Fighting Peer-to-Peer SPAM and Decoys with Object Reputation

Kevin Walsh
Department of Computer Science
Cornell University
Ithaca, NY 14853
kwalsh@cs.cornell.edu

Emin Gün Sirer
Department of Computer Science
Cornell University
Ithaca, NY 14853
egs@cs.cornell.edu

ABSTRACT

Peer-to-peer filesharing is now commonplace and its traffic now dominates bandwidth consumption at many Internet peering points. Recent studies indicate that much of this filesharing activity involves corrupt and polluted files. This paper describes Credence, a new object-based reputation system, and shows how it can counteract content pollution in peer-to-peer filesharing networks. Credence allows honest peers to assess the authenticity of online content by securely tabulating and managing endorsements from other peers. We employ a novel voter correlation scheme to weigh the opinions of peers, which gives rise to favorable incentives and system dynamics. We present simulation results indicating that our system is scalable, efficient, and robust.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Distributed networks; C.2.4 [**Network Architecture and Design**]: Distributed applications

General Terms

Algorithms, Measurement, Security

Keywords

Reputation Systems, Pollution, File Sharing

1. INTRODUCTION

Peer-to-peer filesharing has become a significant feature of the Internet, consuming a large fraction of network bandwidth. However, many filesharing networks are rife with corrupt and mislabeled content, which waste network resources and make it difficult for clients to find sought objects. Corrupt content can also be a source of security vulnerabilities, viruses, worms and other malware. Recent research confirms these vulnerabilities [2], and also indicates that much of the pollution in existing networks is deliberate [14]. Despite this, there are currently no mechanisms for clients to confidently gauge if an object is polluted.

The underlying problem faced by clients is that they routinely interact with peers about which nothing is known. If clients interact only with a small set of peers, or belong to a community of clients all of which tend to interact with the same peers, then a client can learn peer reputations by collecting local observations over time, and querying the observations made of members of the client's community. But in many peer-to-peer systems, interactions span large and dynamic groups of peers. The presence of file replication and locality-based peer selection can further decrease the chances that groups of clients will interact with the same set of peers. These networks require new measures that do not rely on per-interaction pairwise observations.

This paper introduces Credence, a robust and decentralized system that enables peers to confidently gauge *object authenticity*, the degree to which an object's data matches its advertised description. Credence encompasses three basic techniques to obtain reliable estimates of authenticity. First, we employ a simple network-wide voting scheme, where users contribute positive and negative evaluations of objects. Second, we enable clients to weigh votes according to the statistical correlation between the client and its peers. And third, we allow clients to extend the scope of their correlations through selective information sharing.

The fundamental insight driving our work is that communities of peers with similar experiences can arise naturally if, instead of tracking the direct interactions between *peers*, we base reputation only on a client's experiences with *objects*. We find many technical and practical reasons to prefer a system based on object reputation. First, client evaluations of objects are final, since they depend only on the intrinsic properties of each immutable object. By contrast, peer behavior is dynamic, and so a client's evaluation of peers must change over time. Second, object reputations are an efficient and natural complement to swarming downloads, since a client can validate an object's authenticity in a single operation, then immediately initiate parallel connections to several peers offering the object for download. And finally, because an object's authenticity does not depend on user preferences or tastes, we can expect most honest clients to have the same view of object authenticity. Thus, in an object-based reputation system, differing evaluations are an indicator of malicious behavior. By contrast, in peer-based reputation systems, which track the outcome of pairwise interactions over time, each client has a unique and changing view of the network, and a client's own reputation is sensitive to many factors unrelated of malicious behavior, such as network connectivity, availability, and overall filesharing performance.

We have implemented Credence as an extension to LimeWire [15], a popular filesharing client running on the Gnutella network, and our initial implementation has seen over 4000 downloads. In this paper we describe prior work on reputation systems for peer-to-peer networks, detail the design of Credence, and discuss its performance in simulated and live networks.

