FairTrust: Toward Secure and High Performance P2P Networks

Haiying (Helen) Shen
Dept. of Computer Science and Engineering
University of Arkansas, Fayetteville, AR 72701
hshen@uark.edu

Yingwu Zhu
Dept. of Computer Science & Software Engineering
Seattle University, Seattle, WA 98122
zhuy@seattleu.edu

Abstract

Peer-to-Peer (P2P) networks are becoming increasingly popular and are an exciting new class of innovative, internet-based resource sharing systems. In a P2P network, a reputation system is essential to evaluate the trustworthiness of participating peers and to combat the selfish, dishonest, and malicious peer behaviors. The system collects locally-generated peer feedbacks and aggregates them to yield the global reputation values to represent peer trustworthiness. Reputation system guides peers in choosing a server for services/resources. Most of the approaches for peers to exploit reputation metrics are to select the one with the highest reputation value as a providing server. However, it may lead to unexpectedly low efficiency for high-reputed peers, and prevents P2P networks from taking full advantage of all resources for high performance. Other approaches restrict a peer to select a server in the same or lower reputation values. However, these approaches prevent P2P networks from achieving their goal of widely resource sharing and service exchanges. In this paper, we introduce a trust-based fairness-oriented server selection policy, FairTrust, for a peer to choose anther peer to interact. FairTrust takes into account both reputation and capacity factors in server selection. It helps to create a secure P2P communication environment, and meanwhile to take full advantage of system resources for high performance by fair load distribution. Simulation results show the superiority of the FairTrust policy in achieving both high security and high performance in P2P networks in comparison with other related policies.

1 Introduction

Over the last a few year, the immense popularity of P2P networks has produced a significant stimulus to the tremendous advance of distributed P2P applications, such as data sharing, computational grids, navigation systems, multi-party video conferencing, multimedia and telecommunications. P2P networks interconnect computers, clusters, storage systems to make possible the sharing of existing resources, such as CPU time, storage, network bandwidth, equipment, data, software applications. Currently, among over 1 billion documented Internet global users [14], the average simultaneous users of P2P resource sharing application is already over 9 million [1]. P2P resource sharing applications have evolved to the major traffic sources, approximately 80%, in the Internet [13]. The BitTorrent distributed file sharing system [2] alone constitutes roughly 35% of all traffic on the Internet. The eDonkey2000 P2P file sharing system [4] and its derivatives [3, 5] are causing high amounts of traffic volume [7].

In spite of their scalability and reliability, ubiquitous users in the system have posed a challenge of security to P2P networks, where many diverse, autonomous parties without preexisting trust relationships wish to pool their resources. Many technologies have been proposed to prevent, detect and combat malicious attacks. One technology is reputation system. A reputation system collects, distributes, and aggregates feedback about participants' past behavior to help peers decide whom to trust, encourage trustworthy behavior, and deter participation by those who are unskilled or dishonest. Some reputation systems use a centralized authority to maintain and distribute node reputation values, while others operate in a fully distributed manner for high scalability [6, 8, 11, 17, 23, 24, 27, 28, 30] . In P2P networks, the reputation systems provide incentive to nodes to contribute their resources, and help nodes to communicate to others in a secure manner. In a P2P system, a node needs to choose a node (server) from a candidate pool for a service such as forwarding a query or providing a file. In a reputation system, a straightforward approach for nodes to exploit the reputation metrics is to select the node with the highest reputation value as a providing server. Biasing high-reputed nodes may overload them and lead to poor system performance. Efficiency should be an important factor to consider in the server selection process given the fact that P2P file downloading constitutes the major traffic source of Internet. An effective server selection policy needs to avoid generating bottlenecks for efficient file downloading. In addition, biasing policy prevents a P2P system from taking advantage of all resources in the system to achieve system-wide high performance.

