Gossip-Based Reputation Management for Unstructured Peer-to-Peer Networks*

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

Abstract: To build an efficient reputation system for *peer-to-peer* (P2P) networks, we need fast mechanisms to aggregate peer evaluations and to disseminate updated scores to a large number of peer nodes. Unfortunately, unstructured P2P networks are short of secure hashing and fast lookup mechanisms as in structured P2P systems like the DHT-based Chord. In light of this shortcoming, we propose a gossiping mechanism to support unstructured P2P communications. This paper reports the design principles and performance results of a *GossipTrust system.* Gossip-based communications are shown effective to support fast peer scoring and distribution. Gossiping enables light-weight aggregation of local scores and spreading of updated scores to all nodes in $O(\log_2 n)$ time, where n is the P2P network size.

The GossipTrust works as follows: Each peer gossips with a few randomly selected other nodes to share reputation data, periodically. Gossip-based protocols do not require error recovery and thus enjoy the simplicity with a moderate processing overhead. Bloom filters are specially tailored for storing and retrieving global reputation rankings on each node. We apply identity-based cryptography to secure gossiped reputation data. We report simulation results on distributed file-sharing and parameter-sweeping applications on GossipTrust. These results prove the claimed advantages in low aggregation overhead, storage efficiency, and scoring accuracy in unstructured P2P networks. With minor modifications, the system is also applicable to structured P2P systems with projected even better performance, which is beyond the scope of this paper.

Index Terms: Peer-to-peer networks, gossip protocol, reputation system, information aggregation, identity-based cryptography, Bloom filter, distributed file sharing, parameter sweeping applications, and system performance evaluation.

---

* Manuscript submitted to *IEEE-Transactions on Knowledge and Data Engineering.,* Jan. 3, 2007. This work was supported by NSF Grant ITR-0325409 at the University of Southern California. The initial concept and some preliminary results will be presented in *IEEE IPDPS-2007* in March 2007. This journal version is significantly extended in both theories and experimental results, with more than 70% new material compared with the conference version. Corresponding author is Kai Hwang at kaihwang@usc.edu, Tel: 213 740 4470, and Fax: 213 740 4418.

1. Introduction

The growth of *peer-to-peer* (P2P) traffic is largely attributed to the applications in P2P file-sharing [23], digital content delivery [20], and P2P Grid services [12]. Peer anonymity and dynamic joining/departure make P2P networks quite vulnerable to attacks by selfish and malicious peers [7][33][9]. Most P2P file-sharing networks like Gnutella, KaZaA and BitTorrent are built with autonomous peers with different or even conflicting self-interests [13]. Selfish peers tend to take advantage of other peers without contributing their own resources. Malicious peers attack the network by spreading viruses or corrupted software, pirated music, or pornographic video files [18]. Reputation systems are designed to cope with these difficulties and help legitimate clients choose reliable peers in P2P transactions [1].

In the past, several reputation systems like EigenTrust [16], PeerTrust [32], and PowerTrust [36] were proposed to evaluate peer behavior in structured P2P networks like Chord or CAN. Reputation management in P2P networks support resource sharing among legitimate peers and combate selfish or malicious ones. A good on-line reputation system enables peers to assess each other to establish trusted interactions. Without reputation management, peers may hesitate to interact with unknown peers due to the concern of receiving poisoned files or being compromised by malwares. Furthermore, identifying trustworthy peers is required in P2P auctions, trusted content delivery, pay-per-transaction, and P2P service discovery [15].

For example, after a peer completes downloading a music file, the peer rates the provider based on his or her experience in the transaction. The reputation system [24] computes the global reputation score of a peer by aggregating the local rates (i.e. feedbacks) from those who had interacted with that peer, previously. 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 the following requirements in *aggregation overhead*, *storage efficiency*, and *reputation accuracy*. First, the overheads in computation and communication for global reputation aggregation have to be considerably low. Second, the amount of space for each node to store previously collected reputation data has to be reduced as small as possible, preferably a constant number of bytes allocated. Third, the calculation of global reputation scores must be sufficiently accurate. Our

goal is to achieve an error probability within a small window $[(1-\varepsilon)v, (1+\varepsilon)v]$, where v is the actual reputation score and ε is the small error fraction. The high accuracy is achieved with significant increase in aggregation time and storage overhead. The tradeoffs among these three requirements are explored through proper choice of various design parameters.

To meet these requirements, we propose a scalable, robust and secure reputation management system called *GossipTrust*, specifically designed for unstructured P2P networks. This system leverages a gossip-based protocol to aggregate global reputation scores. In GossipTrust, each peer randomly contacts others and exchanges reputation data periodically. Gossip-based protocols do not require any error recovery mechanism, and thus enjoy the simplicity with moderate overhead, compared with optimal deterministic protocols [17].

GossipTrust is built around a fast reputation aggregation module with enhanced security support. We aim to strengthen the robustness of the gossip protocol under disturbances by malicious peers. The system has a novel data management scheme to answer reputation queries and to store reputation data with small overhead. We apply identity-based cryptography to ensure the confidentiality, integrity and authenticity of the exchanged reputation data, without using certified public keys or pre-shared secret.

The remaining parts of this paper are 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 the performance of GossipTrust in Section 5. We propose a Bloom filter-based storage scheme for storing and retrieving reputation data in Section 6. The security enhancement mechanisms are presented in Section 7 to combat malicious peers. We report simulation results of GossipTrust and P2P benchmark results in Section 8. Finally, we conclude with a summary of contributions and make suggestions for further research at the end.

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 [6], [16], [19], [25], [30], [32], [36]. Two important tasks are performed in this process, i.e. reputation score aggregation and reputation data dissemination.

These two tasks involve the scoring, collection, storage, and distribution of peer reputation scores, that are dynamically changing. Most existing reputation systems apply *distributed hash table* (DHT) [16], [32], [36] to support trust management tasks.

How to perform efficient reputation aggregation and distribution thus becomes a major challenge in unstructured P2P networks. Several reputation systems have been proposed to discourage malicious peers and to promote the trustworthiness and cooperation among legitimate peers. Xiong and Liu [32] presented PeerTrust, which computes the peer reputation based on five contributing factors. Srivatsa [27] took a further step to improve the robustness of PeerTrust. They proposed methods to especially counter vulnerabilities in reputation management. The EigenTrust algorithm [16] aggregates trust information by having peers perform a distributed calculation of the eigenvector of the trust matrix.

We have proposed PowerTrust [36] for DHT-based networks, which leverages the power law distribution of peer feedbacks. Trust management in unstructured P2P networks was pioneered by Aberer and Despotovic [1] in 2001. Their approach is based on a decentralized storage method (P-Grid). The trust information provided by P-Grid is used to assess the probability that an agent will cheat in the future. This approach suffers from the fact that trust is evaluated by referrals, not by global trust aggregation.

Another approach uses Bayesian learning [6], by which the first-hand information is exchanged frequently and second-hand information is merged with current reputation ratings. TrustMe [25] uses a random assignment of reputation-holding peers and employs smart public key mechanisms to prevent the loss of anonymity. This method imposes a lot of messages in the network. When network size becomes very large, it will experience long overhead to disseminate the peer reports and to obtain a peer's global reputation. In [19], peer reputation scores are aggregated by limited local peers.

The idea of gossiping in cyberspace communication is very similar to gossiping to reach consensus in the human society. Gossip protocols were proposed for randomized communication [2] and for aggregation of large amounts of distributed information [17]. These 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.

They have been widely used in various network applications as membership management, information dissemination and resource discovery [22].

We have first presented the gossiping idea and preliminary results on GossipTrust in an *IPDPS-2007* paper [37]. The idea of using Bloom filtering for trust information management was inspired by the work reported in [3], [5]. This paper is significantly extended from the conference presentation. We report the gossiping principles for P2P reputation aggregation and dissemination, the GossipTrust architecture, Bloom filters for building the reputation database, and performance results through simulated P2P experiments. To benefit our readers, we summarize in Table 1 the notations, basic definitions and indexing conventions used. All time indices represent discrete time instants. They will be defined further in subsequent sections.

