Examining Metrics for Peer-to-Peer Reputation Systems*

Sergio Marti and Hector Garcia-Molina
Stanford University
{smarti, hector}@cs.stanford.edu

Abstract

As the popularity of resource-sharing through peer-to-peer networks increases, so does the threat of agents seeking to weaken the network by propagating bad information and services. This paper presents advantages and disadvantages of resource selection techniques based on peer reputation. We evaluate the effect of limited reputation information sharing on the efficiency and load distribution of a peer-to-peer system. We also investigate the cost in efficiency of two identity models for peer-to-peer reputation systems. Our results show that, using some simple mechanisms, reputation systems can provide a factor of 20 improvement in performance over no reputation system.

1 Introduction

The increasing availability of high bandwidth Internet connections and low-cost, commodity computers has stimulated the use of resource sharing peer-to-peer (P2P) networks. These systems employ a simple scalable mechanism that allows anyone to offer content and services to other users, as well as search for and request resources from the network. However, the open accessability of these systems make them vulnerable to malicious users wishing to poison the system with corrupted data or harmful services. Because of this danger, users must be wary of the quality or validity of the resources they access.

Determining the validity of a resource is a costly operation. Verifying the authenticity of a file or document requires downloading it from the provider. In the simplest case, a digest may be available for the file from a trusted authority who "owns" the file. Checking the file involves calculating a hash and comparing it to the digest. But often, locating the resource's authority (if one even exists) is more difficult than locating the resource itself.

This problem has led to the development of reputation systems as a means to detect misbehavior and circumvent (or punish) malicious nodes. Peer-to-peer reputation systems collect information on the trustworthiness of resource

*This research is supported in part by NSF Grant (IIS-9817799).

providers and propagate it to improve peers' chances of locating good providers.

There are two classes of misbehavior in P2P networks: selfishness and maliciousness. Selfish nodes wish to use the resources of the network without offering any themselves. Reputation-based incentive schemes aim to discourage freeriders and other selfish nodes by only offering services to nodes that reciprocate ([19] [24]).

Other systems combat malicious nodes who do not care about access to other peers' resources, but only want to propagate their invalid resources. Such systems often employ a shared history of node interactions to signal possibly malicious resource providers that users should avoid. EigenTrust [17], for example, collects statistics on node behaviors and computes a global trust rating for each node. Yet global history schemes are complicated, requiring long periods of time to collect statistics and compute a global rating. They also suffer from the transience of nodes and continual anonymity afforded malicious nodes through zero-cost identities.

In this paper, we evaluate the performance of a peer-to-peer resource-sharing network in the presence of malicious nodes, which, in contrast to global history schemes, uses only limited or no information sharing between nodes. We develop various techniques based on collecting reputation information. We present several interesting side-effects resulting from some of the techniques.

We also study the trade-offs of two identity management schemes for peer-to-peer networks: a trusted central login server and self-managed identities. We analyze the performance of each scenario and compare it to the base case with no reputation system. We look at how new nodes should be treated, and present some mechanisms to further improve system efficiency.

In Section 2 we present our system model and its assumptions. Section 3 describes the two threat models we consider. Then, Section 4 discusses the reputation systems used in the experiments and their options. Section 5 describes the metrics used for evaluating our experiments. In Section 6 we specify the details of the simulation environment used for the experiments, and present the results in Section 7. Section 8 discusses related work. Finally, we conclude in Section 9.

2 System Model

A peer-to-peer system is composed of n peer nodes arranged in an overlay network. In a resource-sharing network each node offers a set of resources to its peers, such as multimedia files, documents, or services. When a node desires a resource, it queries all or a subset of the peers in the network (depending on the system protocol), collects responses from available resource providers, and selects a provider from which to access or retrieve the resource.

Locating a willing resource provider does not guarantee the user will be satisfied with its service. Selfish peers

may offer resources to maintain the impression of cooperation, but not put in the necessary effort to provide the service. Worse, certain nodes may join the network, not to use other peers' resources, but to propagate false files or information for their own benefits.

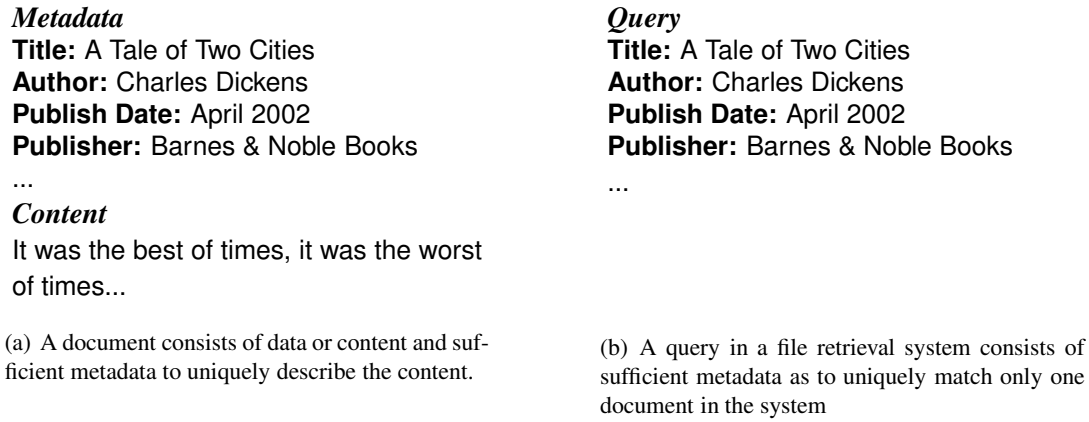
In our model, each peer verifies the validity of any resource it uses. Accessing invalid or falsified resources can be expensive in terms of time and money. A system may implement a micropayment scheme requiring users to pay a provider before being able to verify the validity of the resource. In most cases the user must wait for a file to be downloaded or a remote computation to conclude and then verify the correctness of the result. Checking the validity of the file or service response may itself be a costly but necessary operation in the presence of malicious nodes. Because such an operation is highly domain-specific, we assume the existence of a global verification function, $V(R)$ which checks whether resource R is valid. Any node can perform this verification, but it is indeterminately expensive to compute and may require human interaction (such as listening to a song after downloading it from a music service to ensure it is the correct song and uncorrupted) or even a third-party. A resource must be downloaded or accessed before it can be verified, which costs time and bandwidth. We include this cost in the verification function, so that it represents the full price of accessing a bad resource.

To simplify the discussion we present our work in the context of a file-sharing system, where users query the network, fetch files from other peers, and verify the files' content is correct. Nodes hearing the query reply to the query originator if they have a copy of the file. The originator then fetches copies of the file from the responders until a valid, or *authentic* copy is located. File-sharing networks have existed for some time and their characteristics have been thoroughly studied, allowing us to more accurately model deployed, working systems. Though we use the term "files" in the rest of the paper, most concepts apply to generic resources.

What does it mean for a file to be invalid or fake? The issue of file authenticity is discussed in the following section. The behavior of peers in the system with respect to the authenticity of the files they send each other is captured in the threat model, which is discussed in Section 3. Reputation systems, which track node behavior in order to mitigate the problem of inauthentic files are covered in Section 4.

2.1 Authenticity

In our model the unit of storage and retrieval is the *document*. Every document D consists of some content data C_D and metadata M_D which uniquely describes the content. If two documents contained the same metadata but different content, there must be some information pertaining to their differences that should be included in the documents' metadata to make them unique. For example, different editions of a book should include the edition and the year published in their metadata. Figure 1(a) illustrates a sample document for a specific edition of the Dickens' novel *A*



(a) A document consists of data or content and sufficient metadata to uniquely describe the content.

(b) A query in a file retrieval system consists of sufficient metadata as to uniquely match only one document in the system

Figure 1: Sample document and matching query

Tale of Two Cities. If the only metadata provided were the title and author, the document may not be unique, since the other editions of the same book exist in other languages, or may include notes or pictures.

Given the definition of a document, we can now define document authenticity. A document is considered authentic if and only if its metadata fields are consistent with each other and the content. If any information in the metadata does not "agree" with the content or the rest of the metadata, then the document is considered to be inauthentic, or fake. For example in Figure 1(a), if the Author field were changed to Charles Darwin, this document would be considered inauthentic, since Barnes & Noble Books has never published a book titled *A Tale of Two Cities* written by Charles Darwin that begins "It was the best of times".

We assume the existence of a global authenticity function, $A(D)$ which enables one to verify the authenticity of a document D. Evaluating the function is likely to be very expensive and may require human user interaction or even a third party. An example would be if Alice were to download a song from a music sharing service, she could determine whether it is the correct song by listening to it. We generalize the document authenticity function to the resource verification function $V(R)$. We also use the terms "file" and "document" interchangeably.

3 Threat Models

As stated above, the threat we are studying is that of a group of malicious nodes that wish to propagate inauthentic (or fake) copies of certain files. They do not care if they themselves are unable to query the system for files, thus incentive schemes fail to deter them. In addition, we assume they may pass false information to other nodes to encourage them to fetch bad files. We consider three behaviors for malicious nodes; abbreviated as N, L and C:

N: No misinformation is shared. All nodes give true opinions.

L: Malicious nodes lie independently for their own gain. They give a bad opinion of everyone else.

C: Malicious nodes collude. They give good opinions of each other and bad opinions of well-behaved nodes. For this model, we will briefly consider the situation where some malicious nodes act as "front" nodes by providing only authentic files (but never from the subversion set) in an attempt to gain the trust of other nodes and spread their malicious opinions.

The percentage of nodes in the network that are malicious is given by the parameter π_B. We propose two distinct threat models, one in which malicious nodes target specific files, and another in which they act maliciously towards other nodes by a certain probability. In both threat models, we assume no other malicious activity, such as denial-of-service style attacks, are occurring in the network.

3.1 Document-based Threat Model

The first threat model is designed to emulate expected real-world malicious activity. We randomly select a set of files, called the *subversion set*, that all malicious nodes wish to subvert by disseminating invalid copies. Each unique file has an equal probability of being in the subversion set, specified by the parameter p_B. We assume no correlation exists between a file's popularity and its likelihood to be targeted for subversion. Malicious nodes also share valid copies of files not in the subversion set. The effects on performance of varying both π_B and p_B are discussed in Sections 7.1.3 and 7.2.2.

We assume well-behaved nodes always verify the authenticity of any file they have before sharing it in the network. Though this assumption may be unrealistic for many peer-to-peer systems, experiments in which a small fraction of the files provided by good nodes were invalid demonstrated little effect on our experimental results.

3.2 Node-based Threat Model

Our second threat model performs equivalently to the former, but provides us interesting avenues of research. We choose to model the node behavior described above as a probability that a given node will send an authentic copy of a file to another node requesting the file. For example, good nodes may reply correctly 95% of the time (0.95)[1], while malicious nodes only reply correctly 10% of the time (0.1). These probabilities can be arranged in a *threat matrix*, T, where $T_{i,j}$ contains the probability that node j will reply with an authentic file to request from node i. Section 5 discusses the advantage of modelling the threat model in this form. The threat matrix characterizes the threat model at a specific time, $T_{i,j}(t)$, since nodes may behave well at first and then begin acting maliciously. Though most of

[1]The 5% accounts for the fake files that are shared before they are verified as authentic by the user

our experiments use a static threat model, some look at dynamic node behavior, such as malicious nodes behaving well for a period of time, then turning bad (see Sec. 7.3.4).

For the results in this paper we assume all malicious nodes use the same probability of replying with a fake file, regardless of the query originator. We reuse the parameter p_B to indicate this probability. Therefore, for any malicious node m, $\forall i, T_{i,m} = 1 - p_B$. Similarly, we use p_G to be the probability of a good node sending an authentic file, thus assuming that a fraction equal to $1 - p_G$ of the files on the average well-behaved node are corrupted. For any good node g, $\forall i, T_{i,g} = p_G$. In Section 7.3.2 we look at the effects of having varying node threat values among the well-behaved nodes and the malicious nodes.

4 Reputation Systems

When a node queries the system for a file, it collects all replies (and their source IDs) in a *response set*. The node repeatedly selects responses from the set, fetches the copy of the file offered by the responder and verifies it (using the verification function) until an authentic copy is found.

As nodes interact with each other, they record the outcome, such as whether the file received was authentic of not. As a node collects statistics, it develops an opinion, or *reputation rating*, for each node. We make no assumptions of how this rating should be computed, but since it is used to compare and rank nodes, it should be scalar (see below for an example).

Each node records statistics and ratings in a *reputation vector* of length n, where n is the total number of nodes in the network.[2] When a node first enters the system all entries are *undefined*. As the node receives and verifies files from peers, it updates the corresponding entry. Nodes may also share their opinions about other nodes with each other and incorporate them in their ratings. The reputation vectors can be viewed as an $n \times n$ *reputation matrix*, R, where the ith row is node i's reputation vector. Cell $R_{i,j}$ would contain node i's "opinion" of node j.

When a node has collected replies to a query, the reputation system calls a *selection procedure*, which takes as input the query response set and the node's reputation vector, and selects and fetches a file. The verification function is then calculated on the selected file. As stated earlier, this may be done programmatically if possible, but most likely requires presenting the file to the user. The system updates its statistics for the selected response provider based on the verification result. If verification failed, the selection procedure is called again with a decremented response set. This is repeated until a valid file is located, the response set is empty, or the selection procedure deems there are no responses worth selecting (such as if the remaining responders' ratings are too low).

[2] Or more accurately the number of identities in the network (see Section 4.1).

For this paper we study variants on two reputation systems, one in which peers share their opinion and one in which only local statistics are used. They are compared against a random selection algorithm.

Random Selection: Our base case for comparison is an algorithm which randomly chooses from the query responses until an authentic file is located. Since no knowledge or state about previous interactions is stored, shared or used, this algorithm models the performance of a system with no reputation system.

Local Reputation System: With this reputation system each node maintains statistics on how many files it has verified from each peer and how many of those were authentic. Each peer's reputation rating is calculated as the fraction of verified files which were authentic. This results in a rating ranging from 0 to 1, with 0 meaning no authenticity check passed and 1 meaning all authenticity checks passed. When processing a query, these ratings are used in the selection procedure to select the peer from which to fetch the file. We consider two procedures in our experiments:

- The *Select-Best* selection procedure selects the response from the response node with the highest rating. If the selected response is invalid, the procedure chooses the next highest-rated node.
- Select-Best will prefer to choose good nodes it has previously encountered and thus may overload a small subset of reputable peers. To spread out file requests we propose the *Weighted* selection procedure, which probabilistically selects the file to fetch weighted by the provider's rating. For example, if nodes i and j both provide replies to node q and $R(q,i) = 0.1$ and $R(q,j) = 0.9$, then j is nine times as likely to be chosen as i. We study load distribution in Section 7.2.3.

The Select-Best method requires a node maintain an ordered list of the most reputable nodes it knows. We call this list a *Friend-Cache* of maximum size $\overline{FC}$. There are additional benefits to maintaining a Friend-Cache in the local reputation system. By sending queries directly to nodes in the Friend-Cache before propagating the query normally, the message traffic of query floods in flat unstructured networks can be greatly reduced. We call this the *Friends-First* technique and evaluate it in Section 7.1.4.