Current researches on reputation system mainly focus on accurate reflection of node trustworthiness, and high scalability of reputation system. However, even the accurate calculation of reputation values offered by high scalable reputation system may not be adequate as a mechanism to provide the right incentives for nodes to choose servers, and to improve the achievable efficiency of P2P systems. Reputation system needs to be complemented by an unbiased trust-based node selection policy for secure and efficient node communication. To address this problem, Ranganathan [21] proposed to restrict nodes to download files only from others with lower or equal reputation levels, and Papaioannou [25] proposed to restrict nodes to communicate with other nodes at the same reputation levels. Restricting the eligibility of a node to interact with others prevents nodes from sharing resources freely, which is one ultimate goal of P2P resource sharing systems. In this paper, we propose FairTrust, a trust-based fairness-oriented server selection policy to guide nodes in choosing trustworthy nodes while avoiding bottlenecks. FairTrust policy takes into account both reputation and capacity factors in node selection. It helps to create an secure P2P communication environment, and meanwhile provides flexibility in resource sharing between nodes. Furthermore, it facilitates to take full advantage of system resources for high performance of P2P systems.

The remainder of this paper is structured as follows. Section 2 presents a concise review of related work. Section 3 introduced the FairTrust server selection policy. Section 4 analyzes the performance of FairTrust policy with comparison of other related policies using a variety of metrics. Section 5 concludes this paper with remarks on possible future work.

2 Related Work

Because of the openness and decentralization features of P2P networks, it is usually not desirable to constrain the membership of the nodes in the system. As a result, P2P distributed applications are faced with severe threats such as denial-of-service, masquerading and tampering. In the past a few years, a variety of reputation systems with different characteristics have been proposed in order to provide a secure environment for the P2P applications. Most work on reputation systems can be categorized into two groups: those for scalability without central control [6, 8, 16, 27, 30] and those for accurate reflection of node trustworthiness [8, 16, 26, 27]. Aberer and Despotovic [6] presented scalable data structures and algorithms that require no central control for high scalability reputation system. Exclusively relying on a node's direct observations cannot make use of all the information available. To address the problem, Buchegger and Boudec [8] proposed a fully distributed reputation system that can cope with falsely disseminated information. Kamvar *et al.* [16] presented a distributed and secure method to compute global trust values based on Power iteration. Xiong and Liu [27] presented a reputation-based trust supporting framework, PeerTrust. It 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 network. Papaioannou and Jsang [20, 15] studied the effect of reputation system in trading market environment, while Dingledine *et al.* [10] studied reputation system in decentralized anonymity systems. These studies all confirmed the benefits brought about by reputation systems. Gupta *et al.* [12] investigated the design of a reputation system for decentralized unstructured P2P networks like Gnutella. Zhou and Hwang [30] proposed PowerTrust, a robust and scalable reputation system for trusted P2P. PowerTrust significantly improves global reputation accuracy and aggregation speed by using look-ahead random walk strategy and leveraging the power nodes. It is also adaptable to dynamics in peer joining and leaving and robust to disturbance by malicious peers. Zhang *et al.* [29] developed a trust-incentive resource management (TIM) framework in CROWN Grid for dynamic resource management. TIM integrates values of prices, trust, and incentive and integrates weighted voting scheme to secure the grid system by declining the join request from malicious nodes. Wang and Vassileva [26] pointed out that trust is multi-faceted, and nodes need to develop differentiated trust in different aspects of other peers' capability. They proposed a flexible method to present differentiated trust and to combine different aspects of trust. These works help to achieve accuracy of reputation values and higher scalability of reputation systems. How to effectively use reputation values is also very critical to security. Most of the approaches for nodes to exploit reputation metrics are to select the one with the highest reputation as a providing peer. However, it may lead to unexpectedly low efficiency of high-reputed nodes and prevents P2P systems from making full use of system resources. To address this problem, "peer-approved" policy was proposed in [21] in which nodes can download files only from others with lower or equal rating. This policy encourages a node to provide high quality service in order to improve its reputation. However, a node's received quality is questionable, as it may select services from lower reputed-nodes. To handle this problem, Papaioannou [25] proposed "comparable reputation" policy aiming to restrict nodes to communicate with others at the same reputation level. Restricting the eligibility of a node to interact with others prevents node from sharing resources freely, which is a main goal of P2P systems. We develop FairTrust server selection policy that not only enables nodes to share resources freely, but also ensures secure resource allocation. It avoids overloading high-reputed nodes and takes full advantages of all resources in the system.