Table 1 Notations and Conventions used in this Paper

Notation	Definition	Notation	Definition
n	P2P network size (n peer nodes)	α	Greedy factor of peer random walking
s_{ij}	Local trust score of peer j by peer i	θ	Peer selection threshold
$V(t) = \{v_i(t)\}$	Global trust vector over n peers	ε	Gossiping error threshold
$k \le g$	Gossiping step and upper bound	δ	Aggregation error threshold
$t \le d$	Aggregation cycle and upper bound	γ	Percentage of malicious peers in system
$x_i(k)$	Weighted score by peer i at step k	$x_i(k)$	Consensus factor by peer i at step k
$\beta_i(k)$	Gossiped score at peer i at step k	p	Number of power nodes in system
m	Bloom filter array size (bit count)	b_i	Category boundary at Bloom filter
E	RMS global aggregation error	c	Number of score ranking categories

3. GossipTrust System Architecture

In this section, we formulate the reputation aggregation problem in P2P systems. 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.

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 from node i to j, r_{ij} is set to 0. For global reputation aggregation, each node must normalize all local scores issued by itself. The *normalized local score* s_{ij} is defined as follows:

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

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 up to be 1.

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:

$$V(t+1) = S^T \times V(t) \quad \textbf{(2)}$$

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 [36] 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 often predefined by system designers. In other words, 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 the current distribution of local trust scores. The stationary distribution of the Markov chain is the converged global reputation vector.

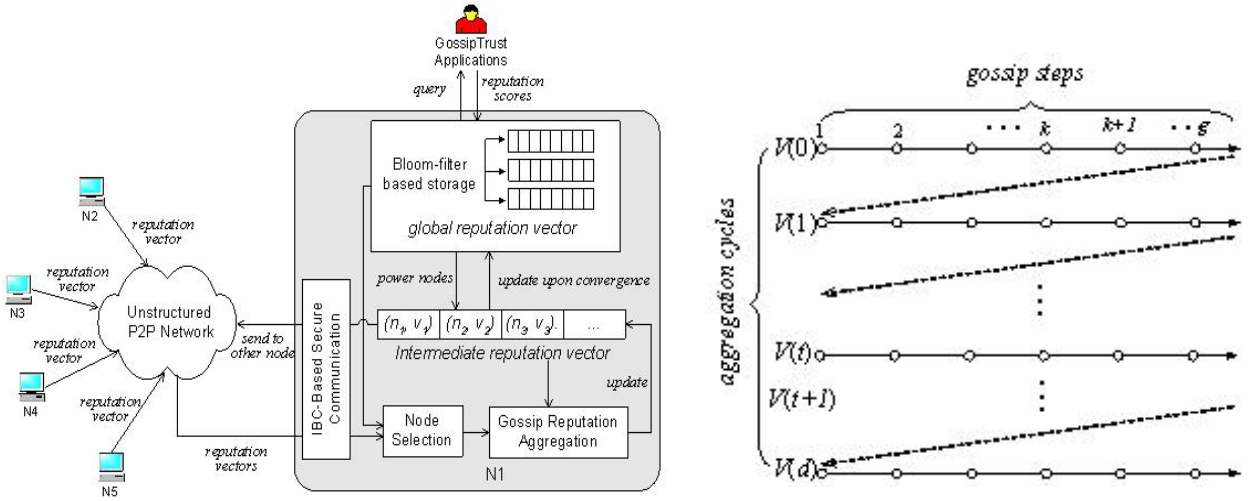
Both EigenTrust and PowerTrust have developed scalable algorithms to calculate the global reputation vector $V(t)$ for DHT-based P2P networks. In this paper, we propose a GossipTrust system for global reputation aggregation in unstructured P2P networks. The new scheme is fully

distributed without using any topological structure among the nodes. It is also proven fast in convergence rate and secure in dynamic peer participations.

3.2 The GossipTrust Architecture

We have developed a simulated *GossipTrust* reputation system at USC Internet and Grid Computing Laboratory. Figure 1 shows the architecture of the GossipTrust system running on a typical node N_i, where i=1,2,…,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.



(a) GossipTrust system architecture **(b) Global reputation aggregation | through iterative gossip steps**

Figure 1. The GossipTrust architecture and global reputation aggregation process

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. Figure 1(b) illustrates the process of reputation aggregation in GossipTrust.

To reduce the storage complexity on each node, GossipTrust uses a novel Bloom filter based scheme for the storage and retrieval of global reputation scores. A Bloom filter is a space-efficient randomized data structure for representing a set to support membership queries [3]. This storage scheme discretizes all global reputation scores into categories and summarizes the set of nodes in each category with Bloom filters. The converged global reputation vector is updated and properly stored in the Bloom filters with a small storage overhead. So each node can easily identify the *power nodes,* which have the highest reputation scores in the system.

Section 7.2 shows how power nodes are used in GossipTrust to combat peer collusions. The applications of GossipTrust can also query the local storage to retrieve the reputation score of a given node. In addition, all communications among GossipTrust nodes are encrypted with *identity based cryptography* (IBC) [4], [28] for better security and reliability of exchanged reputation data. By using publicly identifiable information (e.g. node id) as the public key, the IBC scheme does not require any centralized CA to verify the binding between a node and its public key, which is well suited for decentralized P2P networks.

4. Gossip Protocol for 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 single node in Algorithm 1 and for aggregating the global reputation scores for all nodes in Algorithm 2, concurrently.

4.1 Gossip-based Aggregation Protocol

Gossiping is done iteratively in a small number steps. We reserve the index k to indicate the gossip step. According to Kempe, et al [17], k is upper by a final step $g = O(\log_2 n)$. We use index t to refer to discrete times for aggregation cycles. The upper bound for t is d iterations specified in section 3.1. Associated with each peer node i is a *gossip pair* $\{x_i(k), w_i(k)\}$ at each gossip step k.

At time t, we have the *weighted score* $x_i(t) = s_{ij} \times v_i(t)$ as the local score s_{ij} weighted by the global score $v_i(t)$ of node i. The $w_i(k)$ is called the *consensus factor* of node i at step k. During

each gossip step, every node i executes two computing threads: One thread sends the halved gossip pair $\{½\, x_i(k), ½\, w_i(k)\}$ to itself (node i) and to a randomly selected node in the network. Another thread receives the halved pairs from other nodes and computes the updated $x_i(k+1)$ and $w_i(k+1)$ as follows, where r refers the index of remote nodes that have sent the halved gossip pairs in step k:

$$x_i(k+1) = \sum_r ½\, x_r(k) \quad \textbf{(3)}$$

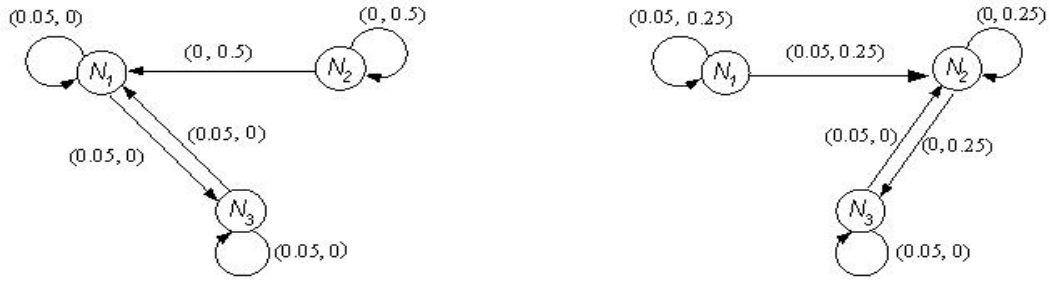
$$w_i(k+1) = \sum_r ½\, w_r(k) \quad \textbf{(4)}$$

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

$$v_i(t+1) = x_i(g)/w_i(g) = \beta_i(g) \quad \textbf{(5)}$$

The above gossiping process is best illustrated by a small example in Fig.2. Consider a P2P network with three nodes. At time t, the global scores are given: $v_1(t) = 1/2$, $v_2(t) = 1/3$, and $v_3(t)=1/6$. Given also normalized local score $s_{12} = 0.2$, $s_{12} = 0$, and $s_{32} = 0.6$. By Eq.(2), the updated global score of node *N2* is calculated as:

$$v_2(t+1) = v_1(t)\times 0.2 + v_2(t)\times 0 + v_3(t)\times 0.6 = (1/2)\times 0.2 + (1/6)\times 0.6 = 0.2 \quad \textbf{(6)}$$



(a) First gossip step (k = 1) **(b)Second gossip step (k = 2)**

Figure 2. Halved score sharing among 3 nodes in two gossiping steps to aggregate the global scores on all nodes, concurrently

We use Table 2 to illustrate the gossiping procedure in Fig.2. The end purpose is to generate the global score $v_2(t+1) = 0.2$ at all 3 nodes in 2 steps. In general, gossip protocols are used to calculate any aggregate function such as *sum, maximum,* or *average* of the numeric values distributed over many nodes [17]. Here, we concentrate on the gossiped calculation of the global score of node *N2*. Initially at step 0, we thus assume $w_2(0) = 1$ and $w_1(0) = w_3(0) = 0$. The initial weighted scores $x_1(0) = (1/2)\times 0.2 = 0.1$, $x_2(0) = (1/3)\times 0 = 0$, and $x_3(0) = (1/6)\times 0.6 = 0.1$.

