Refining Reputation to Truly Select High-QoS Servers in Peer-to-Peer Networks

Haiying Shen*, *Member, IEEE*, Yuhua Lin, Ze Li, *Student Member, IEEE*

Abstract—Peer-to-peer networks (P2Ps) use reputation systems to provide incentives for nodes to offer high quality of service (QoS) and thwart the intentions of dishonest or selfish nodes. Existing reputation systems have two problems. First, they directly regard node reputation as trust. Rather, reputation represents the opinions formed by others about a node's QoS behavior, while trust represents a node's honesty and willingness to cooperate. In addition to trust, factors such as node capacity and lifetime also influence reputation. Due to these factors' heterogeneity and variance over time, reputation cannot directly reflect a node's trust or current QoS. Second, existing reputation systems guide a node to select the server with the highest reputation, which may not actually select the highest QoS server and would overload the highest-reputed nodes. This work aims to accurately reflect node trust and provide guidance for high QoS server selection. Through experimental study, we find that node trust, available capacity, and lifetime positively affect node reputation. Based on this observation, we first propose a manual trust model and an automatic trust model that remove the influence of additional factors on reputation to truly reflect node trust. We then propose a high-QoS server selection algorithm that separately considers node trust, current available capacity, and lifetime. Extensive simulation results demonstrate the effectiveness of the trust models in accurate node trust reflection compared with an existing reputation system. Moreover, the server selection algorithm dramatically increases the success rate of service requests and avoids overloading nodes.

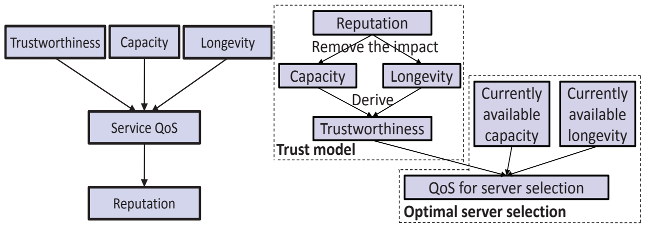
Index Terms—Peer to peer networks, Reputation systems, Server selection, Bayesian networks, Quality of service

1 Introduction

Peer-to-peer networks (P2Ps) enable the sharing of globally-scattered computer resources, allowing them to be collectively used in a cooperative manner for different applications such as file sharing [1–4], instant messaging [5], audio conferencing [6], and distributed computing [7]. P2P applications scale to a large community of users and take full advantage of heterogeneous resources widely scattered all over the world. Node cooperation is critical in achieving reliable performance of P2Ps. However, cooperation is challenging in P2Ps, where many diverse and autonomous parties without preexisting trust relationships work together. Some internal nodes may be compromised, misbehaving, selfish, or even malicious.

Reputation systems [8–15] such as those in eBay [16] and Amazon [17] are a main method used to tackle this problem. These systems collect, distribute and aggregate feedback on participants' past behavior and evaluate a node's trustworthiness to help nodes decide whom to trust, encourage cooperative and honest behavior, and deter uncooperative and dishonest participants. The ability to accurately reflect node trust and provide guidance for high-QoS server selection is vital to the effectiveness of a reputation system.

In reputation systems, a service receiver (i.e., client) usually rates a service provider (i.e., server) based on its received QoS. That is, *reputation* represents the opinions formed by other nodes on a node's QoS behavior. Existing reputation systems usually guide a node to select the server with the highest reputation. However, this



(a) Reputation system (b) The proposed trust system
Fig. 1: The traditional and the proposed trust systems.

server selection strategy may not actually select a high-QoS server and would also overload the highest-reputed nodes. We present the reasons below.

We define *trust* as a measure of a node's willingness to be cooperative in providing service. We call a node's lifetime in the network *longevity*. A node's available longevity is the time it will stay in the network from the present. If a node does not have sufficient available capacity or available longevity (hence serving time), it cannot provide high QoS, while high available capacity and longevity enable a server to provide high QoS. As shown in Figure 1(a), in addition to node trust, node reputation is determined by additional factors such as node capacity and longevity. Different nodes have different capacities and longevities and the available capacity and longevity of a node vary over time. Due to these *heterogeneous* and *time-varying* attributes of the factors, reputation cannot directly reflect a node's trust.

Altruistic but low-longevity, low-capacity, or overloaded nodes may have low reputation values. Nodes that reject requests sometimes but have high-longevity and high-capacity may gain high reputation values. A high-reputed node that had significant available capacity to handle requests in the past does not necessarily have sufficient capacity for requests in the present. Similarly, a high-reputed node that had previously stayed in the system for a long time in the past does not necessarily stay for a long time from the present. Thus, servers selected by the highest-reputed node selection policy may not offer high QoS. The selected highest-reputed node may

• * *Corresponding Author. Email: shenh@clemson.edu; Phone: (864) 656 5931; Fax: (864) 656 5910.*

• *The authors are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, 29634.*
E-mail: {shenh, yuhual, zel}@clemson.edu

be (1) a high-trust node that currently has insufficient available capacity or low longevity, or (2) a low-trust node that has high capacity or high longevity. Also, the policy may miss high-trust nodes with sufficient available capacity and longevity for the requested service. In addition, it would overload the highest-reputed nodes by biasing them.

Therefore, neglecting additional factors such as capacity and longevity will result in an inaccurate node trust measurement and an inadequate capability to provide correct guidance for high-QoS node selection, leading to suboptimal or degraded overall system performance. This work aims to address these neglected issues. The contributions of this work can be summarized as follows.

(1) We study the relationship between node trust, reputation, and additional factors through experiments and analysis, and observe that the additional factors positively impact node reputation.
(2) Based on the previous observation, we develop two trust models (one manual and one automatic) that remove the impact of additional factors on reputation in order to derive accurate node trust (Figure 1(b)).
(3) We propose an optimal server selection algorithm that separately considers node trust and the current values of additional factors to ensure high QoS (Figure 1(b)). By optimal server, we mean that the selected server has high trustworthiness and sufficient available capacity and longevity for the requested services.

As far as we know, this is the first work that focuses on deriving accurate trust by removing the impact of additional factors on reputation and providing accurate guidance for high-QoS server selection.

The rest of this paper is organized as follows. In Section 2, we study the relationship between additional factors and reputation, and introduce the trust models and server selection policy. Section 3 presents the experimental performance of the trust models in comparison to a reputation system, and the server selection algorithm in comparison to other server selection algorithms. Section 4 presents a review of related work on reputation systems. The final section concludes with a summary of contributions and discussions on further work.

2 Reputation-based Trust Models

A P2P network usually has a reputation system for reliable operation and high performance. Our proposed trust system, built upon a reputation system, includes two components: trust models and optimal server selection. Based on the reputation values offered by the reputation system, *trust models* conduct a trust evaluation that excludes additional factors. By considering the derived trustworthiness with the additional factors, the *optimal server selection* algorithm provides the correct guidance for server selection. This work contributes not only to the reliability of the P2Ps, but also to their efficiency. For reliability, it enhances the accurate guidance of the reputation system. For efficiency, it helps to enable full utilization of all node resources and avoid overloading high-reputed nodes.