2. RELATED WORK

A recent study [2] of four large filesharing networks concluded that these systems are vulnerable to content pollution and attacks using replicated decoys. Ross [14] finds evidence of rampant pollution in the decentralized Kazaa network. Previous work on peer-to-peer reputation, however, focuses mainly on the *freeloader* problem, where some peers do not contribute a fair share of resources. Consequently, most previous work relies on peer reputation, rather than object reputation as we advocate.

Eigentrust [13] performs a distributed computation of a single, network-wide reputation for each peer, based on the outcomes of past pairwise interactions. Byzantine and colluding participants can potentially slow or prevent full convergence of the eigenvalue computation. Zhang et al. [23] show how to make centralized eigenvector-based approaches more resistant to collusion, but the techniques are not applicable to the distributed Eigentrust computation. Eigentrust also relies on a set of fixed, trusted nodes at which to root the computation of trust.

Gupta, Judge, and Ammar [12] give an alternative method of managing peer contributions to the network, based on an economic model of earning credits during pairwise interactions and storing the accumulated reputation in the network. In the general case, micropayments can be used to induce cooperation (e.g. [19, 21, 16]). These schemes do not address the content pollution problem, and are therefore complementary our approach.

Bouchegger and Boudec [1] advocate a peer reputation approach, but clearly state the need for a separation between peer performance in the underlying filesharing network, and trustworthiness in the ratings system independent of filesharing performance. Pairwise trustworthiness ratings are computed as a function of the agreement or disagreement in past votes, but the original votes are still subjective evaluations of pairwise filesharing interactions. Similarly PeerTrust [20], and the reputation scheme included in the Free Haven Protocol [7] make a distinction between the reputation of a peer based on performance, and credibility based on past voting history.

P2PRep [3] similarly uses peer-based ratings and a trust metric derived from past votes. The trust ratings are based on a simple threshold-based agreement count that is vulnerable to manipulation by collusion. XRep [6] and X^2Rep [5] extend P2PRep, adding object reputations and computing peer weights based on past voting behavior. These protocols require peers to be online during the evaluation phase in order to compute and transmit their votes, and do not share trust ratings among peers. Information sharing and offline operation are critical features for a peer-to-peer reputation system due to the sparse workloads and session lengths observed in peer-to-peer networks [18].

In contrast, Credence does not address freeloading or peer selection, but rather content pollution, a fundamental weakness in open peer-to-peer networks that is only recently gaining attention [2, 14]. Recommender systems (e.g. [17]), which filter content based on user preferences or tastes, and reputation systems for online marketplaces (e.g. those in [22]), which seek to identify trustworthy vendors and customers, both address problems similar to content pollution. The techniques used in these systems are not directly applicable to peer-to-peer networks because they rely on centralized components.

Guha et al. [10] examine how both positive and negative evaluations might be propagated through a web of pairwise observations made by peers in the network. We share a similar model of information propagation through transitive, pairwise relationships between peers.

3. APPROACH

We consider networks where clients share objects, each consisting of *data* and a *descriptor*. The descriptor contains meta-data to facilitate searching, such as the object name and encoding, and a unique identifier, such as a hash of the data. A client issues queries to its peers in the network in the form of keywords. Peers respond by sending matching object descriptors back to the client.

At this point, the client must judge the authenticity of each descriptor before attempting to fetch the associated data, since some may contain invalid meta-data or point to corrupt or malicious data. Lacking any reliable evidence, a client typically uses ad hoc object reputation indicators, such as the frequency each descriptor is encountered, or revert to random selection. Credence addresses content pollution by providing reliable estimates of object authenticity, and includes incentives for peers to contribute honestly in this evaluation process.

3.1 Endorsements

The basic mechanism in Credence is a simple weighted voting protocol in which any client may vote positively or negatively on any object. A vote $(\langle \text{objectID}, \text{value} \rangle_K, \text{cert}_K)$ is a pair containing a cryptographic signature, under the client's key K, of the object identifier and a vote value from the set $\{-1, +1\}$, along with a certificate of authenticity for key K. The object identifier consists of a hash of the object descriptor and contents. The key and certificate are used by peers to ensure vote authenticity and uniqueness. To help avoid Sybil attacks [8], we require each client to download a large file at installation time before providing the key certificate. Other mechanisms are also possible, such as solving captchas or cryptographic puzzles during installation. Client keys need not be bound to real-world identities, but instead may rely on the same anonymous pseudonyms commonly used in filesharing networks.