Table 2 Gossiped Scores Aggregated Concurrently on All Nodes in Fig.2

Gossip Step, k	Node $N1$			Node $N2$			Node $N3$		
	$x_1(k)$	$w_1(k)$	$\beta_1(k) = x_1(k)/w_1(k)$	$x_2(k)$	$w_2(k)$	$\beta_2(k) = x_2(k)/w_2(k)$	$x_3(k)$	$w_3(k)$	$\beta_3(k) = x_3(k)/w_3(k)$
1	0.1	0.5	0.2	0	0.5	0	0.1	0	∞
2	0.05	0.25	**0.2**	0.1	0.5	**0.2**	0.05	0.25	**0.2**

At the first gossip step, as shown in Fig.2(a), *N1* sends the pair $(½x_1(0), ½w_1(0)) = (0.05, 0)$ to *N1* and to a randomly chosen node *N3* The node *N2* sends the pair (0, 0.5) to *N2* and a random node *N1*. Node *N3* sends the pair (0.05, 0) to *N3* and a random node *N1*. Then *N1* updates $x_1(1) = 0+0.05+0.05=0.1$ and updates $w_1(1) = 0+0.5+0 =0.5$. After first gossip step, *N1* has the pair $\{x_1(1), w_1(1)\}= (0.1, 0.5)$. The gossiped score $x_1/w_1 = 0.2$ on *N1*. Similarly, nodes *N2* and *N3* go through the same gossiping process to produce $x_2/w_2 =0$ and $x_3/w_3 = \infty$.

Figure 2(b) illustrates the halved score sharing by the same gossiping process in step 2. After step 2, we have reached the consensus that $x_1/w_1 = x_2/w_2 = x_3/w_3 = 0.2$. Thus, we accept the updated global score for node *N2* as $v_2(t+1) = x_1/w_1 = x_2/w_2 = x_3/w_3 = 0.2$, which agree with the dot product calculation in Eq.(6). Thus, gossiped scores are equalized in all 3 nodes at the end of gossiping process. Suppose we extend the gossiping process to another step, we will see no more changes in the consensus values of $x_1/w_1 = x_2/w_2 = x_3/w_3 = 0.2$. Lacking centralized control in the system, the consensus must be determined on distributed nodes locally. This additional step is thus performed to determine the consensus in a distributed manner.

4.2 Distributed Reputation Aggregation Algorithm

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 s_{ij} for each peer $j= 1, 2, ...,n$ in Algorithm 1, where the normalized global scores $\{v_i(t-1)\}$ 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 $j = 1,2,\ldots, n$.

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

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, even we specify the protocol only one node below .

Algorithm 1: Gossip Protocol to Compute Any Single Peer Score

INPUT: local score s_{ij}, global score $v_i(t\text{-}1)$ at time t-1, where $i = 1,2,\ldots,n$ and gossip threshold ε
OUTPUT: global reputation score $v_j(t)$ of node j at time t
forall $i = 1, 2, \ldots n$ **do** { simultaneously }
  $x_i \leftarrow s_{ij} \times v_i(t\text{-}1)$ { initialize weighted score x_i }
  if $(i == j)$, set $w_i \leftarrow 1$, **else** $w_i = 0$ { initialize consensus factor w_i }
  $k \leftarrow 0$ *{ initialize gossip step k}*
  repeat
    $u \leftarrow x_i/w_i$ { save previous score before convergence }
    let $\{(x_r, w_r)\}$ be all gossip pairs sent to i in previous step
    $x_i \leftarrow \sum_r x_r$, $w_i \leftarrow \sum_r w_r$ { update x_i and w_i }
    choose a random node q
    send the pair (½ x_i, ½ w_i) to node q and node i itself
    $k \leftarrow k+1$ { increase gossip step by 1}
  until $|x_i/w_i - u| \le \varepsilon$ { ε is a preset error tolerance or the gossip threshold}
endfor
output $v_j(t) \leftarrow x_i/w_i$

The gossip protocol converges to the consent global score. In GossipTrust, every node has a unique identifier and keeps a global reputation vector. The global-scale *gossip-based reputation aggregation* is specified in Algorithm 2. During each aggregation cycle t, each vector element is internally represented by a triplet (x_{id}, id, w_{id}), where id is a node identifier, x_{id} is the weighted score defined in Eq.(3) and w_{id} is the consensus factor defined in Eq.(4). We aggregate all triplets in the reputation vector, concurrently. 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 the reputation vector.

At the end of every aggregation cycle t, the gossip process converges to an equalized gossiped 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

INPUT: local trust matrix $S=(s_{ij})$ and tolerable aggregation error δ
OUTPUT: Converged global reputation vector $V(t)$
forall $i = 1, 2, \ldots n$ **do** { Concurrently on n nodes }
  $t \leftarrow 0$ {initialize the aggregation cycle}
  Repeat
    forall local score s_{ij} **do**
      { initialize weighted score and consensus factor }
      if $(t == 0)$ **then** $x_j \leftarrow s_{ij} / n$
      else $x_j \leftarrow s_{ij} \times v_i(t\text{-}1)$ where $v_i(t\text{-}1)$ is node i's reputation at time instance $t\text{-}1$
      if $(j == i)$ **then** $w_j \leftarrow 1$ **else** $w_j \leftarrow 0$
      add the triplet $<x_j, j, w_j>$ to global reputation vector $V(t)$
    endfor
    repeat
      let $\{ V_r \}$ be all received vectors from previous gossip step
      forall j in V_r **do**
        $x_j \leftarrow \sum_r x_j$, $w_j \leftarrow \sum_r w_j$,
        update the triplet $<x_j, j, w_j>$ in $V(t)$
      endforall
      choose another node q randomly
      send ½ $V(t)$ to node q and to node i itself
    until all n gossiped scores $\{ \beta_i = x_j / w_j \}$ are equalized and converged to $v_i(t)$.
    forall j=1, 2, …, n **do**
      $v_j \leftarrow x_j / w_j$ and replace every triplet $<x_j, j, w_j>$ with the updated pair $<v_j, j>$
    endfor
    $t \leftarrow t + 1$ {increase time instance by 1}
  until $|V(t) - V(t\text{-}1)| < \delta$ {Test with tolerable convergence threshold}
endfor
output The updated global scores $V(t) = \{v_1(t), v_2(t), \ldots, v_n(t)\}$

5. Analysis of Gossip Algorithms

The errors incurred with gossip communication are accumulative over the entire reputation aggregation process. In Section 5.1, we prove that the gossiped error is upper bounded by a very small value. We prove the convergence of the GossipTrust algorithm in Section 5.2. 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 Global Scores

Let $V(t) = \langle v_1(t), v_2(t), ..., v_n(t) \rangle^T$, be the reputation vector of all global scores, computed by Eq.(2) at the t-th aggregation cycle. Let $U(t) = \langle u_1(t), u_2(t), ..., u_n(t) \rangle^T$ be the gossiped reputation vector through the process specified in Algorithm 2. In other words, $v_i(t)$ and $u_i(t)$ are the actual

and estimated global scores of node i at aggregation cycle t , respectively. Let $\varepsilon_i(d) = (u_i(d)-v_i(d)) / v_i(d)$ be the *gossiped aggregation error* of node i score after d cycles, where d is final cycle of the reputation aggregation process.

***Theorem1*:**

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

$$\varepsilon_i(d) = [u_i(d) - v_i(d)] / v_i(d) \leq d\varepsilon + O(\varepsilon^2) \quad \textbf{(8)}$$

where ε is the preset gossiping error threshold and d denotes the final convergence cycle.

