f_1 + f_2 + \ldots + f_c$ Bloom filters to be used.

We store score ranks of all peers in each node, not their numerical score values. This is very different from traditional score storage, that keeps only each peer score locally. The global ranking information is thus replicated in all nodes. All local storages are updated concurrently and independently. When the gossip messages reach a consensus, all local ranking stores need an update.

6.2 Performance of Bloom Filter Storage

To evaluate the Bloom filter performance, we measure the *space reduction factor*, *false positive rate*, and *time complexity* under different storage configurations. We consider two network sizes. Table 3 summarizes the design parameters used in our simulation experiments on using these two Bloom filter designs. In practice, the ranking takes about $c = 10$ to 20 classes for a network up to one million of peer nodes.

The number of hash functions applied is 7 in both designs. Each Boom filter requires m = 1 K bits for the small network and 512 K bits for a very large network. The *space reduction* is the ratio of memory bits to store all global scores in conventional memory array to that needed in a Bloom-filter storage. In our design, 3.2 times of reduction in memory was achieved.

Table 3 : Performance of Two Bloom Filter Configurations for Global Reputation Storage

Network Size, *N*	1,000	1,000,000
Reputation Class, *c*	10 classes	20 classes
Hash Function, ϕ	7 functions	7 functions
Bloom Filter No., *m*	1 K bits	512 K bits
False-Positive Rate,	7%	15%
Space Reduction	3.2 times	3.2 times
Time complexity	7 steps	140 steps

The false positive rate is the probability to return a wrong rank index to a reputation query. This rate should be made as small as possible. Theorem 3 reports the results on estimation of false-positive rate in a general n-node P2P system.

Theorem 3:

The probability that a rank i Bloom filter returns a false-positive ranking is approximated by:

$$Prob_{fp}=1-(1-(1-e^{-\varphi n/cm})^{\varphi})^{\lceil n_i c/n\rceil}(1-(1-e^{-\varphi(n_i-\lceil n_i c/n\rceil n/c)})^{\varphi}) \quad (11)$$

where n_i is the number of nodes in rank i and c is the number of classes in the ranking storage.

Proof: The false positive probability of the first $\lceil n_i c/n\rceil$ Bloom filters in rank i is $(1-e^{-\varphi n/gm})^{\varphi}$, and of the last Bloom filter is $(1-e^{-\varphi(n_i-\lceil n_i c/n\rceil n/c)})^{\varphi}$. Then for all the Bloom filters in rank i, the probability of not all the bits indexed by φ hash functions being set to 1 is thus calculated by $(1-(1-e^{-\varphi n/cm})^{\varphi})^{\lceil n_i c/n\rceil}(1-(1-e^{-\varphi(n_i-\lceil n_i c/n\rceil n/g)})^{\varphi})$. Thus, the probability of all bits being set to 1 is estimated by $1-(1-(1-e^{-\varphi n/cm})^{\varphi})^{\lceil n_i c/n\rceil}(1-(1-e^{-\varphi(n_i-\lceil n_i c/n\rceil n/c)})^{\varphi})$. ***Q.E.D.***

Plugging in practical values into Theorem 3, we find that the false-positive rate is indeed very small 7 % in Table 3. The memory bit count is decreased 3.2 times. A maximum of 7 hashing operations are applied to query the reputation class for one million of nodes.

Increasing the rank class number c to 20 demands significant increase in m and in search steps. The optimal configuration for a 1,000,000-node P2P network is set at c=20, φ = 7 and m= 512 K bits. These results suggest that Bloom filters can reduce the memory space by 3.2 times compared to using a conventional RAM memory array for the same purposes.

6.3 Peer Collusions and Traffic Overhead

We have revealed the fact that peer feedbacks in eBay are Power-law distributed [40]. This phenomenon is primarily caused by the fact that new users joining the reputation system prefer to interact with the most reputable nodes, called *Power nodes* [10]. This distribution is also observed by Yang, et al in the Maze reputation system [38]. One can enhance the aggregation process of global reputations by leveraging the Power nodes. They play a major role in combating peer collusions.

In reality, the peer nodes continuously join and leave a P2P network. This is known the *churn problem.* Our use of dynamically elected Power nodes can handle this problem nicely, because Power nodes predominate the reputation aggregation process and rarely leaving the scene. Furthermore, Kempe, et al [17] have proved that gossip communication is robust against message loss and link or peer failure.

Consider a system with p power nodes. If node j is a power node, the probability of choosing this node is set to be $1/p$. Ordinary peers are unlikely to be chosen and thus has a zero probability, initially. A peer acts as a random-knowledge surfer to search for power nodes in the P2P network. With a probability a, the surfer attaches itself to a power node. In GossipTrust, the power nodes are re-elected periodically based on the updated global scores after each round of reputation aggregations.

In GossipTrust, each aggregation cycle consists of g = O(log$_2$ n) gossip steps. Reputation vector converges at the very last aggregation cycle. The number of aggregation cycle depends on the gap between first largest Eigenvalue and second largest Eigenvalue of the trust matrix, independent of the P2P network size. This proves the scalability of the GossipTrust system.