A client interprets a positive vote as an endorsement of the object's authenticity (i.e., that the contents match the descriptor). Since object descriptors in practice typically contain only factual and easily verifiable information, we assume that among honest clients, a large fraction will generate votes of equal value for a given object. Under this assumption, a client judges the authenticity of an object by estimating its reputation among the client's peers.

3.2 Vote Evaluation

Votes are collected, evaluated, and aggregated by a client wishing to estimate the reputation of a given object. Simply tabulating votes would be prone to manipulation. Instead, each Credence client computes a trust metric for each vote, and uses weighted averaging to compute an estimate of the object's overall reputation.

The weight r of a peer's vote depends most directly on the relationship between client and peer, and so the client weighs votes according to the observed strength and bias of this relationship. Intuitively, two peers that tend to vote identically (or inversely) on objects should develop over time a strong positive (or negative) weight for each other's votes, while peers having uncorrelated voting histories should disregard each other's votes.

Statistical correlation captures precisely this notion of the historical relationship between a pair of peers. We compute a coefficient using the method of Phi correlation as follows. Given the set of objects on which two peers A and B have voted, let a and b be the fraction where A and B voted positively, respectively, and similarly let p be the fraction where both voted positively. Then $\theta = (p - ab) / \sqrt{a(1-a)b(1-b)}$ is the *coefficient of correlation*, which takes on values in the range $[-1, 1]$. A positive value for θ indicates that A and B tend to agree, negative θ indicates that they tend to disagree, and $|\theta| < 0.5$ indicates weak or no correlation. We normally use weight $r_{AB} = \theta$ whenever $|\theta| \geq 0.5$ and $r_{AB} = 0$ otherwise. We use two heuristics to address defunct cases that arise in practice. For pairs of peers with little overlap, not enough data is available to make a robust estimate of θ, so we take $r_{AB} = 0$. For peers whose votes are all negative or positive, θ is undefined even if the peers are mostly or completely in agreement. In this case, we use a simple vote agreement metric with maximum $|r_{AB}| = 0.75$.

3.3 Voting Protocol

We now describe the protocol a client uses to estimate the reputation of a given object. First, a client issues a *vote-gather* query to collect votes on an object; peers respond with matching votes, if any, they possess. Each response contains a subset of the votes known to the responding peer, potentially including the peer's own vote. The impact of this query on the network is bounded by having each peer sub-sample its vote information, rather than sending all known votes. The sub-sampling is biased in favor of votes having the highest local weight in order to disseminate the most useful votes further in the network. Depending on the underlying network structure, the vote gather query may be implemented as a flood followed by a sequence of fetches, as a DHT lookup, or as a single round of interaction in combination with the initial user query.

Once client A has obtained a set of votes for the object, each vote is cryptographically verified, stored for later use, and then tabulated using a weighted average. The weight for a vote from peer B is r_{AB}, the correlation between A and B, computed from data gathered in earlier rounds of the protocol. The client interprets the result as a personalized estimate of the reputation, and hence authenticity, of the object, and can then make a more informed decision to accept (and fetch) or reject the object. In cases where no votes can be found in the network, such as for new objects, the user is forced to resort to ad hoc estimates of popularity, as in existing networks. Such cases are common to the bootstrap phase of any reputation system.

The voting protocol described above works most accurately if the client can compute accurate peer correlations, which is possible only when a sufficient number of both positive and negative local votes have been cast. This provides a strong incentive for users to participate in voting.

