DIFFERENTIATED ADMISSION FOR PEER-TO-PEER SYSTEMS: INCENTIVIZING PEERS TO CONTRIBUTE THEIR RESOURCES

H. T. Kung[1] and Chun-Hsin Wu[2]

[1]Harvard University
Division of Engineering and Applied Sciences
Cambridge, Massachusetts, USA

[2]Academia Sinica
Institute of Information Science
Taiwan, ROC

Abstract

A peer-to-peer (P2P) system will starve from resources if every user is a freeloader who only takes resources from the system but never contributes any. It is useful to develop a scheme that can incentivize users to contribute resources such as content, CPU, storage and network bandwidth. This paper describes an admission control system capable of providing differentiated services to users based on their past contributions or reputations. A user desiring a certain level of service will need to make a comparable level of contributions to the P2P system. The admission system uses a distributed eigenvector-based method to compute user reputations, a sampling heuristic to reduce computation and communication costs, and a distributed trust-enhancement scheme to provide system security.

1 Introduction

A well-known issue in P2P systems concerns the fair contribution of resources by participating users (see, e.g., [1], [2]). In particular, it is important to avoid the situation that none of the users will contribute resources. In this paper, we present an admission system aimed at solving this free riding problem and demonstrate simulation results.

The admission system embodies five main ideas:

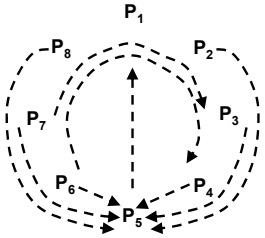
- A reputation-based, differentiated admission control that allows a node to receive a level of service based on its reputation or contributions in the past. (In this paper, we use the terms "nodes" and "users" interchangeably.)
- An eigenvector-based method that derives service and usage reputations of nodes by computing the largest eigenvalue/eigenvector pairs of the credit matrix associated with past transactions.
- An adaptation system that allows a node to take care of its own interest by contributing resources at a level just sufficient for its desired level of service.
- A sampling technique that uses top service and usage nodes, as well as benchmark nodes, to reduce the cost of computing service and usage reputations of nodes.
- A distributed trust-enhancement scheme that uses third-party nodes to manage and store credits required by reputation computations.

2 Eigenvector-based Reputation System

We give a brief overview of the eigenvector-based reputation system used in our admission control. We will use the formulations and notations introduced in this section throughout this paper.

We consider a P2P system for providing content service. Denote the nodes in the P2P system as P_1, …, P_n. We define a *transaction* as the transport of a piece of content from a node to another, possibly through some other intermediate nodes.

(a) Transaction paths



(b) Service credit matrix

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
P_1	0	0	1	1	0	0	0	0
P_2	0	0	1	1	2	0	0	0
P_3	0	0	0	1	3	0	0	0
P_4	0	0	0	0	4	0	0	0
P_5	2	0	0	0	0	0	0	0
P_6	0	0	0	2	4	0	0	0
P_7	0	0	2	1	3	0	0	0
P_8	0	0	1	1	2	0	0	0

(c) Eigenvector-based rankings

	Service Reputation	Usage Reputation
P_1	7	4
P_2	5	4
P_3	4	3
P_4	2	2
P_5	8	1
P_6	1	4
P_7	3	4
P_8	5	4

Figure 1: (a) A set of transactions, each represented by a dotted arrow. An arrow from P_i to P_j represents a transaction of sending content from P_i to P_j. (b) The corresponding service credit matrix. (c) Eigenvector-based rankings of nodes in service and usage reputations.

We define the *service credit matrix* S associated with a set of transactions as follows. The entry at position (P_i, P_j) of S is the amount of service credits node P_i receives in providing or transporting content for node P_j. Figure 1 (b)

shows the service credit matrix associated with the 8-node transaction example of Figure 1 (a). In this example, we assume that for each transaction a node will receive 2 credits for providing content and 1 credit for transporting content.

We define the *usage credit matrix* U to be S^T, the transpose of S. The entry at position (P_i, P_j) of U represents the amount of usage credits node P_i receives in receiving content or in using transporting service provided by node P_j. Thus by defining $U = S^T$ we assume that whenever P_i receives service credits in providing or transporting content for P_j, P_j will receive usage credits in the same amount.