In each aggregation cycle, the maximum network traffic is thus bounded by O($n$log$_2$$n$) messages for an n-node network. Since the total number of aggregation cycles (the parameter d in Table 1) is independent of the system size. The total *network traffic overhead* to update the global reputation vector is estimated by O(dnlog$_2$$n$) messages. These theoretical results are validated by simulation experiments in the next section.

In using the *identity-based encryption* (IBC) to secure gossip message passing between nodes, the sender encrypts the message and signs the encrypt message. the receiver verifies the message and decrypt message. Some small delays may be introduced from signing, encryption, verification, and decryption. Analysis of IBC overhead is beyond the scope of this paper.

7 Simulated Performance Results

We evaluate the performance of GossipTrust system through extensive simulation experiments. We present simulated results to demonstrate the performance gains of GossipTrust in selected benchmark applications. In P2P file-sharing experiments, we assume a Power-law query distribution in Section 7.4. We execute the PSA (*parameter sweeping application*) benchmark workload in large-scale P2P grid experiments in Section 7.5.

7.1 GossipTrust Simulator and Environment

We designed an event-driven simulator of GossipTrust. We construct a Gnutella-like flat unstructured P2P network initially consisting of 1,000 nodes. The simulation experiments run on a dual-processor Dell server with Linux kernel 2.6.9. Each data reported is averaged over 10 simulation runs. The base setting and default values used in experiments are summarized in Table 4.

Table 4 Parameters and Default Values Used

Notation	Parameter Meaning	Default Value
n, c	n nodes divided in c reputation ranks	1000, 5
α	Greedy factor to select a power node	0.15
d_{max}, d_{ave}	Max. and ave. peer feedback amounts	200, 20
γ	Percent of malicious peers	10%
p	No. of power nodes in a system	1%
δ, ε	Tolerable thresholds of aggregation and gossip errors	10^{-3}, 10^{-4}
θ	Peer selection threshold	3×10^{-4}
λ	Query popularity rate i	1.2
M	Number of jobs in PSA workload	40,000

Initially the maximum feedback d_{max} is 200 and the average feedback d_{avg} is 20. We choose a greedy factor α = 0.15 by default. The system selects 1% of nodes as the Power nodes. The parameter θ is the *peer selection threshold* with a default value 3×10^{-4}. We measure the reputation aggregation time and gossip aggregation error. We use 5 Bloom filters in each score-ranking class.

7.2 Convergence Rate and Aggregation Errors

Two convergence processes for gossiping and reputation aggregation are simulated. We measure both the number of reputation convergence cycles and the number of gossip aggregation iterations. Our studies prove that GossipTrust only runs a small number of aggregation cycles before convergence.

Given an arbitrary trust matrix and a fixed greedy factor $a > 0.1$. The larger the number of gossip steps, the higher is the convergence overhead. Figure 5 shows the effects of various gossiping error threshold ε and network size n on the number of gossip steps. The convergence overhead increases with the decrease of gossiping error threshold and growth of network size.

When the gossiping error threshold ε is very small, as $\varepsilon < 10^{-4}$, the gossiping error threshold dominates the convergence time. While when ε is large, as $\varepsilon > 10^{-2}$, the network size dominates the convergence overhead. With a fixed small ε, the convergence is caped with almost constant time, which means that GossipTrust is able to scale well when the reputation system grows. The relative error incurred is propagated across cycles.

Figure 6(a) shows the convergence overhead under different percentage of message loss and ε, in a network of 1000 nodes. We observed that when the percentage of message loss is less than 15%, the gossip step count (d) grows slowly for decreasing ε. As the percentage of message loss increases, the convergence overhead increases faster .

Under the condition of the fixed gossip error threshold ($\varepsilon = 10^{-4}$) shown in Fig.6(b), we observed the slow convergence when the percentage of message loss is higher than 15% for various network sizes. GossipTrust can tolerate moderate percentage of message losses up to 15% caused by the dynamism or unreliability of peers and link failure in the network.

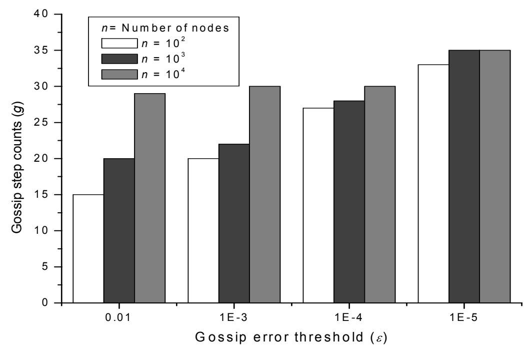


Figure 5. Gossip step counts of three GossipTrust configurations for various gossip error thresholds