3.4 Client Local State

Conceptually, each client maintains a *vote database* of votes it has encountered during recent vote-gather queries. Votes are stored regardless of if the client accepted the corresponding object, or whether the client's own vote, if any, agrees with the votes gathered. The database is used to respond to vote-gather queries, and as a dataset for computing peer correlations. For each object, the database contains a row with a timestamp, the client's own vote, if any, and a list of all other votes encountered for the object. Database size is bounded by retaining only recent additions, and by sub-sampling during vote collection. The resulting database size is proportional to a client's gossip rate, frequency of voting, and number of locally shared files, and independent of the number of files in the network.

Peer correlations are stored in a separate *correlation table*, which is periodically updated by scanning the vote database. For each peer in the vote database, the client determines the set of objects for which it knows both the peer's vote and its own. These votes are then used to derive a peer correlation value, with weak correlations immediately discarded. Any strong correlations discovered are cached in the correlation table for use later, both for weighing votes during estimation and for selecting which votes to send in response to vote-gather queries.

3.5 Transitive Correlation

Computing correlations directly from the local vote database works well for peers that vote on the same objects, but will not discover relationships between peers with few common interests. However, clients can leverage the correlations discovered by peers to expand their view of the network. *Transitive correlation* captures the notion that a strong positive correlation between A and B, and again between B and C, should be taken as an indication that all three peers tend to be correlated.

To effect this computation, each client maintains a directed graph in which nodes represent peers, and a directed edge (A, B) with weight r_{AB} represents a correlation between a peers A and B. Initially, a client populates its graph with the correlations it computes directly from its local vote database. The rest of the graph is built by randomly selecting peers in the network and gossiping correlation coefficients. Gossip selection is biased towards peers with known positive correlations to preferentially expand the most useful parts of the graph.

A client computes a transitive correlation by multiplying weights along a path in the graph. One way to view this computation is that votes from distant peers in the correlation graph are propagated back towards the client using weighted voting at each step of the process. As a simplification and optimization, instead of performing a computationally expensive graph flow computation, each client periodically computes only the max-path to every other node in the graph and caches the results.

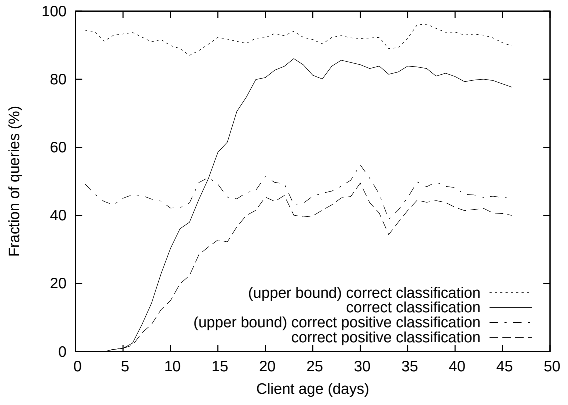

Figure 1: Classification success rate.

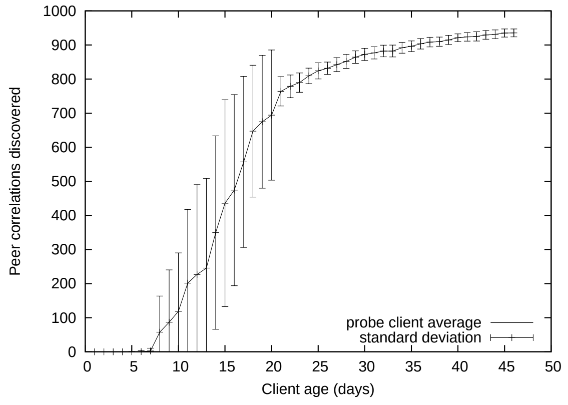

Figure 2: Correlation table size for probe clients.

Credence employs two strategies to deal with peers that lie about correlations during gossip. First, a client can request an audit of the correlations gathered during each gossip. Because peers only gossip results computed locally from the vote database, a client can request some or all of the correlation inputs and verify the reported value. Second, the local correlation graph itself can be audited, since it will typically contain significant redundancies. For instance, weights should be consistent between pairs of peers and in well connected portions of the graph.