Voting Reputation System: This system collects statistics and determines local peer ratings just as the local system does. It extends the previous system by considering the opinions of other peers in the selection stage. When a node, q, has received a set of responses to a query, it contacts a set of nodes, Q, for their own local opinion of the responders. Each polled node, or *voter* $v \in Q$, replies with its rating (from 0 to 1) for any responder it has interacted with and thus has gathered statistics. The final rating for each responder is calculated by the formula

$$\rho_r = (1 - w_Q)R(q,r) + w_Q \frac{\sum_{v \in Q} R(q,v)R(v,r)}{\sum_{v \in Q} R(q,v)} \quad (1)$$

For each responder r, the querying node q sums each voter's (v) rating of r weighed by q's rating for v. This result is the *quorum rating*. If node q has no prior knowledge of r, it uses the quorum rating as r's rating in the selection procedure. If q already has statistics from prior interaction with node r, the rating for node r is the combination of the local statistics and the quorum rating, by some given weight called the *quorumweight*, w_Q. Note that when w_Q = 0 the voting system works exactly like the local system.

Until now we have not discussed how the nodes in the quorum Q are selected to give their opinion. We consider two methods of selecting voters. The first method is to ask one's neighbors in the overlay topology. These are typically the first peers a node is introduced to in the network and, though neighbors may come and go, the number of voters will remain relatively constant. The other method is to ask peers from whom one has fetched files and who have proven to be reputable. This group would consist of the peers with the f highest local ratings at node q. The former quorum selection we call *Neighbor-voting* while the latter is referred to as *Friend-voting*. In the Friend-voting scheme we reuse the Friend-Cache described above to maintain our list of voters. The cache has a maximum size of $\overline{FC}$. We study the effects of varying $\overline{FC}$ in Section 7.2.1.

Above, we describe the source node as contacting each voter for their opinion for each and every query once it has collected the responses. Realistically, nodes may instead periodically exchange reputation vectors with each other. If the rate at which reputation vectors are exchanged is as frequent as once per query, then the two methods are equivalent. For simplicity, we assume this equivalence in our simulator and model the system as acquiring voter opinions at the time of the query.

Both reputation systems have two additional parameters. Since all entries in R are initially undefined, an initial reputation rating ρ_0 must be assigned to nodes for which no statistics are available, to be used for comparing response nodes in the selection stage. Analysis of different values for ρ_0 is provided in Section 7.1.1.

In some domains it may be easy for malicious nodes to automatically generate fake responses to queries. In situations where a node is querying for a rare document, it may receive many replies, all of which are bad. To prevent the node from fetching every false document and calculating $V(R)$, we introduce a *selection threshold value* (ρ_T). Any response from a node whose reputation rating is below this threshold is automatically discarded and never considered for selection.[3] In Section 7.1.2 we analyze the effects of varying the threshold value on performance.

In the weighted selection procedure a response from a node with a rating of 0 would never be chosen since it has a weight of 0.[4] The Weighted procedure differs from the Select-Best procedure because it may choose *any* response from the response set, with some probability (albeit small). To prevent nodes from being permanently

[3]New nodes are automatically exempt from being discarded, even if $\rho_0 < \rho_T$.

[4]A node would receive a rating of 0 if the first file fetched from it were inauthentic

excluded from the selection process by the Weighted procedure, all nodes with a reputation rating of 0 are assigned an artificial weight we call the *zero-weight* (w_0) in the selection procedure. In addition a positive value avoids issues when all nodes in the response set had a rating of 0. For the experiments performed the ideal and local Weighted reputation systems used a w_0 of 0.01 in their weighted selection procedure. The value 0.01 was chosen because it is a positive value, but significantly smaller than any other node behavior value, such as p_B. Experiments were performed using both a zero-weight of 0 (no zero-weight) and 0.01. The results were similar.

Finally, we present a prescient reputation system, which is applicable only when using the node-based threat model:

Ideal: The ideal case uses T to base its selection decision. It represents the best possible performance a reputation system can achieve by using the actual threat model in selecting the document to present. Both the Select-Best and Weighted selection procedures are evaluated for the ideal system.

This system is not realistic, but is used as a guide for the ideal performance of the previously defined reputation systems, if the reputation matrix R converges to the actual threat matrix T. Results for this system are presented in the third results section, relating to the node-based threat model.

4.1 Identity

Maintaining statistics of node behavior requires some form of persistent node identification. In order to build reputation, a user or node must have some form of identity which is valid over a period of time. The longer this period of time, and the more resistant the identity is to spoofing, the more accurately the reputation system can rate nodes [26].

The simplest way to identify a node is to use its IP address. This method is severely limited because addresses are vulnerable to IP-spoofing and peers are often dynamically assigned temporary IP addresses by their ISPs. Instead, a more reliable method may be to use self-signed certificates. This technique allows well-behaved nodes to build trust between each other over a series of disconnections and reconnections from different IP addresses. Although malicious nodes can always generate new certificates making it difficult to distinguish them from new users, this technique prevents them from impersonating existing well-behaved nodes.

Some argue that the only effective solution to the identity problem in the presence of malicious nodes is to use a central trusted login server, which assigns a node identity based on a verifiable real-world identity. This would limit a malicious node's ability to masquerade as several nodes and to change identities when their misbehavior is detected. It would also allow the system to impose more severe penalties for abuse of the system.[5]

[5]For example, a person might have to use a valid credit card to enter the system, allowing the system auditors to debit their card if they are caught misbehaving.

For simplicity, we generally assume that all nodes use the same identity for their lifetime. This mimics a system with a centralized login server, assigning unforgeable IDs based on real-world identities. This scheme ensures users cannot (easily) change identities to hide their misbehavior, by limiting each real-world entity to one network ID. In this system the trusted server need not know which system ID refers to which real-world ID [13].

The second model relies on users generating their own certificates and public/private key pairs as forms of identification. Though robust to spoofing, any user can easily discard an identity and generate a new one. Using self-managed identities makes the system vulnerable to *whitewashing*, where malicious nodes periodically change their identities to hide their misbehavior [19]. This is modelled by erasing all information gathered on a malicious node after it sends an invalid document to the query source node for verification. If node M sends node S a fake document, all information collected by nodes (including S) about M is erased. Essentially all nodes "forget" about bad nodes. We abbreviate the references to the login server and self-managed identities scenarios as *Login* and *Self-Mgd*, respectively. These two identity schemes are compared in Section 7.1 of the results.

In Section 7.2 we experiment with a slightly different whitewashing scenario. Instead of each malicious node constantly changing identities, malicious nodes whitewash periodically. We conduct experiments using both identity models and distinguish the two as the whitewashing and static scenarios. In our results, the default identity model is the static model, unless whitewashing (WW) is specified.

5 Metrics

The main objective of a reputation system is to reduce the number of documents the user must look at before finding the correct document for their query. We call this the *efficiency* of the reputation system. This is equivalent to minimizing the number of times the authenticity function is calculated in the selection stage. This metric seems the most practical and direct measure of a particular selection heuristic's performance.

While systems reduce the number of document fetches and authenticity function computations to be more efficient, it often comes at the sacrifice of *effectiveness*. The effectiveness of a search system relates to its ability to locate an answer given that one exists somewhere in the network. A reputation system's effectiveness is measured by the fraction of queries for which an authentic document is selected, given that one exists in the response set. We call this metric the *miss rate*. This measurement of effectiveness is only accurate for systems in which the reputation algorithm does not interfere with query response or query/response propagation, but it is accurate for the systems described here.

When studying reputation systems it is necessary to determine what metrics best measure the success of a

Table 1: Simulation statistics and metrics

Metric	***Description***
q_{tot}	# of queries generated
q_{good}	# of queries with an authentic file in at least one response
q_{succ}	# of successful queries where the selection procedure located an authentic file
V_i	# of verification function evaluations performed on files fetched from node i
n_G	Number of good nodes in the network
$\overline{n}_{fld}$	Average number of nodes that receive a query through flooding
q_{FC}	Number of queries successfully answered by a node in the Friend-Cache
V	# of verification function evaluations
V_G	Total number of verification function evaluations of files fetched from good nodes
r_V	Verification ratio
d_{TR}	Threat-reputation distance
r_{miss}	Miss rate
ℓ_i	Load on node i
ℓ_G	Average load on good nodes
MT_{rel}	Relative message traffic of Friends First w.r.t. flooding

particular system. Here we present the metrics we use to evaluate our experimental results. We ran simulations of our system model for a period of time and gathered statistics at the end. These statistics are used to compute the metrics. They are summarized in Table 1.

From among all the queries generated during execution (q_{tot}) we are specifically interested in the number of good queries (q_{good}) and the number of successful queries (q_{succ}). A *good query* is any query whose response set includes at least one authentic copy of the queried file, even if no authentic copy was located by the selection procedure. A *successful query* is a query that results in an authentic copy of the requested file being selected by the selection procedure. The relation between the three statistics is given by the following equation:

$$q_{tot} \geq q_{good} \geq q_{succ} \tag{2}$$

For the reputation systems we are testing, if q_{succ} always equals q_{good} then the system is considered to be 100% effective.

5.1 Efficiency

When designing reputation systems our primary concern is to reduce the number of files which must be fetched and verified before locating a valid query response. During execution we record the number of file verifications supplied by each node i, which we refer to as V_i. From this data we compute the total number of verification function evaluations, V, as

$$V = \sum_{i=1}^{n} V_i \tag{3}$$

But V alone is insufficient. A system could ignore every response, report failure on every query, and have $V = 0$. To account for the fact that some systems may incur more verification checks, but locate valid files to more queries, we divide V by the number of successful queries (q_{succ}). We call this metric the *verification ratio* (r_V).

$$r_V = \frac{V}{q_{succ}} \tag{4}$$

The lower the value of r_V, the more efficient the system is. The best possible performance would be a prescient algorithm which always chose a valid file if one was available in the response set, and ignored all responses if not. This would give an r_V of 1. The verification ratio measures the *efficiency* of a reputation system and is our principal metric of system performance.

5.2 Effectiveness

While systems reduce the number of file fetches and authenticity function computations to be more efficient, it often comes at the sacrifice of *effectiveness*. The effectiveness of a search system relates to its ability to locate an answer, given that one exists somewhere in the network. A reputation system's effectiveness can be considered to be the fraction of queries for which an authentic file is selected, given that one exists in the response set. We call this metric the *miss rate*. This measurement of effectiveness is only accurate for systems in which the reputation algorithm does not interfere with query response or query/response propagation, but it is accurate for the systems described here.

Some reputation systems with selection thresholds may not locate an authentic file even when one is available, and thus are not completely effective. We are interested in measuring how often such systems report a failure to a good query. We introduce the *miss rate* (r_{miss}), given by the equation

$$r_{miss} = \frac{q_{good} - q_{succ}}{q_{good}} \tag{5}$$

The miss rate gives the fraction of good queries that were missed. A system which returns a valid file for every good query will have a miss rate of 0. A system which never returns a good response would have a miss rate of 1. Therefore, the miss rate is inversely related to the effectiveness of the reputation system.

5.3 Load

We are also interested in measuring the load on the network under the various reputation systems and threat models. We are primarily concerned with the load on the well-behaved nodes in the network from file fetches. If each file is transferred only when it is selected to be verified, then the number of files a node has uploaded is equal to the number of verification function evaluations of files from that node. We define the load on node i (ℓ_i) as the number

of verification checks on files it supplies normalized by the total number of queries, or

$$\ell_i = \frac{V_i}{q_{tot}} \tag{6}$$

We measure the average load on the network as the average load across well-behaved nodes, ℓ_G. Let G be the set of all good nodes in the network and let n_G be the total number of good nodes. Therefore, the average load is

$$\ell_G = \frac{\sum_{i \in G} \ell_i}{n_G} \tag{7}$$

Network load is analyzed in Section 7.2.3.

5.4 Message Traffic

To measure the message efficiency of the Friends-First method we compare the network query message traffic generated by this method to the default practice in unstructured networks of flooding the network for each query. We calculate the relative message traffic as

$$MT_{rel} = \frac{\text{Number of Friends-First Messages}}{\text{Number of Flooding Messages}} \tag{8}$$

and compute it using the system parameters and the statistics gathered from the Select-Best experiments. Note that the number of Friends-First messages includes messages sent directly to friends and messages from query floods, resulting when the query goes unanswered by friends.

5.5 Threat-Reputation Distance

The final metric we introduce applies only to the node-based threat model defined in Section 3.2. If a node's reputation rating is expressed as the perceived probability that a node returns an authentic file, then the reputation matrix R, approximates the threat matrix T. Let the *standardized reputation matrix* R', be a an $n \times n$ matrix such that $R'_{i,j}$ is the probability with which node N_i expects a file from N_j to be authentic.[6] For many reputation systems R' is equal to or easily derived from R, and R' may converge to T.[7] For some reputation systems, R converges to T. How quickly convergence takes place may be a useful metric. Since R is most likely a sparse matrix, an appropriate matrix distance algorithm must be used.

We sum the square of the differences between each defined value of R' and T, take the square root, and divide by the number of defined values in R'. This metric we call the *threat-reputation distance*, or *T-R distance* (d_{TR}) for

[6] For the local and voting reputation systems we simulate $R' = R$.

[7] $T_{i,j}$ is the *a priori* probability that j sends an authentic file to i. $R_{i,j}$ is the probability with which i expects j to reply to it with an valid file based on past experience.

Table 2: Configuration parameters, and default values

Parameter	***Description***	***Default Value***
τ	Simulation runtime	1000
n	Number of nodes	10000
d_{max}	Maximum allowed degree of a node in the network	150
d_{avg}	Average degree of a node in the network	≈ 3.5
TTL	Distance from source queries are propagated	5
π_B	Percentage of malicious nodes in the network	0.3
p_G	Probability of a good node replying with an authentic file	0.99
p_B	Probability of a malicious node replying with a fake file	0.9
ρ_0	Initial reputation rating used for nodes with no prior interaction	0.4
ρ_T	Selection threshold. Nodes with reputation ratings below ρ_T are not considered in the selection procedure.	0.15
w_0	Weight assigned to nodes with a reputation rating of 0	0.01
α_F	Zipf exponent for file popularity distribution	1.2
α_Q	Zipf exponent for query popularity distribution	1.24
PQR	Number of popular queries	200
α_{PQ}	Zipf exponent for most popular queries	0.63
$\overline{FC}$	Size of Friend-Cache used with Friends-First method	-
w_Q	Quorumweight - Weight given to voters' opinions with respect to local statistics	0.1

short, and can be mathematically expressed as

$$d_{TR} = \frac{\sqrt{\sum\limits_{R'_{i,j}\text{defined}}^{n}\sum_{j}^{n} (T_{i,j} - R'_{i,j})^2}}{\sum\limits_{R'_{i,j}\text{defined}}^{n}\sum_{j}^{n} 1} \tag{9}$$

If no cells in R' are defined then the T-R distance is undefined. For the ideal reputation system the T-R distance is always 0 (by definition of the ideal system).

6 Simulation Details

The following section describes the specific component models, parameters, and metrics used in the simulations. The key parameters for the simulations are summarized in Table 2 along with their default values. Table 3 lists the various statistical distributions used, along with the default values for their parameters. The results of the simulations are presented and discussed in the following section.