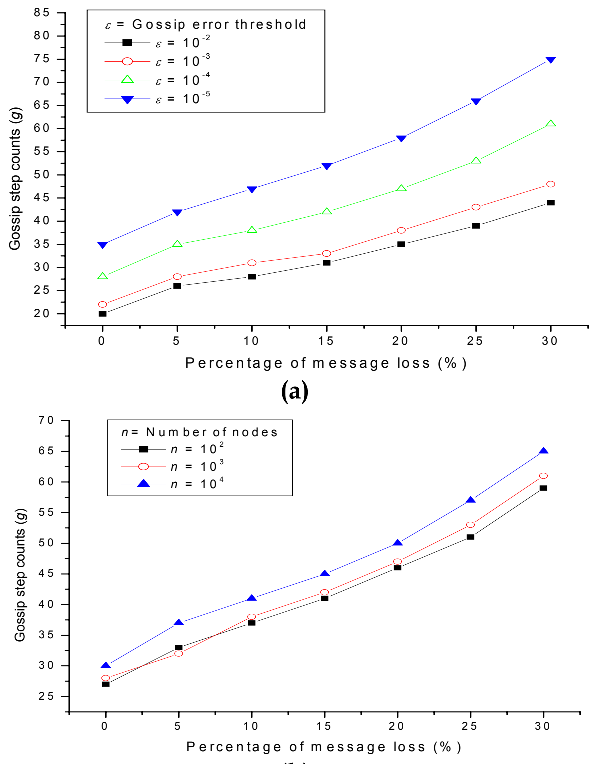


Figure 6 Gossip step counts of GossipTrust under various message loss rates. a) Network n= 1,000 nodes. (b) Fixed error threshold $\varepsilon = 10^{-4}$

Tradeoffs do exist between computational efficiency and scoring accuracy. The smaller is the aggregation error δ and gossiping error ε, the less is the propagation of computation error. Table 5 reports propagation and computation errors under various aggregation error threshold δ and gossip error threshold ε. We measure the propagation error by the relative error in global reputation scores caused by gossip protocol.

The computation error is measured as the distance between actual and estimated global scores. The traffic size increases linearly with the network size. For a network of 1,000 nodes, we choose $\varepsilon = 10^{-4}$ and $\delta = 10^{-3}$ to balance the design between convergence overhead and computational accuracy. The bottom row shows the number of gossip messages, representing the traffic overhead in global aggregation. These results verified the theoretical bounds given in Section 6.3.

Table 5: Gossip and Aggregation Errors in Three GossipTrust P2P Network Configurations

Threshold Values in System Design	$\varepsilon = 10^{-5}$, $\delta = 10^{-4}$	$\varepsilon = 10^{-4}$, $\delta = 10^{-3}$	$\varepsilon = 10^{-3}$, $\delta = 10^{-2}$
Network Size, *n*	100	1,000	10,000
Aggregation Cycles, *d*	**33**	**28**	**29**
Gossip Steps, *g*	**23**	15	6
Gossip Error	2×10^{-5}	7×10^{-6}	2.3×10^{-4}
Aggregation Error	4.2×10^{-5}	7.3×10^{-4}	4.3×10^{-3}
Network Traffic Overhead	7.6×10^{4} messages	4.2×10^{5} messages	1.7×10^{5} messages

For the 10,000-node network, only 6 gossip steps are enough per aggregation cycle with a 0.43% error. If the gossip error threshold ε is set small enough, i.e., as small as less than 10^{-4}, the propagation error is very low for different network size. This indicates small relative error caused by gossip protocols will not affect the accuracy of the reputation aggregation scheme.

7.3 Performance under Different Threat Models

We evaluate the robustness of GossipTrust against malicious peer behaviors. The experiments were performed to compare non-collusive and collusive peer operations. In a non-collusive setting, malicious peers report dishonest trust scores independently.

In a collusive setting, abusers collaborate with each other to boost up their own ratings. They rate the peers in their collusion group very high and rate outsiders very low. The probability of a node behaving maliciously is inversely proportional to its global reputation, because a node providing corrupted services is highly likely to issue dishonest local trust scores.

We compute below the *root-mean-square* (RMS) error E in aggregated global scores under different percentage of malicious peers in a P2P network. Lower RMS error implies the system is more robust to attacks by malicious peers. The RMS error is defined by:

$$\textit{RMS aggregation error} \quad \mathbf{E} = \sqrt{\frac{\sum((v_i - u_i)/v_i)^2}{n}} \qquad (12)$$

where v_i and u_i are the calculated and gossiped global reputation scores of peer *i*, respectively.

The greedy factor α indicates the eagerness for a peer to work with selected power nodes [37]. We plot in Fig.7(a) the RMS errors under different values of α and various percentages of independent malicious peers. By leveraging power nodes, we set the greedy factor a = 0.15. This gives 20% less aggregation error than treating all peers equally with a = 0.

However, increasing the greedy factor α to 0.3 does not lead to higher performance. This is because relying too much on the power nodes will miss the global view of the reputation data provided by majority nodes in the system. Therefore, setting α = 0.15 is a very good choice.

Figure 7(b) reports the RMS aggregation error under collusive peers working collectively to abuse the system. The plot represents the effects of various collusion group sizes, defined by the number of malicious peers in a group. Leveraging power nodes with α = 0.15 makes the system more robust against peer collusions.

With 5% collusive peers, using power nodes has resulted in 30% less errors when collusion group size is greater than 6. The message being conveyed is proper use of power nodes are indeed effective to cope with peer collusions. In general, the possibility of a peer behaving maliciously increases inversely with respect to its global reputation score.