4. EVALUATION

We evaluate Credence through simulations that approximate the behavior of filesharing users [11]. We model the system as a growing set of objects, and a fixed set of clients following a synthetic workload. We use a randomized topology with a fixed search width to model the underlying peer-to-peer network. The effects of higher level user behavior, such as intermittent network connectivity and churn, are the subject of an ongoing deployment study.

4.1 Clients, Objects, and Workload

We follow the abstract model of the Kazaa filesharing network described in [11]. The network consists of 1000 clients initiating 5 queries per day on average. Clients draw queries from a set of 40000 objects, with new objects introduced at a rate of 5475 objects per year. We model a highly polluted network, where only half the objects are authentic, and the rest consist of pollution.

In order to match the strong clustering observed in deployed networks [9], we partition the objects into 20 *genres* and randomly assign clients to 4 genres each such that genre popularity, both in terms of object and assigned clients, follows a Zipf distribution. Object popularity within a genre follows a Zipf distribution as well. A client makes a query by selecting first a genre, then drawing without repeats an object from the genre. The resulting overall popularity distribution is consistent with [11].

We add noise to model user mistakes and non-deterministic decisions as follows. Having computed a reputation estimate v, a client will accept the object if $v = 1.0$, reject if $v = -1.0$, and accept with a linearly varying probability otherwise. Clients generate votes on all objects that are accepted, but vote correctly with only 90% probability, and randomly the rest of the time.

4.2 Overall Success in Estimation

We evaluate the convergence and accuracy of our approach by tracking 20 probe clients inserted into the network starting on day 50. We first measure the total fraction of correct classifications across these probe clients. A correct classification is when a client computes a correct, strong ($|v| > 0.5$) estimate for an object.

Figure 1 shows how the success rate varies as clients participate in the system. Also shown are upper bounds, the values that would have been attained had the clients shared complete global knowledge of all votes in the system. The bound does not approach 100%, because a fraction of queries are for never-before-seen content, which cannot be classified even using global information. By day 15, our scheme achieves a high rate of success, maintaining roughly 80% correct classification.

Of the queries not classified correctly, almost all are instances where no estimate is obtained at all, and only a few are due to misclassification. These are split evenly between authentic and polluted objects, and less than 9% are for the 2000 most popular objects. This implies that our system gives very accurate estimates for the popular objects, and rarely misclassifies even unpopular ones.

4.3 Correlation Table Behavior

To explain the factors driving the overall success rate, we turn our attention to the clients' correlation tables. Estimation accuracy and robustness is dependent on the table size, since a peer's vote is used only if a correlation for the peer can be found in the table. Figure 2 shows the average size of probe clients' correlation tables, which can be seen to closely match the overall success rate observed earlier in Figure 1.

After an initial period, probe nodes discover correlations quickly, discovering on average 500 strong correlations after 16 days. For comparison, non-probe clients have somewhat smoother and slower growth, discovering 500 correlations only at day 40. The difference in behavior highlights the key role played by transitive correlations. The initial clients do not have the advantage of existing, established peers from which to gather correlation data, but a probe client can leverage the work done by earlier joining clients.

The initial lag time for probe clients is due to a lack of locally computed correlations. Until the node votes on several objects, it cannot compute local correlations and thus

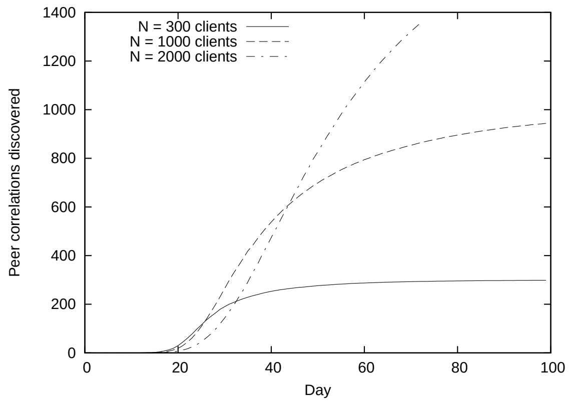

Figure 3: Correlation table size for all clients.

cannot compute any transitive correlations. Once the node establishes a few local correlations, it uses them to quickly compute many more transitive correlations. This lag time can be reduced in practice using an existing, semi-trusted "sponsor" client to establish the first few correlations.