We evaluate the reputation systems using our own P2P Simulator based on our system model. The simulations were run on a Dual 2.4Ghz Xeon processor machine with 2GB of RAM. Each data point presented in the results section represents the average of approximately 10 simulation runs with different seeds.

Table 3: Distributions and their parameters with default values

Description	***Distribution***	***Parameters (with default values)***
Network topology	Power-Law	$n = 10000, k_{max} = 150, \beta \approx 1.9$
Query popularity	Zipf	$\alpha_{PQ} = 0.63$, $PQR = 250$, $\alpha_Q = 1.24$
Query selection power	Zipf	$\alpha_F = 1.2$

Though most of our findings apply to any peer-to-peer network, for our experiments we construct a Gnutella-like flat unstructured network. Specifying the overlay topology is necessary for studying certain issues, such as Neighbor-voting and message traffic reduction. Studies of unstructured peer-to-peer networks have shown their topologies are power-law networks [12]. We use randomly generated, fully connected power-law networks with $n = 1000$ nodes, a maximum node degree of $d_{max} = 50$ and an average node degree of $d_{avg} \approx 3.1$.[8] Queries are propagated to a TTL of 5. For simplicity we assume the network structure does not change, though we simulate a node leaving and a new node taking its place in the network.

Each timestep a query is generated and completely evaluated before the next query/timestep. Therefore, a simulation run of 100 timesteps processes 100 queries. For the results dealing solely with the local reputation system, which does not exchange reputation information between peers, all queries are sent from a single node randomly chosen at startup. Each simulation seed selects a different node. For experiments using the voting-based system a node is randomly chosen as the query source at each timestep.

The simulation component most specific to file-sharing (as opposed to general resource-sharing) is our query model. It is similar to the one proposed in [30]. We assume a total of 100,000 unique files. The number of copies of each file in the system is determined by a Zipf distribution with $\alpha = 1.2$. Each node is assigned a number of files based on the distribution of shared files collected by Saroiu et al [27]. The query popularity distribution determines which file each query searches for. For this distribution we use a two-part Zipf distribution with an α of 0.63 from rank 1 to 250 and an α of 1.24. This distribution better models query popularity in existing peer-to-peer systems [28]. Though our query model is based on data collected on today's file-sharing networks, we expect networks providing other content or services to have similar distributions.

In Section 7.2, we model node turnover by having a random node leave the network and a new node enter on average once per query from a single node. Therefore, a turnover occurs every timestep for the single query source experiments and every 1000 timesteps for the multiple query source experiments in a 1000 node network. For the reputation system, this is equivalent to clearing all information in the ith row and column of R, when node i leaves. For the whitewash experiments in Sectio 7.1, each malicious node changes its identity after uploading a fake file to

[8] We have experimented with larger networks. Results are not shown due to space limitations but observed trends are similar to what is reported here.

any node, by clearing all the column of R relating to the malicious node. In Section 7.2, all malicious nodes change identity every 10 queries from a single node (or every 10000 queries with multiple query sources),

Unless otherwise stated, we use a selection threshold of 0.2 in all experiments reported in this paper. We use an initial reputation rating of 0 for the whitewashing experiments, and 0.3 otherwise. All experiments with constant π_B, p_B, and p_G, were run with $\pi_B = 0.3$, $p_B = 0.9$ and $p_G = 0.99$.

7 Results

The results section is divided into three components. First, we look solely at the local reputation system and compare the two identity models in detail and measure the effects of the parameters common to both reputations systems. We also evaluate the Friends-First technique for message traffic reduction.

In Section 7.2, we focus on the voting-based reputation model, look at the effects of the system parameters specific to it and also analyze the distribution of load on the well-behaved peers in the network. We also look at the effects of the malicious opinion-sharing (N, L, C).

The first two parts use only the document-based threat model defined in Section 3.1. In the final part, we look at the performance of the node-based threat model (see Sec. 3.2) and compare it to our results from the document-based threat model.

7.1 Local Reputation System

In this section we address several of the questions brought up in the previous sections. Specifically:

1. Is an initial reputation rating of zero always preferable to nonzero?
2. What is the cost in efficiency (as defined here) for using self-managed identities in lieu of a trusted login server?
3. Is there a benefit to using a selection threshold?
4. Can maintaining a Friend-Cache reduce message traffic?

Here we compare the two extreme identity models, the *Login* model, in which nodes do not change identities over time, and the *Self-Mgd* model, in which malicious nodes change identities after every fake file they upload to a peer. The experiments in this section were all conducted with no node turnover. Each simulation was run for 1000 timesteps (unless otherwise noted).

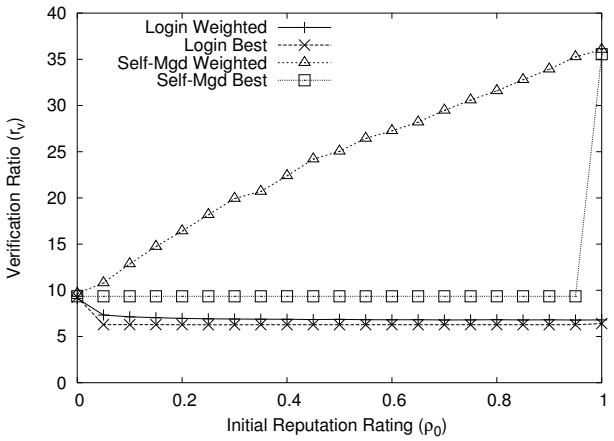


Figure 2: Efficiency for varying ρ_0. Lower value is better. 1 is optimal.

7.1.1 New Node Reputation

In this experiment we varied the initial reputation rating (ρ_0) used by the local reputation system for any node from which we have not received a document and checked its authenticity. Our experiments demonstrate that, though a reputation system performs similarly for both identity models for a ρ_0 of 0, efficiency in the login server scenario can improve substantially by increasing ρ_0, while performance in the self-managed identities scenario will only worsen.

Figure 2 shows that for the *Login* scenario, a nonzero initial reputation rating (eg. $\rho_0 = 0.4$) performs better by a factor of 1.5 in terms of minimizing the number of authenticity checks computed. If malicious nodes cannot change their identities to pose as new nodes after misbehaving, there is a benefit to selecting new nodes over previously encountered malicious nodes.

If malicious nodes are allowed to change their identities, as in the self-managed identities scenario, they will usually be treated as new nodes with a reputation rating of ρ_0 in the selection procedures. We would expect that varying ρ_0 would have a significant effect for *Self-Mgd*. Figure 2 shows that increasing ρ_0 decreases the efficiency when using the Weighted procedure, though unexpectedly, the Select-Best procedure is not affected (until $\rho_0 = 1$). For example, from a ρ_0 of 0.0 to 0.5, the verification ratio (the average number of authenticity checks performed per query) of the Weighted method goes from 9.7 to 25.1, while Select-Best stays constant at 9.4. Since the Weighted method considers all nodes (weighted by their ratings) in the selection stage, it is important to lower the weight of new nodes, which are more likely to be malicious nodes in the scenario of self-managed identities than in that of a login server. The results support our intuition. The Select-Best method's unvaried performance across all values of ρ_0 can be attributed to the fact that often a node receives a reply from a peer which has previously provided an authentic document, in which case the node will always choose the reputable source over any unknown peer.

From these experiments we selected 0.3 as the default value for ρ_0 for *Login*. Many of the following experiments

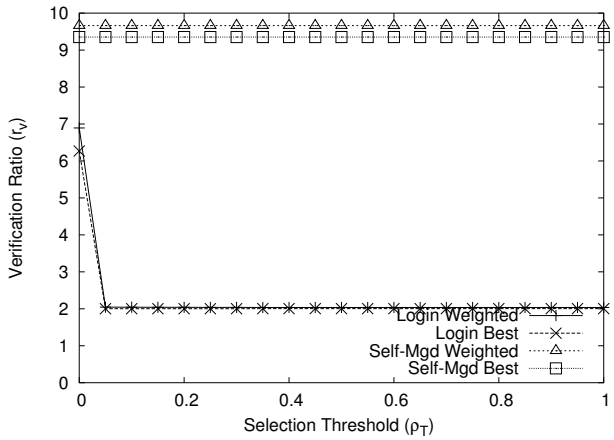


Figure 3: Varying selection threshold values.

were additionally performed with other values of ρ_0, but the results did not vary noticeably from those at $\rho_0 = 0.3$ and are not discussed. For *Self-Mgd* simulations we use only ρ_0 = 0, which clearly performed best for the Weighted method.

7.1.2 Selection Threshold

Figure 3 shows tests varying the value of the selection threshold for both the Weighted and Select-Best variants of the local reputation system. The verification ratio is plotted as a function of ρ_T. As stated above, ρ_0 was set to 0.3 for *Login* and 0 for *Self-Mgd*.

The result is surprising. For *Login* all values of ρ_T above 0 resulted in almost equal performance, yet significantly better than ρ_T = 0 (r_v of 6.3 for ρ_T = 0 down to 2.0 for $\rho_T > 0$).[9] Because malicious nodes always reply with a copy when a document in the subversion set is queried for, the vast majority of responses in the response set come from malicious nodes supplying bad copies. When searching for rare content, it is common to receive only bad copies from malicious nodes. The threshold prevents nodes from repeatedly fetching and testing documents from peers which have proven malicious or unreliable in the past. The drawback of the selection threshold is a decrease in query effectiveness (discussed in the following section).

For *Self-Mgd* varying ρ_T had no effect. Remembering which nodes have lied in the past is of no use if those nodes can immediately change their identities to hide their misbehavior. The threshold may be useful if nodes were motivated to maintain their identities, perhaps by providing incentives for building reputations.

In successive tests any system variant using a selection threshold uses a ρ_T value of 0.2 unless otherwise stated.

[9]Though almost the same, the values of r_v for different nonzero ρ_T for a given reputation system variant are not exactly identical.

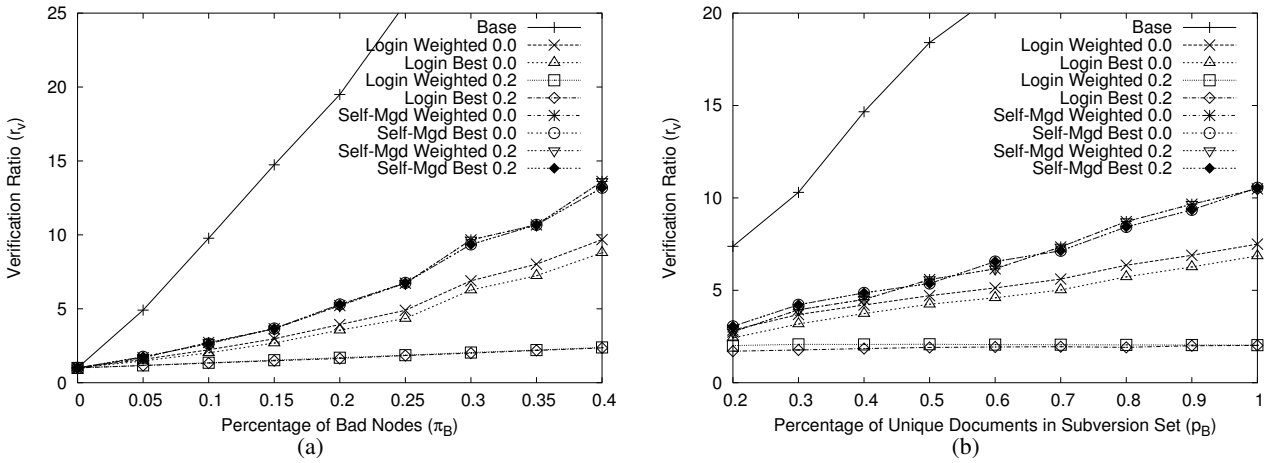


Figure 4: Efficiency comparison.

7.1.3 Performance under Various Threat Conditions

In this section we look at system performance under different threat model parameter values. Specifically we demonstrate how overall efficiency is affected by varying the percentage of malicious nodes in the system (π_B) and the probability of a unique document being in the subversion set (p_B). Eight different variants of the local reputation system were tested. These eight variants are derived from three system parameters: the identity model (*Login* or *Self-Mgd*), ρ_T (0 or 0.2), and the selection procedure (Weighted or Select-Best).

The graphs in Figure 4 present the system performance for varying π_B and p_B. The results show that overall a trusted login server significantly reduces the cost of insuring authenticity over self-managed identities roughly by a factor of 5.5. Yet, using a reputation system with the *Self-Mgd* model outperforms having no reputation system at all (Base curve in Figure 4) by an additional factor of 3.5.

For both graphs, the curve corresponding to the base case, of purely random selection, quickly climbs out of the range of the graphs. In Figure 4(a) the base curve increased steadily to 46 at π_B = 0.4; 3.5 times the verification ratio of the *Self-Mgd* variants, and up to 20 times the r_v of *Login* using a selection threshold. In Figure 4(b) the base curve climbed to 35 at p_B = 1; resulting in 3.4 times the r_v of the *Self-Mgd* variants, and approximately 17 times that of *Login* with ρ_T = 0.2. This means one would expect to have to fetch and test on average 20 times as many query responses in order to find a valid response! Even using self-managed identities, a rudimentary reputation system provides significant performance improvements over no reputation system. Even then users would expect to fetch over ten bad copies for every good copy they locate (for $\pi_B > 0.3$). In contrast, a peer using a selection threshold in a login server environment would only expect to encounter one or two fakes for every authentic file, no matter the level of malicious activity in the network.

Figures 4(a) and 4(b) show that the Select-Best and the Weighted procedures perform similarly. Overall the Select-Best method outperformed the Weighted method, especially in the *Login* model. Though the Select-Best performed well and served to mitigate the performance variance of other parameters (such as the initial reputation rating), it does have drawbacks. A study of the load on well-behaved nodes (measured as the number of documents fetched *from* a node) showed a much more skewed distribution for the Select-Best variants than the Weighted variants. In fact, the highest loaded good nodes in the Select-Best simulations were being asked for 2.5 times as many documents as the highest loaded nodes in the Weighted simulations. At the bottom of the distribution, hundreds of nodes were never accessed in the Select-Best simulations that were in the Weighted simulations. This dramatic skew in load distribution can result in unfair overloading, especially in a relatively homogeneous peer-to-peer network. We study load distribution in detail in Section 7.2.3.

Both graphs illustrate that the selection threshold is useless in the *Self-Mgd* scenarios, but provides a large performance boost for *Login*. This supports our findings in the previous section, and demonstrates it was not an artifact of the selected values of the threat parameters. Using a selection threshold system efficiency is relatively unaffected by variations in π_B and p_B.

Measurements of effectiveness in these experiments (only applicable to a nonzero selection threshold) resulted in a miss rate well below 0.001 (0.1 of 1%) for the experiments varying π_B at a constant p_B of 0.9. For the experiments varying p_B, the miss rate increases as p_B decreases, but always remains below 0.0025 (0.25 of 1%). As p_B decreases, the subversion set decreases. Because malicious nodes become more likely to provide authentic documents, but tend to fall under the threshold, the effectiveness of the system deceases. For most applications these miss rates are acceptable, especially when compared to the increased efficiency offered by the selection threshold.

