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

Haiying Shen and Lianyu Zhao
Department of Electrical and Computer Engineering
Clemson University, Clemson, SC 29631
{shenh, lianyuz}@clemson.edu

***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. Reputation represents the opinion 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 the heterogeneous and time-varying attributes of these factors, reputation actually 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 accurate 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 and an automatic trust model that removes 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. Simulation results demonstrate the effectiveness of the trust models in accurate node trust reflection. Moreover, the server selection algorithm dramatically increases the success rate of service requests and avoids overloading nodes.**

I. 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], [6], [9], [13]. Node cooperation is critical to achieving reliable performance of P2Ps. However, it is faced with a challenge posed by the users in the P2Ps, where many diverse and autonomous parties without preexisting trust relationships work together. A selfish node may not be willing to provide resources if it won't receive benefits in return.

Reputation systems are a main method used to tackle this problem. The method collects, distributes and aggregates feedback about participants' past behavior and evaluates a node's trustworthiness (trust and trustworthiness are interchangeable terms in this paper) to help nodes decide whom to trust, encourage cooperative and honest behavior, and deter uncooperative and dishonest participants. The abilities of accurately reflecting node trust and providing the right guidance for server selection are vital to the effectiveness of a reputation system. By right guidance, we mean the selected servers under the guidance truly provide high QoS.

However, existing reputation systems [5], [10]–[12], [15], [16], [20] fall short in these abilities mainly due to two reasons. First, they directly regard node reputation as trust, which we claim is not appropriate in general. Second, they guide a node to select the server with the highest reputation, which may not actually select a high-QoS server and would overload the highest-reputed nodes. The details are in below.

Reputation cannot directly reflect trust.

- *Reputation* represents the opinion formed by other nodes about a node's QoS behavior.
- *Trust* is essentially an assessment of a node's honesty and willingness to be cooperative.

A node's reputation is measured by its reputation value calculated in the reputation system. In addition to node trust, node reputation is determined by factors such as node capacity and longevity (i.e., service length or lifetime). Due to the *heterogeneous* and *time-varying* attributes of these factors, reputation cannot directly reflect a node's trust. Heterogeneity means different nodes have different capacities and longevities. Time variance means the available capacity and longevity of a node vary over time. A node's available longevity is the time it will stay in the network. Altruistic but low-longevity, low-capacity or overloaded nodes may have low reputation values.

Highest-reputed servers may not provide high QoS. Though no explicit server selection method has been proposed, current reputation systems often guide server selection policy to select the highest-reputed node. This cannot provide the right guidance for choosing high-QoS servers due to the time-varying attribute of the capacity and longevity. Since nodes prefer to choose high-reputed nodes as servers, a high-reputed node that had significant available capacity to handle a request in the past does not necessarily mean it currently has sufficient capacity for a request. The server selected by this selection policy may not offer high QoS.

In addition, this policy would lead to detrimental *reputation fluctuation*. Because of a node's high reputation, it receives 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 repeatedly occurs.

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 the additional factors through experiments and analysis, and observe that the additional factors positively impact node reputation.

(2) Based on the observation, we develop two trust models (one manual and one automatic) that remove the impact of additional factors on the reputation in order to derive accurate node trust.

(3) We develop an optimal server selection algorithm that separately considers node trust and the current values of additional factors to ensure high QoS. 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 the reputation and providing the accurate guidance for high-QoS server selection.

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

II. Related Work

One group of recently proposed reputation systems focuses on improving the scalability and accuracy of reputation by relying on structured P2Ps to collect locally-generated peer feedbacks and calculate the global reputation scores. Zhou and Hwang [20] proposed *PowerTrust*, which dynamically selects a small number of the most reputable power nodes to collect feedback in order to improve aggregation speed. The authors also proposed *GossipTrust* [19], which adapts to peer dynamics and is robust to disturbance from malicious peers by resorting to a gossip protocol and leveraging the power nodes. Song *et al.* [11] presented a P2P reputation system based on fuzzy logic inferences, which can better handle uncertainty, fuzziness, and incomplete information in peer trust reports.