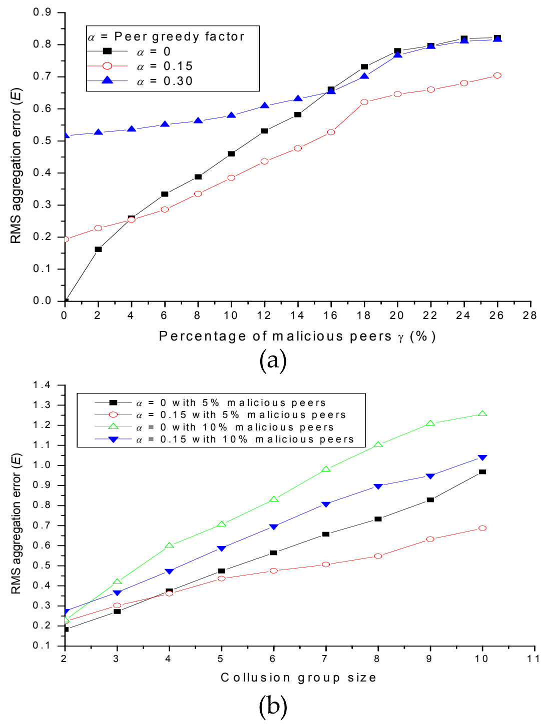


Figure 7: Effects on global aggregation errors by fake trust scores from malicious peers in a 1,000-node P2P network. a) Independent peers. (b) Collusive peers

Figure 8 plots the RMS aggregation error with respect to different percentage of malicious peers in a 1000-node P2P network. When the selection threshold θ is 0, it implies that all nodes will accept the reported scores from other nodes without suspicion. The RMS aggregation error grows rapidly from 0 to 0.915 with increase of malicious nodes. The RMS error increases slowly when θ is low 0.0001 under the condition of 10% malicious peers.

We test the case by setting θ = 0.0005. Figure 8 shows that the RMS error increases to 30%, even when the malicious peers are just a few, say less than 5%. Therefore, a good choice of the peer selection threshold is $\theta = 3\times10^{-4}$ under the experimental setting. By Fig. 7 and Fig.8, we see that proper choices of a and θ can control the error in the global reputation aggregation process.

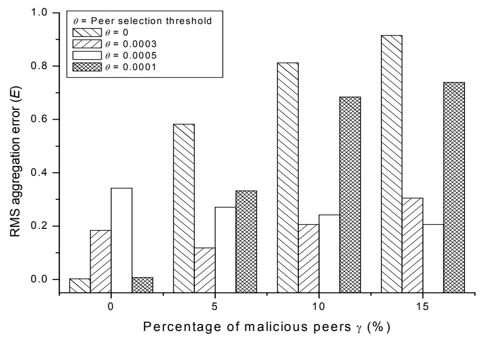


Figure 8. Effects on RMS aggregation errors due tofake scores by malicious peers in a 1000-node P2P network

In summary, the RMS error increases rather slowly as θ increases beyond 0.0003. The plot shows a 60% less RMS error, compared with the case of $\theta = 0$, when malicious peers are higher than 5% of the total population. The message is that if the peers only accept the scores from the reputable nodes whose global scores are higher than 0.0003, the system will be able to handle malicious peers effectively during gossip communication.

7.4 P2P File Sharing using GossipTrust

We conducted extensive simulation experiments to measure the performance of using GossipTrust in P2P file-sharing applications. We choose the same file sharing model used by Stanford EigenTrust group [20]. Over 100,000 files are simulated. This experiment proves the effectiveness of using reputation information to boot up query success rate in unstructured P2P systems.

The number of copies of each file is characterized by a Power-law distribution. Each peer is assigned with a number of files based on the Sarioiu distribution [20]. At each step, a query is randomly generated by a peer and completely executed before the next step. The query popularity reflects which file to request.

We rank the queries according to their popularity. We use a power law distribution on queries of all ranks. This distribution models the query popularity distribution of Gnutella network. After a query for a file is flooded over the entire P2P network, a list of nodes having this file is generated and the one with the highest global score is selected to download the file. The system updates global reputation scores after 1,000 queries.

The *query success rate* is the percentage of successful queries over total number of queries issued. Each node is modeled to submit a query with a rate inversely proportional to the nodal global reputation. We also consider a NoTrust system randomly selecting a node to download the file without considering node reputation. We plot the result of using GossipTrust and NoTrust in simulated P2P file sharing experiments in Fig.9.

The malicious peers issue unreliable scores and provide corrupted files. The performance of GossipTrust drops only slightly with increasing malicious peers, while performance of NoTrust drops sharply with more malicious peers. With the help of GossipTrust, even when the system has 20% malicious peers, it can still maintain around 80% query success rate.

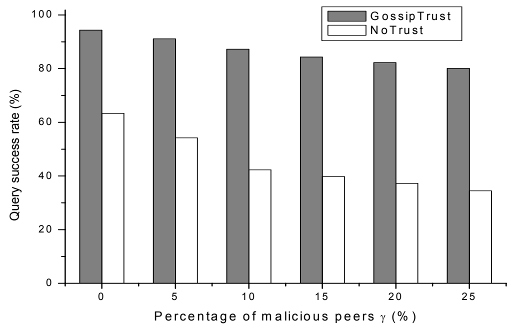


Figure 9 Query success rate of a 1000-node GossipTrust system, compared with a no-trust system

7.5 P2P Grid PSA Benchmark Results

Unstructured P2P Grid computing supports distributed execution of parallel jobs in the PSA benchmark on a P2P computational grid [12], [39], [40]. Denote the total number of parallel jobs as M. We use two metrics to evaluate the GossipTrust performance in P2P Grid job execution over the PSA workload [27].