Let vectors s and u denote the *service reputations* and *usage reputations* of nodes in the P2P system. That is, the i-th component of s or u is the service or usage reputation of node P_i, respectively.

We define s to be the converged value computed iteratively by $s^{(i+1)} = Su^{(i)}$ and $u^{(i)} = Us^{(i)}$, with initial $u^{(0)}$ and $s^{(0)}$ being any values. Note that $s^{(i+1)} = SUs^{(i)}$, or equivalently $s^{(i+1)} = SS^Ts^{(i)}$. We can view the latter iteration as the power method [3] of computing the largest eigenvalue/eigenvector pair of SS^T. This means that s is the eigenvector of SS^T corresponding to the largest eigenvalue of SS^T. Similarly, we define u to be the converged value computed iteratively by $u^{(i+1)} = Us^{(i)}$ and $s^{(i)} = Su^{(i)}$. Thus, u is the eigenvector of S^TS corresponding to the largest eigenvalue of S^TS. This matrix formulism parallels to that of the HIT algorithm [4] for identifying influential articles from webpage hyperlink structures. As discussed in [5], such kinds of algorithms give stable rankings under certain perturbations to the hyperlink patterns.

Figure 1 (c) shows the service and usage reputation rankings of nodes based on the computed largest eigenvalue/eigenvector pairs of SS^T and S^TS, respectively. Note that the eigenvector-based rankings in Figure 1 (c) are what one would expect. Consider, e.g., P_5 in Figure 1 (a), which receives content from six other nodes, but only sends content to one node and does not provide any content transport. We see from Figure 1 (c) that P_5 has a high usage ranking (ranking: 1) and low service ranking (ranking: 8). In contrast, P_6 sends content to two other nodes and provides content transport for two transactions, while not receiving any content. We see that P_6 has a high service ranking (ranking: 1) and low usage ranking (ranking: 4).

3 Reputation-based, Differentiated Admission

When a node X receives a request for content from a node Y, X will determine whether to send its content to Y based on Y's service and usage reputations. For the rest of the paper, let u or s denote, respectively, the usage or service reputation-ranking percentile of the node in question. Suppose that Y's u is above A% while its s is below B%, where A and B, with A > B, are certain preconfigured system-wide parameters. (For example, A and B can be 80 and 20, respectively.) Then X will deny Y's request. This denial represents a reasonable scenario in view of fact that Y has had a relatively high usage of services but has only made relatively low contributions in serving others. While being denied of receiving content, Y can continue providing content and transport services, thereby improving its service reputation. This makes it possible for Y to receive content in the future.

Note that admission decisions allow or disallow certain transactions to take place. After an allowed transaction is complete, the participating nodes that request, provide or transport the content will update their credits to reflect their roles in the transaction. Thus, the service and usage credit matrices will change accordingly. These changes will in turn affect future reputation rankings of the nodes.

In this reputation-based admission system, a node desiring to receive content from other nodes will need to contribute to the P2P systems at a certain level. A freeloader, which has a low service reputation and a high usage reputation, will be denied of service. The node will need to either provide an increased level of service or reduce its content usage in order to be readmitted again.

4 Nodes' Adaptation in Willingness to Serve

The admission control allows a node to be greedy by searching for a minimum level of contribution it needs to provide in order to receive a desired level of service from other nodes. For this purpose, the node monitors its *success rate* σ, which is the percentage of its content or transport requests that are granted by requested nodes. Based on the σ value, the node adjusts its *willingness-to-serve probability* ρ, which is the probability at which the node will accept arriving requests. Note that a node using a higher value of ρ means that the node is more willing to provide service. When a node finds that its σ is above a desired level C for a certain period T of time, it will decrease its ρ. On the other hand, when the node finds that its σ is below C for a period T of time, it will increase its ρ.

We describe a method of controlling the increase and decrease of ρ. As depicted in Figure 2, a node will be in the "Deny" state when its u is above A% while its s is below B%. The node will increase ρ when in states "Deny" and "Admit & More", and decrease ρ when in state "Admit & Not More". Note that when in the "Deny" state, a node can get out of the state by increasing ρ. A node can also raise its σ value by increasing ρ. Moreover, when a node in-

creases ρ, it will improve its ranking in service reputation. This will, in turn, cause some of the other nodes to drop their rankings in service reputation. If these nodes have sufficiently high usage reputations, then they will enter the "Deny" state. These nodes will then increase their ρ, thereby allowing the current node to improve its σ.