Liu [16] 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.* [5] proposed a distributed and secure method to compute global trust values based on Power iteration. Zhang *et al.* [18] developed a trust-incentive resource management framework that integrates values of prices, trust, and incentive, and a weighted voting scheme to secure the grid system by declining join requests from malicious nodes. Zhang and Fang [17] presented a reputation system built upon the multivariate Bayesian inference theory for reliable service selection.

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

III. Reputation-based Trust Models

A. Study of Reputation and Trust

Untrustworthy nodes are selfish or dishonest nodes that are not willing to offer service, while trustworthy nodes are altruistic or honest nodes that are willing to supply high QoS. As explained previously, a node's reputation cannot reflect its trust and its current QoS. We use experiments to study the relationship of reputation with capacity and longevity. The experiment parameters and settings are presented in Section IV. In this experiment, all nodes are trustworthy with a 100% probability of serving requests, and they receive approximately the same number of requests. 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.

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 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 1 shows the reputation of each node versus its capacity. We can see that the reputation increases linearly as capacity grows. That is, for fully trustworthy nodes, the higher-capacity node has a higher reputation. These 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's available capacity is high enough to complete the services requested from it. In this case, a server's reputation should be evaluated according to the percent of the requested services it has completed. Thus, the reputation value r_l equals

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

where l_a is the available longevity of a server and l_r is the time requirement of a requested service. Figure 2 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. 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 factors of capacity and longevity. In this case, the reputation value given is $r = r_c \cdot r_l$. Figure 3 illustrates the joint effect of both factors on reputation. The series of nodes indicated with $A - J$ are the nodes with the same

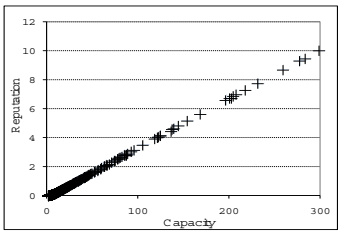


Fig. 1. Reputation vs. capacity.

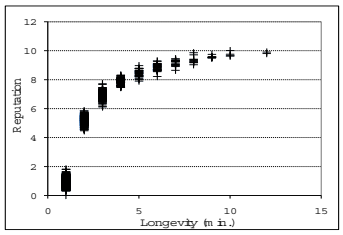


Fig. 2. Reputation vs. longevity.

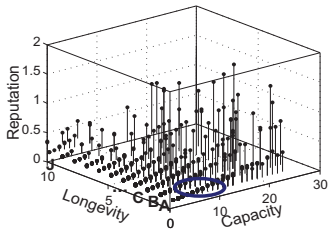


Fig. 3. Reputation vs. longevity & capacity.

longevity. We make a number of observations. First, for the nodes with the same longevity, those with a higher capacity have a higher reputation. For example, in each of the series indicated by $A - J$, nodes with a larger capacity have a higher reputation. Second, a node's low longevity significantly reduces its reputation. Comparing nodes with capacity 10 in series A and B in the circled region, we see that the nodes with higher longevity have higher reputations than those with lower longevity. The reason is that 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 experiment results confirm that a reputation value itself cannot directly reflect node trust along. It is also affected by capacity and longevity. Since a node's available capacity and longevity constantly change, 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.