3 FairTrust: Trust-based Fairness-oriented Node Selection Policy

The overall performance of P2P systems depends on the performance of the individual nodes in service. Reputation system is a widely used means to get a quantifiable measure of each node's trustworthiness based on the evaluation from others about its performance. Each node's trustworthiness is evaluated by the reputation value gained based on its performance since joining the system. In a P2P system, a node needs to choose a node from a candidate pool for a service such as forwarding a query or providing a file. With a reputation system, a straightforward approach for nodes to exploit the reputation metrics is to select the one with the highest reputation value as a providing server. Such an approach may give rise to unexpected problems. First, biasing high-reputed nodes may overload them and cause inefficiency. Second, some high-reputed but non-highest-reputed nodes are excluded from service offering. Consequently, their resources cannot be fully utilized, and they are deprived of the opportunities to earn their reputation. Eventually, it will result in gradual decomposition of the nodes in the system. On one hand, some nodes are high overloaded, and may even not have enough capacity to handle the requests timely. On the other hand, some nodes are idle, while they also can offer high-quality services. Therefore, even the accurate calculation of reputation values by itself may not be adequate as a mechanism to improve the achievable efficiency of high-performing nodes, and to provide the right incentives for nodes to choose servers for services of high quality. The reputation values have to be complemented by an unbiased trust-based policy that helps a node to decide a candidate for service with consideration of both security and efficiency.

To avoid overloading high-reputed nodes, Ranganathan [21] proposed "peer-approved" policy in which nodes can download files only from others with lower or equal rating. This policy encourages a node to offer high-quality service in order to improve its reputation. However, a node's received quality is questionable, because it may select services from lower reputed-nodes. To handle this problem, "comparable reputation" policy was proposed in [25] aiming to restrict nodes communicate with other nodes at the same reputation level. The policy results in layered communities. That is, services of similar quality are exchanged among nodes of the same layer. The quality of offered services is high in the top layer, while in the bottom layer, the services offered are in most cases useless or even harmful for other nodes. The policy helps high-reputed nodes in avoiding low-reputed nodes, but it has a number of drawbacks. First, newly-joint nodes will be in a hazard environment surrounded by low-reputed nodes or even malicious nodes for communication. Second, newly-joint nodes are provided few opportunities to increase their reputation. Third, the policy deprives middle-reputed nodes of the accesses to services provided by high-reputed nodes, and also deprives high-reputed nodes of the accesses to services provided by middle-reputed and low-reputed nodes. It prevents P2P systems from achieving an ultimate goal of widely resource sharing. To address the problem, we introduce a trust-based fairness-oriented node selection policy called FairTrust. FairTrust takes into account node capacity and reputation simultaneously to choose a providing node from a candidate pool. The policy contributes not only to the security of the system but also the overall high performance by making full utilization of all node resources and avoiding flooding high-reputed nodes.