The *average job turnaround time* is defined by averaging all job completion time 3_i $\{c_i\}/n$, where c_i is the *completion time* of the i-th job for $i = 1, 2, \ldots M$. The *average job success rate* is defined by the ratio $F_{rate} = (1\text{-}M_{fail})/M$, where M_{fail} accounts the number of failed jobs. Figure 10 shows the performance results of our GossipTrust reputation system over the PSA workload, compared with a system with the no-trust support.

We apply a realistic PSA workload of 20,000 to 100,000 jobs in Fig.10(a) over a P2P Grid, consisting of 4,000 resource sites. We assumed an average job execution time 5 sec/job and an average 2 jobs/sec arrival rate. The PSA benchmark runs independent jobs.

The execution model essentially involves parallel execution of *M* independent jobs on *N*, where $M >> N$. Per each job, 10% of the peer sites will respond to the job assignment. The peer sites having the shortest *expected time-to-completion* (ETC) are selected for executing the job.

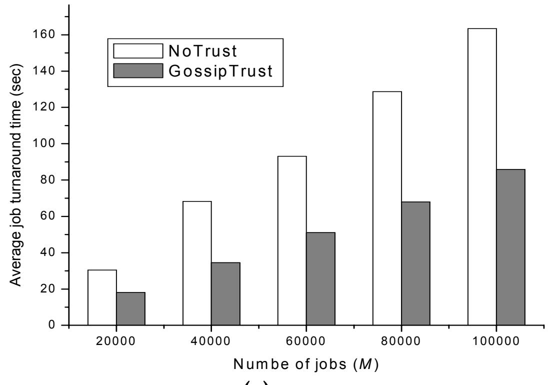
GossipTrust computes ETC = *real_etc*/(1- *fail_rate*), where the *real_etc* is the actual ETC of a peer site and the fail_rate is the failing rate experienced with the Grid site, which is determined by the site's global reputation. After each job execution, the Grid site will update the local trust scores of other sites according to job execution result.

GossipTrust updates the global scores for all sites every hour. A job is executed if it was not rejected for 3 times. The NoTrust in white bars corresponds to the worst case that the peer site reputations are not used in job scheduling. The grey bars correspond to job scheduling using the reputation scores by GossipTrust.

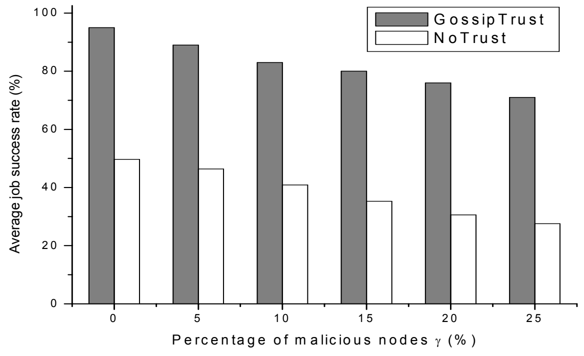
Figure 10 reports the PSA simulation results involving massive parallelism for executing independent jobs. The *average job turnaround time* plotted in Fig.10*(a)* and the *average job success* rate in Fig.10(b) offer two good indicators whether the jobs are assigned properly for parallel execution over all the nodes.

The good peer nodes have higher job completion rate and contribute to lower execution time. The bad nodes may fail the job easily and thus prolong the job execution time. The GossipTrust system is used here to provide peer reputations based on evaluating their past job execution results. We check in Fig.10(a) the effects of total PSA job number (*M* in Table 4) on the average job turnaround time. Figure 10(b) shows the effects on the average job success rate by malicious peers.

The shaded bars are resulted from using GossipTrust to aid the job scheduling process. The white bars (labeled as No Trust) are resulted from no peer reputation information in job scheduling. In fact, the two metrics are dependent of each other. They grow inversely proportional to each other. This can be seen by comparing the corresponding bar heights in Fig.10(a, b).



(a)



(b)

Figure 10. Average job turnaround time in PSA benchmark tests on a simulated P2P Grid. (a) Effect of job number. (b) Effect of malicious peer density.

Figure 10(a) shows that the *average job turnaround time* in PSA experiments grows linearly in both GossipTrust (gray bars) and NoTrust (white bars) systems. For 20,000 jobs, the gray shaded bar is about 30% higher than the white bar. For 1000,000 jobs, the GossipTrust outperforms the NoTrust system by 50%.

We plot in Fig.10(b) the *average job success rate* against different percentage of malicious peers under the default value of having M = 40,000 jobs. We experimented 10 peers in a collusion group. Without trust information, the job succeeds 48% of the time. GossipTrust has 40% performance gains over the NoTrust system under 10% malicious peers. Even when the malicious peers increase to 25%, the GossipTrust has a 75% performance gain. These results prove the effectiveness of using global reputation to support trusted P2P Grid computing.