4.4 Scaling

In order to assess the scalability of our protocol as the size of the network varies, we ran simulations with proportionally larger and smaller networks. Figure 3 shows the average correlation table size of the initial clients for each network. The total convergence time appears to grow approximately linearly in the size of the network when measured in terms of the time needed to derive correlations for half of the clients. However, note that the overall success rate grows quickly as soon as clients discover a few strong correlations, and that in practice a limited number of entries is likely to be sufficient to compute robust estimates, due to the high degree of clustering of client interests expected in practice.

4.5 Dynamic Behavior

Our system contains subtle feedback loops that give rise to resilience against attack and help explain the system's overall dynamics. For instance, an authentic object tends to follow an exponential increase in reputation, since correct positive votes induce a positive feedback cycle: Each such vote increases the object's reputation among correlated clients. This in turn leads to more likely acceptance and retrieval, and thus to additional positive votes. An attacker voting negatively on this same object, however, induces a damped response, since a lowering of perceived value will not result in more negative results, but instead will simply lower slightly the rate of acceptance.

For an object on which honest clients tend to vote negatively, but an attacker votes positively, the system will react strongly against the attacker. Each positive vote in this case raises the expectations of the honest clients, who are then more likely to download the file. This leading eventually to an increase in correct negative votes, which counteracts the initial, incorrect vote. Additionally, the attacker will see a decrease in its correlation values with honest clients. The net effect is a disincentive for an attacker to be dishonest too often.

4.6 Attacks

We have investigated the impact of several attacks on our system, both analytically and through simulation. Due to the dynamic effects discussed above, we can immediately see that several attacks that are quite effective in existing systems have little effect and, in many cases, actually provide a tangible benefit to the system. For example, a peer that consistently lies about the authenticity of objects is just as useful as a peer that consistently generates honest votes, since in the former case the votes will simply be multiplied by a negative weight. Voting randomly will lead to the votes being essentially discarded, as peer correlation values will tend to zero in this case.

A rational attacker has an incentive to vote honestly in order to keep from approaching either of these extremes, and so must carefully balance the amount of information leaked to the network. The best known attack on Credence is a *whitewashing attack*, where an attacker votes correctly on a large set of files before endorsing a small set of new decoy files. But here, the damage caused by the attacker and decoys before detection is partly offset by the larger number of correct votes required before the attack begins. Moreover, multiple independent whitewashing attacks can easily negate each other.

An attacker may try to increase the dissemination of incorrect votes, whether they come from the attacker or honest but mistaken users. This attack also turns out to be beneficial to honest clients, since it allows them to more quickly identify those peers with which they are less correlated. Similar attacks that slow down the rate of diffusion of positive votes do not alter the overall trends presented above.

4.7 LimeWire Deployment

We have deployed Credence on the Gnutella network. To date, there have been more than 4000 Credence downloads. A snapshot of users active during the third month revealed 184 active Credence users who have cast 8221 votes on 7691 files. These users form a connected component of size 113 through the common files they have voted on. The remaining active users vote only rarely, at an average of 8 times compared with 68 overall, and have little overlap with any peers. While there is much more analysis to be performed on user interaction in reputation systems, the early evidence from Credence users is that the system is able to distinguish between peers voting randomly or inconsistently and those voting correctly and honestly, and it can correctly identify pollution with few false negatives and positives.

5. SUMMARY AND FUTURE WORK

Existing peer-to-peer filesharing networks are severely polluted with decoys, malware, and other damaged content. Credence is a new object-based, decentralized reputation system to combat pollution in such networks. It requires no trusted entities in the network, resists attacks and effectively identifies trustworthy content and pollution. The Credence approach is practical; an implementation for Gnutella is freely available [4] and has an active user community. The underlying problem in filesharing networks will be common to any large network system in which clients interact with previously unknown peers. We are currently studying the emergent behavior of users in such systems, as well as the applicability of the Credence approach to domains besides filesharing.