2.1 Study of Reputation and Trust

We conduct experiments to study the relationship of reputation with capacity and longevity. The experiment parameters and settings are presented in Section 3. In this experiment, all nodes are trustworthy with a 100% probability of serving requests, and they receive approximately the same number of requests. As the work in [18], we assume node capacity can be represented by one metric. A node's available capacity equals $c_a = c - w$, where c is its capacity represented by the number of service requests it can handle during a time unit, and w is its workload represented by the number of its received requests during a time unit. We assume that every node is rational and it gives reputation rating based on its received QoS. Also, a node's offered QoS is determined by its available capacity [19]; higher available capacity enabling it to offer higher QoS and vice versa.

Firstly, we test the relationship between node reputation and capacity with the assumption that each server's longevity is high enough to complete the requested services. In this case, since a higher available capacity enables a server to offer higher QoS, a node gives its server a reputation value of r_c, which equals the server's available capacity c_a. Figure 2 shows the reputation of each node versus its capacity. We see that the reputation increases linearly as capacity grows. That is, for fully trustworthy nodes, a higher-capacity node has a higher reputation because such a node can offer higher QoS hence receives higher reputation rating. The results mean that node capacity positively influences a node's reputation.

Secondly, we test the relationship between node reputation and node longevity with the assumption that each server has the same amount of available capacity that is high enough to complete a service requested from it. We use l_a to denote the available longevity of a server and use l_r to denote the time requirement of a requested service. In this case, when $l_a \geq l_r$, the requested service can always be fully completed, and when $l_a < l_r$, only l_a/l_r percent of a requested service can be completed. Since each server has the same amount of available capacity for a service, to evaluate a server's QoS based only on longevity, a server's reputation should be evaluated according to the percent of a requested service it has completed. Thus, the reputation value r_l equals

$$r_l = \begin{cases} l_a/l_r & \text{if } l_a/l_r < 1 \\ 1 & \text{otherwise.} \end{cases} \qquad (1)$$

Figure 3 shows the reputation of each node versus its longevity. We observe that the relationship between reputation and longevity exhibits a logarithmic trend, and higher-longevity nodes have higher reputations. Here, the relationship is non-linear because the reputation evaluation is based on the percent of requested services completed by a server. As long as a server's available longevity exceeds the requested service's required time, they receive the same reputation regardless of the differences in their available longevities. For those servers whose available longevities are lower than the required time, lower available longevity leads to lower reputation.

Thirdly, we test the relationship between node reputation and both capacity and longevity. In this case, the reputation value given is $r = r_c \cdot r_l$. Figure 4 illustrates the joint effect of both factors on reputation. The series of nodes indicated with $A - J$ are the nodes with the same longevity. We make a number of observations from the figure. First, for nodes with the same longevities, those with higher capacities have a higher reputation. For example, in each of the series indicated by $A - J$, nodes with larger capacities have higher reputations. Second, a node's low longevity significantly reduces its

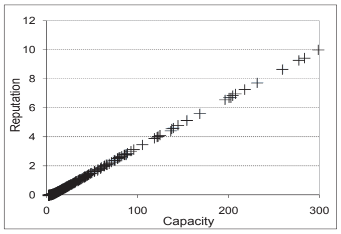


Fig. 2: Reputation vs. capacity.

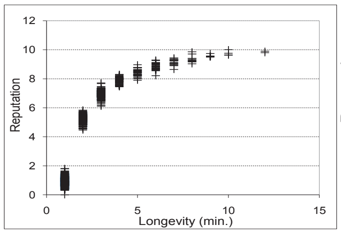


Fig. 3: Reputation vs. longevity.

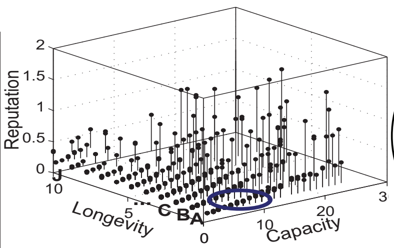


Fig. 4: Reputation vs. longevity & capacity.

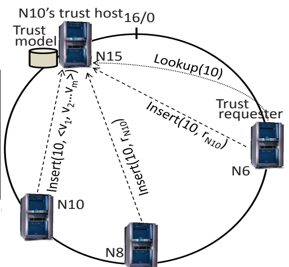


Fig. 5: Reputation information aggregation.

reputation. Comparing nodes with capacity 10 in series A and B in the circled region, we see that the nodes with higher longevities have higher reputations than those with lower longevities. This is because higher longevity decreases the probability that a server leaves while performing a requested service. Nodes with low capacity but high longevity still have low reputations. This is because low-capacity nodes become overloaded easily; thus, no matter how long a node stays in the system, it cannot gain a high reputation.

The experimental results confirm that a reputation value alone cannot directly reflect node trust. It is also affected by capacity and longevity. Thus, reputation cannot be directly used for optimal server selection since a node's available capacity and longevity constantly change, so high-reputed nodes in the past may not have sufficient available capacity and longevity to provide high QoS. Therefore, the effect of a node's previous capacity and longevity on the reputation should be removed when evaluating a node's trustworthiness, and the currently available capacity and longevity should be considered with trust in choosing a service provider. The experimental results in Section 3 confirm the effectiveness of the combined consideration of currently available values of additional factors in server selection.

2.2 Reputation Information Aggregation

Our proposed trust models can be built on any reputation system that collects feedback and calculates global node reputation values. The proposed methods for efficiently collecting feedback and calculating reputation values in [9, 10, 14, 15] are orthogonal to our study. We use PowerTrust [8] as an example for the underlying reputation system. PowerTrust uses a number of high-reputed power nodes for reputation aggregation and calculation. We call these power nodes *trust hosts* (THs). We use ID_i to represent the structured P2P ID of node i, that is the consistent hash value of node i's IP address. The TH for reputation feedback of node i is the P2P owner of ID_i. The nodes use P2P functions $\texttt{Insert}(ID_i, r_i)$ to send the feedback of node i (r_i) to its TH, and $\texttt{Lookup}(ID_i)$ to query the reputation value of node i from its TH. A TH periodically collects the feedback and computes the reputation values of its nodes.

The THs need information on additional factors when evaluating node trust. When a client selects a server from a number of server options, it needs to query the server options' available capacity and available longevity to make the final decision. We assume that a node knows its approximate departure time in this paper. Thus, when a client reports a server's reputation rating, it also reports the server's available capacity (c_a) and available longevity (l_a) using the function $\texttt{Insert}(ID_i, r_i + V_i)$, where $V_i =< c_a, l_a, \cdots >$. The server's TH uses the reported reputation rating and V_i to derive its trust by applying the proposed trust models. Using $\texttt{Lookup}(ID_i)$, a node's query for node i's trust is forwarded to the TH, which returns node i's trust.

A node is unlikely to misreport its V_i to a queried client with values lower than its actual values in order to receive a high reputation. First, low c_a and l_a will prevent it from attracting service requests to increase reputation. Second, if a server cannot provide a service that corresponds to its trust, the client can report this to the server's TH. The server's TH can also notice this from other feedbacks on the server. The TH will then punish the server by greatly reducing its reputation. For example, in our proposed methods, a client chooses a server with sufficient capacity and longevity. If a node's TH notices that many reputation evaluations on this server do not match its trust value, the server is regarded as a cheater and its trust value is reduced. Thus, a rational node will not misreport its information, since this is unlikely to bring about benefits but will lead to a reputation penalty if detected. Admittedly, clever cheaters can find a way to avoid being punished. We will study the possible methods used to circumvent punishment and explore strategies to counteract these malicious behaviors in our future work. Techniques to deal with dishonest reports [20, 21] can also be adopted.