8 CONCLUSIONS AND FURTHER WORK

In any P2P network, global reputation aggregation is quite expensive when the network grows to reach millions of nodes. To our best knowledge, GossipTrust offers the very first attempt to extend the gossip protocol for reputation aggregation in P2P networks without any structured overlay support. We summarize below the major contributions and discuss some open research issues and further efforts needed.

8.1 Summary of Contributions

GossipTrust is shown very fast and accurate in aggregating local trust scores into global reputation ranks. The technical innovations of this reputation system are summarized in four aspects:

(1) ***Fast gossip-based reputation aggregation algorithms with small aggregation error*** **:** The O($\log_2 n$) time steps make the gossip search equally attractive as in DHT-based table lookup. The gossip and aggregation errors are maintained rather low as proven in Theorems 1 and 2 and numerically verified in Table 5 with at most 0.43 % error for 10,000 nodes.

(2) ***Efficient reputation storage with Bloom filters with low false-positive error :*** Even for a network of one million nodes, the ranking memory in Bloom filters is limited to 512 KB per node. The false-positive error in using the Bloom filters is very small as proven in Theorem 3 and reported in Table 3 with at most 15% for up to one million nodes.

(3) ***Limited network traffic overhead in gossip message spreading:*** The total network traffic increase with O($n\log_2 n$) is considerably low, compared to multicast or broadcast approaches. As numerically reported in Table 5, the messaging overhead is limited to 1.7× 10^5 messages for a network of 10,000 nodes.

(4) ***Combating against peer collusions by using the power nodes dynamically:*** Like the PowerTrust system [40], the GossipTrust system also leverages the ranking of all nodes in terms of their relative standing in the global reputation vector. The effects of peer collusion are reduced to the minimum due their low standing in the ranking order.

GossipTrust enables peers to compute global reputation scores in a fully distributed, secure, scalable and robust fashion. The simulation results show that the system scales well with the increase of network size. The system can also tolerate link failures and peer collusions.

The benchmark experiments on P2P file-sharing applications and P2P Grid PSA workloads demonstrate significant performance gains in using GossipTrust system, compared with an unstructured P2P networks without any reputation services.

The benchmark experiments result in performance gains in 6 metrics : (i) reduced number of aggregation cycles, (ii) fast convergence rate, (iii) low RMS aggregation errors, (iv) high query success rate in distributed file sharing, (v) improved job success rate, and (vi) shortened turnaround time in PSA applications.

8.2 Open Issues and Further Research

Both gossip protocol and Power nodes play a crucial role in speeding up the reputation aggregation process. Randomized gossiping can reach the consensus among all nodes in a fully distributed manner. This approach results in massive concurrency being exploited in gossip-based communications among millions of active nodes in a very large P2P network.

We have to point out that the GossipTrust system is not restricted to apply only in unstructured P2P systems. With minor modifications, this reputation system performs even better in a structured P2P system. The gossip steps and reputation aggregation process reported here can be further accelerated by fast hashing and search mechanisms built in DHT-based overlay networks. That is considered a good extension of this work.

A peer providing corrupted services is highly likely to issue dishonest reputation scores. To probe further, we suggest to keep two kinds of reputation scores on each peer node: one to measure the *quality-of-service* (QoS) performance measures reported here and another for *quality-of-feedback* (QoF) by participating peers. It is true that good service providers are not always good feedback providers. In general, weighting the two factors offers a possible approach to solving the problem.

In fact the QoF is already reflected in the global score ranking. How to analyze [18] and filter out unfair feedbacks [34] posts a good further research issue. We suggest integrating these two scores together and address the tradeoffs between them in future research challenges. With file replication, P2P systems are vulnerable to content pollution and attacks using replicated decoys or index poisoning [18].

In eBay, a user issues either a positive or a negative score after each transaction. As the global reputation score is accumulative by adding up all local scores in history. The trust matrix we use is used by many reputation systems. Xiong and Liu suggested using a personalized trust model [35], which deserves more work by the research community. There are suggestions to use a personalized trust model [35]. This approach deserves more work by the cybertrust research community.

Another concern is that vulnerability in peer scoring can create more problems [29]. This has something more to do with the problem of peer collusions. There are suggestions to filtering out unfair ratings in Bayesian reputation systems [34] and analysis of ratings on trust inference in open environments [18]. Both are good topics worthy of future research challenges.

Further research is also encouraged to apply reputations systems to enforce copyright protection in P2P systems [19]. The file-poisoning protection could be aided by reputation ranking. With the help of object reputation [32], a client can validate the authenticity of an object before initiating parallel file download from multiple peers. This opens up a meaningful direction to extend gossip-based trust management at the object-or file-level or using role-based credential chains [6].

Acknowledgements: This work was fully supported by NSF ITR Grant ACI-0325409, performed at the Internet and Grid Research Laboratory, University of Southern California.

REFERENCE:

[1] K. Aberer and Z. Despotovic, "Managing Trust in a Peer-2-Peer Information System", *Tenth Int'l Conf. on Information and Knowledge Management*, New York, 2001.

[2] S. Boyd, A. Ghosh, B. Prabhakar, D. Shah, "Randomized Gossip Algorithms", *IEEE Trans. on Information Theory*, June 2006, 52(6):2508-2530.