6. REFERENCES

[1] S. Buchegger and J.-Y. L. Boudec. A Robust Reputation System for P2P and Mobile Ad-hoc Networks. In *Workshop on the Economics of Peer-to-Peer Systems*, Boston, MA, June 2004.

[2] N. Christin, A. S. Weigend, and J. Chuang. Content Availability, Pollution and Poisoining in File Sharing Peer-to-Peer Networks. In *ACM Conference on Electronic Commerce*, Vancouver, Canada, June 2005.

[3] F. Cornelli, E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. Choosing Reputable Servents in a P2P Network. In *International World Wide Web Conference*, Honolulu, HI, May 2002.

[4] Credence. http://www.cs.cornell.edu/People/egs/credence/.

[5] N. Curtis, R. Safavi-Naini, and W. Susilo. X^2Rep: Enhanced Trust Semantics for the XRep Protocol. In *Applied Cryptography and Network Security*, Yellow Mountain, China, June 2004.

[6] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks. In *ACM Conference on Computers and Communications Security*, Washington, DC, October 2002.

[7] R. Dingledine, M. Freedman, and D. Molnar. The Free Haven Project: Distributed Anonymous Storage Service. In *Workshop on Design Issues in Anonymity and Unobservability*, Berkeley, CA, July 2000.

[8] J. R. Douceur. The Sybil Attack. In *International Workshop on Peer-to-Peer Systems*, Cambridge, MA, March 2002.

[9] F. L. Fessant, S. Handurukande, A.-M. Kermarrec, and L. Massoulié. Clustering in Peer-to-Peer File Sharing Workloads. In *International Workshop on Peer-to-Peer Systems*, La Jolla, CA, February 2004.

[10] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of Trust and Distrust. In *International World Wide Web Conference*, New York, NY, May 2004.

[11] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload. In *ACM Symposium on Operating Systems Principles*, Bolton Landing, NY, October 2003.

[12] M. Gupta, P. Judge, and M. Ammar. A Reputation System for Peer-to-Peer Networks. In *ACM Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, Monterey, CA, June 2003.

[13] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *International World Wide Web Conference*, Budapest, Hungary, May 2003.

[14] J. Liang, R. Kumar, Y. Xi, and K. W. Ross. Pollution in P2P File Sharing Systems. In *IEEE INFOCOM*, Miami, FL, March 2005.

[15] LimeWire. http://www.limewire.com/.

[16] MojoNation. http://www.mojonation.net/.

[17] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *ACM Conference on Computer Supported Cooperative Work*, Chapel Hill, NC, October 1994.

[18] S. Saroiu, K. P. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Multimedia Computing and Networking*, San Jose, CA, January 2002.

[19] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer. KARMA: A Secure Economic Framework for P2P Resource Sharing. In *Workshop on the Economics of Peer-to-Peer Systems*, Berkeley, CA, June 2003.

[20] L. Xiong and L. Liu. PeerTrust: Supporting Reputation-Based Trust in Peer-to-Peer Communities. *IEEE Transactions on Knowledge and Data Engineering, Special Issue on Peer-to-Peer Based Data Management*, 16(7), July 2004.

[21] B. Yang and H. Garcia-Molina. PPay: Micropayments for Peer-to-Peer Systems. In *ACM Conference on Computers and Communications Security*, Washington, DC, October 2003.

[22] G. Zacharia, A. Moukas, and P. Maes. Collaborative Reputation Mechanisms in Electronic Marketplaces. In *Hawaii International Conference on System Sciences*, Maui, HI, January 1999.

[23] H. Zhang, A. Goel, R. Govindan, K. Mason, and B. V. Roy. Making Eigenvector-Based Reputation Systems Robust To Collusion. In *Workshop on Algorithms and Models for the Web-Graph*, Rome, Italy, October 2004.