B. Manual Trust Model

According to the experimental results, we regard nodes whose reputations are high corresponding 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 1, we know that reputation has a 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 4 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 2, 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 6. Hence, reputation has a linear relationship with $loglog(l+1)$. Similar to capacity, depending on a coordinate as in Figure 4, 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:

$$u = \alpha \times (c \cdot loglog(l+1)),$$

where α is a constant factor. 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. A reputation system usually uses a number of nodes, called trust hosts, to collect reputation feedbacks and calculate the reputation values of other nodes. A node's trust host refers to the one collecting and calculating its reputation value. Each node periodically reports information on its available capacity and longevity to its trust host. Each trust host builds a trust model as shown in Figure 5. 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 locality-preserving hashing [3] to normalize the trust value in order to get trust level t_l. That is, $t_l = m \cdot t_v/(\max(t_v) - \min(t_v))$, where m is the number of trust levels in the system, $\max(t_v)$ and $\min(t_v)$ are the maximum and minimum values of t_v in the system, respectively. In order to get $\max(t_v)$ and $\min(t_v)$, trust hosts form a d-nary tree [21] 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 these values, each trust host calculates the trust levels of its nodes.

In Figure 5, 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

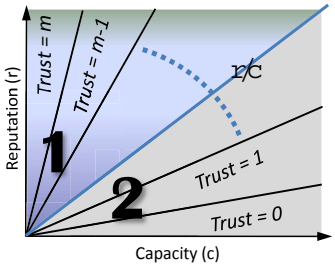


Fig. 4. Capacity-based trust.

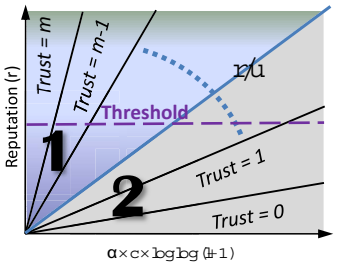


Fig. 5. Capacity & longevity-based trust.

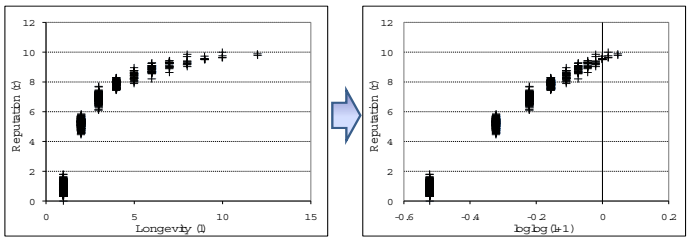


Fig. 6. Reputation vs. longevity and $\log\log(l+1)$.

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 trust host first locates the node's position in the two-dimensional trust model based on its reputation and spatial/temporal value. The level where the node locates is its trust level.

C. Automatic Trust Model

The proposed automatic trust model uses a neural network technique [2] for node trust evaluation by catching the non-linear relationship between reputation, trust, and additional factors including capacity and longevity.

We build a neural network with one layer of hidden units [4] as shown in Figure 7. 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 $W(w_{ji} : 1 \le i \le n, 1 \le j \le k)$. 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 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 their desired trust to train the neural network to learn how to evaluate a node's trustworthiness based on its attribute variables. The

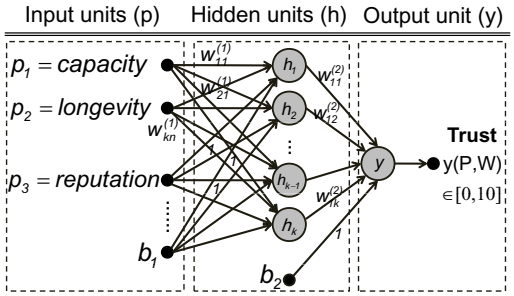


Fig. 7. Neural network based trust model.

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. After training, a trust host 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 of a node, the neural network can automatically output the node's trust.

D. Optimal Server Selection Algorithm

Our proposed optimal server selection algorithm considers the trust derived by the trust model and the current available capacity and longevity. When selecting a server from b ($b > 1$) options, a client queries the trust and the information of available capacity and longevity of each server from its trust host. 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. 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 [14] 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.