In the following, we will introduce the details of FairTrust. We assume the existence of a reputation system in a P2P system that is scalable and can accurately calculate the trustworthiness for each node. As the methods for scalability and trustworthiness accuracy of reputation systems are beyond the scope of this paper, we will not discuss the topics in the paper. Please refer to the papers [8, 16, 26, 27] for accurate reflection of node trustworthiness, and papers [6, 8, 16, 27, 30] for scalability improvement. We note that different tasks and information have different security requirements. For example, a node needs to choose very high trustworthy nodes for forwarding confidential information, but does not necessarily depend on such nodes for public news. On the other hand, a node may prefer to download a file from a high-reputed node rather than from the highest-reputed node in order to avoid request flux in the highest-reputed node. Based on this observation, we propose FairTrust to trade surplus security for efficiency by mapping tasks of different desired security levels to nodes with corresponding trustworthiness levels. The FairTrust policy enhances system efficiency by distributing load among trustworthy nodes based on their available capacities. Specifically, when node i needs to choose a server from a number of candidates, it firstly determines reputation value r based on its desired security level. It then acquires the reputation value of each candidate, and filters out candidates whose reputation values are below r. The rest of candidates include both the highest-reputed nodes and nodes that are trustworthy enough based on the requester's desired security level. The question remaining is how to choose a server from the rest of the candidates. Efficiency is an indispensable factor to consider in the server choosing process for high performance of P2P systems. It is important for a selection policy to ensure that no node becomes a bottleneck for data downloading. A straightforward way is to select the server with the highest available capacity among the trustworthy servers. However, checking each server's load status is not efficient. To reduce the overhead, instead of probing all of the neighbors to find the best candidate, FairTrust restricts the search scope to a small set of size b. That is, firstly b servers are randomly chosen and then the nodes in the set are probed sequentially, until a node with the highest available capacity is found. Probing all b servers is still a costly process. How to find a good candidate from b servers at a relatively low cost? The server selection problem can be regarded as load scheduling among a number of servers in order to achieve a fair load distribution without overloading any server. In [22], we proved the equivalence between the load scheduling and supermarket customer service model [18], and theoretically and experimentally proved that 2-way randomized probing could achieve a more balanced load distribution with faster speed over one-way probing, but $d(> 2)$-way probing may not result in much additional improvement. Based on the theorem, FairTrust adopts the 2-way randomized probing. That is, two servers are randomly probed at a time, and the server with higher available capacity is selected.

To cope with the problem mentioned above that high reputation nodes may get low quality service if they choose low reputation nodes for service, FairTrust policy uses fair trading method. That is, a client pays a server for the service it requested, and a server prices its services based on its reputation and service quality. We define that only cash payment method is used in the market, and the cash is measured by credit unit. Nodes earn credits by offering services to others. Security aside, efficiency can also be controlled by the fair trading method. For example, servers can adaptively raise their service prices to discourage clients to ask service from them when being overloaded, and offer discounts in the case of being lightly loaded. The adaptive load control not only prevents servers from being overloaded, but also helps to take full advantage of server capacities. To address the newly joined node problem mentioned above, we define that each node will be given a certain amount of credits once it joins in the system, which can be used for transactions to increase its reputation. The method makes all service accessible to all nodes and provides freedom in service requesting and offering, and meanwhile uses barter exchange to protect from malicious nodes and leech nodes. Algorithm 1 shows the pseudocode of the FairTrust policy. We define *congestion* of node i as $g_i = L_i/C_i$, where L_i represents the load of node i and C_i represents the capacity of node i. We refer to the node whose actual load is larger than its capacity (*i.e.* $L_i > C_i$) as a *overloaded node*; otherwise a light node.

Algorithm 1 Pseudo-code for FairTrust policy.

```
1: get a pool of service provider $S = \{s_1, s_2 \ldots\}$
2: determine $r$ based on required security level
3: probe reputation system for each server's reputation value
4: remove the servers whose reputation value is $< r$ from S
5: //use 2-way randomized probing
6: randomly choose two nodes $s_a$ and $s_b$ from S
7: probe $s_a$ and $s_b$ for available capacities
8: if $s_a$ and $s_b$ are all overloaded servers then
9:     select the server with less congestion
10: else
11:     if $s_a$ and $s_b$ are all light servers then
12:         use fair trading method
13:     end if
14: else
15:     select the light server
16: end if
```

4 Performance Evaluation

This section demonstrates the distinguished properties of FairTrust. We conducted experiments on a Cycloid network. For comparison, we also include the results of the "comparable reputation" policy in [25] denoted by sameTrust, and Random and MaxTrust policies. Given a number of nodes, Random chooses a node randomly and MaxTrust chooses the node with the highest reputation. Simulation results verified the superiority of FairTrust policy in comparison with the other policies toward achieving security goal and high performance at

Table 1. Simulated environment and algorithm parameters.