Figure 5 presents a 4-node reputation system built on top of Chord [3] with a 4-bit identifier space. Node $N10$ periodically reports its V to its TH $N15$ by $\texttt{Insert(10,}V_{N10}\texttt{)}$. Others report to $N15$ about $N10$'s local reputation by $\texttt{Insert(10,}r_{N10}\texttt{)}$. $N15$ calculates $N10$'s reputation value and then calculates its trust value based on its V using our proposed *trust models*. When a node, say $N6$, wants to select a server from several candidates, it queries for the trust value and V of each server. For example, it uses $\texttt{Lookup(10)}$ to query $N10$'s trust value and V. Then, it uses the proposed *optimal server selection algorithm* to choose a server.

2.3 Manual Trust Model

The overall QoS of P2P applications depends on the cooperation of the individual nodes in service. A reputation system is a widely used means to get a quantifiable measure of each node's trust based on evaluations from others about its performance. However, as demonstrated in Section 2.1, the reputation actually cannot directly represent node trust. This is essentially because a server's QoS or reputation is also affected by its capacity and longevity, in addition to its trust. To address the problem, we introduce two trust models: manual and automatic. The models help to remove the influence of node capacity and longevity on reputation when determining node trust.

As verified by the experimental results, for trustworthy nodes with a willingness to contribute their complete resources, those with higher capacities have higher reputation values and vice versa. Similarly, nodes with higher longevities receive higher reputations. Therefore, we regard nodes whose reputations are high with respect to their capacity and longevity as trustworthy nodes. This means medium-reputed and even low-reputed nodes with low capacity or longevity but a correspondingly high reputation should also be regarded as trustworthy.

Based on this rationale, we propose a manual trust model for node trust evaluation. From Figure 2, we know that reputation has an approximately linear relationship with capacity. Thus, if there are no other factors that influence reputation except capacity, we can use the ratio of a node's reputation over its capacity to measure its trust, i.e., $t_c = r/c$, where r and c denote the node's reputation and capacity, respectively. t_c represents the reputation earned by a node for each unit of capacity it has contributed to providing service. We assume there are m levels of trust in the system. The normalized t_c determines a node's trust level. Figure 6 shows a coordinate graph with the x axis representing capacity and the y axis representing reputation. The space is divided into different sections, each representing a trust level. A higher trust level means a node's reputation is high relative to its capacity. We map a node's normalized t_c to the graph according to its capacity. Based on its coordinate location, the node's trust level is determined.

From Figure 3, we know that reputation has a logarithmic relationship with longevity. Using MATLAB, we transform the curve to a line by changing longevity l to $loglog(l+1)$ as demonstrated in Figure 8. Hence, reputation has a linear relationship with $loglog(l+1)$. Similar to capacity, depending on a coordinate as in Figure 6, we can use $r/loglog(l+1)$ to measure a node's trust level when there are no other additional factors except longevity. By considering both capacity and longevity, we introduce spatial/temporal values. By viewing node capacity in a spatial domain and node longevity in a temporal domain, the spatial/temporal value (u) is defined as:

$$u = \alpha \times (c \cdot loglog(l+1)), \tag{2}$$

where α is a constant factor. A higher α leads to lower absolute trust value and vice versa, but the α value does not affect the relative trust levels between nodes. Based on the above analysis, the relationship between reputation r and u can be approximately regarded as a linear relationship. A node with a higher u should have higher reputation and vice versa. Thus, each TH builds a trust model as shown in Figure 7. The model is a two-dimensional space where spatial/temporal value and reputation are coordinates. A node's trust value is calculated by $t_v = r/u$. We use a locality-preserving hashing [22] to normalize the trust value in order to get trust level t_l. That is, $t_l = m \cdot (t_v - \min(t_v))/(\max(t_v) - \min(t_v))$, where m is the number of trust levels in the system and $\max(t_v)$ and $\min(t_v)$ are the maximum and minimum values of t_v in the system, respectively. We explain how THs can get $\max(t_v)$ and $\min(t_v)$ later on.

Previous reputation systems always set a reputation threshold. The nodes with reputations higher than the threshold (i.e., nodes above the horizontal dotted line) are trustworthy nodes, and others are untrustworthy

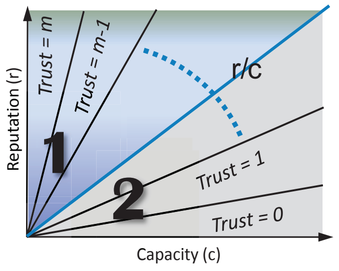


Fig. 6: Capacity-based trust.

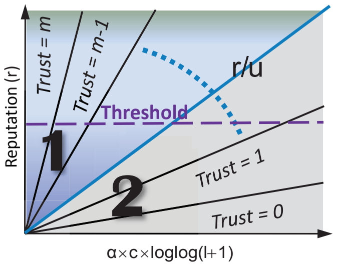


Fig. 7: Capacity & longevity-based trust.

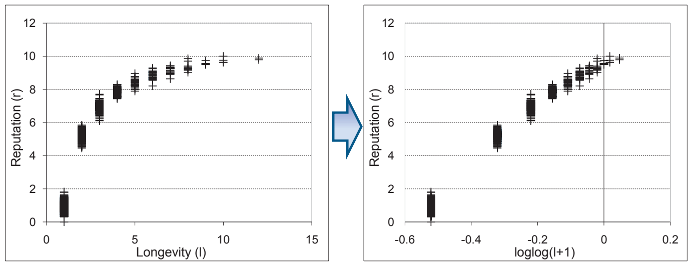


Fig. 8: Reputation vs. longevity and $\log\log(l+1)$.

nodes. Rather than using an absolute reputation value for node trust evaluation, the proposed trust model considers the relative reputation increase rate of the node corresponding to node capacity and longevity, and derives accurate node trust by removing the influence of capacity and longevity from node reputation. In Figure 7, the nodes located in area 1 are the nodes whose reputations are higher corresponding to their spatial/temporal values than the average t_v rate. These nodes have relatively higher trust than the average trust in the system. Among these nodes, some have low reputations only because of their low longevity or low capacity. In contrast, nodes in area 2, whose t_v is lower than the average level, are the nodes whose reputations are low corresponding to their spatial/temporal values. These nodes are relatively untrustworthy. A node with high longevity, large capacity and low reputation has low trust, while a node with low longevity, low capacity and high reputation has high trust. To evaluate a node's trust in the system, its TH first locates the node's position in the two-dimensional trust model based on its reputation and spatial/temporal value; then, that level is the node's trust level.