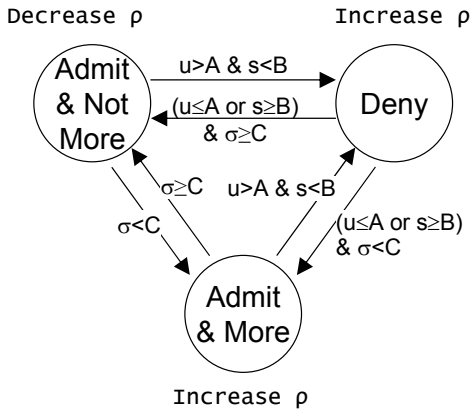


Figure 2: The three states of a node and state transitions. Depending on its state, a node may increase or decrease its willingness-to-serve probability ρ.

When a node is in the "Admit & Not More" state with σ ≥ C, it will decrease ρ until reaching a point when σ is less than C or the node enters the "Deny" state.

In summary, a node will adapt its willingness-to-serve probability ρ based on the observed success rate σ. If σ is less than the desired level C, the node will try to raise σ by increasing ρ. When σ has reached C, the node will try to reduce the level of service it provides by decreasing ρ. Thus this is a feedback control system parameterized by the desired service level C.

The desired service level C is constrained by the system-wide parameters A and B. Suppose, for example, that nodes in the P2P system make uniform requests to other nodes. Then it is easy to see that each node has the probability of B*(1-A) being in the "Deny" state. This means that C should not be set to exceed 1 - B*(1-A).

Note that it is important to have separate "Admit & Not More" and "Admit & More" states rather than a single combined "Admit" state, in order to avoid a scenario which we call the "poor man's equilibrium". This is the scenario when many nodes are in this combined "Admit" state with small σ and zero or very low ρ. These nodes are expected to have low u values since many of their peers themselves are likely also in the same situation, i.e., having small σ and zero or very low ρ. This means that these nodes will never be able to increase their ρ since they cannot enter the "Deny" state due to their u being below A%. We have indeed seen this deadlock scenario in our simulation.

5 Convergence of Nodes' Adaptation

We demonstrate by simulation that nodes' adaptation converges in the sense that when the desired level of service C changes, the success rate σ will converge to C's new value. Figure 3 depicts the simulation result showing the adaptation of σ and the willingness-to-serve probability ρ in response to changes in C over time for a randomly chosen node.

The simulation configuration is as follows. There are 128 nodes in the P2P system, each constantly making content requests to other nodes. The requesting and requested nodes as well as intermediate nodes used in transporting contents and their numbers are all uniformly distributed. The simulation uses the following parameters: A = .8, B = .2 and the number of transactions per round = 10,000. For all nodes, C changes its value from .8 to .4 or vice versa every 100 rounds. When a node is in the "Admit & Not More" state, it decreases ρ using ρ ← .95 * ρ, else it increases ρ using ρ ← max (ρ + . 05, 1). Note that except when ρ = 1, the decreasing rate is smaller than the increasing rate. Figure 3 depicts the adaptation of ρ and the observed σ, in response to changes in C. We note that σ generally tracks changes in C. There are short periods when the node is in the "Deny" state, as shown by sharp downward spikes in σ.

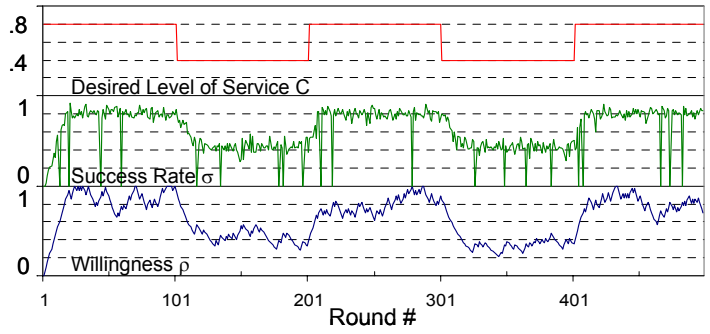


Figure 3. Adaptation of willingness-to-serve probability ρ and the observed success rate σ, in response to changes in the desired service level C.