Environment Parameter	Default value
Cycloid dimension d	8
Number of nodes n	Fixed at 2048
Node capacity c	Bounded Pareto: shape 2 lower bound: 500 upper bound: 50000
Number of reputation levels	5
Request processing latency in a light/overloaded server	0.2/1 second

the same time. In the simulation, the file requests are consecutively generated according to a Poisson process at a rate of one per second, with a random source node and a random target key. We assume that a client has five servers in a candidate pool for its file request, and the client can locate the five servers successfully. Table 1 lists the parameters of the simulation and their default values. We assumed a bounded Pareto distribution for the capacity of nodes. This distribution reflects real world situations where machines' capacities vary by different orders of magnitude.

We define node i can handle $c_i = \lfloor 0.5+\alpha C_i \rfloor$ queries at one time, where C_i is the capacity of node i and $\alpha = 11 \cdot \frac{nC_i}{\sum_i C_i}$ in this experiment. We define node i's load as the number of requests in its request processing queue. If node i has more than c_i queries in its queue, it is overloaded. We assume there are five levels of reputation, with the probability of successfully providing service ranging from 0.2 to 1, with 0.2 increase in each level. Each node is randomly assigned one of the five reputation levels.

We evaluate the effectiveness of the server selection policies in the following metrics:

- Success rate. After a request arrives at a server, the possibility that a request can be resolved depends on the server's reputation. For example, 0.2 reputation server will have 20% probability to provide file. The success rate is the successfully resolved service requests over the total requests. The metric shows the security performance on getting successful service from servers.

- Congestion. Ideally, congestion is no more 1, which means that the node is not overloaded. The higher the congestion is above 1, the worse efficiency performance. We get the maximum congestion of each node during the experiment process, and use the the average and the 99th percentile of these maximum congestions to show the performance.

- Number of overloaded servers chosen. This metric shows the effectiveness of a policy in reducing overloaded nodes in server selection in order to reduce request processing latency.

- Request processing time. The metric represents the efficiency of server selection policies in avoiding overloaded nodes to achieve fair load distribution, and ultimately speeding up service provision.

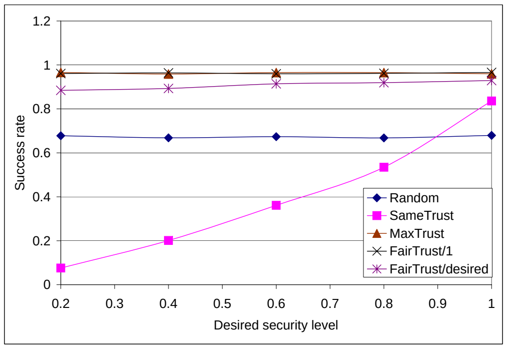
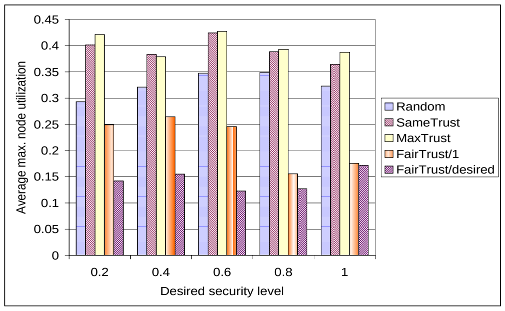
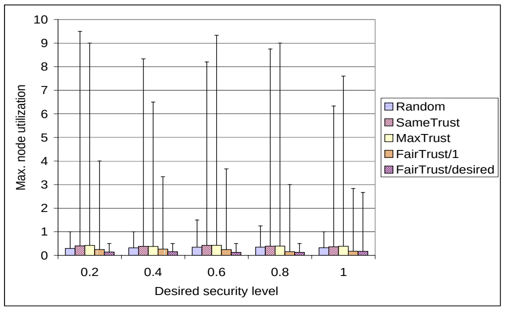


Figure 1. Success rate of service requests of different server selection policies.