Proof: After the first gossip step, we have

$$V(1) = \begin{bmatrix} v_1(1) \\ v_2(1) \\ \vdots \\ v_n(1) \end{bmatrix} = S^T V^0 = \begin{bmatrix} s_{11} & s_{21} & \dots & s_{n1} \\ s_{12} & s_{22} & \dots & s_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ s_{1n} & s_{2n} & \dots & s_{nn} \end{bmatrix} \begin{bmatrix} v_1(0) \\ v_2(0) \\ \vdots \\ v_n(0) \end{bmatrix} \qquad U(1) = \begin{bmatrix} v_1(1) \pm \varepsilon_1(1) v_1(1) \\ v_2(1) \pm \varepsilon_2(1) v_2(1) \\ \vdots \\ v_n(1) \pm \varepsilon_n(1) v_n(1) \end{bmatrix}$$

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

After convergence cycle 2, we obtain

$$S^T U(1) = \begin{bmatrix} s_{11} & s_{21} & \dots & s_{n1} \\ s_{12} & s_{22} & \dots & s_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ s_{1n} & s_{2n} & \dots & s_{nn} \end{bmatrix} \begin{bmatrix} v_1(1) \pm \varepsilon_1(1) v_1(1) \\ v_2(1) \pm \varepsilon_2(1) v_2(1) \\ \vdots \\ v_n(1) \pm \varepsilon_n(1) v_n(1) \end{bmatrix} \quad \begin{matrix} s_{1i} v_1(1) + s_{2i} v_2(1) + \dots + s_{ni} v_n(1) = v_i(2) \\ -\varepsilon v_i(2) \leq s_{1i} \varepsilon_1(1) v_1(1) + s_{2i} \varepsilon_2(1) v_2(1) + \dots + s_{ni} \varepsilon_n(1) v_n(1) \leq \varepsilon v_i(2) \end{matrix}$$

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

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

Hence, after d aggregation cycles, we have $(1-\varepsilon)^d v_i(d) \leq u_i(d) \leq (1+\varepsilon)^d v_i(d)$. Since the gossiping error threshold ε is set very small, e.g. $\varepsilon < 10^{-4}$, the relative gossip aggregation error is bounded by $\varepsilon_i(d) = [u_i(d)-v_i(d)] / v_i(d) \leq (1+\varepsilon)^d - 1 = d\varepsilon + O(\varepsilon^2)$. ***Q.E.D.***

5.2 Convergence of Global Reputation 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 $U(d) = \langle u_1(d), u_2(d), \dots, u_n(d) \rangle$ at the d-th cycle as follows:

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

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 $U(d)$.

Theorem 2:

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

$$\bar{\delta} < \delta + 3d\varepsilon \quad \textbf{(10)}$$

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

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

$$(1- d\varepsilon)v_i(d) \le u_i(d) \le (1+ d\varepsilon)v_i(d) \quad \text{and}$$

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

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.***

Since both δ *and* ε are very small fractions, the above result proves that the gossiped aggregation process is guaranteed to converge with tolerable error. In reality, the peer nodes continuously join and leave a P2P network. This is commonly called a churn problem. Kempe, et al [17] have proved that gossip communication is robust against message loss and link or peer failure. This implies that GossipTrust can leverage this property. This also implies that our gossip algorithm can survive from the churn problem.

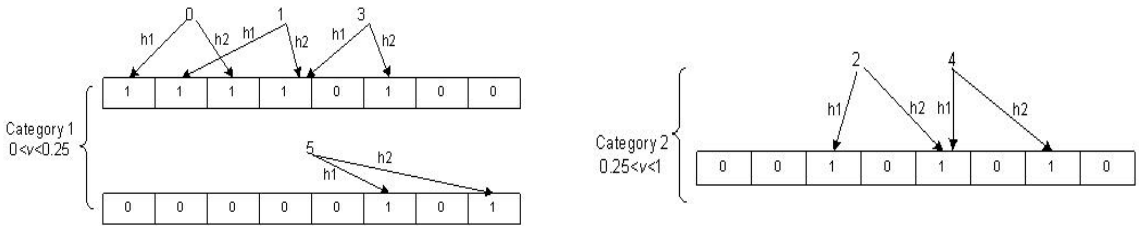
6. Reputation Storage and Retrieval with Bloom Filters

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]. We suggest storing ranked peer membership in various categories of 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. The architecture and performance of using Bloom filters are given in this section.

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 several hashed peer encodings into the same category. In this example, we divide six peers into c = 2 categories: those nodes (0, 1, 3, and 5) with scores below 0.25 forming Category 1 and those above 0.25 (nodes 2 and 4) forming Category 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 filter array. We define two hash functions: $h_1(x) = x$ Mod (8) and $h_2(x) = x$+2 to encode the peer identifiers into bit strings in the Bloom filters.



(a) Category 1 encoded by two bloom filters **(b) Category 2 encoded by one bloom filter**

Figure 3 Bloom Filter based storage scheme for a 6-node P2P network

Initially, every category has one bloom filter, and all bits in the filter array are set to 0. For example, to put node 3 to category 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 Category 1. In total, Category 1 requires 2 Bloom filters in Fig.3 (a). Similarly,

nodes 2 and 4 are encoded into one Bloom filter in Category 2 in Fig. 3(b).

Consider a user downloads files from two available peer nodes 0 and 4. The user applies the 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 Category 1. Since $h_1(4) = 4$ and $h_2(4) = 6$, in the first bloom filter in Category 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 Category 2 using the same method and find 1 in bits 4 and 6, indicating the membership of node 4 in Category 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 categorical ranks. Therefore, the storage requirement 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 categories with at most $\lceil n_i c / n \rceil$ Bloom filters per category, where n_i is the number of nodes in category i. The categories are separated by score boundaries $\{b_i\}$, where $0 \le i \le c$ and $0 = b_0 < b_1 <\ldots< b_c = 1$. Category 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. When a filter is in full state, additional Bloom filters are needed to accommodate more peer identifiers in the same category.

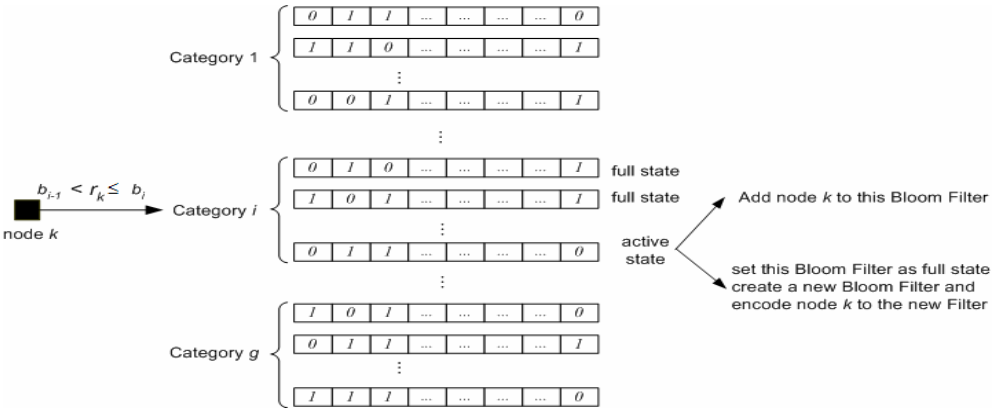


Figure 4 Bloom filter based storage scheme for efficient storage and retrieval of peer reputation ranking information

The time needed to save a node to a proper category is $O(\varphi+c)$, where φ is the number of hash functions applied and c is number of categories. To locate the category 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 in each category. Then the system returns the category number when there is a match. The time and space requirements of a Bloom-filter based storage system are given in Table 2, compared with those required in a conventional storage scheme for peer membership in various categories. The notation f_i refers the number of Bloom filters in each category. In total, there are $F = f_1 + f_2 + \ldots + f_c$ Bloom filters used.

6.2 Performance of Bloom Filter Storage

To estimate the performance of the storage scheme, we measure space reduction factor, false positive rate, and time complexity under different storage configurations. We consider the network size n ranging from 1,000 to 1 million. Table 3 summarizes the design parameters used in our simulation experiments on using Bloom filters. In practice, the category number c should be maintained small, less than 10 up to 1000 nodes and around 20 for a million nodes. The number of hash functions applied is less than 10. Each Boom filter has m = 1K to 0.4M bits.

The *space reduction* is defined by the ratio of memory bits to store all global scores in conventional memory array to that needed in a Bloom-filter storage scheme. The time complexity is the time to read a peer ranking. We have to check all mapped bit positions are set to 1 by application of all hash functions $h_i(id_j)$ $(1 \le i \le \varphi)$ in each category. The time to save (write) a peer identifier into a category is $O(\varphi+c)$.