Below, we present how a TH can get the $\max(t_v)$ and $\min(t_v)$ in the system. To solve this problem, THs form a d-nary tree to collect t_v values of nodes in the system in a bottom-up manner. After the tree root receives all the ratio values, it finds the $\max(t_v)$ and $\min(t_v)$ and then propagates the information along the tree in a top-down manner. Based on the values, each TH calculates the trust levels of its nodes. We briefly describe how to dynamically arrange THs into a d-nary tree for information collection. For more details, please refer to [23]. For an ID space $[0, n-1]$, the tree root is the TH whose ID is the closest to $\frac{n}{2}$, holding the whole ID space. This ID space is partitioned into d parts with equal size. The TH whose ID is the closest to the middle ID of each part is selected as the tree node to hold this part of ID space. These d representatives become the children of the root. Each part is further partitioned into d parts with equal size, and so on, until there is only one TH in the ID part. In this way, each TH can calculate the IDs of its children and parent in the d-nary tree. Using the P2P function `Insert(ID,object)`, children

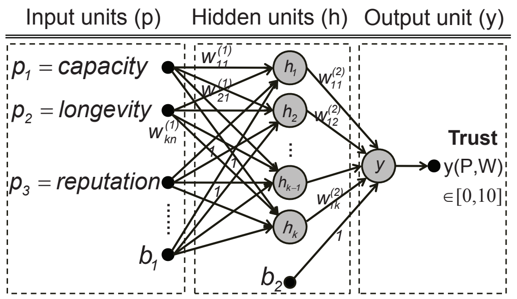


Fig. 9: Neural network based trust model.

report their t_v values to their parents. Then, the tree root collects all t_v and propagates the information of $\max(t_v)$ and $\min(t_v)$ from the top to the bottom along the tree, such that all THs receive the values.

2.4 Automatic Trust Model

A node's capacity, longevity, trust, and other factors determine its reputation value. It is a challenge to determine the weight of each factor's influence on reputation. The proposed automatic trust model uses a neural network technique [24] for node trust evaluation by capturing the nonlinear relationship between reputation, trust, and additional factors including capacity and longevity. The neural network can have any number of layers. It has been indicated that only one layer of hidden units can successfully approximate any function [25]. Thus, we build a neural network with one layer of hidden units [25] as shown in Figure 9. It is a nonlinear function from a set of input variables $P = \{p_1, p_2, \cdots\}$ to output variable $y \in [0, 10]$ controlled by a set of vectors of adjustable weight parameters $W(w_{ji} : 1 \leq i \leq n, 1 \leq j \leq k)$. The input units are a node's attribute variables including reputation, longevity, capacity and other additional factors, and the output is the node's trust level. The inputs are fed into the hidden units, which are merely "fan-out" units; no processing takes place within these units. The output of the hidden units is fed into the y output unit. The activation of a hidden unit is a function F_i of the weighted inputs plus a bias as given in the following equation:

$$y(P, W) = F_j(\sum_{j=1}^{k} w_{1j}^{(2)} F_i(\sum_{i=1}^{n} w_{ji}^{(1)} p_i + b_1) + b_2),$$

where

$$F(x) = \frac{1}{1 + e^{-x}}.$$

After building the neural network, we use training data, including the nodes' capacities, longevities, and desired trust to train the neural network to learn how to evaluate a node's trustworthiness based on its attribute variables. The neural network adaptively changes its structure based upon multiple inputs that flow through the network during the training phase in order to reduce errors in training.

Errors are the differences between the generated trust values by the output layer unit and the desired trust values. We use the sum-of-squares error function [24] to measure the errors. That is, given a training set comprised of P_i together with a corresponding set of desired values d_i, the error function is:

$$E(W) = \frac{1}{2} \sum_{i=1}^{n} |y_i(P_i, W) - d_i|^2.$$

During the training, after the outputs are computed, we use the backpropagation algorithm [26] to minimize the sum-of-squares error in training. In the algorithm, the errors are propagated backward through the neural network to adjust connection weights W. Every time the weights are updated, the approximation errors are decreased. The weight update process continues until the neural network provides satisfying trustworthiness estimations.

After training, a TH can directly use the trained neural network for trust evaluation of its responsible nodes. Given the attribute values of calculated reputation, capacity, longevity, and other additional factors, the neural network can automatically output the node's trust.

The automatic trust model has two distinguishing features. First, the weight parameters determined through training are precise indicators to measure the influence of different factors on trust. Second, the training of the neural network is an adaptive process and the results of trust evaluation are more objective.

2.5 Optimal Server Selection Algorithm

An effective server selection policy is needed for true high-QoS server selection. Our proposed optimal server selection algorithm considers the trust derived by the trust model and the current available capacity and longevity. Recall that in the system, each node periodically reports information on its available capacity and longevity to its TH using `Insert(ID,object)`. When selecting a server from b ($b > 1$) options, a client sends `Lookup(`ID_i`)`$(1 \leq i \leq b)$ to query the trust and V of each server. Each query will reach the TH of each server, where the client can retrieve the trust level, available capacity, and current longevity of the server.

when choosing a server, a client first identifies all servers with sufficient capacity and longevity to meet its needs. It then chooses the nodes from these options that have the highest trust. As the available capacity and longevity are time-varying, the client probes the selected servers and chooses the nodes with sufficient current capacity and longevity. These selected servers have sufficiently high capacity to serve the request, and sufficiently high longevity to complete the requested service before their departure. They can reliably satisfy the client's request. Second, in order to distribute the load among nodes according to their available capacity without overloading a server, lottery scheduling [27] is adopted in the final server selection so that servers owning higher available capacity receive more requests and vice versa. Specifically, a client assigns tickets to the servers according to their available capacity units. A server having e units of available capacity receives e tickets. Then, the client randomly chooses a ticket and selects the server holding this ticket.

Combining the components of the proposed trust system, Algorithm 1 shows the pseudocode for a node's operations in the system.

2.6 Reliable Trust Management

Recall that this proposed system depends on a number of THs to collect information, calculate trust values, and respond to trust queries. However, the trust system's function is interrupted if THs are compromised by malicious nodes and the trust information is tampered with or falsified. The malicious nodes can change their own low trust values to high trust values. Then, the nodes requiring services would be misled by the falsified

Algorithm 1 Pseudocode for node i's operation in the trust system.

```
1:  //Executed by node $i$ in the system
2:  while receive a service from node $j$ do
3:      Evaluate the service
4:      $Insert(ID_j, r_j)$ //Report the local reputation value
5:  end while
6:
7:  //Periodically report its status of additional factors
8:  $Insert(ID_i, V_i)$
9:
10: if it is a trust host then
11:     for each of its responsible node $j \in \mathcal{I}$ do
12:         Calculate node $j$'s global reputation value using existing
            reputation system approach
13:         Calculate node $j$'s trust level using the proposed automatic
            or manual trust model
14:     end for
15:     Report $t_{v_j}$ $(j \in \mathcal{I})$ to its parent in the dynamic tree
16: end if
17:
18: //Server selection from a group of servers $S_k$ $(1 \leq k \leq b)$ for a service
    with required duration $L_s$ and capacity $C_s$
19: Choose servers $\hat{S} \in S$ with $C_a > C_s$ and $L_a > L_s$
20: for each node $j \in \hat{S}$ do
21:     $t_{l_j} = Lookup(ID_j)$ //Query trust level
22: end for
23: Order $t_{l_j}$ $(j \in \hat{S})$ in a descending order
24: Choose the servers with the highest $t_{l_j}$
25: Choose one server using the lottery-scheduling algorithm
```

information and may select low-reputed servers. We can use previously proposed security techniques to prevent this security attack. Here, we introduce a method to reinforce the system's ability to handle this problem. This method uses redundant THs for a node to help ensure the uninterrupted function of the trust system. Specifically, a reporting node applies c consistent hash functions on n_i's IP address to generate c IDs. Then, it uses Insert(ID_{i_k}, R_i) $(1 \leq k \leq c)$ to report n_i's reputation to the c THs of n_i. These THs update n_i's trust level. When a requester queries the trust of node n_i, it also generates c IDs using the c consistent hash functions, and then executes Lookup(ID_k) $(1 \leq k \leq c)$ to retrieve the values. The requester first calculates the average of the c returned values. The TH whose returned trust value greatly deviates from the average is suspected as a compromised or malevolent TH. The node then regards the average value of the trust values from the other THs as n_i's trust value. It reports the potentially malevolent TH to other THs. The suspicious TH is then dismissed by other THs from the trust system and replaced by the highest reputed node in the system. Since the THs are chosen from the highest reputed nodes, we assume that most of the c THs of a node are trustworthy. We will explore a method to handle the case when most of them are malevolent or compromised in our future work.