An efficient server selection policy should distribute the different amount of load among the servers with the consideration of security and efficiency. In this experiment, we study the effect of FairTrust in security and efficiency compared to other related policies. In the experiment, 1000 different files were requested and each file was requested 10 times. We varied the desired security level in FairTrust from 0.2 to 1, with 0.2 increase in each step. We also take this rate as the reputation level of requesters in SameTrust. We compare the success rate between FairTrust, Random, MaxTrust, FairTrust/1, and FairTrust/desired. FairTrust/1 represents FairTrust policy with the highest reputation level 1 as desired security level, and FairTrust/desired is the policy with different desired security levels as set in the experiment. Figure 1 shows that the success rates of FairTrust/1 and MaxTrust are the highest compared with others, and the rates close to 1. The rate of FairTrust/desired is marginally lower than their rates and it increases as the desired security level increases. The results imply that FairTrust and MaxTrust achieve high security performance. We expected that the curve of FairTrust/desired is y=x. It is surprising to see that FairTrust/desired achieves much higher success rate than expected. This is because FairTrust/desired guarantees the desired security level first, and then seeks for higher security level combined



(a) The average of max. congestions



(b) The 99.9th percentile max. congestion

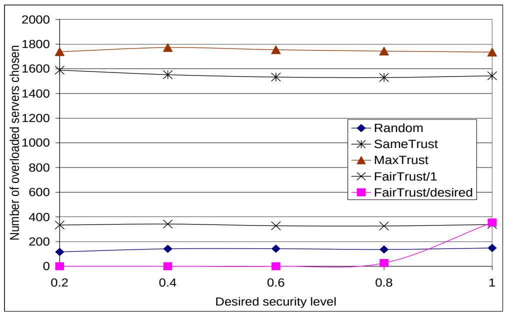
Figure 2. Node congestions of different server selection policies.

with efficiency consideration. Random does not consider reputation in node selection, so it has lower success rate. Unlike others, the success rate of SameTrust increases dramatically fast with the desire security level. Recall that SameTrust restricts communication between nodes in the same reputation level. Therefore, in addition to the server reputation, whether or not there exists a server with the same reputation level as the requester also determines the successfulness of a request. For example, if a requester has reputation level 0.2, but there's no server of the requested file having reputation level 0.2, the requester has nowhere to request the file. If there is such a server, the probability of successfully file provision is only 0.2. Consequently, the success rate increases with the requester reputation level increases, and it is much lower than others due to the constraint of same level communication. The experiment results show the effectiveness of FairTrust in security performance in comparison with other related policies. Next, let's see the performance of each policy in efficiency performance.

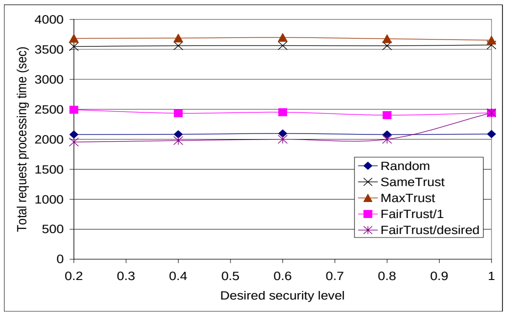
We measured each node's maximum congestion during all test cases and calculated the average and 99th percentile of the maximum node congestions. Figure 2(a) shows the average of maximum congestion rate versus desired security level. We can see that MaxTrust and SameTrust generate the highest rates. It is within expectation because MaxTrust and SameTrust strongly bias on highest-reputed or same reputation level nodes for serve provision and those nodes may turn out to be overloaded. The results indicate that MaxTrust and SameTrust have poor efficiency performance although MaxTrust has high security performance as illustrated in Figure 1. In contrast, FairTrust/1 and FairTrust/desired have much less rates. One question is that FairTrust/1 has the highest reputation level 1 as its desired security level, why it can get lower congestion rate than MaxTrust? It is because using 2-way randomized probing, FairTrust/1 distributes load between nodes among the highest-reputed nodes rather than biasing on a single node. On the other hand, with lower desired security level, FairTrust/desired expands server candidate pool, and achieves more balanced load distribution, and then higher efficiency.