[3] B. H. Bloom, "Space/Time Trade-offs in Hash coding with Allowable Errors", *Comm. of the ACM*, vol.13, no.7, pp. 422-426, 1970.

[4] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A survey", *Conf. on Comm., Control, and Computing*, 2002.

[5] S. Buchegger and J. Y. Boudec, "A Robust Reputation System for P2P and Mobile Ad-hoc Networks", *Second Workshop on Economics of P2P Systems*, Boston, June 2004.

[6] K. Chen, K. Hwang, and G. Chen, "Heuristic Discovery of Role-based Trust Chains in Peer-to-Peer Networks", *IEEE Trans. on Parallel and Distributed Systems, submitted Se*pt. 2007and revised Jan. 2008.

[7] N. Christin, A.S.Weigend, and J. Chuang, "Content Availability, Pollution and Poisoning in File Sharing Peer-to-Peer Networks", *ACM Conf. on E-Commerce*, Vancouver, June 2005.

[8] E. Damiani, S. Vimercati, S. Paraboschi, P. Samarati, "Managing and Sharing Servants' Reputations in P2P system", *IEEE Trans. on Knowledge and Data Engineering*, Vol. 15, Issue 4, July 2003.

[9] D. Dumitiu, E. Knightly, A. Kuzmanovic, I. Stoica, and W. Zwaenepoel, "Denial-of-Service Resilience in P2P File Sharing Systems", *ACM/IEEE Sigmetrics'05*, Alberta, June 2005

[10] C. Gkantsidis, M. Mihail, and A. Saberi, "Conductance and Congestion in Power Law Graphs", *ACM/IEEE Sifmetrics*, San Diego, June. 2003.

[11] K.Gummadi, R.Dunn, R. Dunn, "Measurement, Modeling and Analysis of a Peer-to-Peer File-Sharing Workload", *ACM Symp.on Operating Systems Principles*, Bolton Landing, NY, 2003.

[12] J. Hu and R. Klefstad, "Decentralized Load Balancing on Unstructured Peer-2-Peer Computing Grids", *Fifth IEEE Int'l Symp. on Network Computing and Applications* (NCA'06), Boston, July, 2006

[13] D. Hughes, G. Coulson, and J. Walkerdine, "Free Riding on Gnutella Revisited: The Bell Tolls", *IEEE Distributed Systems Online*, Volume 6, June 2005.

[14] M. Jelasity, A. Montresor and O.Babaoglu, "Gossip-Based Aggregation in Large Dynamic Networks", *ACM Trans. on Computer Systems*, Vol.23, No.3, August 2005.

[15] V. Kalogeraki, A. Delis and D. Gunopulos, "Peer-to-Peer Architectures for Scalable, Efficient and Reliable Media Services", *Int'l Parallel & Distributed Processing Symposium* (IPDPS-2003), France, April 2003.

[16] S. Kamvar, M. Schlosser, and H. Garcia-Molina, "The Eigentrust Algorithm for Reputation Management in P2P Networks", *ACM WWW'03*, Budapest, Hungary, May 2003.

[17] D. Kempe, A. Dobra and J. Gehrke, "Gossip-based Computation of Aggregate Information", *Proc. of IEEE Symposium on Foundations of Computer Science*, Cambridge, MA, Oct.2003.

[18] Z. Liang and W. Shi, "Analysis of Recommendations on Trust Inference in Open Environment", *Journal of Performance Evaluation*, Elsevier 2007.

[19] X. Lou and K. Hwang, "Collusive Piracy Prevention in P2P Content Delivery Networks", *IEEE Trans. on Computers*, submitted Sept.27, 2007 and revised Dec. 2007.

[20] S. Marti and H. Garcia-Molina, "Limited Reputation Sharing in P2P Systems", *Proc. of the 5th ACM Conference on Electronic Commerce*, New York, May 2004.

[21] J. Meserve, "P2P Traffic Still Dominates the 'Net", *Network World*, 2005.

[22] R. L. Page, S.Brin and T. Winograd, "the Pagerank Citation Ranking: Bringing Order to the Web", *Technical Report*, Stanford Digital Library Technologies Project, 1998.

[23] S. Nandy, L. Carter and J. Ferrante, "GUARD: Gossip Used for Autonomous Resource Detection", *19th Int'l Parallel &Distributed*

Processing Symposium, Colorado, Apr. 2005.

[24] G. Pallis and A. Vakali, "Insight and Perspectives for Content Delivery Networks," *Comm. of The ACM,* Jan. 2006, pp.101-106.

[25] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks", *ACM/IEEE Sigcomm 2004*, Portland, USA, Aug-Sep, 2004.

[26] P. Resnick, R. Zeckhauser, E. Friedman, and K.Kuwabra, "Reputation Systems", *Communications of the ACM*, 43(12), pp.45-48, 2000.

[27] A. Singh and L. Liu, "TrustMe: Anonymous Management of Trust Relationships in Decentralized P2P Systems", *IEEE Intl. Conf. on Peer-to-Peer Computing,* Sep. 2003.

[28] S. Song, K. Hwang, and Y.K. Kwok, "Risk-Resilient Heuristics and Genetic Algorithms for Security-Assured Grid Job Scheduling", IEEE *Trans. on Computers,* August 2006.