The admission system can bootstrap itself out from the initial state when all the nodes or most of them are new nodes. The initial state of a new node is set to be "Admit & More" with the initial value of ρ being 0. That means it can request services without providing services initially. If a new node requests too many services such that its u is above A and its s is below B, it will enter the "Deny" state. Otherwise, if its success rate σ does not reach its desired

level of service C, it will stay at the "Admit & More" state and increase its ρ to raise σ. When its σ reaches C, it will enter the "Admit & Not More" state and try to minimize its ρ. While bootstrapping the system, the service or usage credit matrix is initially a zero matrix and the success rate of every node is zero, since there have not been any transactions. Before the success rate σ of each node reaches C, it will increase its ρ to raise σ. That means a node will try to receive desired level of service by providing more services. As illustrated in Figure 3, within about 16 rounds, a new node will be able to adapt its ρ to respond to its desired level of service C.

6 Top-node Sampling and Benchmark Nodes

We describe an efficient heuristic for determining the current state of a node X. There are two ideas in the method. First, we only compare X to two *benchmark nodes* in usage and service reputations. Second, in computing reputation, we only use a set of *top nodes*, consisting of top usage nodes and top service nodes, together with node X and the two benchmark nodes.

We select two benchmark nodes, P_A and P_B, as follows: P_A has its usage-reputation ranking percentile at A%, whereas P_B has its service-reputation ranking percentile at B%. Next, we define a *sampling set* consisting of top nodes, node X, and the two selected benchmark nodes P_A and P_B. Considering only transactions involving nodes in this sampling set, we compute the reputations of P_A, P_B and X. Based on the computed reputations, X is compared with P_A in usage reputation and with P_B in service reputation. Finally, X's state is determined based on the state transitions depicted in Figure 4. Periodically, the set of top nodes and the two benchmark nodes are updated. We only need to perform these updates relatively infrequently.

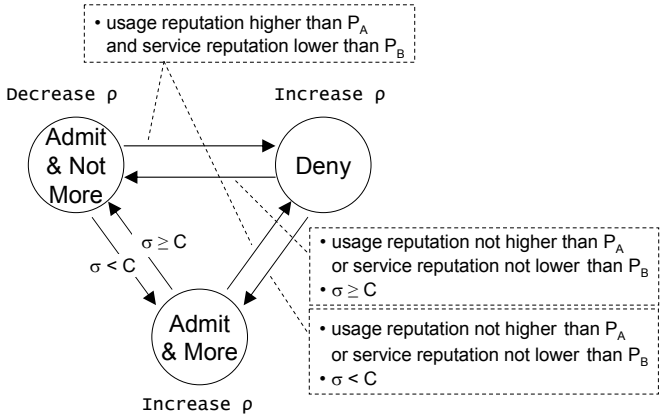


Figure 4: Node state transitions using the top-node sampling heuristic and benchmark nodes P_A and P_B. Compare this diagram with that in Figure 2 which uses no sampling heuristics.

To quantify the effectiveness of the top-node sampling heuristic, we consider error rates in two categories: FalseDeny and FalseAdmit. FalseDeny or FalseAdmit means that the method incorrectly classifies a node to be in the state "Deny" or one of the two "Admit" states, respectively. Figure 5 depicts simulation results on error rates as a function of the number of top nodes used in the sampling. The configuration and parameters used in this simulation are the same as those for Figure 3, except that this time we use Laplace distributions rather than uniform distributions to model transaction sources and destinations. In particular, nodes requesting content follow a distribution with three peaks, formed by a mixture of three Laplace distributions. The use of such Laplace distributions allows increased variations in usage and service reputations among nodes, thereby being able to take advantage of the top-node sampling heuristic.

We have simulated this sampling heuristic. The simulation results, e.g., those shown in Figure 5, suggest that it generally suffices to use a small number of top nodes such as ten top nodes. That is, we can use a small service matrix corresponding to just the top nodes, the two benchmark nodes and the current node in computing reputations, Thus this sampling approach reduces the computation and communication costs.

As one would expect, the heuristic is effective when there are large "hub nodes" present which have dominant usage or service reputations. It appears that real-world P2P systems do often exhibit the hub nodes phenomenon [6].