Table 3 : Performance of Various Configurations of the Global Reputation Score Storage Unit based on Bloom Filtering

Network Size, *n*	Category Number, *c*	No. of Hash Functions, φ	Bloom Filter Size, *m*	False-Positive Rate	Space Reduction	Time Complexity
10^3	5	7	2 K bits	4%	3.2	35
10^3	10	7	1 K bits	7%	3.2	70
10^3	5	6	1.5 K bits	13%	4.3	30
10^6	20	9	0.6 M bits	6%	2.7	180
10^6	20	7	0.5 M bits	15%	3.2	140
10^6	30	9	0.4 M bits	8%	2.7	270

The *false positive rate* is the probability to return a wrong category 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 category *i* returns a false positive is approximated by:

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

where n_i is the number of nodes in category *i* and *c* is the number of categories.

Proof*:* The false positive probability of the first $\lceil n_i c/n \rceil$ Bloom filters in category *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 category *i*, the probability of not all the bits indexed by φ hash functions being set to 1 is $(1-(1-e^{-\varphi n/cm})^{\varphi})^{\lceil n_i c/n \rceil}(1-(1-e^{-\varphi(n_i-\lceil n_i c/n \rceil n/g)})^{\varphi})$. 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, ranging 4% to 15% in Table 3. The memory bit count is decreased 3.2 times with 4% false positive rate. A maximum of 35 hashing operations are applied to query the node category for one million of nodes. Increasing the category number *c* to 10 will incur smaller *m* but higher time complexity. Decreasing *m* to 1500 will lead to higher false positive rate. The optimal configuration for a 10^6-node P2P network is set at *c=20,* φ = 9 and *m= 0.6 M bit*. These results suggest that Bloom filters can reduce the memory space by 2-4 times compared to using conventional RAM memory arrays.

7. Security Enforcement in GossipTrust

We extend the basic GossipTrust algorithms with IBC-based secure communication and some selective peer aggregations. By leveraging power nodes, we extend the operations in GossipTrust to combat the problem of peer collusions.

7.1 Identity-based Signature for Secure Peer Communication

In this section, we present a security enhancement mechanism to guarantee security and integrity of the reputation data obtained through global gossiping. Current implementations rely heavily on traditional PKI as a way of supporting security services [32]. Xiong and Li assumed that all nodes have obtained their certificates from a trusted CA and additionally assume prior knowledge of the certificates of other nodes. This requirement adds considerable cost and

complexity, especially when the system membership is highly dynamic with nodes constantly joining and leaving, as many of the certificates can quickly become invalidated. Figure 5 illustrates how to use *identity-based cryptography* (IBC) to secure peer communications.

We use IBC-based [4][28] scheme to increase the security and reliability of the exchanged reputation data. In IBC, the public key can be generated from publicly identifiable information, such as a peer *id*. The corresponding private key is generated and maintained by a *Private Key Generator* (PKG). The potential of IBC to provide more immediate flexibility to entities in a security infrastructure may well match the qualities demanded by P2P computing. The central PKG is not a performance bottleneck because it does not participate in the P2P protocol and it can even be offline. In particular, IBC allows generation of key information on the fly, which reduces the overhead and complexity significantly.

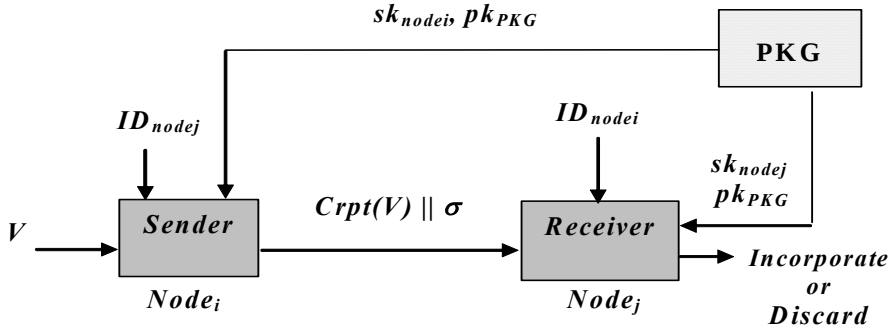


Figure 5 Use of identity-based cryptograph to secure peer communication

Suppose node *i*, through an IBC-based cryptosystem, wants to send an encrypted message to node *j*. Node *i* encrypts the message *V* by using the identifier of node *j* and PKG public key pk_{PKG}. Node *i* signs the encrypt message *Crpt*(*V*) with the signature σ using its private key $sk_{nodei.}$ Upon receiving the message, node *j* verifies signature σ using the identifier of node *i* and pk_{PKG}. After the authentication verification, node *j* decides to accept or discard the message. Node *j* can decrypt the message using its private key sk_{nodej} and pk_{PKG}. Figure 5 shows that node *i* neither needs node *j* public key certificate nor verifies its identity as the authentication task is always performed in traditional PKI. The use of IBC to generate public keys requires no certificate lookup or verification. This offers the flexibility to match with node dynamics.

Malicious node may not follow the gossip protocol to evaluate other nodes. They behave in

essentially three ways. First, it can communicate more frequently than prescribed by the protocol, pumping unwanted information into the system. Second, it can communicate less frequently than prescribed. Third, a malicious node can send incorrect information to other nodes [17]. We introduce a selection threshold for each node to decide whether to accept or discard the received reputation vectors. Nodes will only incorporate the reputation data originating from those whose global reputation scores are higher than the *peer selection threshold θ*.

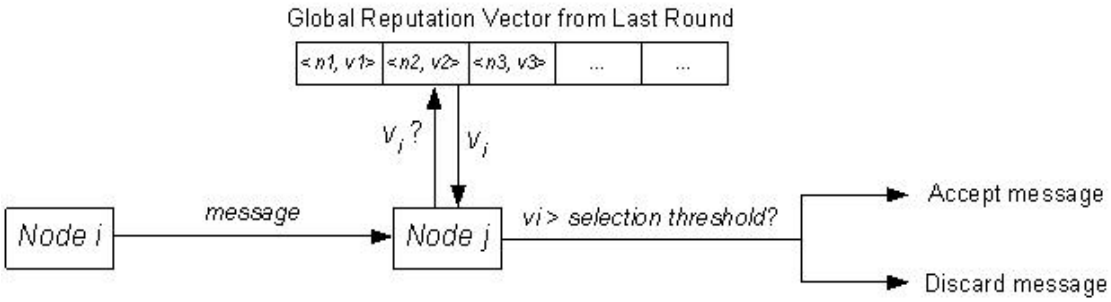


Figure 6: Peer selection threshold to optimize the reputation aggregation process

7.2 Combating Peer Collusions with Power Nodes

In out PowerTrust paper [36], we revealed that peer feedbacks in eBay are power-law distributed. The fundamental cause of this phenomenon is that new users keep joining the reputation system and they are willing to interact with more reputable nodes. So power-law feedback distribution is applied to P2P reputation systems, which was also observed by Yang et al in the Maze reputation system [34]. In Algorithm 3, we enhance the aggregation process of global reputations by leveraging the power nodes to play a major role in combating peer collusions through weighted gossipping.

Power-law feedback distribution implies the existence of some reputable peers called power nodes [10]. The new clients are encouraged to deal primarily with the power nodes to guarantee fast convergence speed [21] and to handle the problem of peer collusions. If node j is a power node, the probability p_j in Algorithm 3 is set to $1/p$. Otherwise, p_j is set to 0. Algorithm 3 specifies how a peer acting as a random-knowledge surfer to search for reputable nodes in the P2P network. With a probability α, 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.