[29] M. Srivatsa, L. Xiong, and L. Liu, "Trustguard: Countering Vulnerabilities in Reputation Management for Decentralized Overlay networks", *Proc. of the 14th International World Wide Web Conference,* pages 422–431, 2005.

[30] T. Stading, "Secure Communication in a Distributed System Using Identity Based Encryption", *Proceedings of 3rd IEEE Int'l Symp. on Cluster Computing and the Grid* (CCGrid), May 2003.

[31] E. Sit and R. Morris, "Security Considerations for P2P Distributed Hash Tables", *Proc. IPTPS 2002,* Cambridge, MA, March 2003.

[32] S. Song, K. Hwang, R Zhou, and Y. K. Kwok, "Trusted P2P Transactions with Fuzzy Reputation Aggregation", *IEEE Internet Computing,* Nov/Dec. 2005, pp.18-28

[33] K. Walsh and E. Sirer, "Experience with an Object Reputation System for Peer-to-Peer File-sharing", *NSDI' Symp.on Networked Systems Design & Implementation*, San Jose, May 8-10, 2006

[34] A. Whitby, A. Josang and J. Indulska, "Filtering Out Unfair Ratings in Bayesian Reputation Systems", *Workshop on Trust in Agent Societies*, New York, July 2004.

[35] L. Xiong and L. Liu, "PeerTrust: Supporting Reputation-based Trust for Peer-to-Peer Electronic Communities", *IEEE Trans. Knowledge and Data Engineering,* Vol.16, No.7, 2004, pp. 843-857.

[36] L. Xiao, Y. Liu and L. M. Ni, "Improving Unstructured Peer-to-Peer Systems by Adaptive Connection Establishment," *IEEE Trans. on Computers,* Sept. 2005, pp. 1091-1103.

[37] B. Yang, T. Condie, S. Kamvar and H. Garcia-Molina, "Non-Cooperation in Competitive P2P Networks", Proc. of the 25th *IEEE Int'l Conf.on Distributed Computing Systems,* Columbus, Ohio, 2005

[38] M. Yang, Z. Zhang, X. Li and Y. Dai, "An Empirical Study of Free-Riding Behavior in the Maze P2P File-Sharing System", *Proceedings of IPTPS*, Ithaca, NY. Feb, 2005

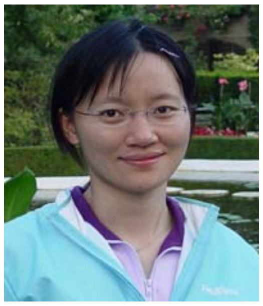
[39] H. Zhang, A. Goel and R. Govindan, "Making Eigenvector-based Reputation Systems Robust to Collusion", *Third Workshop on Economic Issues in P2P Systems,* Berkeley, June 2003.

[40] R. Zhou and K. Hwang, "PowerTrust: A Robust and Scalable Reputation System for Trusted P2P Computing", *IEEE Trans. on Parallel and Distributed Systems,* April 2007, pp.460-473.

[41] R. Zhou and K. Hwang, "Gossip-based Reputation Aggregation in Unstructured P2P Networks", *IEEE Int'l Parallel and Distributed Processing Symposium,* March 2007.

[42] R. Zhou, "Scalable Reputation Systems for Peer-to-Perer Networks", *Ph.D.Thesis*, Dept. of Computer Science, Univ. of Southern California, Los Angeles, May 2007.

[43] H. Zhuge, X. Sun, J. Liu, E. Yao and X. Chen, "A Scalable P2P Platform for the Knowledge Grid", *IEEE Trans. Knowledge and Data Engineering,* Dec. 2005, pp. 1721- 1736.



Runfang Zhou received the B.S. and M.S. in computer science from Southeast University, China. She earned the Ph.D. in computer science at the Univ. of Southern California in May 2007. She is presently with Microsoft Corp., Mountain View, CA. Her research covers peer-to-peer reputation systems, overlay networks, web services performance, and trusted Grid computing. Contact her at Email: ruzhou@microsoft.com

Kai Hwang is a Professor of Electrical Engineering and Computer Science and Director of Internet and Grid Research Laboratory at USC. He received the Ph.D. degree from the Univ. of California, Berkeley in 1972. An IEEE Fellow, he specializes in computer architecture, parallel processing, Internet security, P2P, Grid, and distributed computing.

Dr. Hwang is the founding Editor-in-Chief of the *Journal of Parallel and Distributed Computing* published by Elsevier. He is also on the editorial boards of IEEE *Transactions on Parallel and Distributed Systems*. He has published over 200 papers and 7 books. His research group at USC have developed security-binding techniques, P2P reputation systems, distributed defense systems against network worms and DDoS attacks, P2P, Grid, and Internet applications. Contact him at Email: kaihwang@usc.edu or URL: //GridSec.usc.edu/Hwang.html

Min Cai received BS and MS in Computer Science from Southeast University, China in 1998 and 2001, respectively. In December 2006, he received the Ph.D. degree in Computer Science at the Univ. of Southern California. He is a member of technical staff with the VMware, Inc., Palo Alto, CA. His research interests include P2P and grid computing, network security, and web services. Contact him by Email: mcai@umware.com