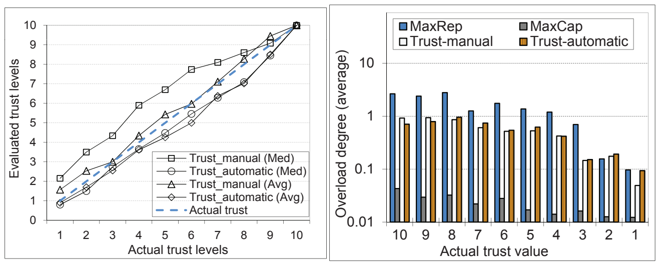
3 Performance Evaluation

We used P2PSim [28] as our testbed for performance evaluation. We built an unstructured P2P network of 1000 nodes, each with 200 randomly chosen neighbors. We implemented the PowerTrust [8] reputation system, upon which we built our proposed trust models. 10 power nodes form a structured P2P for calculating reputation and trust in a distributed manner. Table 1 shows the default experiment parameters. We assume a bounded Pareto distribution for node capacity. This distribution reflects real world situations where machine capacities vary by different orders of magnitude [29]. Node longevity is exponentially distributed [30]. A node joins and leaves the network at the interval of its longevity. We set α in Equation (2) to 1.

Network size	1000 nodes
Reputation evaluation phase	400 min
Transaction phase	1000 min
Num. of transactions	10^6
Request interval	$60s$
Request service time	$[15-60]$s
Reputation/Trust level	$[0, 10]$
Node capacity	Bounded Pareto distribution max=300, min=3, shape=1
Node longevity	Exponential distribution mean=240s

TABLE 1: Default experiment parameters.

From eBay, we randomly chose 100 sellers from each seller group with a final rating in $[0.1x, 0.1x+0.1)(x \in [1, 9])$. We then randomly mapped the 1000 sellers to the 1000 nodes in the P2P. We regarded a node's final rating as its *actual trust level*. A node with actual trust level t has t probability of successfully providing service. In the experiment, each node generated a service request every 60s until the specified testing time duration had elapsed. A request's service time was randomly chosen from $[15, 60]$s. There are two occasions in which a server drops a request: when it does not have sufficient capacity or longevity, and when it is not willing to provide service (determined by its trust). In both occasions, a client gives a reputation of 0 to the server. Otherwise, a client gives a reputation based on the QoS provided by the server according to the method introduced in Section 2.1.



(a) Accuracy of trust evaluation (b) Overload status
Fig. 10: Trust evaluation and node load.

The experiment has two phases: the *reputation evaluation phase* and the *transaction phase*. The reputation evaluation phase is used to build each node's reputation and trust. We normalized the reputation and trust levels to $[0, 10]$ with locality-preserving hashing. In this phase, each node randomly chose one node from other nodes in the system as the server for its requests. In the subsequent transaction phase, a node's neighbors are the server options for its requests. A node selected a server from the options using different server selection policies. Since there is no work specifically for server selection and all reputation systems encourage nodes to choose the highest-reputed server, we compared the performance of our proposed optimal server selection algorithm with *MaxRep* [8–15] and *MaxCap* [31] algorithms, which select the server with the highest reputation and the highest available capacity, respectively. We use *Trust-manual* and *Trust-automatic* to represent our proposed manual and automatic trust models, respectively. We also use them to represent the optimal server selection algorithm using the trust values calculated by the two

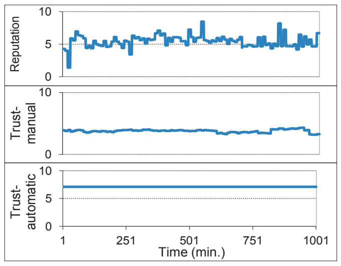


Fig. 11: Reputation/trust over time.

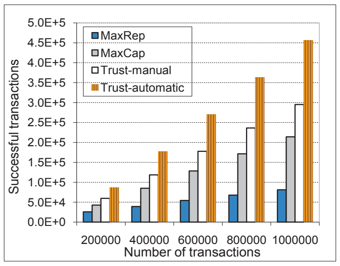


Fig. 12: Number of successful transactions.

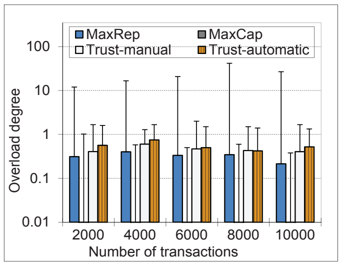


Fig. 13: Overload status.

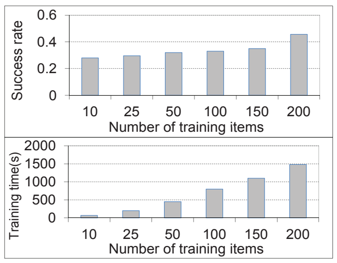


Fig. 14: Success rate / training time vs. amount of training data.

trust models. The final result of each experiment is the average of 3 runs of the experiment. We collected 260 data items from the reputation evaluation phase for the neural network training in *Trust-automatic*. Supplementary material in Section 6 presents additional experimental results for comprehensive performance evaluation.

3.1 Accuracy of Trust Evaluation

Figure 10(a) illustrates the median and average values of the evaluated node trust levels in both *Trust-manual* and *Trust-automatic* versus the actual node trust level. The median and average values of the evaluated trust levels of both models increase approximately linearly as the actual trust level increases, though they sometimes fluctuate slightly. This implies that the evaluated trust level can represent actual node trust level. In *Trust-manual*, the increase rate of the median value slows down when the actual trust level increases from 6 to 9, while the median value of *Trust-automatic* grows relatively more smoothly in the entire range of [1,10]. Also, the median value of *Trust-manual* deviates from the actual trust level more than other values. Both the median and average values of *Trust-automatic* are close to the actual trust levels. The experimental results show that both proposed trust models can generally derive trust levels that reflect actual node trust.

Figure 10(b) shows the average overload degree of each group of nodes with the same actual trust value. We see that nodes with a higher trust level tend to have higher average overload degrees. The overload degree for a node is determined by both the number of received requests and the node's trust. A node with a higher trust level is more likely to receive requests and less likely to drop requests, thus producing a higher overload degree. In *MaxRep*, a client always chooses the server with the highest reputation in server options. *Trust-manual* and *Trust-automatic* choose the highest-trust node that is lightly loaded. Therefore, servers with higher trust levels attract more service requests and produce higher overload degrees. As *MaxRep* is biased toward the highest-reputed node, it does not distribute the load among different groups as evenly as *Trust-manual* and *Trust-automatic* and many of its average overload degrees are greater than 1. *MaxCap* always chooses the server with the highest available capacity. As high-trust nodes have lower probabilities of dropping requests, higher-trust nodes have higher overload degrees than lower-trust nodes in *MaxCap*. Nodes in *MaxCap* have much lower overload degrees than in all other methods because many requests are dropped since *MaxCap* does not consider node trust in server selection.