IV. Performance Evaluation

We used P2PSim [8] as our testbed for performance evaluation. We built an unstructured P2P network with 1000 nodes and each node has 200 randomly chosen neighbors. We implemented the PowerTrust [20] reputation system, upon which we built our proposed trust models. Ten power nodes formed a structured P2P for calculating reputation and trust in a distributed manner. Table I shows the default experiment parameters. Node longevity is exponentially distributed [7]. A node joins and leaves the network at the interval of its longevity.

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

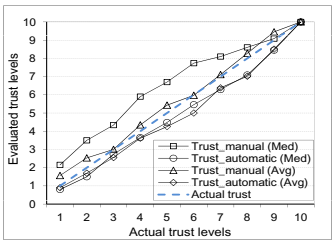
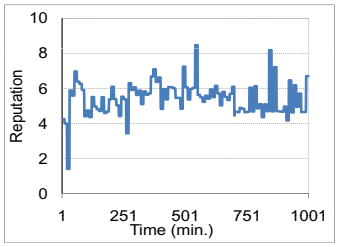
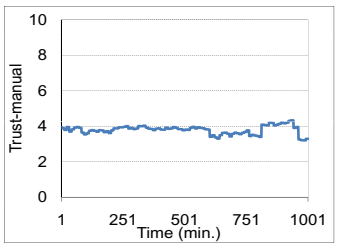


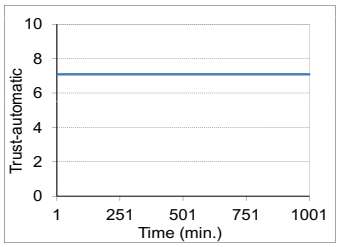
Fig. 8. Accuracy of trust evaluation.



(a) MaxRep



(b) Trust-manual



(c) Trust-automatic

Fig. 9. Reputation/trust over time.

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 I
DEFAULT EXPERIMENT PARAMETERS.

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 is overloaded 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 III-A.

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 $[0, 10]$ by locality-preserving hashing. In this phase, each node randomly chose one node from other nodes in the system as the server for its request. 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. We compared the performance of our proposed optimal server selection algorithm with *MaxRep* and *MaxCap* algorithms, which select the server with the highest reputation and 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 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*.

A. Accuracy of Trust Evaluation

Figure 8 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.

We indicated that *MaxRep* would lead to node reputation fluctuations. 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 9 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 trust. The trust level only changes slightly over time in *Trust-manual* and remains 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 reflection by reputation.

B. Effectiveness of Server Selection Algorithm

Figure 10 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 algorithm considers node trust, current available capacity, and longevity to ensure a high transaction success rate. The figure also shows that *Trust-automatic* achieves more successful transactions than *Trust-manual*. This is because when the non-linear neural network is trained properly, it can better reflect the joint influence of both capacity and longevity of nodes on

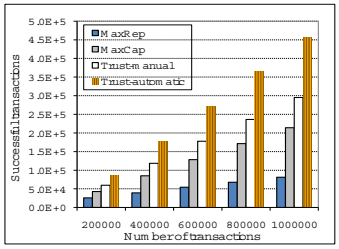


Fig. 10. # of successful transactions.

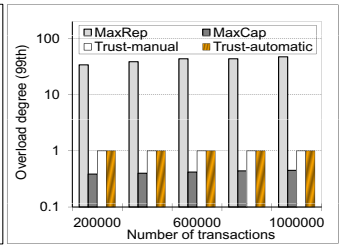


Fig. 11. Overload status.

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. Therefore, 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. Also, *MaxRep* cannot make full use of other trustworthy resources in the system. Similarly, by biasing the server with the highest available capacity without considering node trust and longevity, *MaxCap* leads to less 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.

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. Figure 11 shows the 99^{th} percentile overload degree in the highest degrees versus the number of transactions. From the figure, we see that *MaxRep* has a significantly higher overload degree than others since it does not take into account the available capacity when selecting a server. In contrast, *MaxCap* produces the least 99^{th} percentile overload degree because it always chooses the server with the highest available capacity. However, as illustrated in Figure 10, without considering node trust, it cannot guarantee the success of transactions and cannot fully utilize the resources of trustworthy nodes. *Trust-manual* and *Trust-automatic* maintain a 99^{th} percentile overload degree no more than 1 and avoid overloading the servers thanks to the proposed optimal server selection algorithm, which selects nodes with high trust and enough capacity and longevity.