7.1.4 Message Traffic

Now, we present our experiments on mitigating message traffic using the Friends-First technique. As explained earlier, Friends-First takes advantage of the Friend-Cache to try and locate a positive query response among the known reputable nodes, before querying the entire system. As we will see, in a flood-based querying system, this can result in 85% less message traffic!

Before presenting the results, we redefine the general formula for relative message traffic, given in Equation 8, in terms specific to our model. The numerator is the total message traffic for Friends-First. For all queries, messages are sent to all nodes in the Friend-Cache ($q_{tot} \cdot \overline{FC}$). In addition there is the cost in messages of flooding the network when a valid response is not located from the Friend-Cache. The number of messages generated in the network to propagate a query will be at least equal to the number of nodes which hear the query, and most likely much larger

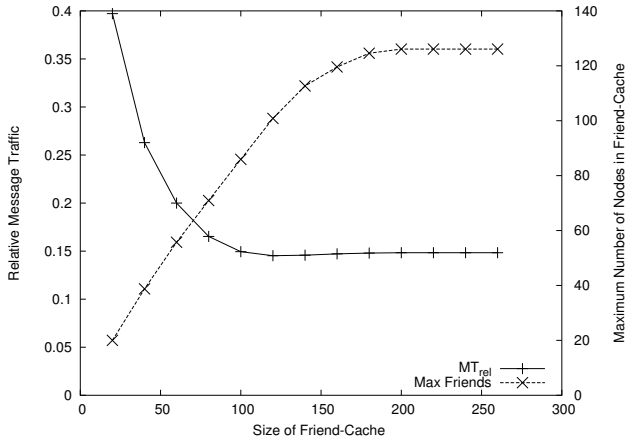


Figure 5: Relative message traffic of Friends-First and maximum Friend-Cache utilization as a function of the size of the cache.

due to several occurrences of two nodes forwarding the query to the same node. We roughly estimate the number of messages generated by a query flood as the average number of nodes reached by a query flood ($\overline{n}_{fld}$). Therefore, the additional cost of flooding for Friends-First would be the number of queries *not* answered by a node in the Friend-Cache ($q_{tot} - q_{FC}$) times $\overline{n}_{fld}$. The denominator is the number of messages generated assuming every query is a flood ($q_{tot} \cdot \overline{n}_{fld}$).

Note that $\overline{FC}$ is greater than or equal to the actual number of nodes in the Friend-Cache at any time, so not all queries will have $\overline{FC}$ nodes to query directly. Let $\overline{FC}_i$ be the number of nodes in the Friend-Cache after $i-1$ queries. $\overline{FC}_i$ is the number of messages sent directly to reputable nodes for the ith query. Note that for all i $\overline{FC}_i \leq \overline{FC}$ and $\overline{FC}_1 = 0$ since all nodes are initially unknown. We can define our message traffic metric as

$$MT_{rel} = \frac{\sum_{i=1}^{q_{tot}} \overline{FC}_i + (q_{tot} - q_{FC})\overline{n}_{fld}}{q_{tot} \cdot \overline{n}_{fld}} \tag{10}$$

Note that this is still a conservative calculation of relative traffic since $\overline{n}_{fld}$ is less than or equal to the total number of messages generated due to a query flood.

We conducted these experiments using the local reputation system and the single-source query generator. For the results in this section, we ignored whitewashing and node turnover. We ran simulations for various numbers of queries (1000, 10,000, 50,000, etc).

The solid line in Figure 5 plots the relative message traffic of Friends-First with respect to regular flooding (MT_{rel}) as a function of the maximum Friend-Cache size, after 50,000 queries. We see that, as the size of the Friend-Cache increases, the query message traffic drops quickly to approximately MT_{rel}=0.15 until it reaches a point where growing the cache no longer provides any benefit. This means that the Friends-First method is generating only 15% as much message traffic as flooding without any loss in effectiveness!

Interestingly, for cache sizes greater than 120, the traffic overhead actually *increases* slightly, before levelling off

at around 200. For small $\overline{FC}$, increasing the cache size greatly reduces message traffic because of the high likelihood of locating future query answers at the additional nodes stored in the cache. Every additional query satisfied by a node in the cache saves the system a query flood, outweighing the cost of the additional messages sent to the new nodes in the Friend-Cache for every query. But when $\overline{FC}$ is large, any node added to the cache will likely be sharing few files (thus rarely provide a response in the future). If the node had more files, it would have been located earlier and already be in the Friend-Cache. We find that well-behaved nodes sharing many files tend to be located quickly and be placed in the Friend-Cache early. Nodes added later offer fewer (approx. 5) files and do not offer any more query responses, thus wasting bandwidth on query messages sent directly to them.

As stated earlier, we performed experiments for varying lengths of time. In our shorter simulations (e.g. 1000 queries) there were no rise in relative traffic for large $\overline{FC}$. Instead MT_{rel} drops quickly and levels off, with no single minimum. These shorter simulations end before the Friend-Cache begins collecting useless nodes with very few files. Runs of 20,000 and 100,000 queries, on the other hand, also showed a preferred $\overline{FC}$ around 130. This result supports our hypothesis that, once only small nodes remain outside the cache, adding a node to the cache increases overall traffic because the cost of sending them a direct query outweighs the slim probability of their answering a request and avoiding a query flood.

In studying the efficiency of Friends-First, it is useful to consider the utilization of the Friend-Cache. The right y-axis of Figure 5 corresponds to the number of reputable nodes in the Friend-Cache when the simulation ended, represented in the graph by the points on the dashed line. Notice that the number of nodes in the cache increases linearly until MT_{rel} reaches the minimum, and levels off when MT_{rel} levels off. Interestingly, the value it reaches is 126, approximately the same value as the optimal cache size. We believe this is not a coincidence. This value is an average of several simulation runs with different seeds. Some runs had lower values and others higher, but it does indicate that, on average, the system did not use responses from more than 130 reputable nodes. Thus, in the simulations where more than 130 reputable nodes were located and placed in the cache, we would not expect them to provide any further useful unique responses. Therefore, limiting the Friend-Cache to a size of 130 prevents useless nodes from entering the cache and worsening performance.

7.2 Voting-System

In this section we discuss the following important issues:

1. How well does the voting-based system perform? How do the parameters w_Q and $\overline{FC}$ affect the performance?

2. How does the voting system compare to the local reputation system? Remember that the voting system with a

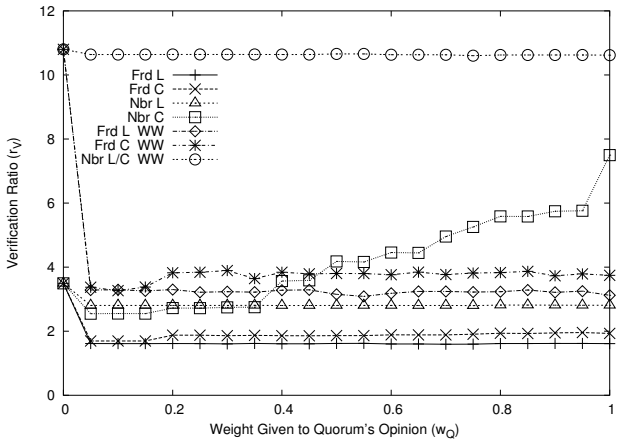


Figure 6: Efficiency of the voting reputation system with respect to varying quorumweight (w_Q). Lower r_V is better. 1 is optimal.

quorumweight of 0 is equivalent to the local system.

3. How do the Select-Best and Weighted methods compare in terms of overall efficiency?

4. How does the reputation system affect the distribution of load across well-behaved nodes?

As stated before, all experiments use the document-based threat model. We also slightly relax the whitewashing scenario. Instead of a malicious node changing identities after each false file upload, all malicious nodes whitewash after after an average of 10 queries per node in the network. To underscore this difference we refer to the two identity models as the whitewash (WW) and no whitewash scenarios, as opposed to *Login* and *Self-Mgd*, as done in the previous section.

7.2.1 Voting System Parameters

In this section, we analyze the performance of the voting-based reputation system for various parameter values. All experiments were performed using the multi-source query generator for a total of 100,000 queries. For this scenario the random algorithm obtained an r_V = 28.2, off the scale of the graphs. The relative performance of the local reputation system is given by the data point for a quorumweight of 0.

Figure 6 presents the effects of varying the quorumweight, w_Q. It shows results for both with whitewashing (WW) and without, both Neighbor (Nbr) and Friend (Frd) voting, and both the selfish lying (L) and colluding (C) malicious opinion-sharing models. No (N) opinion-sharing misbehavior mirrored the L curves, performing only marginally better across all experiments, and are not graphed. In the selfish lying model, malicious nodes give themselves a rating of 1 and all others a rating of 0. Since malicious nodes cannot vote for themselves and give everyone else an equal rating of 0, they do not greatly impact a vote in favor of malicious nodes.

Note that the values of r_V in Figure 6 are relatively high. For example, an r_V value of only 3 means we would expect to download and verify three files for each query. In an actual system, it may not be feasible for a node to thoroughly check each downloaded file's authenticity. The node may simply trust the file to be valid. In this case, r_V can be viewed as the inverse probability that such a file is valid. Accounting for well-behaved nodes offering bad copies of files complicates the threat model. We have conducted experiments with this assumption and, as long as the probability of a good node offering a bad file is small, it does not noticeably affect our results.

Observing the drop in r_V from $w_Q = 0$ to $w_Q = 0.05$, we conclude that incorporating other nodes' opinions tends to improve the efficiency of the system. Except when malicious nodes collude to subvert the voting process, varying the weight of the voters opinions beyond $w_Q = 0.05$ has no effect on the system performance. This behavior indicates that the greatest benefit from voting is in the situation where the local node has no opinion of their own. When bad nodes collude (C), system performance decreases as the weight given to the quorum's opinion increases, reinforcing that there is no substitute for personal experience in an untrusted environment.

Comparing the Frd family of curves to the Nbr curves within the same whitewash scenario (e.g. $Frd\ L$ vs. $Nbr\ L$), we clearly see that Friend-voting outperforms Neighbor-voting. Nodes that have given you good service in the past have demonstrated some effort to be reliable and well-behaved. Asking them for their opinions is more reasonable than relying on one's neighbors, a third of which, in this scenario, are likely to be bad. Not only does Neighbor-voting not perform as well, but it is more susceptible to malicious collusion as neighbors' opinions are given more weight (see $Nbr\ C$ curve). Friend-voting, however, tends to avoid asking malicious nodes for their opinions, mitigating the effects of collusion.

Though the whitewash scenario performs worse than no whitewashing, it can benefit more from opinion-sharing. As the $Frd\ WW$ curves between $w_Q = 0$ and $w_Q > 0$ illustrate, efficiency for Friend-voting improves by a factor of 3 over the local reputation system. The $Nbr\ L/C\ WW$ curve shows that Neighbor-voting in the WW scenario is almost completely unaffected by opinion-sharing, no matter the malicious opinion-sharing model. As stated before, in the WW scenarios an initial reputation rating of 0 is assigned to unknown nodes. Since this value is used for weighing the opinions of the voting nodes, any unknown peer in the neighbor quorum (including malicious nodes that have whitewashed) will have their votes ignored. Because the average number of neighbors is small (approx. 3.1) the probability of a well-behaved neighbor providing a query response that is tested, and thus becoming "known" and having their opinion used, is rare. In contrast, in the no WW scenario, since ρ_0 = 0.3, even untested peers' opinions are considered, explaining its poor performance when bad nodes collude.

In summary, this experiment shows that choosing a relatively small quorumweight around 0.1 with Friend-voting improves performance by a factor of 2 or more across all scenarios. But how many reputable nodes should one keep

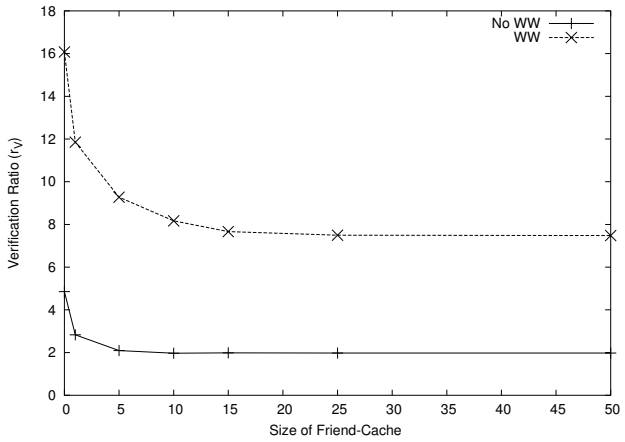


Figure 7: Efficiency of the voting reputation system with respect to Friend-Cache size ($\overline{FC}$).

in the Friend-Cache? Does increasing the size of the Friend-Cache always result in better efficiency? In a real system, a larger cache means greater maintenance cost periodically checking the liveness of the nodes in the cache. Is this cost always justified?

Figure 7 shows the performance of both the whitewashing and no whitewashing scenarios for various Friend-Cache sizes ($\overline{FC}$) with no bad opinion-sharing (N).[10] Both scenarios stabilize so that increasing the size of the cache yields no performance improvement, but a system dealing with whitewashing benefits from a larger cache. For instance, while a Friend-Cache of 10 is sufficient when there is no whitewashing, the whitewash scenario can benefit from a cache as large as 25. As expected, when tested with the malicious opinion-sharing models (N, L, C), all three models produced similar r_V values, with the C values being slightly greater than that of the N and L values by about 0.4 in the no WW scenario. Thus, we only plot the N curve in Figure 7. A surprisingly small cache is needed for this technique to be efficient.

Friend-voting is effective against collusion because it only considers the opinions of nodes that have shown to behave well by providing good files. Given our threat model, this quickly bars malicious nodes from the Friend-Cache. One technique malicious nodes may employ to defeat Friend-voting would be to set up *front nodes*. These nodes properly trade only authentic files, but when asked for their opinion of other nodes, act according to the collusion model, C, promoting only malicious nodes.

We have run simulations where a fraction of the malicious nodes are set to be front nodes. We present the results for both a quorumweight of 0.1 and 0.8 in Figure 8. These experiments show that, in the case of $w_Q = 0.8$, front nodes can cause considerable harm to the system. The damage peaks when 40% of the malicious nodes are front nodes, decreasing the system performance by more than a factor of 3! For a larger number of front nodes,

[10] $\overline{FC}$ = 0 corresponds to the local reputation system

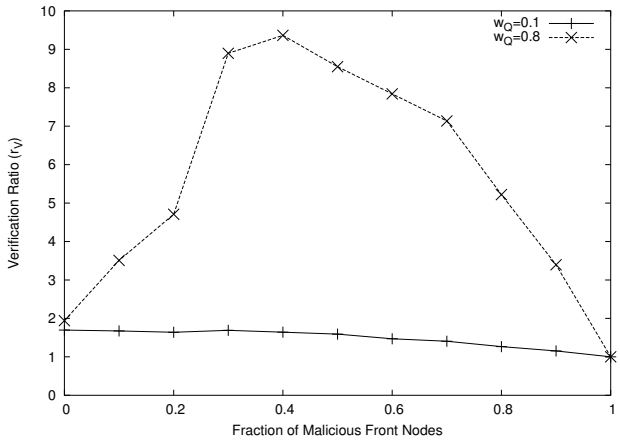


Figure 8: Effects of front nodes on efficiency.

r_V steadily drops, indicating that too many malicious nodes are behaving well to promote a smaller group causing actual damage. To be optimally effective, attackers would need to use the right balance of front nodes and actively malicious nodes. Surprisingly, front nodes appear to have no adverse effect when $w_Q = 0.1$. We believe this shows that a very low quorumweight limits the impact of front nodes' bad opinions sufficiently that the damage caused by front nodes is negated by the benefit of having fewer actively malicious nodes.