In addition to overloading the highest-reputed nodes, *MaxRep* would also lead to detrimental *reputation fluctuation*. A node with a high reputation can receive many requests over its available capacity. It is then unable to offer high-QoS, resulting in a low reputation. Later, due to its low reputation, it receives few requests, which enables it to offer high-QoS, resulting in a high reputation value. This cycle occurs repeatedly. To confirm our analysis, we chose a node with high capacity and average longevity and kept track of its reputation and trust levels during the entire transaction phase. Figure 11 shows the node's reputation level with *MaxRep* and trust level with the optimal server selection algorithm over time. As expected, the node's reputation fluctuates dramatically and the change spans the entire time duration. Therefore, reputation is not a stable measurement of nodes' innate trustworthiness. The trust level only changes slightly over time in *Trust-manual* and stays almost constant in *Trust-automatic*. The results show that the trust derived in our proposed methods can accurately reflect nodes' willingness to offer service, and *MaxRep* leads to reputation fluctuation, which impairs the accuracy of trust reflected by reputation.

3.2 Effectiveness of Server Selection Algorithm

Figure 12 shows the number of successful transactions in the *MaxRep*, *MaxCap*, *Trust-manual* and *Trust-automatic* server selection algorithms. We see that in each algorithm, the number of successful transactions increases as the total number of transactions grows. Also, *Trust-manual* and *Trust-automatic* achieve more successful transactions than the others, which confirms the effectiveness of our proposed trust evaluation methods and server selection algorithm. The figure also shows that *Trust-automatic* achieves more successful transactions than *Trust-manual*. This is because the properly trained non-linear neural network can better reflect the joint influence of both capacity and longevity of nodes on reputation. In this experiment, the workload generated by all transactions was much higher than the capacity of the nodes with trust 10 in the system. This is the reason that some transactions cannot be performed. We can see that *MaxRep* generates the least number of successful transactions. This is because *MaxRep* chooses the highest-reputed server regardless of its current available capacity and longevity. The server may be overloaded or leave the system before completing the requested service. Similarly, by biasing the server with the highest available capacity without considering node trust and longevity, *MaxCap* leads to fewer successful transactions. *MaxRep* performs worse than *MaxCap* because a high workload severely overloads the highest-reputed nodes. Though nodes chosen in *MaxCap* may be untrustworthy, they still have a certain probability of accepting requests and providing service with sufficient capacity.

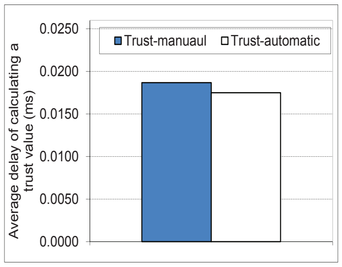


Fig. 15: Delay for trust value calculation.

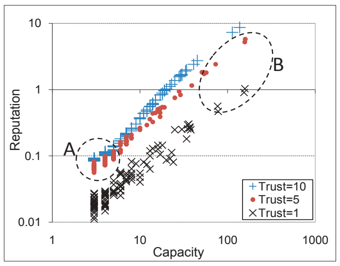


Fig. 16: Reputation vs. capacity.

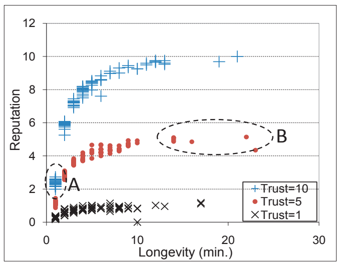


Fig. 17: Reputation vs. longevity.

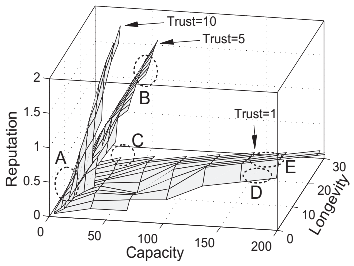


Fig. 18: Reputation vs. capacity & longevity.

We define a node's *overload degree* as the ratio of its workload to its capacity. We measured the highest overload degree that each node in the system has ever experienced during the entire testing, and calculated the 1^{st}, 50^{th}, and 99^{th} percentiles of overload degree for all the methods. Figure 13 shows the experimental results versus the number of transactions. We see that *MaxRep* has a significantly higher 99^{th} percentile overload degree than the others since it does not take into account the available capacity when selecting a server. The highest-reputed servers are chosen by many clients and become very overloaded. In contrast, *MaxCap* produces the least 99^{th} percentile overload degree because it always chooses the server with the highest available capacity. However, it fails to achieve load balance, with a median and 1^{st} percentile of 0 for overload degree. The 99th percentile overload degrees for nodes in *Trust-manual* and *Trust-automatic* stay around 1, avoiding server overload thanks to the proposed optimal server selection algorithm, which selects nodes with high trust and enough capacity and longevity.

3.3 Effectiveness and Efficiency of the Automatic Trust Model

The amount of training data affects the accuracy of the derived trust values and the efficiency of the training process. We define the *success rate* as the percentage of successful transactions out of all transactions. The upper and lower half of Figure 14 shows the success rate and training time versus the amount of training data, respectively. We use the training data to train the neural network repeatedly for 5×10^6 times. By increasing the training times, training error will converge to a small value. From the figure, we observe that the success rate increases as the number of training items increases. This result shows that more training data can reduce errors and better train the neural network for accurate trust derivation. From the lower half of the figure, it can be observed that the training time of the neural network increases almost linearly as the number of training items increases. This is because the training time is directly determined by the number of training items. More training items cause the neuron weights to be adjusted for more times. Combining the success rate result, we can conclude that it is critical to choose a proper amount of training data for high effectiveness and efficiency of training. We also can see that the training time is acceptable even when the amount of training data is large.

Figure 15 shows the average time delay to calculate the trust value of a node in *Trust-manual* and *Trust-automatic*. The time delay stays around 0.018ms. *Trust-manual* needs to calculate the trust value for each server manually. *Trust-automatic* calculates the trust value through a neural network. As a result, *Trust-manual* generates slightly higher time delay than *Trust-automatic*.

3.4 Accuracy of Trust Reflection by Reputation

In order to see the sole impact of capacity on reputation, we first assume that all servers have high enough longevities to complete requested services. Figure 16 shows each node's reputation versus its capacity for three groups of nodes with trust levels 1, 5 and 10, respectively. We observe that node reputation has a linear relationship with node capacity in each group. That is, if two nodes have the same trust level, the higher-capacity node has the higher reputation. This result implies that node capacity affects node reputation, and a node's reputation is not sufficiently accurate to reflect its trust. Directly using reputation to evaluate a node's trust is unfair to low-capacity nodes.

Comparing the different groups, we observe that for equal capacity nodes, higher-trust nodes receive higher reputations. This means that without the impact of additional factors, node reputation can accurately reflect node trustworthiness. Higher-trust nodes offer a higher probability of successful service, thus earning higher reputations. Most importantly, we see that some nodes (e.g., nodes in region A) with high trust and low capacity have low reputations, whereas some nodes (e.g., nodes in region B) with low trust and high capacity have high reputations. This result confirms that node reputation is affected by node capacity. Even though a node has a low willingness to serve requests, because it has a high capacity to satisfy a request when it chooses to serve a request, it still earns a high reputation. In contrast, even though a node has 100% willingness to serve requests, it has to drop some requests when overloaded due to its limited capacity, resulting in low reputation. The phenomenon that low-trust nodes in region B exhibit higher reputations than high-trust nodes in region A shows that under the influence of capacity, reputation cannot directly reflect trust.