V. 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 the right 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. In the manual trust model, we manually find the relationship between reputation, trust, and additional factors for trust derivation. In the automatic trust model, we use a neural network for trust derivation. To provide correct guidance for nodes to choose high-QoS servers, we further propose an optimal server selection algorithm. It considers node trust and currently available values of the additional factors to ensure successful and efficient transactions. Experimental results confirm that reputation cannot directly reflect node trust and the superior performance of our proposed approaches. In our future work, we will further explore other additional factors and study their influence on node reputation. We will also explore methods for measuring node longevity and capacity.

Acknowledgment

The authors are grateful to Mr. Ze Li for his help on the work of neural network. This research was supported in part by U.S. NSF grants OCI-1064230, CNS-1049947, CNS-1025652, CNS-1025649, CNS-1057530 and CNS-0917056, Microsoft Research Faculty Fellowship 8300751, and Sandia National Laboratories grant 10002282.

References

[1] Bittorrent sites. http://www.bittorrent.com/.
[2] C. M. Bishop. Pattern Recognition and Machine Learning . *Information Science and statistics, Springer*, 2006.
[3] M. Cai, M. Frank, and P. Szekely. MAAN: A multi-attribute addressable network for grid information services. *JGC*, 2:3–14, 2004.
[4] K. Hornik, M. Stinchcombe, and H. White. Multilayer Feed-forward Networks are Universal Approximators. *Neural Networks*, 1989.
[5] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust Algorithm for Reputation Management in P2P Networks. In *Proc. of WWW*, 2003.
[6] Kazaa, 2001. http://www.kazaa.com.
[7] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the Evolution of Peer-to-Peer Systems. In *Proc. of PODC*, 2002.
[8] p2psim. http://pdos.csail.mit.edu/p2psim/.
[9] 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.
[10] A. Singh and L. Liu. TrustMe: Anonymous Management of Trust Relationships in Decentralized P2P Systems. In *Proc. of P2P*, 2003.
[11] S. Song, K. Hwang, R. Zhou, and Y. K. Kwok. Trusted P2P Transactions with Fuzzy Reputation Aggregation. *IEEE Internet Computing*, 2005.
[12] M. Srivatsa, L. Xiong, and L. Liu. Trustguard: Countering Vulnerabilities in Reputation Management for Decentralized Overlay Networks. In *Proc. of WWW*, 2005.
[13] 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.
[14] C. Waldspurger and W. Weihl. Lottery Scheduling: Flexible Proportional-Share Resource Management. In *Proc. of OSDI*, 1994.
[15] Y. Wang and J. Vassileva. Trust and Reputation Model in Peer-to-Peer Networks. In *Proc. of P2P*, 2003.
[16] L. Xiong and L. Liu. Peertrust: Supporting Reputation-based Trust for Peer-to-Peer Electronic Communities. *TKDE*, 16(7):843–857, 2004.
[17] Y. Zhang and Y. Fang. A Fine-Grained Reputation System for Reliable Service Selection in Peer-to-Peer Networks. *IEEE TPDS*, 2007.
[18] 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.
[19] R. Zhou and K. Hwang. Gossip-based Reputation Aggregation for Unstructured Peer-to-Peer Networks. In *Proc. of IPDPS*, 2007.
[20] R. Zhou and K. Hwang. Powertrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing. *TPDS*, 2007.
[21] Y. Zhu and Y. Hu. Efficient, Proximity-Aware Load Balancing for DHT-Based P2P Systems. *IEEE TPDS*, 16(4), 2005.