Figure 2(b) shows the 99th percentile maximum congestion of each policy based on the average rates. We can observe that the rates of MaxTrust and SameTrust are significantly higher than others, and FairTrust/1 has lower rates than them due to the same reason of the observations of Figure 2(a). Random keeps the rate around 1. It confirms that Random achieves relatively balanced load distribution among servers by random selection. FairTrust/desired still keeps low congestion and it slightly increases as the desired security level increases. This is because FairTrust/desired expands the server candidates to the servers above the desired reputation level rather than biasing on the highest-reputed node, which increases the possibility of getting a light server. The results confirm the high efficiency performance of FairTrust.

We tested the service request processing efficiency in different server selection policies. Figure 3(a) and (b) show the number of overloaded servers chosen and service request processing time versus desired security level, respectively. The figures show that the MaxTrust and SameTrust have significantly higher results than others. It is due to the side-effect of high security performance by biasing the highest-reputed or same reputation level nodes. Biasing policy overburdens those nodes, and hence generates more overloaded nodes, resulting in longer request processing time. In contrast, FairTrust and Random have much less overloaded servers and processing time, and FairTrust/1 has slightly higher results than Random. This is because FairTrust/1 has a candidate pool only consisting of the highest-reputed nodes, and randomized selection can achieve relatively balanced load distribution among all servers. The figures also illustrate that FairTrust/desired has less results than



(a) Number of overloaded servers chosen



(b) Service request processing time

Figure 3. Request processing efficiency of different server selection policies.

Random and FairTrust/1, and it has 0 overloaded servers chosen in most cases. FairTrust/desired trades surplus security for efficiency by mapping tasks of different desired security levels to nodes with corresponding trustworthiness levels. The results imply the efficiency of FairTrust in avoiding overloaded nodes and speeding up service request processing.

The simulation results show that MaxTrust and SameTrust are good at achieving high security, but have difficulty in improving efficiency. Random is good at achieving high efficiency, but does not consider security. FairTrust is superior to them by achieving high performance in both security and efficiency.

5 Conclusions

Tremendous advances in P2P systems enable different distributed P2P applications take advantage of resources and enhance collaborations world-wide. Due to lack of central management, P2P systems need security technologies to protect their operations. Today's advanced reputation system enhance system trustworthiness. However, current approaches for nodes to exploit the reputation metrics degrade the system performance. This paper presents a reputation-based fairness-oriented server selection and service exchange policy, FairTrust, that treats both security and efficiency as a first class entity in order to meet the high trustworthiness and high performance of a diversified wealth of distributed P2P applications. Simulation results show the superiority of the FairTrust policy compared with other related policies in P2P systems. FairTrust helps to guarantee the high successful rate of service requests, to achieve fair load balance, to improve request processing efficiency, and to make full use of each node's capacity while control each node's load below its capacity, leading to high system performance.

Anonymity approaches to hide the identification of nodes are also very important for secure environment in P2P public networks. However, P2P performance is compromised while exploring anonymity technologies for distributed P2P applications. For example, Freenet [9] and Mute [19] achieve anonymity by requiring all data to be forwarded back hop-by-hop from a data provider to a requester rather than direct communication, which introduces high communication overhead and results in low performance. We plan to explore anonymity approaches with little performance compromise.