Similarly, in order to see the sole impact of longevity on reputation, we assume all servers have sufficient available capacities to handle all requests. Figure 17 shows the relationship between reputation and longevity for the three groups of nodes. Unlike the linear relationship between reputation and capacity, the reputation increases logarithmically with longevity. Recall that as long as the longevities of two nodes are high enough to complete a request, they receive the same reputation regardless of the difference in longevities. Given the same longevity, higher-trust nodes receive higher reputations due to the same reason as in Figure 16.

Figure 18 shows the joint effect of capacity and longevity on the reputation of three groups of nodes with trust levels 1, 5 and 10, respectively. We can see that the reputation results of each of the three node groups form into one layer. For the nodes with the same capacity or longevity, higher-trust nodes receive a higher reputation. We can also observe that some nodes with high trust (e.g., nodes in region A) have a low reputation, and some nodes with relatively low trust (e.g., nodes in region B) gain a high reputation. These results are consistent with the results in Figures 16 and 17. The results confirm that reputation is influenced by capacity and longevity, and cannot directly reflect node trust.

Moreover, for nodes with the same trust in each layer, nodes with high capacity but low longevity (e.g., nodes in region D) do not obtain reputations as high as the nodes with the same capacity but higher longevity (e.g., nodes in region E). This is because low-longevity nodes have a high probability of leaving the system before completing a requested service, thus earning low reputations. It is interesting to see that nodes with the lowest capacity but the highest longevity (e.g., nodes in region C) have lower reputations than nodes with the highest capacity but the lowest longevity (e.g., nodes in region D). This is because nodes with the lowest capacity and the highest longevity drop most of the requests due to insufficient capacity, which causes them to receive low reputations most of the time. The nodes with the highest capacity and lowest longevity can partially process some requests, which helps them obtain a relatively higher reputation.

4 Related Work

One group of recently proposed reputation systems focus on improving the scalability and reputation accuracy in P2P networks. Zhou and Hwang [8] proposed *PowerTrust*, which dynamically selects some most reputable power nodes to collect feedback to improve aggregation speed. The authors also proposed *GossipTrust* [32], which adapts to peer dynamics and is robust to disturbance from malicious peers by resorting to a gossip protocol and leveraging power nodes. Song *et al.* [12] presented a P2P reputation system based on fuzzy logic inferences, which can better handle uncertainty, fuzziness, and incomplete information in peer trust reports.

Liu [9] presented *PeerTrust*, which includes a coherent adaptive trust model for quantifying and comparing the trustworthiness of nodes based on a transaction-based feedback system and a decentralized implementation of such a model over a structured P2P. Kamvar *et al.* [10] proposed a distributed and secure method to compute global trust values based on Power iteration. Zhang *et al.* [33] developed a trust-incentive resource management framework that integrates values of prices, trust, and incentives, and a weighted voting scheme to secure the grid system by declining join requests from malicious nodes. Zhang and Fang [34] presented a reputation system built upon the multivariate Bayesian inference theory for reliable service selection. The system offers a theoretically sound basis for clients to predict the reliability of candidate servers along with a fine-grained QoS differentiation method to satisfy the diverse QoS needs of individual nodes. Wang and Vassileva [14] pointed out that trust is multi-faceted, and nodes need to develop differentiated trust in different aspects of nodes capability. They proposed a flexible method to present differentiated trust and to combine different aspects of trust.

Sonnek *et al.* [35] presented a model in which reliability is not a binary property but a statistical one based on a node's prior performance and behavior. The authors used this model to construct several reputation-based scheduling algorithms that employ estimated reliability ratings of nodes for efficient task allocation. Piatek *et al.* [36] proposed a one hop reputation protocol for P2P networks, which limits propagation to at most one intermediary. This protocol improves performance and incentives compared to when contribution information is globally visible and tit-for-tat in BitTorrent. In order to motivate peers to cooperate, Satsiou and Tassiulas [37] proposed a distributed reputation-based system according to which peers earn reputations analogous to their contributions. In this way, each node has to trade off the capacity it will dedicate for serving other nodes in order to increase its reputation for receiving services.

Zhang *et al.* [38] proposed a trust model called SFTrust based on a double trust metric: service providing and feedback. Compared with the single trust metric model, SFTrust can take full advantages of all the peers' service abilities for high performance. Gutowska and Buckley [39] proposed an improved distributed agent-based reputation mechanism which contributes to measuring the reputations of online providers. The system considers a number of the parameters that have a bearing on trust and reputation. Credential chains are needed in trusted P2P applications, where trust delegation must be established between each pair of peers at a specific role level. Chen *et al.* [40] proposed a heuristic-weighting approach for selecting the most likely path to construct a role-based trust chain. Dewan and Dasgupta [41] proposed a cryptographic protocol for ensuring secure and timely availability of the reputation data of a peer to other peers at extremely low cost. The cryptographic protocol is coupled with self-certification and cryptographic mechanisms for identity management and countering Sybil attacks. Our previous work [42] proposed SocialTrust, which adaptively adjusts the weight of ratings based on the social distance and interest relationship between peers to enhance the capability of reputation systems in combating collusion.

In spite of these efforts, there has been no research devoted to finding an accurate reflection of node trustworthiness by removing the influence of node capacity and longevity. Our proposed methods can complement the existing reputation systems for more accurate node trustworthiness calculation and for true high-QoS server selection.

5 Conclusions

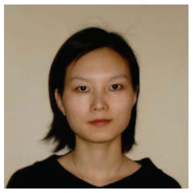
In this paper, we studied the relationship between reputation and trust, and found that in addition to node trust, factors such as capacity and longevity also influence node reputation. Since a node's available capacity and longevity are heterogeneous and time-varying, reputation cannot provide accurate guidance for nodes in choosing a high-QoS server. Thus, we propose a manual trust model and an automatic trust model to accurately derive node trust by removing the influence of additional factors on reputation. To enable nodes to choose high-QoS servers, we further propose an optimal server selection algorithm. It considers node trust and the current values of additional factors to ensure successful and efficient transactions. We also propose a method to handle compromise attacks on trust hosts for reliable trust management. Experimental results confirm the superior performance of our proposed approaches. In our future work, we will explore other additional factors and study their influence on node reputation, and explore methods for measuring node longevity and capacity.

Acknowledgment

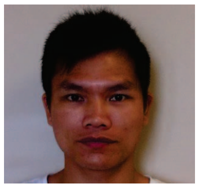
This research was supported in part by U.S. NSF grants CNS-1249603, OCI-1064230, CNS-1049947, CNS-1156875, CNS-0917056 and CNS-1057530, CNS-1025652, CNS-0938189, CSR-2008826, CSR-2008827, Microsoft Research Faculty Fellowship 8300751, and U.S. Department of Energy's Oak Ridge National Laboratory including the Extreme Scale Systems Center located at ORNL and DoD 4000111689. An early version of this work was presented in the Proceedings of ICCCN'11 [43].