Algorithm 3: Global Reputation Aggregation with Security Enhancement

INPUT: local trust matrix $S=(s_{ij})$, aggregation convergence threshold δ, and selection threshold θ
OUTPUT: converged global reputation vector $\boldsymbol{V_g}$
forall $i = 1, 2, \ldots, n$ **do**
  Repeat
    forall local trust score s_{ij}
    if ($t == 0$) **then** $x_j \leftarrow ((1-\alpha)s_{ij} + \alpha \cdot p_j) / n$
    else $x_j \leftarrow ((1-\alpha)s_{ij} + \alpha \cdot p_j) \times v_i(t-1)$
    if ($j == i$) $w_j \leftarrow 1$ **else** $w_j \leftarrow 0$
    add the triple $<s_j, j, w_j>$ to global reputation vector $V(t)$
    endforall
    send $\boldsymbol{V}(t)$ to node j itself
  Repeat
    let { $\boldsymbol{V}_j$ } be all vectors sent to $\underline{i}$ in previous gossip iteration
    forall $\boldsymbol{V}_j$ in { $\boldsymbol{V}_j$ }
      verify $\boldsymbol{V}_j$ signature by using the identifier of node j
      decrypt $\boldsymbol{V}_j$ with the private key of node i
      if ($v_j > \theta$) **then** accept V_j **else** discard V_j
    endforall
    forall node k in { $\boldsymbol{V}_j$ }
      update x_k and w_j
      update the triplet $<x_k, k, w_k>$ in global reputation vector $V(t)$
    endforall
    send ½ $V(t)$ to node i
    choose a random node k
    encrypt ½ $V(t)$ with id_k and sign it with private key of node i
    send the encrypted vector ½ $V(t)$ to node k
  Until every x_k / w_k in $V(t)$ converge
    forall $k = 1, 2, \ldots, n$
      $v_k \leftarrow x_k / w_k$ and replace every triplet $<x_k, k, w_k>$ with the pair $< v_k, k>$
    endforall
  Until $|V(t) - V(t-1)| < \delta$
endforall

8. Simulation Experiments and Performance Results

We evaluate the performance of GossipTrust through extensive simulation experiments. We designed an event-driven simulator of GossipTrust system of various sizes. In our experiments, we construct a Gnutella-like flat unstructured 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 point reported represents the average of at least 10 simulation runs.

The base setting and default values used in experiments are summarized in Table 4. As proven in our earlier paper [37], the number of peer feedback is power law distributed. Initially the maximum feedback amount $d_{\max}$ is 200 and the average feedback amount d_{avg} is 20. We choose a greedy factor $\alpha = 0.15$ as a default value. The system selects 1% of nodes as the power

nodes. The parameter θ represents the selection threshold defined in Section 7.1 and its default value is set at 3×10^{-4}. We measure the *reputation aggregation time* and *gossip aggregation error* under variable system parameters and threat models. We present simulated experiments to explore the performance gains of GossipTrust in representative P2P applications.

Table 4 Parameters and Their Default Values used in Simulation Experiments

Parameter	Meaning	Default Value
n	Number of initial peer nodes in a P2P system	1000
α	Greedy factor of a peer to choose power nodes	0.15
d_{max}	Maximum peer feedback amount	200
d_{avg}	Average peer feedback amount	20
γ	Percentage of malicious peers in a P2P system	10%
p	Maximum No. of power nodes in a P2P systems	1%
δ	Threshold for controlling global aggregation error	10^{-3}
ε	Threshold for controlling gossiping error	10^{-4}
c	Number of categories every node keeps	5
θ	Peer selection threshold	3×10^{-4}
λ	Query popularity rate in Power law distribution	1.2
M	Number of parallel jobs in PSA workload	40,000

On the average, we assume 5 Bloom filters used in each score-ranking category. We study three malicious peer behaviors. First, malicious peers cheat in transactions and issue dishonest feedbacks to others, independently. Second, with collusions among peers, they may collaborate with each other to boost up their own ratings. They may rate the peers in their collusion group very high and rate outsiders very low. The third kind of malicious behavior is caused by peers to act purposely not to reach consensus in gossip communication. In P2P file-sharing experiments, we assume the Power-law query distribution in Section 8.3. We execute the PSA (*parameter sweeping application*) benchmark workload in the large-scale P2P grid experiments reported in Section 8.4.

8.1 Convergence Rate and Aggregation Errors

The objective of this set of experiments is to evaluate computational efficiency and scalability of GossipTrust. As indicated in Section 4.2, there are two convergence processes in GossipTrust: one is the convergence of the reputation computation round and another is during every aggregation cycle, the convergence of gossip protocol to aggregate the weighted sum. 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 the reputation scores converge, given an arbitrary trust matrix and a fixed greedy factor $\alpha > 0.1$. The larger the number of gossip steps, the higher is the convergence overhead.

Figure 7 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 overhead, regardless of the network size. While when ε is large, as $\varepsilon > 10^{-2}$, the network size dominates the convergence overhead. With a fixed small ε, the convergence overhead remains closes for different network sizes, which means that GossipTrust is able to scale well when reputation system grows.

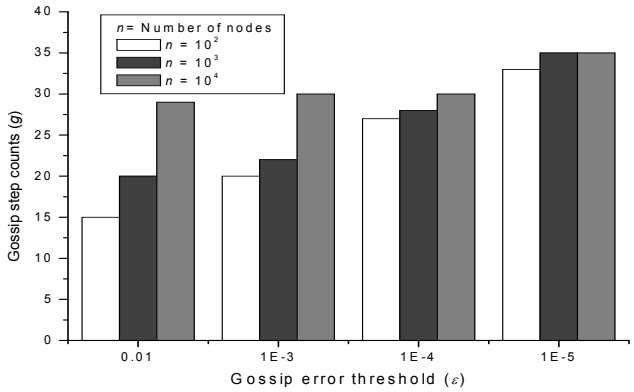
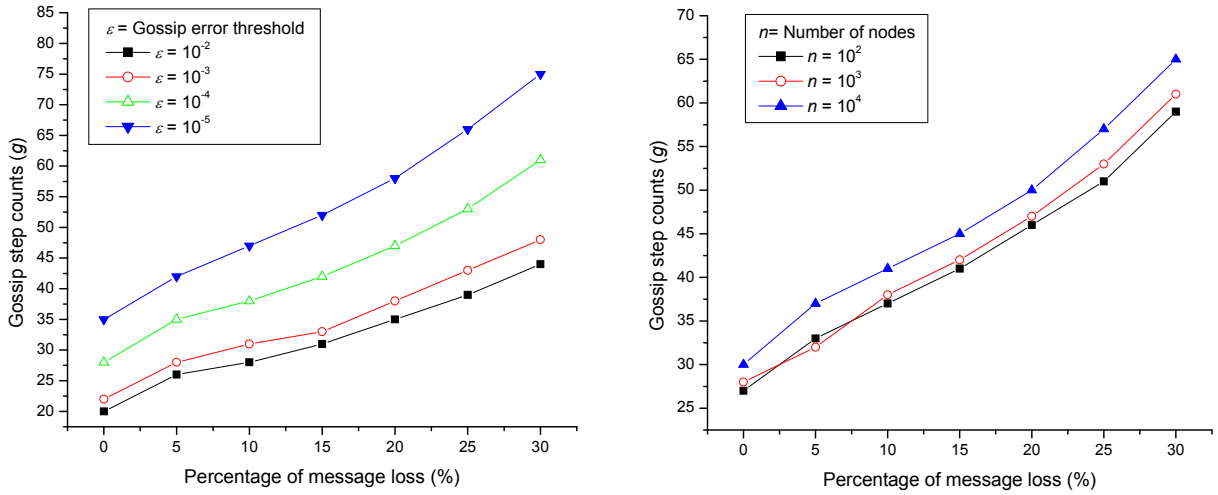


Figure 7. Gossip step count of three GossipTrust configurations under various gossip error thresholds

Figure 8(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 becomes very sensitive. Under the condition of the fixed gossiping error threshold ($\varepsilon = 10^{-4}$), shown in Fig.8(b), we observed the slow convergence when the percentage of message loss is more than 15% for different network size n. As we have expected, GossipTrust can tolerate moderate percentage of message loss (up to 15%) caused by the dynamism or unreliability of peers and links in the network.

The relative error incurred by gossip protocols in every aggregation cycle is propagated during the aggregation process. Table 5 reports both propagation error and computation error under different aggregation error threshold δ and gossiping error threshold ε. We access the

propagation error by measuring the relative error of the global reputation scores caused by gossip protocols. The computation error is measured as the distance between actual and estimated global reputation vectors.



(a) Network n= 1,000 nodes **(b) Fixed threshold $\varepsilon = 10^{-4}$**

Figure 8 Gossip step counts of GossipTrust under various allowable gossip errors

Table 5 shows the convergence overhead and aggregation error for different network size, under three combinations of gossiping error threshold and aggregation error threshold respectively. If the gossiping error 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.