Acknowledgments

This research was supported in part by U.S. Acxiom Corporation.

References

[1] Average simultaneous global P2P users. http://www.slyck.com/misc/p2p_history_sep-06_average.xls.

[2] Bittorrent. http://en.wikipedia.org/wiki/Bittorrent.

[3] eMule Project Team Web Site. http://www.emule-project.net/.

[4] Meta Search Inc.- eDonkey2000 Home Page. http://www.edonkey2000.com/.

[5] mlDonkey Web Site. http://mldonkey.org/.

[6] K. Aberer and Z. Despotovic. Managing trust in a peer-2-peer information system. In *Proc. of CIKM*, pages 310–317, 2001.

[7] N. Azzouna and F. Guillemin. Analysis of ADSL traffic on an IP backbone link. In *Proc. of IEEE GLOBECOM*, 2003.

[8] S. Buchegger and J. L. Boudec. A robust reputation system for p2p and mobile ad-hoc networks. In *Proc. of P2PECON*, 2004.

[9] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proc. International Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, 2001.

[10] R. Dingledine, N. Mathewson, and P. Syverson. Reputation in p2p anonymity systems. In *Proc. of P2PECON*, 2003.

[11] D. Dutta, A. Goel, R. Govindan, and H. Zhang. The design of a distributed rating scheme for peer-to-peer systems. In *Proc. of P2PECON*, 2003.

[12] M. Gupta, P. Judge, and M. Ammar. A reputation system for peer-to-peer networks. In *Proc. of NOSSDAV*, 2003.

[13] R. Hartani and J. Neil. P2p optimized traffic control. Technical report, Caspian Networks, 2005.

[14] Internet world stats. http://www.internetworldstats.com/.

[15] A. Josang, S. Hird, and E. Faccer. Simulating the effect of reputation systems on e-markets. In *Proc. of iTrust*, 2003.

[16] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *Proc. of WWW*, 2003.

[17] S. Marti and H. Garcia-Molina. Limited reputation sharing in P2P systems. In *Proc. of Conf. Electronic Commerce*, 2004.

[18] M. Mitzenmacher. On the analysis of randomized load balancing schemes. In *Proc. of SPAA*, pages 292–301, 1997.

[19] The mute file sharing systems. http://mute-net.sourceforge.net/.

[20] T. G. Papaioannou and G. D. Stamoulis. Enforcing credible reporting in peer-to-peer environments. Working paper, AUEB, 2004. Available at http://nes.aueb.gr.

[21] K. Ranganathan, M. Ripeanu, A. Sarin, and I. Foster. ťo share or not to shareán analysis of incentives to contribute in collaborative file sharing environments. In *Proc. of P2PECON*, June 2003.

[22] H. Shen and C. Xu. Locality-aware and churn-resilient load balancing algorithms in structured peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 2007.

[23] A. Singh and L. Liu. Trustme: Anonymous management of trust relationships in decentralized p2p systems. In *Proc. of P2P Computing*, 2003.

[24] M. Srivatsa, L. Xiong, and L. Liu. Trustguard: Countering vulnerabilities in reputation management for decentralized overlay networks. In *Proc. of WWW*, 2005.

[25] G. P. Thanasis and D. S. George. Effective use of reputation in peer-to-peer environments. In *Proc. of GP2PC*, 2004.

[26] Y. Wang and J. Vassileva. Trust and reputation model in peer-to-peer networks. In *Proc. of P2P*, 2003.

[27] L. Xiong and L. Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *TKDE*, 16(7):843–857, 2004.

[28] M. Yang, Y. Dai, and X. Li. Bring reputation system to social network in the maze P2P file-sharing system. In *Proc. of CTS*, 2006.

[29] 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.

[30] R. Zhou and K. Hwang. Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing. *TPDS*. To appear.