7.2.2 Efficiency Comparisons

Given the results of our analysis on the voting parameters, we wish to evaluate the system with respect to varying threat parameters. Specifically, we demonstrate how overall efficiency is affected by varying the percentage of malicious nodes in the system (π_B). We have run similar experiments varying the probability of a unique file being in the subversion set (p_B) and obtained similar results and performance comparisons.

We test the voting system with $w_Q = 0.1$ and $\overline{FC} = 10$, and using the two selection procedures both with and without whitewashing. These experiments were also run using the multi-source query generator for 100,000 queries. We evaluate the efficiency of the reputation systems for values of π_B between 0 and 0.4 using the default p_B of 0.9.

It may seem unlikely that a network would have 40% malicious peers attacking 90% of the files. But in the real world, there are large entities, with access to vast resources, which have an interest in subverting peer-to-peer networks. We have simulated across several degrees of malicious activity (varying both π_B and p_B) and the relative performance of the different reputation system variants is comparable in weaker threat scenarios to those presented here.

Figure 9(a) shows the performance of the voting reputation system. Clearly, using any local statistics when selecting a provider results in significantly better efficiency than purely random selection (base case). While the

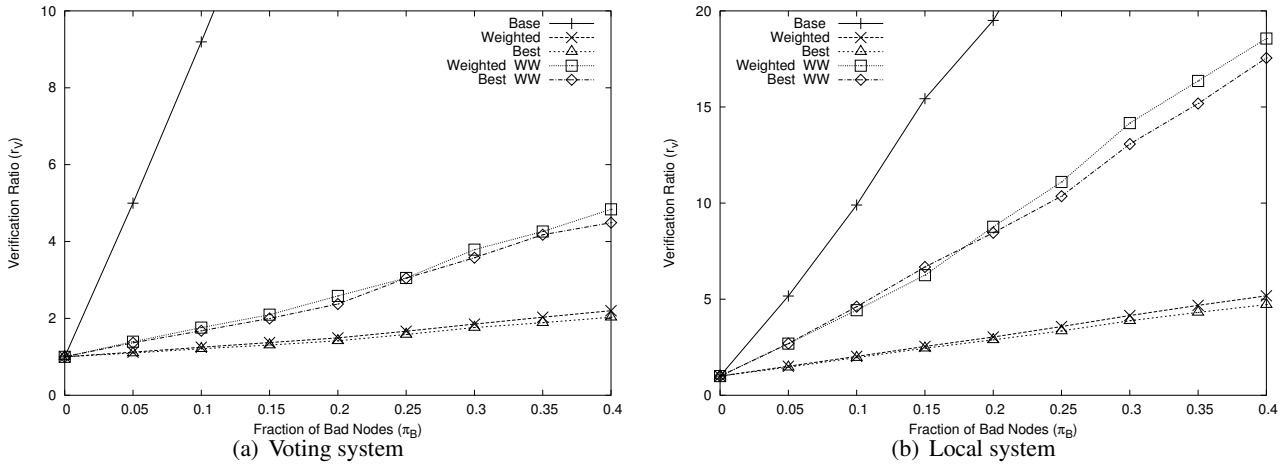


Figure 9: Comparison of the efficiency of the two reputation systems with the random algorithm as a function of π_B.

base case climbed to 42.5 at 40% malicious nodes, the voting reputation system attained an efficiency of 2 (with no whitewashing), a factor of improvement of 21! Whitewashing adversely affects the performance of the system, but not as badly as expected. For example, with a verification ratio of 4.5 the reputation system in the whitewash scenario performs 2.3 times worse than when there are no whitewashers. This means that on average a node would have to fetch more than twice as many copies of a file before finding a valid one, showing a clear advantage to preventing whitewashing by requiring users to log in through a trusted authority that can verify each real user has only one system identity.

We also executed the experiments using the local reputation system under equivalent conditions (100 queries from a single querying node). The results, shown in Figure 9(b), were only a factor of 2 worse performance than the voting system in the non-whitewashing scenario.[11] The performance difference between the two systems was greater in the whitewashing scenario, a factor of 4. These results support our findings in Section 7.2.1 that opinion-sharing is worthwhile in spite of its slightly higher implementation complexity.

When comparing the performance of the Select-Best (*Best*) and Weighted selection procedures in either graph of Figure 9, we see no large efficiency advantage of one procedure over the other, though the Select-Best method outperforms Weighted across all values of π_B. As expected, selecting the best known provider is slightly more efficient than probabilistically choosing a provider, but this comes at a cost, which we discuss in the following section.

[11] Note the difference in scale between the two graphs in Figure 9.

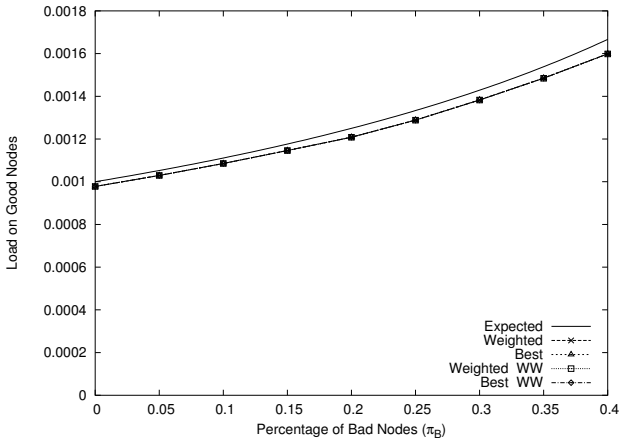


Figure 10: Average load on well-behaved nodes as a function of p_B.

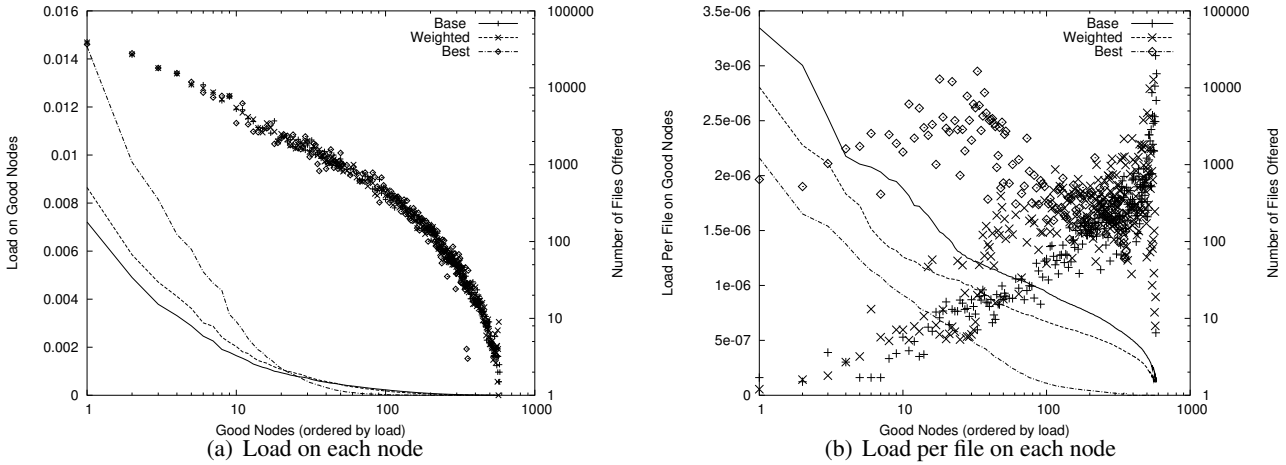
7.2.3 Load on Good Nodes

One critical issue is that reputation systems may unfairly burden some of the good nodes in the network. Thus, we now look at the amount of load placed on well-behaved nodes in the network in terms of the number of files they upload. We are interested only in the effect produced by requests from well-behaved nodes running the algorithms correctly. We use the same setup as above but concentrate on the scenario with no whitewashing.

Figure 10 plots the average load on the well-behaved nodes, as a function of the fraction of malicious nodes in the network. In an ideal system with no malicious nodes, we would expect exactly 1 download per query, giving a value of ℓ_G = 0.001 for a 1000 node network. In our case, when there are no malicious nodes, the value of ℓ_G is 0.00098. This value is less than expected (shown by the *Expected* curve in the graph) because a few queries go unanswered by any node in the network.

As the fraction of malicious nodes increases, so does ℓ_G. For instance, when $\pi_B = 0.3$ the average load is 0.00138. With only 70% as many good nodes to service requests, we would expect $\ell_G = \frac{1}{0.70 \cdot 1000} = 0.00143$. Both the fact that malicious nodes provide some good files, and that the probability of a successful query is lower, account for the difference between the observed and expected loads. Comparing the two selection procedures shows an insignificant difference in average load. Both procedures fetch the same number of files from good nodes overall.

Though there was little difference between the selection procedures in terms of average load, it is important to consider the load distribution. In a homogeneous network where all nodes have similar bandwidth, it is preferable if load is distributed evenly across all nodes, as opposed to a few nodes handling most of the traffic while the majority are idle. To study load distribution we measured the load (using Eq. 6) on each individual node using the two voting-based reputation system selection procedures and the random selection algorithm. The values were then sorted in descending load order. The results, averaged across 10 runs with different seeds, are shown by the three line curves



(a) Load on each node (b) Load per file on each node

Figure 11: Distribution of load on good nodes (and their corresponding number of files shared).

on the left y-axis in Figure 11(a). Here we see that, using the Select-Best selection procedure, the most heavily loaded node (with rank 1) has a load of almost 0.015. This value is more than 10 times the average load of 0.00138.

Though both selection procedures incurred greater load on the highest ranked nodes than the base case, Select-Best concentrated the load on a few nodes while Weighted distributed the load better. The maximum load on a node with the Select-Best method was almost twice that of Weighted. This is expected since Select-Best locates a few good nodes and tries to reuse them when possible, while the Weighted model encourages fetching files from new nodes (broadening its pool of known good nodes). If load-balancing in a homogeneous system is an important requirement, then the Weighted selection procedure would be preferable.

Another factor to consider is how load relates to the number of files shared by each node. It would be expected that good nodes with more files are more likely to be able to answer queries, increasing the number of files they upload and thus their load. Figure 11(a) plots as points the number of files on each good node on the right-hand y-axis. For example, for rank 1, there are three points around 38,000. This means that, for all three systems, the most heavily loaded node shared an average of around 38,000 files. As expected, all distributions show a strong correlation between nodes sharing more files and higher load.

In Figure 11(b) we divide the load on each node by the number of files it provides and reorder the distribution. For instance, the node at rank 1 has a load *per file* of 2.8×10^{-6} for the Weighted selection procedure, but only 2.2×10^{-6} for the Select-Best procedure. The result is surprising. The Select-Best method generated much less load per node than the Weighted or random methods. To understand this result we again plot the number of files offered by each node on the right y-axis. Here we see two trends. The base case and the Weighted method both curve from the bottom left upwards, showing that the nodes with highest load per file offer very few files. This effect is due to the sublinearity of the answering power of a node with respect to the number of files it is offering. For example, if

node i has twice as many files as node j, we expect node i to be able to answer less than twice as many queries as j. In general, given a probability p that any individual file in the system matches a query, the probability that a node with f files can respond to a query equals $1-(1-p)^f$. In a purely random selection model this probability is an indicator of the expected load on a node; as f increases, so does the probability, and thus the likely load. This is corroborated by our results in Figure 11(a). Now if we divide this probability by f we have an indicator for the load per file: $\frac{1-(1-p)^f}{f}$. This equation has a maximum value when $f=1$ and decreases as f increases. This explains the behavior we see from the random base case and the Weighted case in Figure 11(b).

The Select-Best method, on the other hand, shows a different trend. The most heavily loaded (per file) nodes share a very large number of files. The Select-Best procedure selects nodes which have proven reliable in the past. This behavior favors well-behaved nodes which respond to queries early in the simulation and often, nodes sharing many files. This procedure gives nodes with many files an even greater chance of being chosen with respect to the random model.

Whether or not it is desirable to send greater traffic to nodes with more files is dependent on the environment. Some have suggested that in some peer-to-peer systems, the number of files a node offers correlates to its available bandwidth. If so, using a selection procedure that gives preference to nodes with more files may result in more effective bandwidth usage.

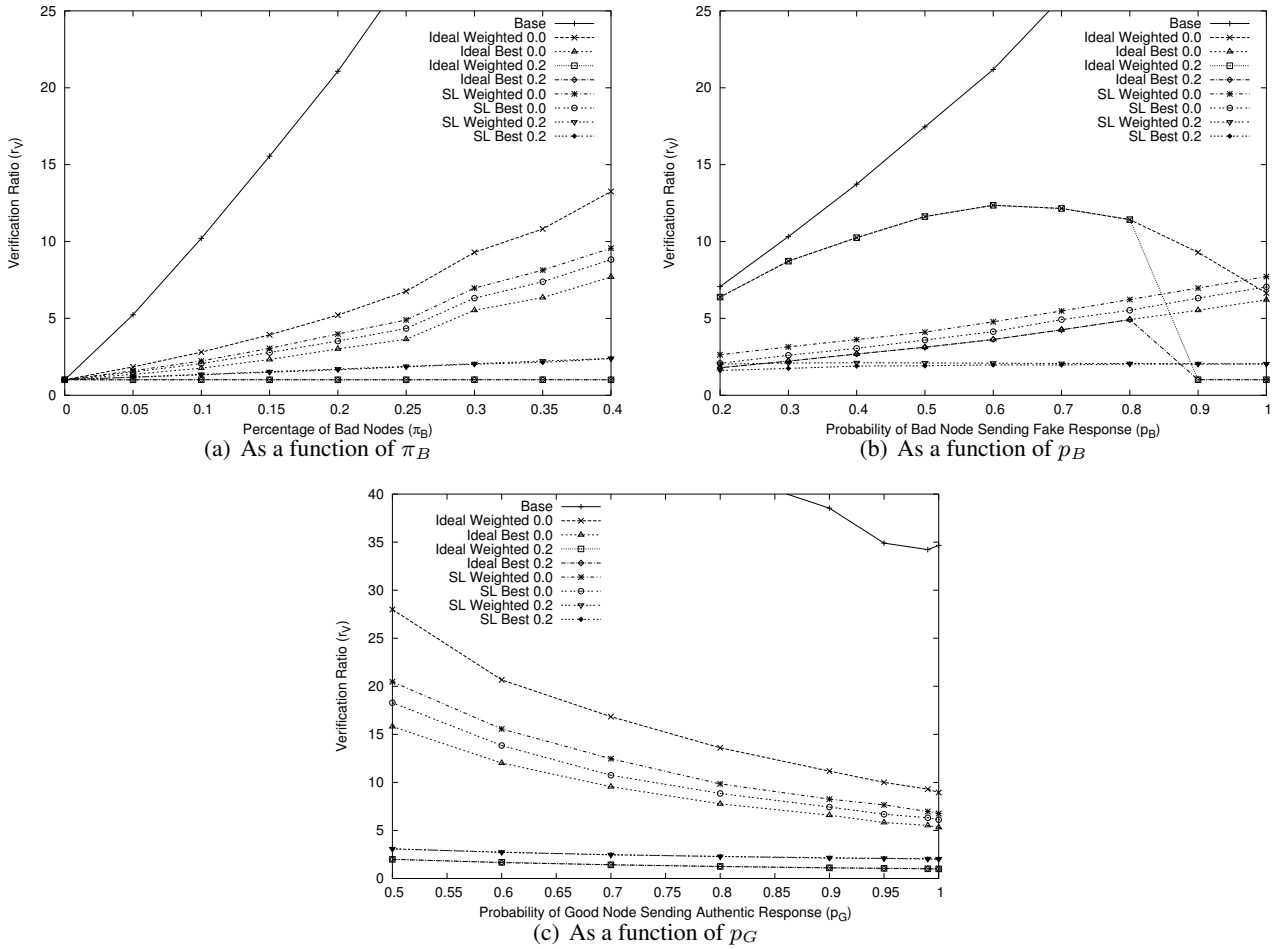
7.3 Node-based Threat Model

Here we present the results of experiments using the threat model, based on the threat matrix. We evaluate the local reputation system and the ideal reputation system, which uses the threat matrix T as its reputation matrix. All results in this section were performed with no node turnover or whitewashing.