Table 5: Gossip and Aggregation Errors under Different Convergence Conditions in Three GossipTrust Configurations

Threshold Values	Network Size	Aggregation Cycles	Gossip Steps	Gossip Error	Aggregation Error
$\varepsilon = 10^{-5}$, $\delta = 10^{-4}$	100	33	23	2×10^{-6}	4.2×10^{-5}
	1000	35	19	1×10^{-6}	1.6×10^{-4}
	10000	35	21	1.4×10^{-6}	3.7×10^{-4}
$\varepsilon = 10^{-4}$, $\delta = 10^{-3}$	100	27	14	1.1×10^{-5}	3.4×10^{-4}
	1000	28	15	7×10^{-6}	7.3×10^{-4}
	10000	30	12	9.1×10^{-6}	8.9×10^{-4}
$\varepsilon = 10^{-3}$, $\delta = 10^{-2}$	100	20	7	1.7×10^{-4}	3.7×10^{-3}
	1000	22	5	1.6×10^{-4}	3.8×10^{-3}
	10000	29	6	2.3×10^{-4}	4.3×10^{-3}

Tradeoffs exist between computational efficiency and accuracy: the smaller are the aggregation error threshold δ and gossiping error ε, the less the propagation of computation error, but the larger the number of gossip steps and reputation aggregation cycles. Based on the result in Table 5, for a network of 1000 nodes, we choose $\varepsilon = 10^{-4}$ and $\delta = 10^{-3}$ to balance the tradeoff between the convergence overhead and computational accuracy.

8. 2 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:

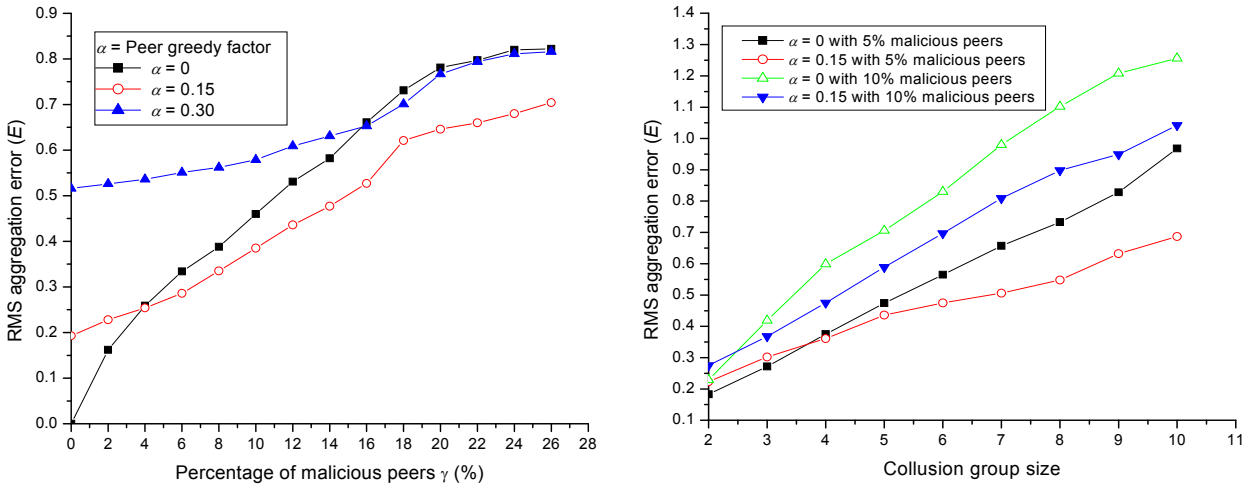
$$\textit{RMS aggregation error} \quad \boldsymbol{E} = \sqrt{\frac{\sum((v_i - u_i)/v_i)^2}{n}} \qquad \textbf{(12)}$$

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.9(a) the RMS errors under different values of α and various percentages of independent malicious peers. By leveraging power nodes, we set the the greedy factor $\alpha = 0.15$. This gives 20% less aggregation error than treating all peers equally with $\alpha = 0$. 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 $\alpha = 0.15$ is indeed a very good choice.

Figure 9(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. In all cases (5% and 10% collusive peers), leveraging the

power nodes with a greedy factor $\alpha = 0.15$ 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.



a) Independent malicious peers **(b) Collusive malicious peers**

Figure 9: Effects on global aggregation errors by fake trust scores from different percentages of malicious peers in a 1,000-node P2P network

Besides reporting unreliable peer scores, malicious peers may deviate from the gossip protocol as indicated in Section 7.1. In Algorithm 3, we have used a *peer selection threshold* θ to filter out malicious peers. The effects of different choices of θ are shown in Fig.10 under independent peer behaviors. The possibility of a peer behaving maliciously increases inversely with respect to its global reputation score. Figure 10 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 the increase of malicious nodes. The RMS error increases slower when θ is set very low 0.0001 under the condition of 10% malicious peers. 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 θ= 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. We test the case by setting θ = 0.0005. Figure 10 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. The conclusion from Fig.9 and Fig.10 is that proper choices of α and θ will make a sharp difference in error control in global reputation aggregation process.

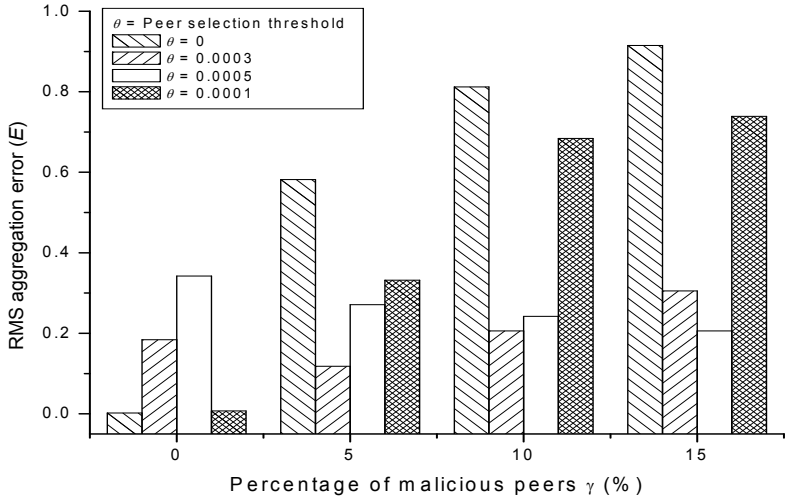


Figure 10. Effects on RMS aggregation errors from fake scores reported by malicious peers in a 1000-node P2P network

8.3 P2P File Sharing Supported by GossipTrust System

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 researchers [19]. There are over 100,000 files simulated in these experiments. The number of copies of each file is determined by a Power-law distribution with a popularity rate λ = 1.2. Each peer is assigned with a number of files based on the Sarioiu distribution [19]. At each time step, a query is randomly generated at a peer and completely executed before the next query step. The query popularity reflects which file the query prefer to request.

We rank the queries according to their popularity. We use a power law distribution with a λ = 0.63 for queries ranked 1 to 250 and λ = 1.24 for lower-ranking queries. This distribution models the query popularity distribution in Gnutella. After a query for a file is issued and 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 at all sites after 1,000 queries.

The *query success rate* is measured by the percentage of successful queries over the total number of queries issued. Every node has a rate to respond a query with inauthentic files. For simplicity, this rate is modeled inversely proportional to node's global reputation. We also consider the case of a *NoTrust system,* which randomly selects a node to download the desired file without considering node reputation. We plot the result of using GossipTrust and NoTrust in simulated P2P file sharing experiments in Fig.11

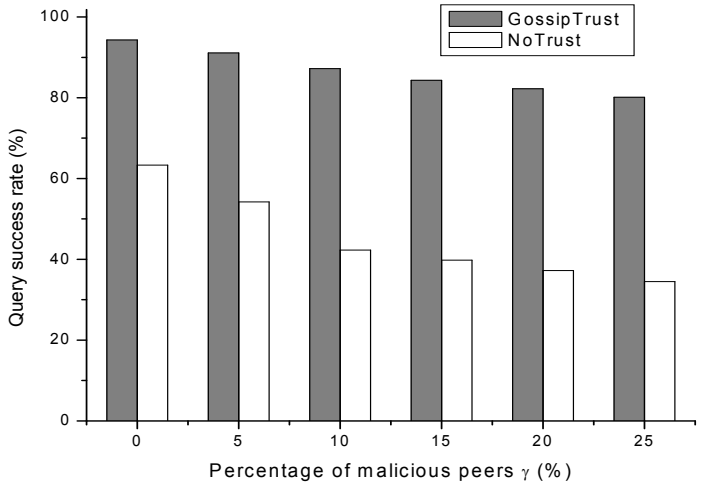


Figure 11 Query success rate of a 1000-node GossipTrust system, compared with an identical system without any reputation support

In this experiment, the malicious peers issue unreliable scores and provide corrupted files. The performance of GossipTrust drops only slightly with the increase of the number of 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. This experiment proves the effectiveness of using reputation-based selection scheme in unstructured P2P file sharing applications

8.4 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], [37], [38]. Denote the total number of parallel jobs as *M.* We use two performance metrics to evaluate the GossipTrust performance in P2P

Grid job execution over the PSA workload [26]. The *average job turnaround time* is defined by averaging all job completion time $\sum_i \{c_i\}/n$, where c_i is the completion time of the i-th job for i = 1 ,2,…M. The *average job success rate* is defined by the ratio $F_{rate} = (1\text{-}M_{fail})/M$, where M_{fail} *ac*counts the number of failed jobs. Figure 12 shows the performance results of our GossipTrust reputation system over the PSA workload, compared with a system with the no-trust support.