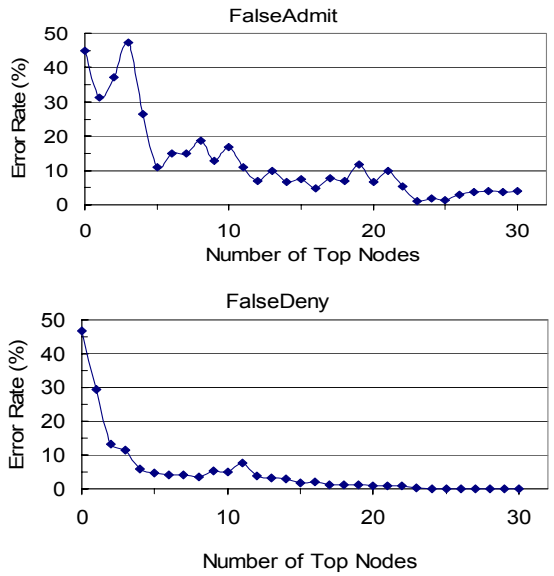


Figure 5: Error rates in FalseDeny and FalseAdmit decrease rapidly as top nodes used in the sampling increase.

7 Distributed Trust-enhancement

We describe a distributed trust-enhancement scheme to provide protection against possible cheating of the reputation system. To illustrate the idea, consider a transaction that sends content from node P_i to P_j under a simplifying assumption that nodes can receive credits only by providing content, not by transporting content. Note that if the credits associated with the transactions are stored at P_i or P_j, then these nodes could potentially alter the credit values for their own benefit.

To enhance the trustworthiness of credits, our scheme will store credits at a *third-party* node P_h determined by h = *hash* (i, j) with *hash* being a hash function known to all nodes. For example, we can use a distributed hash table mechanism (DHT) such as Chord [7] to maintain the credit matrix in a scalable manner. That is, the entry at position (P_i, P_j) of the service matrix is stored at node P_h of the DHT network.

In general, one can expect that P_i and P_j have no management authority on altering credits stored at a third-party node. Moreover, since reputations are based on credits associated with transactions involving potentially many different pairs of P_i and P_j and these credits are stored at different third-party nodes, even if some of these third-party nodes are compromised, the impact to the integrity of the overall reputation system can still be limited.

To make sure that the third-party node P_h will properly increment P_i's credit, P_i will encrypt content sent to P_j in a session key k and send the key to P_h. Node P_j can only decrypt the content by requesting the session key k from P_h and receiving it. This scheme, as depicted in Figure 6, ensures that P_h knows about the transaction and thus can properly increment P_i's credit stored at P_h to reflect P_i's contribution in the transaction.

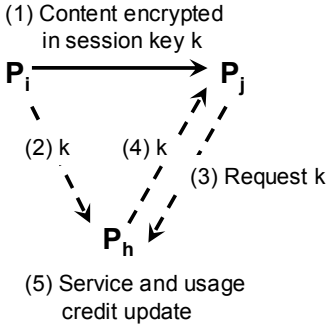


Figure 6: Distributed trust-enhancement using third-party nodes. For the transaction of sending content from P_i to P_j, the associated service and usage credits are stored at a third party node P_h, determined by h = *hash* (i, j) with *hash* being a hash function.

In this scheme, we assume that it is to P_i's advantage that it will deliver the correct content to P_j as requested, for otherwise in the long term others will eventually stop requesting content from P_i. This means that P_j cannot cheat by falsely claiming that encrypted content it receives is incorrect and asking P_h to erase its usage credit.

8 Putting the Whole Thing Together

We give a brief overview of the entire reputation-based admission system. To simplify the presentation, we consider here only the simplified case where nodes can only receive credits by providing content. It is straightforward to extend the scheme here to cover the case where this assumption is removed.

When node X receives a request from node Y for content, X will trigger a series of steps:

a) X accepts the request from Y with a probability based on X's current willingness-to-serve probability ρ (see Section 4). If X decides to accept Y's request, then the steps below take place.

b) X determines whether Y should be "admitted" or "denied:

   - Consider a sampling set, consisting the current top nodes, two benchmark nodes, P_A and P_B (see Section 6), and node Y.
   - For each ordered pair of nodes (P_i, P_j) in the sampling set, retrieve from P_h the credits of P_i in serving P_j, where h = *hash* (i, j) (see Section 7), and form the service credit matrix S.
   - Compute the largest eigenvalue/eigenvector pairs of SS^T and S^TS to obtain the service and usage reputation rankings of nodes in the sampling set, respectively.
   - Y is "denied" if Y has higher usage reputation than P_A and lower service reputation than P_B; otherwise Y is "admitted". For the latter case, the following steps take place.

c) X sends Y the requested content encrypted in a session key, and sends the session key to node Z, where X = $P_{i'}$, Y = $P_{j'}$, Z = $P_{h'}$ and h' = *hash* (i', j'). (See Section 7.)

d) Y requests Z for the session key.

e) Z sends Y the session key and increments the credit of X in serving Y.

f) Y uses the session key to decrypt the content received from X.