This threat model allows us to perform a statistical analysis of the expected long-term performance of the reputation system variants, which we present in Appendix A. This analysis presents the expected system behavior in steady-state after running for sufficiently (perhaps infinitely) long. Comparing the analytical results with those presented in the next section, gives us an understanding of the inherent limitations of each of reputation system variants. The statistical analysis results assume no node whitewashing or node turnover. For more information please see the appendix.

7.3.1 Efficiency of the Reputation Systems

We first present the results of varying the three threat parameters, π_B, p_B and p_G.



(a) As a function of π_B

(b) As a function of p_B

(c) As a function of p_G

Figure 12: Comparison of the efficiency of the local and ideal reputation systems under the node-based threat model. Lower is better. 1 is optimal.

Figure 12 shows that the local reputation system performs quite well, matching the associated ideal system, and even surpassing it in some situations. Though the Weighted and Select-Best methods achieve equal efficiency, the use of a selection threshold dramatically improves performance, allowing it to maintain a verification ratio under 2.5.

The ideal system likewise benefited from a selection threshold, allowing the system to maintain a near perfect ratio of approximately 1.01. The improvement in performance due to the threshold reinforces our observations of the large number of queries that returned no authentic documents, only fakes. As expected, the selection threshold for the ideal case is useless once $p_B > \rho_T$ and it performs as if there is no threshold (Figure 12(b)).

The most interesting observation is the shape of either Ideal Weighted curve. We see that the verification ratio peaks when p_B is around 0.6. The reason the Ideal system performs badly in this situation is because, knowing the values of the threat matrix, it expects bad nodes to reply correctly to 40% of the queries. In reality, malicious

nodes reply falsely to around 60% of the queries, but only reply correctly to a small fraction of the other 40% of the queries. This is because it can only return a valid response to a document it has, and each file has only a small fraction of all the unique documents in the city. This shows that the "ideal" reputation system is not as good as we may have originally believed.

Comparing the SL curves in the graphs in Figure 12 to the corresponding ones the document-based threat model and the login server identity model, shown in Figure 4, we see that relative performance stays same. The behavior the systems under both threat models is quite similar, especially regarding varying the common threat parameters, π_B and p_B. In fact, we repeated the experiments for determining the optimal values for the initial reputation rating and selection threshold (under both identity models) but the results were so similar, we feel it would be redundant to include them.

7.3.2 Distributed Node Ratings

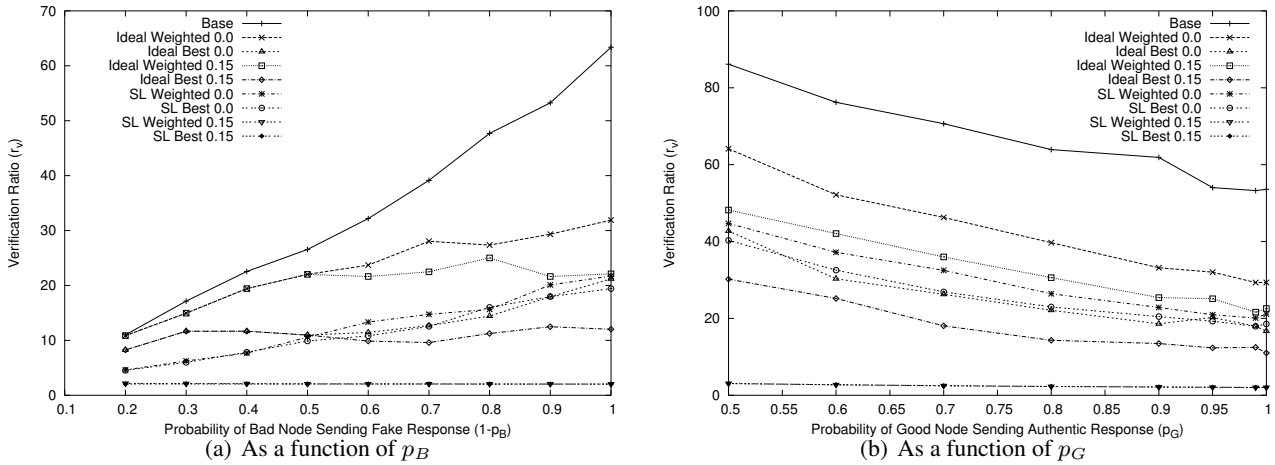
In most experiments all well-behaved nodes have same probability of sending an authentic document, or rating, of p_G. Malicious nodes, likewise, all have the same rating of p_B. These values are expected to be far apart, allowing algorithms to more easily locate and isolate the two. But how do the reputation systems behave when the node behaviors are not so distinct?

To test this, we randomize the node ratings in the threat matrix. Each node's rating is chosen from a uniform distribution with an interval of size 0.7 centered on p_B or p_G, depending on whether it is a good or bad node. For example, if p_B = 0.4 then malicious nodes will be assigned ratings in the range of 0.05 to 0.75 and the average value will be 0.4. For values of p_B and p_G near 0 or 1, the interval cannot extend completely 0.35 in each direction from the center. Rather than shorten the interval equally on both sides of the center, the interval is abruptly cut at 0 or 1. This results in a shift of the interval center (and average) away from the intended center. For example, when p_B = 0.1, then the values of malicious nodes are chosen uniformly from the interval [0, 0.45], resulting in an average value of 0.225.

Figure 13 shows the results of simulations run with the scattered threat ratings, both for different p_B and p_G values. Comparing Figures 13(a) and 13(b) to Figures **??** and **??** respectively, we see little difference in the performance of the base case and all the local variants. The base case

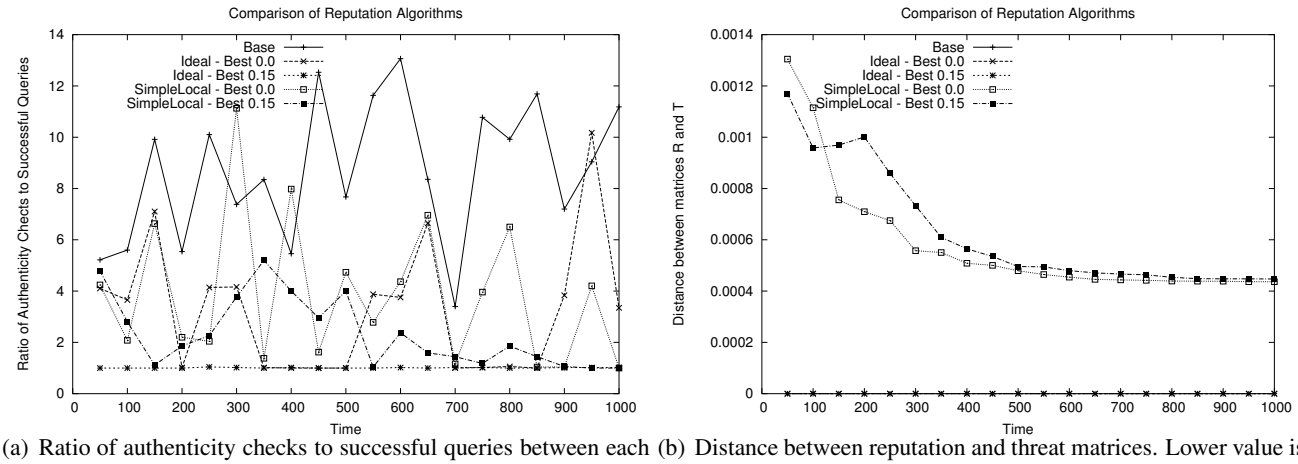
7.3.3 Convergence Over Time

The next tests we ran involved collecting statistics during each simulation run in order to evaluate the change in performance over time and to see if the reputation matrix converges to the threat matrix. In each set of simulations the



(a) As a function of p_B (b) As a function of p_G

Figure 13: Comparison of the efficiency of the reputation systems with node threat ratings uniformly distributed in an interval of length 0.7 around p_B or p_G.



(a) Ratio of authenticity checks to successful queries between each 50 query interval.

(b) Distance between reputation and threat matrices. Lower value is better. 0 is optimal.

Figure 14: Comparison of the local reputation system with ρ_T of 0.0 and 0.15 and the base case over time. The simulation was run for 1000 queries and statistics were collected every 50 queries.

simulation ran for Q total queries and statistics were gathered every δ queries. The graphs measuring the verification ratio over time compute the ratio based only on the number of authenticity checks and successful queries in the last δ queries. The graphs measuring T-R distance give the calculated T-R distance at the current time (i.e after δ queries, 2δ queries, etc.).

Figure 14 ran for Q = 1000 queries with a δ = 50 queries. In it we compare the three reputation systems using only the Select-Best procedure with ρ_T of 0 and 0.15 for the ideal and Simple Local cases.

Though in Figure 14(a) the values appear to vary randomly, notice that the local curves appear to converge towards 1. This indicates the the statistics the reputation system is gathering allows it to make better decisions in

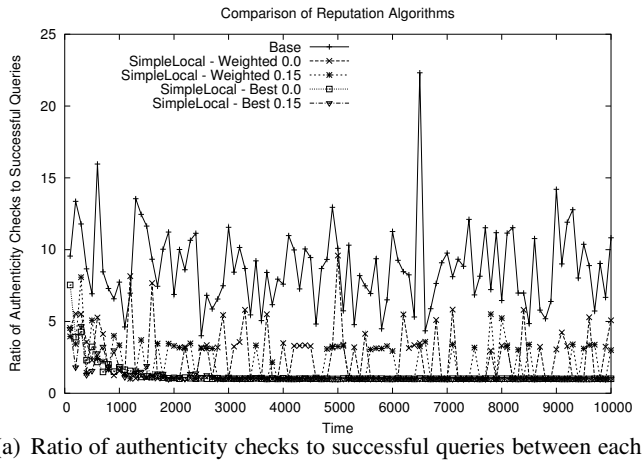
the future when selecting responses. The base case and the ideal case with no threshold, as expected, do not show this convergence since they do not "learn" from previous experiences. The ideal case with threshold performs well enough to stay near 1 for the entire run.

In Figure 14(b) we see the T-R distance during the same test. The ideal case is trivially 0, and the base case is not present (since the Random algorithm does not maintain statistics). We see that both local curves converge to a value of almost 0.0004. The curve corresponding to $\rho_T = 0.0$ seems to converge faster and to a slightly smaller distance than for $\rho_T = 0.15$. Since local using no threshold performs worse in terms of the number of authenticity checks it must perform, this allows it to collect more statistics about other nodes faster and therefore converge faster with a bit more accuracy. But once it locates a pool of good nodes, the Select-Best procedure will always attempt to pick from this group. Therefore it will not choose documents from other nodes if possible and will not collect new statistics, other than refine the reputation ratings of the good nodes.

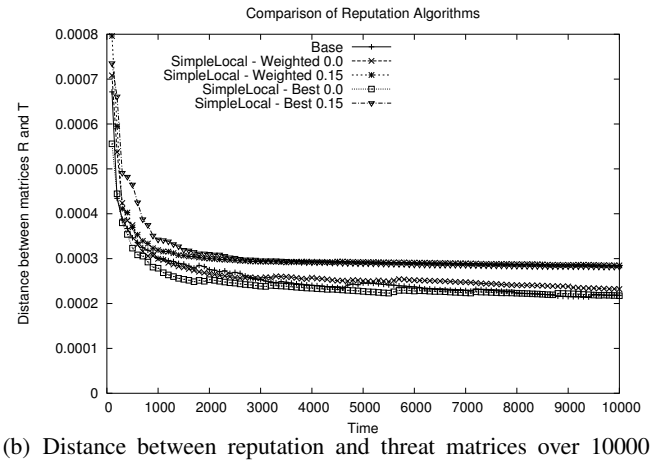
To see if gaining more varied statistics results in faster and better convergence we ran a similar longer test with Q = 10000 queries and δ = 100. We included the Weighted procedure to if its ability to choose a node that is not necessarily the best known node will help it minimize the T-R distance. We also modified the base case to collect statistics in order to calculate a reputation for each node, but still not use the information in the selection process. Since the base case performs the worst by performing the most authenticity checks, we would expect it to collect the most data and converge faster and better than the rest.

Figure 15(a) shows the verification ratio over time. Though the local Select-Best curves converge quickly to 1, the Weighted versions periodically jump to high values, resulting in worse performance.

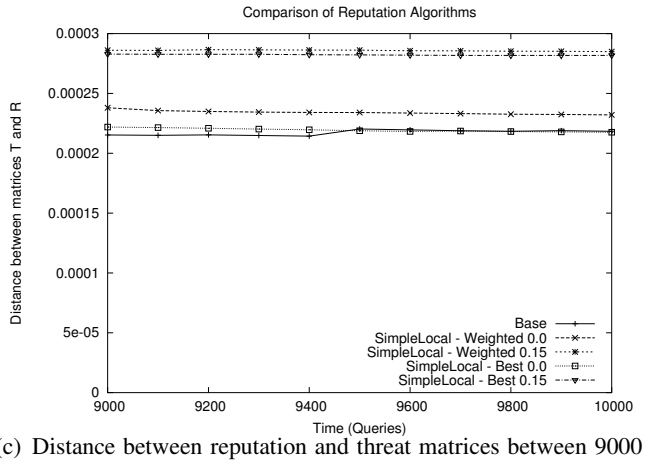
In Figure 15(b), all the curves converge at about the same rate. But both $\rho_T = 0.15$ curves converge to a higher T-R distance than the rest. This is as expected after the previous results. Figure 15(c) shows a more detailed view of the end of the simulation run. Interestingly, the Weighted version with no threshold and the base case did not converge faster or to a lower distance the the Select-Best version with no threshold, even though they both performed many more authenticity checks. This may be because, though the Weighted and base case collect more statistics, the statistics are across a larger number of nodes, while the Select-Best case leaves more nodes as undefined. Since the undefined nodes are not included in our measure of distance, a system would exhibit a lower T-R distance by information on a smaller number of nodes than the same amount of information (or even more) across a much larger number of nodes. A different method of calculating distance which promotes having less undefined nodes would likely improve the measure for the bad performing systems.