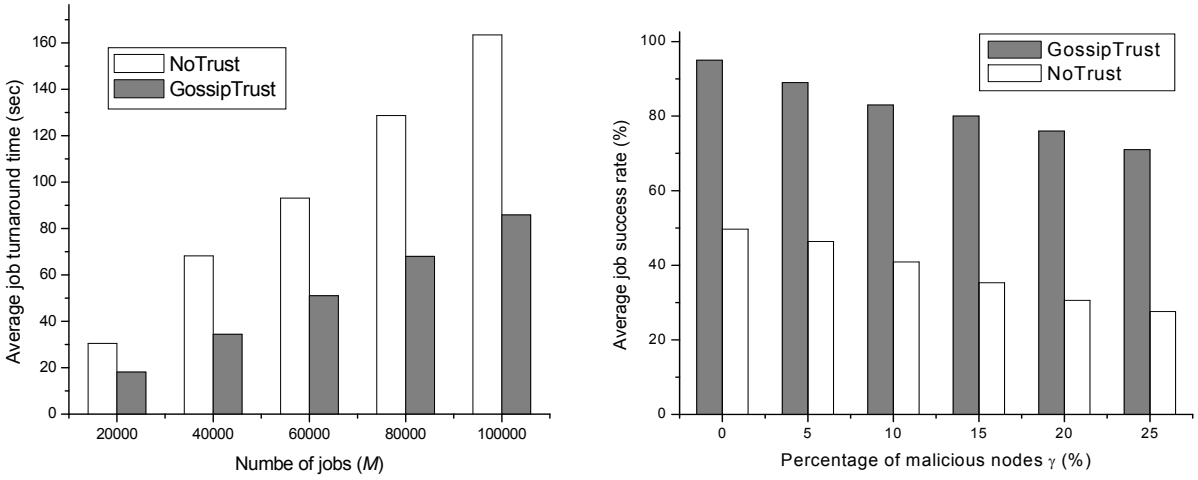
We apply a realistic PSA workload of 20,000 to 100,000 jobs in Fig.12(a) over a P2P Grid, consisting of 4,000 resource sites. We assumed an average job execution time 5 sec/job and an average 2 jobs/sec arrival rate. The PSA benchmark runs independent jobs. The execution model essentially involves parallel 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 actually executing the job.

GossipTrust computes the 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 reputation scores for all sites every hour. A job will be executed if it was not rejected for more than 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 aggregated by GossipTrust.

Figure 12(a) reports the average job turnaround time (average completion time) of the PSA workload executed under both GossipTrust and NoTrust. The turnaround time increases linearly with respect to the increase of the job number. GossipTrust significantly outperforms NoTrust by saving more than 50% job completion time. We plot in Fig.12(b) the average job success rate against different percentage of malicious peers under the default value of M = 40,000 jobs. We experimented 10 peers in a collusion group.

In the PSA experiments, the malicious peers fail the job execution at any time. Without trust information, the job success rate is quite low (around 48%). GossipTrust has more than 40% performance gains over the case of NoTrust support, when there is less than 10% malicious peers. Even when the malicious peers increase to 25%, the GossipTrust is able to maintain a 75%

average job success rate. These results prove the effectiveness of using global reputation to establish trust among the participating peers in a large-scale P2P grid computing grid.



(a) Effect of job number **(b) Effect of malicious peer density**

Figure 12. PSA benchmark results on a simulated P2P Grid configuration under various PSA workload and malicious peer distributions

9. Conclusions and Further Work

In any unstructured P2P network, global reputation aggregation is quite expensive when the network grows very large 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. GossipTrust is shown very fast in aggregating local trust scores into global reputation scores. The major innovations in GossipTrust development are summarized in three aspects: fast gossip-based aggregation algorithms, efficient reputation storage with Bloom filters, and secure communication with identity-based cryptography.

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 tolerates 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 performance gains lie in reduced number of aggregation cycles, faster convergence rate, lower RMS aggregation errors, and

higher query success rate in distributed file sharing, and much improved job success rate and turnaround time in PSA applications.

We have to point out that the GossipTrust system is not restricted to apply only in unstructured P2P systems exclusively. With minor modifications, the system can perform even better in a structured P2P system. The gossip steps and reputation aggregation process reported here can be further accelerated by the fast hashing and search mechanisms built in DHT-based overlay networks. 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) such as those performance measures reported here and another for *quality-of-feedback* (QoF) by participating peers.

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]. Further research is also encouraged to apply reputations systems to enforce copyright protection in P2P systems. With the help of object reputation [31], 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 systems for managing object reputations.

Acknowledgements: This work was fully supported by NSF ITR Grant ACI-0325409 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] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Paring", *Proceedings of 21st Advances in Cryptology*, 2001.

[5] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A survey", *40th Conf. on Communication, Control, and Computing*, 2002.

[6] 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.

[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", *Sigmetrics'05*, Alberta, June 2005

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

[11] K.Gummadi, R.Dunn, R. Dunn, "Measurement, Modeling and Analysis of a Peer-to-Peer File-Sharing Workload", *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, Bolton Landing, NY, 2003.

[12] J. Hu, 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,* 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] X. Lou and K. Hwang, "Adaptive Content Poisoning To Prevent Illegal File Distribution in P2P Networks", *IEEE Trans. Computers,* submitted August 2006.

[19] 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.

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

[21] 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.

[22] S. Nandy, L. Carter and J. Ferrante, "GUARD: Gossip Used for Autonomous Resource Detection", *19th Int'l Parallel &Distributed Processing Symposium*, Colorado, Apr. 2005.

[23] D. Qiu and R.Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer

Networks", *Sigcomm 2004*, Portland, USA, Aug-Sep, 2004.

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

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

[30] 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

[31] 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

[32] 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.

[33] B. Yang, T.Condie, S. Kamvar and H. Garcia-Molina, "Non-Cooperation in Competitive P2P Networks", *Proceedings of the 25th IEEE Int'l Conference on Distributed Computing Systems* (*ICDCS'05*), Columbus, Ohio, 2005

[34] 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

[35] 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.

[36] R. Zhou and K. Hwang, "PowerTrust: A Robust and Scalable Reputation System for Trusted P2P Computing", *IEEE Trans. on Parallel and Distributed Systems*, accepted, March 2006. (in press)

[37] R. Zhou and K. Hwang, "Gossip-based Reputation Aggregation in Unstructured P2P Networks", IEEE Int'l *Parallel and Distributed Processing Symposium* (IPDPS-2007), accepted to be presented in March 2007.

[38] 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.

Biographical Sketches:

Runfang Zhou received the B.S. and M.S. in computer science from Southeast University, China. She is currently pursuing Ph.D. in Computer Science at the University of Southern California. She is expected to complete all Ph.D. degree requirements by Spring 2007. Her research activities cover peer-to-peer reputation systems, overlay network design, web services performance improvement, and trust and secure collaboration in Grid computing. She can be reached at: rzhou@usc.edu.

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 University of California, Berkeley in 1972. An IEEE Fellow, he specializes in computer architecture, parallel processing, Internet and wireless security, P2P, Grid, and distributed computing systems. He has published over 200 original scientific papers and 4 popular textbooks in these areas.

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.* His latest two books, *Scalable Parallel Computing* and *Advanced Computer Architecture* are being adopted worldwide and translated into 4 languages. His research group at USC have developed security-binding techniques, P2P reputation systems, distributed intrusion detection systems against network worms and DDoS attacks in trusted Grid, P2P, and Internet computing applications. Contact him at kaihwang@usc.edu or visit web site: http://GridSec.usc.edu/Hwang.html.

Min Cai received his BS and MS degrees 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 University of Southern California. His research interests include P2P and grid computing, network security, semantic web, and web services technologies. His current email address is mincai@usc.edu.