References

[1] Kazaa, 2001. http://www.kazaa.com.
[2] Bittorrent sites. http://www.bittorrent.com/.
[3] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable Peer-to-Peer Lookup Protocol for Internet Applications. *TON*, 2003.
[4] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-scale Peer-to-Peer Systems. In *Proc. of Middleware*, pages 329–350, 2001.
[5] Skype. http://www.skype.com/.
[6] X. Wang, Z. Yao, Y. Zhang, and D. Loguinov. Enhancing application-layer multicast for p2p conferencing. In *Proc. of CCNC*, 2007.
[7] H. Shen, Z. Li, T. Li, and Y. Zhu. PIRD: P2P-Based Intelligent Resource Discovery in Internet-Based Distributed Systems. In *Proc. of ICDCS*, 2008.
[8] R. Zhou and K. Hwang. Powertrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing. *TPDS*, 2007.
[9] L. Xiong and L. Liu. Peertrust: Supporting Reputation-based Trust for Peer-to-Peer Electronic Communities. *TKDE*, 2004.
[10] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust Algorithm for Reputation Management in P2P Networks. In *Proc. of WWW*, 2003.
[11] A. Singh and L. Liu. TrustMe: Anonymous Management of Trust Relationships in Decentralized P2P Systems. In *Proc. of P2P*, 2003.
[12] S. Song, K. Hwang, R. Zhou, and Y. K. Kwok. Trusted P2P Transactions with Fuzzy Reputation Aggregation. *IEEE Internet Computing*, 2005.
[13] M. Srivatsa, L. Xiong, and L. Liu. Trustguard: Countering Vulnerabilities in Reputation Management for Decentralized Overlay Networks. In *Proc. of WWW*, 2005.
[14] Y. Wang and J. Vassileva. Trust and Reputation Model in Peer-to-Peer Networks. In *Proc. of P2P*, 2003.
[15] K. Aberer and Z. Despotovic. Managing Trust in a Peer-2-Peer Information System. In *Proc. of CIKM*, pages 310–317, 2001.
[16] ebay. http://www.ebay.com.
[17] Amazon. http://www.amazon.com/.
[18] P. Brighten Godfrey and I. Stoica. Heterogeneity and Load Balance in Distributed Hash Tables. In *Proc. of INFOCOM*, 2005.
[19] S. Ran. A model for web services discovery with qos. *ACM SIGecom Exchanges*, 4(1):1–10, 2003.
[20] Q. Feng, Y. Yang, Y. L. Sun, and Y. Dai. Modeling Attack Behaviors in Rating Systems. In *Proc. of ICDCS*, 2008.
[21] T. G. Papaioannou and G. D. Stamoulis. Achieving Honest Ratings with Reputation-Based Fines in Electronic Markets. In *Proc. of INFOCOM*, 2008.
[22] M. Cai, M. Frank, and P. Szekely. MAAN: A multi-attribute addressable network for grid information services. *JGC*, 2004.
[23] Y. Zhu and Y. Hu. Efficient, Proximity-Aware Load Balancing for DHT-Based P2P Systems. *IEEE TPDS*, 16(4), 2005.
[24] C. M. Bishop. Pattern Recognition and Machine Learning . *Information Science and statistics, Springer*, 2006.
[25] K. Hornik, M. Stinchcombe, and H. White. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, 1989.
[26] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
[27] C. Waldspurger and W. Weihl. Lottery Scheduling: Flexible Proportional-Share Resource Management. In *Proc. of OSDI*, 1994.
[28] p2psim. http://pdos.csail.mit.edu/p2psim/.
[29] H. Shen and C. Xu. Locality-aware and churn-resilient load balancing algorithms in structured peer-to-peer networks. *TPDS*, 2007.
[30] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the Evolution of Peer-to-Peer Systems. In *Proc. of PODC*, 2002.
[31] H. Shen and C. Xu. Elastic routing table with provable performance for congestion control in dht networks. *TPDS*, 2009.
[32] R. Zhou, K. Huang, and M. Cai. GossipTrust for Fast Reputation Aggregation in Peer-To-Peer Networks. *TKDE*, 2008.
[33] Y. Zhang, J. Huai, Y. Liu, L. Lin, and B. Yang. A Framework to Provide Trust and Incentive in CROWN Grid for Dynamic Resource Management. In *Proc. of ICCCN*, 2006.
[34] Y. Zhang and Y. Fang. A Fine-Grained Reputation System for Reliable Service Selection in Peer-to-Peer Networks. *TPDS*, 2007.
[35] J. Sonnek, A. Chandra, and J. B. Weissman. Adaptive Reputation-Based Scheduling On Unreliable Distributed Infrastructures. *TPDS*, 2007.
[36] M. Piatek, T. Isdal, A. Krishnamurthy, and T. Anderson. One Hop Reputations For Peer To Peer File Sharing Workloads. In *Proc. of NSDI*, 2008.
[37] A. Satsiou and L. Tassiulas. Reputation-Based Resource Allocation In P2P Systems of Rational Users. *TPDS*, 2010.
[38] Y. Zhang, S. Chen, and G. Yang. SFTrust: A Double Trust Metric Based Trust Model in Unstructured P2P System. In *Proc. of IPDPS*, 2009.
[39] A. Gutowska and K. Buckley. Computing Reputation Metric in Multi-Agent E-Commerce Reputation System. In *Proc. of ICDCS*, 2008.
[40] K. Chen, K. Hwang, and G. Chen. Heuristic Discovery of Role-Based Trust Chains in Peer-to-Peer Networks. *TPDS*, 2009.
[41] P. Dewan and P. Dasgupta. P2P Reputation Management Using Distributed Identities and Decentralized Recommendation Chains. *TKDE*, 22(7):1000–1013, 2010.
[42] Z. Li, H. Shen, and K. Sapra. Leveraging Social Networks to Combat Collusion in Reputation Systems for Peer-to-Peer Networks. In *Proc. of IPDPS*, 2011.
[43] H. Shen and L. Zhao. Refining Reputation to Truly Select High-QoS Servers in Peer-to-Peer Networks. In *Proc. of ICCCN*, 2011.



Haiying Shen Haiying Shen received the BS degree in Computer Science and Engineering from Tongji University, China in 2000, and the MS and Ph.D. degrees in Computer Engineering from Wayne State University in 2004 and 2006, respectively. She is currently an Assistant Professor in the Department of Electrical and Computer Engineering at Clemson University. Her research interests include distributed computer systems and computer networks, with an emphasis on P2P and content delivery networks, mobile computing, wireless sensor networks, and grid and cloud computing. She was the Program Co-Chair for a number of international conferences and member of the Program Committees of many leading conferences. She is a Microsoft Faculty Fellow of 2010 and a member of the IEEE and ACM.



Yuhua Lin Yuhua Lin received both his BS degree in Software Engineering and MS degree in Computer science from Sun Yat-sen University, China in 2009 and 2012 respectively. He is currently a Ph.D student in the Department of Electrical and Computer Engineering of Clemson University. His research interests include social networks and reputation systems.

Ze Li Ze Li received the BS degree in Electronics and Information Engineering from Huazhong University of Science and Technology, China, in 2007. He is currently a Ph.D. student in the Department of Electrical and Computer Engineering of Clemson University. His research interests include distributed networks, with an emphasis on peer-to-peer and content delivery networks, wireless multi-hop cellular networks, game theory and data mining. He is a student member of IEEE.