(a) Ratio of authenticity checks to successful queries between each 100 query interval



(b) Distance between reputation and threat matrices over 10000 queries

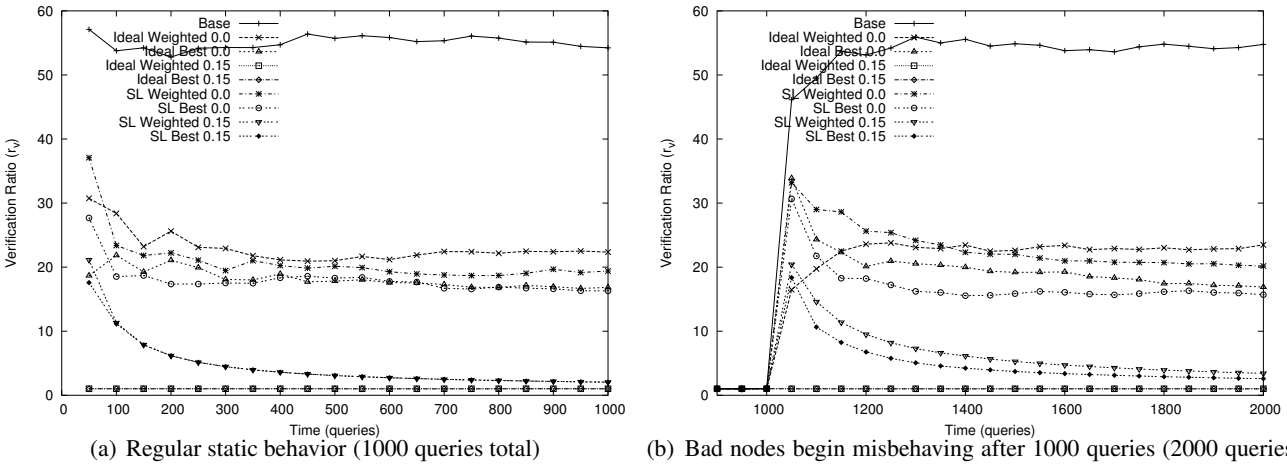


(c) Distance between reputation and threat matrices between 9000 and 10000 queries

Figure 15: Comparison of the local reputation system with both Weighted and Select-Best variants and a selection threshold of 0.0 and 0.15 and the base case over time. The simulation was run for 10000 queries and statistics were collected every 100 queries.

7.3.4 Dynamic Misbehavior

A suggested strategy for malicious nodes in a reputation system is to behave correctly for some period of time and accrue a positive reputation, then begin acting maliciously. What effect does such a scheme have on efficiency? Can good nodes detect such misbehaving nodes? If so, how quickly? To determine the effects of this strategy we devised a test where malicious nodes behaved correctly for 1000 queries from a single source. At that time all bad nodes began misbehaving. The simulation then continued for 1000 more queries. The verification ratio was calculated every 50 timesteps using the cumulative $\overline{A(D)}$ and q_{succ} at that time, except in the case of the dynamic bad nodes, in which case the verification ratio after 1000 queries is calculated using statistics gathered since the nodes began misbehaving at time 1000.



(a) Regular static behavior (1000 queries total)

(b) Bad nodes begin misbehaving after 1000 queries (2000 queries total)

Figure 16: Comparison of the efficiency of the reputation systems over time.

For comparison we graph the behavior of the standard static threat model where the malicious nodes misbehave for the entire simulation of 1000 queries. Figure 16(a) shows that the reputation systems quickly stabilize to a steady-state. The ideal system with threshold performs the best (almost 1) for the entire simulation, but the local system with threshold quickly converges to near optimal.

Comparing it now to the dynamic misbehavior scenario in Figure 16(b) we see that the reputation systems perform just as well, even though the malicious nodes have had time to build up a good reputation. The one interesting difference, is the worse performance of the Weighted procedure as opposed to Select-Best. With Select-Best as soon as the querying node fetches and checks one fake document from a malicious node, that node's rating will be lowered significantly so that other nodes will always be selected before it in the future, if possible. On the other hand, with the Weighted method, a malicious node is likely to be selected multiple times before its reputation rating is lowered sufficiently that it is unlikely to be chosen in future queries.

To illustrate this, let node A be a good node and node M be a dynamic malicious node. Say that both have provided 4 good documents during the period of time that M behaved well to accrue reputation and so both have a reputation of 1.0. Now say that M delivers an inauthentic document. It's rating will drop to $\frac{4}{5} = 0.8$. While Select-Best will never pick M if A has also replied, with the Weighted method A is only 25% more likely to be selected than M. In fact after 3 more false responses from M, its rating has only dropped to 0.5 and is only half as likely to be chosen as A. This demonstrates a weakness of the simple reputation rating function used. In situations with dynamic malicious nodes, reputations based on only the statistics gathered over that last x number of queries would perform better, but require more state be maintained.

The reason why Weighted does not perform too badly in this dynamic test is that in a period of only 1000 queries,

few malicious nodes had the opportunity to provide more than 1 or 2 correct answers to build their reputation statistics. If the misbehaving nodes were to behave better for a longer period, the greater the performance gap between Select-Best and Weighted. But if malicious nodes behave well for very long periods of time, they would be very ineffective at disrupting the network.

8 Related Work

Extensive research has been done on general issues of reputation (eg. [15] [16] [21]). Much work has been done in the area of locating reputable nodes in resource-sharing peer-to-peer networks and many interesting reputation systems have been proposed (eg. [9] [19] [14]). Here we describe a few related examples.

Reference [13] presents a game theoretical model, based on the prisoner's dilemma, for analyzing the social cost of allowing nodes to freely change identities. It proposes a mechanism, based on a centralized trusted intermediary. It ensures each user is assigned only one system identifier, yet protects their anonymity so that even the intermediary does not know which identifier was assigned to which node.

In [10] Douceur discusses the problem of preventing users from using multiple identities in a system with no trusted central agency (the Sybil attack). He presents methods for imposing computational cost on identity creation and lists system conditions necessary to limit the number of identities peers can generate.

In [19], Lai et al. propose a reciprocative incentive strategy to combat freeriders, based on the Evolutionary Prisoners Dilemma [3]. They compare the performance of private history versus shared history and develop an adaptive stranger response strategy that balances punishing whitewashers with overly taxing new nodes.

Reference [17] presents EigenTrust, a system to compute and publish a global reputation rating for each node in a network using an algorithm similar to PageRank [22]. Reputation statistics for each node are maintained at several nodes across a content-addressable network to mitigate the effects of bad nodes colluding.

In [7], Damiani et al enhance their previous work on reputation [5] by proposing the concept of resource reputation, where peers give opinions on a resource's authenticity based on its reported digest. This technique complements the process of maintaining peer reputations, which is still necessary in situations where the resource is rare and no other peers have encountered it.

9 Conclusion

We have compared two practical identity infrastructures for peer-to-peer resource-sharing environments. A centralized trusted login server that ties nodes' network pseudo-identities to their real-world identities provides better support for reputation systems by preventing nodes from quickly changing identities. However, this benefit comes at a high management cost and requires users to disclose information to a level which they may not find acceptable. The decentralized approach, where each node generates its own identity, provides a higher level of anonymity while simultaneously preventing identity hijacking, at the cost of no enforced identity persistence for malicious nodes. Though we have concentrated on two distinct identity models, many practical solutions fall in a spectrum between them (such as providing incentives for persistent identities) and perform accordingly.

Our results show that even simple reputation systems can work well in either of the two identity schemes when compared to no reputation system. In environments where system identities are generated by the peers themselves, all unknown nodes should be regarded as malicious. But, if a centralized login authority enforces identities tied to real world identities, then the optimal reputation for unknown nodes is nonzero. In addition, certain techniques, such as using a selection threshold, provide large benefits in efficiency for one identity scheme, but are ineffectual for others.

We have presented a simple voting-based reputation system that significantly mitigates the deleterious effects of malicious nodes, by sharing information with a small group of nodes. Even with 40% of the network attempting to subvert 90% of the resources, a node would expect to only have to attempt twice before locating a good provider, though it increases to four or five tries if the system is vulnerable to whitewashing.

We compared two methods for selecting providers given reputation information and showed that, while one provides better efficiency, it also significantly skews the load on the well-behaved nodes in the network. Depending on the amount of heterogeneity in the network this may be acceptable. We also show how the Friend-Cache developed for the reputation system can be applied to significantly reduce message traffic in unstructured peer-to-peer networks.

Finally, we present two distinct threat models, allowing us to simulate a variety of malicious behaviors. Both models affect system performance quite similarly and reputation system results in one are equivalently proportioned in the other. This allows us to compare reputation systems using whichever model is most convenient and expect similar results from the other.

A Statistical Analysis of Reputation Systems

In this section we provide a derivation for our statistical analysis of the steady-state behavior of the reputation systems and their variants. We begin by presenting the variables and equations dictating the expected system behavior. Next, we empirically compute values for certain parameters. Finally we calculate the expected performance of the different systems. Specifically, we are interested in the steady-state verification ratio of a system, which is approximately equal to the expected number of documents fetched and checked for each query.

A.1 Equations

In the experiments all the network topologies were static. Therefore, for a given TTL and network topology, the nodes reached by a flood query was always the same for each node, but different between nodes and topologies. Let $n_{flood}(i)$ be the number of nodes reached by a query from node i. Let $\overline{n}_{flood}$ be the average number of nodes queried across all possible nodes in all possible network configurations. We empirically estimate this value in the following section.

For the analysis we are interested in computing the expected probability of a node being able to reply to a query. In the simulations the probability of a node having a document matching a query q is

$$p_N(d, q) = 1 - (1 - p_D(q))^d \tag{11}$$

where $p_D(q)$ is the probability of any document in the system matching query q and d is the number of documents stored at the specified node. The probability of a document matching a query is determined by the query model using the query popularity and query selection power distributions [30]. The number of documents on a node is chosen from the file distribution sample collected by Saroiu [27]. The expected probability of a node answering a query, $\overline{p}_N$, is computed experimentally. More details are given below.

- The expected number of bad nodes that hear a query is $\overline{n}_{flood}\pi_B$.
- The expected number of good nodes that hear a query is $\overline{n}_{flood}(1 - \pi_B)$.
- The probability of good node replying with an authentic document = $\overline{p}_N p_G$.
- The probability of good node replying with a fake document = $\overline{p}_N(1 - p_G)$.
- The probability of bad node replying with an authentic document = $\overline{p}_N p_B$.
- The probability of bad node replying with a fake document = $1 - p_B$.

Notice that the probabilities of sending an authentic document and sending a fake document do not add up to 1. The remainder is the probability of the node not replying. Also note the difference between good and bad nodes in the probability of replying with a fake document. Because bad nodes can generate fake documents even when they do not have a query match, $\overline{p}_N$ is not taken into account.

From these equations we can derive formulas for the expected number of documents in a query's response set, how many come from good or bad nodes, and how many are authentic or not.

- Expected number of authentic documents received for a query
  - From good nodes: $d_{AG} = \overline{n}_{flood}(1 - \pi_B)\overline{p}_N p_G$
  - From bad nodes: $d_{AB} = \overline{n}_{flood}\pi_B\overline{p}_N p_B$
  - Total: $d_A = \overline{n}_{flood}\overline{p}_N(p_G(1 - \pi_B) + p_B\pi_B)$
- Expected number of fake documents received for a query
  - From good nodes: $d_{FG} = \overline{n}_{flood}(1 - \pi_B)\overline{p}_N(1 - p_G)$
  - From bad nodes: $d_{FB} = \overline{n}_{flood}\pi_B(1 - p_B)$
  - Total: $d_F = \overline{n}_{flood}(\overline{p}_N(1 - p_G)(1 - \pi_B) + (1 - p_B)\pi_B)$
- Expected number of total documents received for a query
  - From good nodes: $d_{TG} = \overline{n}_{flood}(1 - \pi_B)\overline{p}_N$
  - From bad nodes: $d_{TB} = \overline{n}_{flood}\pi_B(1 - p_B + p_B\overline{p}_N)$
  - Total: $d_T = \overline{n}_{flood}(\overline{p}_N(1 - \pi_B) + \pi_B(1 - p_B + p_B\overline{p}_N))$

Another set of equations which will be necessary are the probability a document received from a good or bad node is authentic. This is equivalent to the expected number of authentic documents from a good/bad node divided by the expected number of total documents from a good/bad node. This gives the following three equations

$$P(D_G = A) = \frac{d_{AG}}{d_{TG}} = p_G \tag{12}$$

$$P(D_B = A) = \frac{d_{AB}}{d_{TB}} = \frac{p_B\overline{p}_N}{p_B\overline{p}_N + 1 - p_B} \tag{13}$$

$$P(D = A) = \frac{d_A}{d_T} = \frac{\overline{p}_N(p_G(1 - \pi_B) + p_B\pi_B)}{\overline{p}_N(1 - \pi_B) + \pi_B(1 - p_B + p_B\overline{p}_N)} \tag{14}$$

For good nodes the probability is simply p_G since they only reply with documents they own, though a small percentage are fake. Bad nodes, on the other hand, can generate false documents.

A.2 Empirical Estimations

We estimate $\overline{n}_{flood}$ to be 3950 after 10000 samples from 100 nodes in 100 power-law topologies of 10000 nodes with approximately 3.1 degrees on average per node.

To compute $\overline{p}_N$ we generate 4000000 random samples from both our query popularity and query selection power distributions, and the sample document distribution collected by Saroiu [27] (described in Section **??**). For each generated value of $p_D(q)$ and d we calculate $p_N(d, q)$ using Equation 11, and average across all samples. The experimental estimate for $\overline{p}_N$ was 0.1090, or approximately 10% probability that a node can answer a query.

A.3 Long-Term Reputation System Performance

In this section we calculate the expected performance of each of the reputation systems and their variants after running for a very long period and having settled into a steady-state. For simplicity, this analysis makes two assumptions. First, all nodes maintain their public identities for all time. Second, the network topologies do not change. These assumptions are not expected to be realistic. For discussion of system performance in realistic scenarios, see the experimental results. We are interested in the steady-state performance as a guide to optimal performance of each system. A comparison of the experimental results to our statistical analysis should indicate how quickly the systems converges toward the steady-state behavior.