In addition, each node computes periodically service and usage reputation rankings of nodes in the current sampling set. Based on its rankings relative to P_A and P_B, the node

decides whether to increase or decrease its willingness-to-serve probability ρ (see Figure 4).

Further, there are background processes that compute service and usage reputation rankings of all nodes for updating the top nodes and the two benchmark nodes. The update results are advertised to all nodes. The top and benchmark nodes need relatively infrequent updates. In our simulation for the results of Figure 5, these updates are performed once each 10,000 transactions.

For a P2P system with a very large number of nodes, the "link-state" style of computing reputation rankings, where a node first forms a service credit matrix S and then computes the largest eigenvalue/eigenvector pairs of SS^T and S^TS as described in step b) above, can be prohibitively expensive. This is because in selecting the top nodes and the two benchmark nodes, all nodes rather than just the sampling nodes will need to be considered. In this case, we may use a relaxation or "distance-vector" style of algorithm that computes the largest eigenvalue/eigenvector pairs in a distributed manner. Moreover, at any given time, we may consider only a small number of randomly selected nodes outside the top set for their possible replacement of some of the current top nodes. This approximate method could work because the top set is not expected to change substantially over a short period of time.

9 Summary and Concluding Remarks

We have described a reputation-based P2P admission system. It allows only those nodes that have made reasonable service contributions to receive services from others. We use eigenvector methods, similar to those that compute webpage rankings, to compute the service and usage reputations of nodes. We show how a node can adapt its willingness-to-serve probability in order to find a minimum level of services it needs to provide while still being able to receive services provided by others. We give a sampling technique that can substantially reduce the cost of reputation computations. Finally, we show a distributed trust-enhancement scheme based on the use of third-party nodes. Our approach differs from previous work in P2P reputation such as [8], [9], [10], [11] in a number of aspects, including our use of eigenvector methods and focus on reputations of content requesters rather than content providers.

There are several important issues beyond those presented in this paper, on some of which we have already done initial investigation. These include schemes for reputation caching and expiration, and methods that can tolerate network partitioning. We have also found that it could be difficult to achieve convergence when the topology of the P2P network is spare. For this, we may need to develop a better feedback control system.

References

[1] Eytan Adar, and Bernardo A. Huberman, "Free Riding on Gnutella," First Monday, Vol. 5, No. 10, October 2000.

[2] "Punishing Freeloaders Key to Cooperation?" http://www.infoanarchy.org/story/2002/1/11/02517/7790.

[3] G.W. Stewart, Introduction to Matrix Computations, Academic Press, New York, 1973.

[4] Jon M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," In Proc. the 9th ACM-SIAM Symposium on Discrete Algorithms, 1998.

[5] Andrew Ng, Alice Zheng and Michael Jordan, "Link Analysis, Eigenvectors and Stability," in Proc. the 17th International Joint Conference on Artificial Intelligence, 2001.

[6] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," in Proc. Multimedia Computing and Networking, 2002.

[7] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek and Hari Balakrishnany, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in Proc. ACM SIGCOMM, 2001.

[8] Mao Chen and Jaswinder Pal Singh, "Computing and Using Reputations for Internet Ratings," in Proc. the 3rd ACM Conference on Electronic Commerce, 2001.

[9] Fabrizio Cornelli, Ernesto Damiani and Sabrina De Capitani di Vimercai, "Choosing Reputable Servents in a P2P Network," in Proc. World-Wide Web Conference, 2002.

[10] Richard Lethin, "Reputation," in *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, Edited by Andy Oram, O'Reilly and Associates, Inc., 2001, pp. 341-535.

[11] Sepandar Kamvar, Mario Schlosser and Hector Garcia-Molina, "EigenRep: Reputation Management in P2P Networks," in Proc. World-Wide Web Conference, 2003.