To determine the long-term efficiency of the reputation systems we are interested in calculating the expected verification ratio. This ratio is the number of documents that must be verified for authenticity for each successful query before an authentic one is found. Above, we determined the expected number of documents of each type received in response to a query. For all reputation systems, the expected verification ratio will be a variation of the expected number of documents that must be chosen from a subset of the responses, *without replacement*, until an authentic one is found. Since the expected number of documents received in response to a query is large we can approximate this problem to that of *with replacement* [8]. Given that the probability of choosing an authentic document on the first try is p, we assume that each successive attempt at choosing an authentic document is also has a probability p of success. The expected number of documents that must be checked before locating an authentic document, with replacement, is approximated as

$$E(X) = \sum_{x=1}^{\inf} p(1-p)^{x-1} = \frac{1}{p} \tag{15}$$

This is the formula we use in the analysis below.

We also assume (as in most experiments) that all well-behaved nodes have a threat rating of p_G and all malicious nodes have a threat rating of p_B, and that their threat ratings do not change over time (or at least that the systems

reach a steady-state between changes).

The equations in Appendix A.1 give the expected number of documents per query. As stated before, the verification ratio, r_v is proportional to the number of *successful* queries, not *total* queries. We account for this distinction by calculating a factor Q, which is the fraction of total queries expected to be successful. Say we issue θ queries. The expected number of successful queries would be $Q \cdot \theta$. Each query will fetch $\frac{1}{p}$ documents on average. The total number of documents fetched will be $\frac{1}{p} \cdot \theta$. By plugging these values into Equation 4 we get

$$\begin{aligned} r_v &= \frac{\frac{1}{p} \cdot \theta}{Q \cdot \theta} = \frac{\frac{1}{p}}{Q} \\ r_v Q &= \frac{1}{p} \end{aligned} \tag{16}$$

For each system, we give equations for:

- The probability of choosing an authentic document on the first try, p (when applicable).
- The fraction of total queries expected to be successful, Q.
- The expected verification ratio multiplied by the Q factor.

A.3.1 Random base case

All responses will be looked at equally likely.

$$p = \frac{d_A}{d_T} \tag{17}$$

$$Q = 1 - (1 - P(D = A))^{d_T} \tag{18}$$

$$\text{Expected } r_v Q = \frac{1}{p} = \frac{d_T}{d_A} = \frac{\overline{p}_N(p_G(1 - \pi_B) + p_B \pi_B)}{\overline{p}_N(1 - \pi_B) + \pi_B(1 - p_B + p_B \overline{p}_N)} \tag{19}$$

A.3.2 Select-Best/Weighted ideal case with threshold

We assume all malicious nodes' ratings fall below the selection threshold and so only answers from good nodes will be considered. Since we assume all good nodes have the same threat rating, then they will all be equally likely to be chosen by both the Select-Best and Weighted methods.

Probability of choosing an authentic document of the first try is

$$p = \frac{d_{AG}}{d_{TG}} \tag{20}$$

$$Q = 1 - (1 - P(D_G = A))^{d_{TG}} \tag{21}$$

$$\text{Expected } r_v Q = \frac{1}{p} = \frac{d_{TG}}{d_{AG}} = \frac{1}{p_G} \tag{22}$$

A.3.3 Weighted ideal case without threshold

All responses are considered but are weighted based on the rating of the sending node. Given we have received d_{TG} responses from good nodes and d_{TB} responses from bad nodes, then the probability of choosing a response from a good node with the weighted method is $\frac{p_G d_{TG}}{p_G d_{TG} + p_B d_{TB}}$. Likewise, the probability of choosing a response from a bad node is $\frac{p_B d_{TB}}{p_G d_{TG} + p_B d_{TB}}$. The probability that a random document from a good node is authentic is simply $\frac{d_{AG}}{d_{TG}}$, and the probability that a document from a bad node is authentic is $\frac{d_{AB}}{d_{TB}}$. Combining these formulas we get the probability of choosing an authentic document using the weighted method.

$$p = \frac{p_G d_{TG}}{p_G d_{TG} + p_B d_{TB}} \cdot \frac{d_{AG}}{d_{TG}} + \frac{p_B d_{TB}}{p_G d_{TG} + p_B d_{TB}} \cdot \frac{d_{AB}}{d_{TB}} = \frac{p_G d_{AG} + p_B d_{AB}}{p_G d_{TG} + p_B d_{TB}} \tag{23}$$

$$Q = 1 - (1 - P(D = A))^{d_T} \tag{24}$$

$$\text{Expected } r_v Q = \frac{p_G d_{TG} + p_B d_{TB}}{p_G d_{AG} + p_B d_{AB}} = \frac{p_G \overline{p}_N (1 - \pi_B) + p_B \pi_B (1 - p_B + p_B \overline{p}_N)}{p_G^2 \overline{p}_N (1 - \pi_B) + p_B^2 \overline{p}_N \pi_B} \tag{25}$$

A.3.4 Select-Best ideal case without threshold

With no threshold all responses may be looked at. First, the responses from good nodes will be checked one at a time. If no authentic document was found, then the responses from bad nodes are checked. If there are responses from good nodes, then the probability of locating an authentic document on the first try is $P(D_G = A)$, or p_G, on the second try $(1 - p_G)p_G$, on the third try $(1 - p_G)^2 p_G$, and so on. Let γ be the number of responses from good nodes (d_{TG}), and β be the number of responses from bad nodes (d_{TB}). The probability of the first authentic document found being the first document from a bad node checked would be $(1-p_G)^\gamma P(D_B = A)$. The probability of the first authentic document found being the second document from a bad node checked would be $(1 - p_G)^\gamma (1 - P(D_B = A))P(D_B = A)$.

We can now calculate the expected number of documents which must be downloaded and checked per query:

$$\text{Expected } r_v Q = \sum_{k=1}^{\gamma} k p_G (1 - p_G)^{k-1} + \sum_{k=1}^{\beta} (\gamma + k)(1 - p_G)^\gamma P(D_B = A)(1 - P(D_B = A))^{k-1} \tag{26}$$

By applying the well-known equations [18]

$$\sum_{0 \le j \le n} a x^j = a \left(\frac{1 - x^{n+1}}{1 - x} \right) \text{ and } \sum_{0 \le j \le n} a j x^j = a \left(\frac{n x^{n+2} - (n+1)x^{n+1} + x}{(x-1)^2} \right) \tag{27}$$

on Equation 26 and simplifying we obtain

$$\text{Expected } r_v Q = \frac{1}{p_G} + (1-p_G)^{\gamma}\left[\frac{1}{P(D_B=A)} - \frac{1}{p_G} - (1-P(D_B=A))^{\beta}\left(\frac{1}{P(D_B=A)} + \beta + \gamma\right)\right] \quad (28)$$

Substituting d_{TG} and d_{TB} for γ and β, gives us

$$\text{Expected } r_v Q = \frac{1}{p_G} + (1-p_G)^{d_{TG}}\left[\frac{1}{P(D_B=A)} - \frac{1}{p_G} - (1-P(D_B=A))^{d_{TB}}\left(\frac{1}{P(D_B=A)} + d_{TB} + d_{TG}\right)\right] \quad (29)$$

As with any system with no threshold, the Q factor is

$$Q = 1 - (1 - P(D=A))^{d_T} \quad (30)$$

A.3.5 Select-Best/Weighted local reputation system with threshold

In the long run we would expect the local system's standard reputation matrix, R', converge to the threat matrix T. Therefore the local system in steady-state would approximate the ideal case. One exception is in the scenario with a threshold. All bad nodes would eventually be rated below the threshold and ignored, just as in the ideal case. In addition, a fraction of good nodes would fall below this threshold and be ignored based on the fact that, with some probability $(1-p_G)$ good nodes do reply with fake documents. For very large p_G and small ρ_T we approximate the percentage of good nodes which fall below the threshold as the percentage of good nodes which reply with a fake document the first time they reply to a particular node's queries by $(1-p_G)$. Thus the only a $1-p_G$ fraction of the responses from good nodes will be considered for both Select-Best and Weighted with a threshold.

$$p = \frac{p_G d_{AG}}{p_G d_{TG}} \quad (31)$$

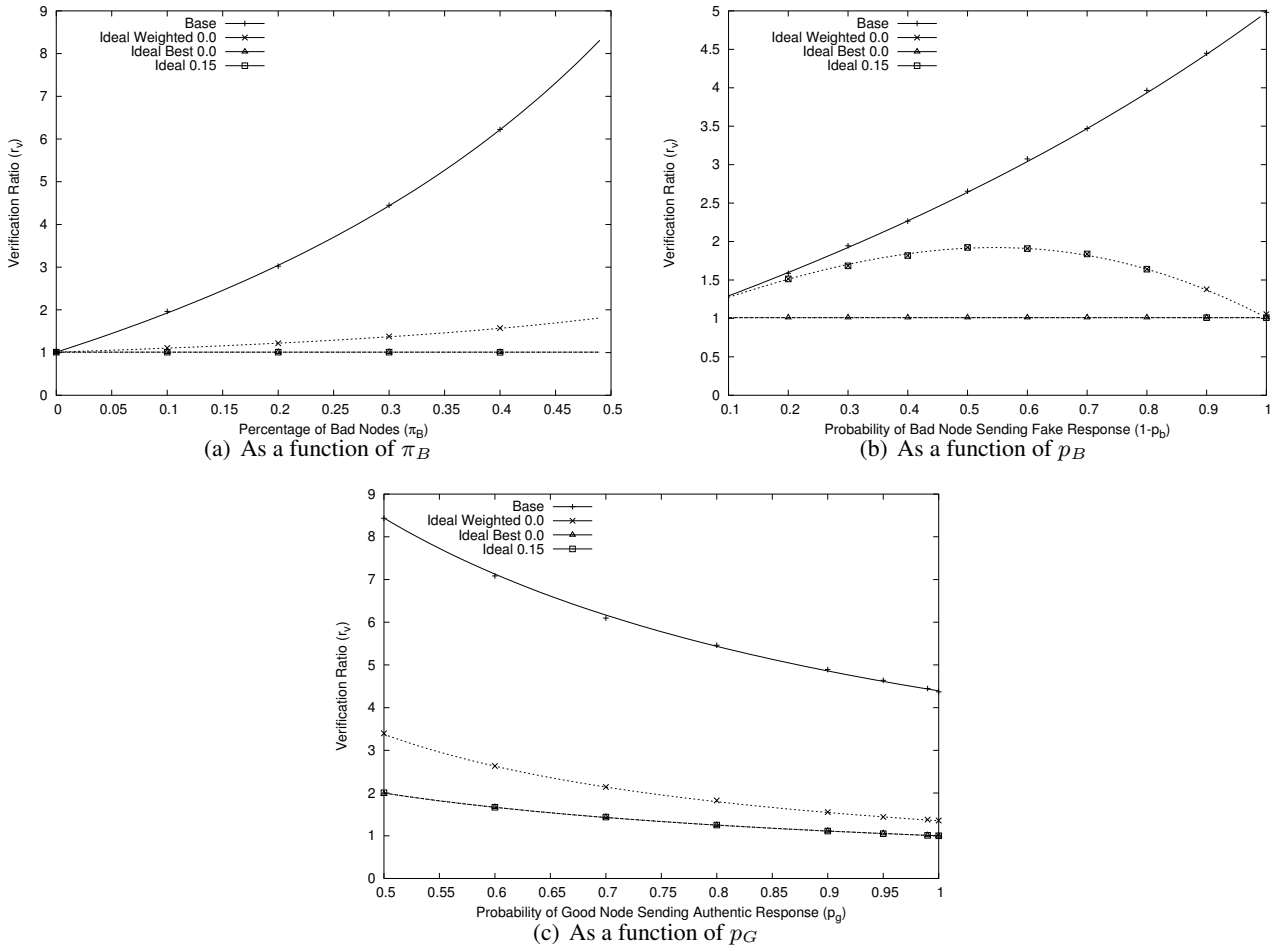
$$Q = 1 - (1 - P(D_G=A))^{p_G d_{TG}} \quad (32)$$

$$\text{Expected } r_v Q = \frac{1}{p} = \frac{p_G d_{TG}}{p_G d_{AG}} = \frac{1}{p_G} \quad (33)$$

Although the long-term efficiency of system is almost equal to that of the ideal system (note the difference in Q), its effectiveness is reduced because of the additional exclusion of the small percentage of good nodes. This is further discussed in Section **??**.

A.3.6 Weighted local system without threshold

With no threshold the local system should more closely approximate T than with a threshold since good nodes which reply with a fake document their first time, will be given a second chance. All responses are considered but



(a) As a function of π_B

(b) As a function of p_B

(c) As a function of p_G

Figure 17: Comparison of expected steady-state system behavior with 1000 query simulation using uniform document base with p_N = 0.1090. Lines represent expected performance based on analysis. Points are results of corresponding simulations.

are weighted based on the rating of the sending node. The equations for p, Q, and the expected $r_v Q$ are given by Eq. 23, 24, and 25, respectively.

A.3.7 Select-Best local system

As with the Weighted procedure the Select-Best local system with no threshold will perform like the corresponding ideal case variant. Thus the equations for expected $r_v Q$ and Q are given by Eq. 29 and 30, respectively.

A.4 Comparison of Statistical Analysis to Simulation Results

Using the equations derived above, we verify the correctness of our simulator by comparing our simulation results to the expected value at steady state for the same parameter values. In Figure 17 we graph the expected steady-state r_v as a function of three threat model parameters for the random base case, and the three variants of the ideal case

discussed above. These equations are represented by the line curves.

We reran the experiments from Section 7.3.1 for the base case and the ideal cases with a modification to the document model, which we discuss below. We chose only to test the base case and the ideal variants because those systems attain steady state from the beginning. Unlike the local system, they gather no new information over time, so their behavior remains constant. We plot these results as the datapoints in the graphs of Figure 17. Clearly, the simulation results closely match the curves of the expected performance.

If we compare the graphs in Figure 17 with those in Figure **??** we see that the curves are similar in shape and relative proportions, but the absolute values much larger in Figure **??**. This difference is due to the modified document base model used for the simulations for Figure 17. In these experiments we use a simplified model where each node had an equal probability of matching any query, p_N. p_N was set to a value of 0.1090, the expected probability of a node matching a query ($\overline{p}_N$) derived experimentally from our realistic document base model used in all other simulations. That model is dependent on three Zipf distributions (query popularity, query selection power, and number of documents per node). When comparing the simulator to our derived equations it is necessary to substitute this simplified document model because of a similar simplification which we made in our equations.

The formulas all use the expected value $\overline{p}_N$ in place of the linearly dependent random variable p_N. This simplification allowed us to derive relatively simple formulas (if not the formulas would be as complex as the simulator itself). Unfortunately, this mitigates the effects of certain conditions, such as when a node receives no authentic replies whatsoever in its response set. With approximately 4000 nodes hearing each query, and each node having, on average, almost an 11% chance of matching the query, it would seem unlikely that no authentic document would be received. But because of the heavy-tail nature of Zipf distributions, this condition occurs much more frequently than one might expect and should not be ignored. This shows the importance of running complex simulations and not simply relying on statistical derivations which may lead to seemingly insignificant simplifications that actually greatly affect the validity of one's conclusions.

References

[1] ADAMIC, L. Search in power law networks. *Physical Review E 64* (2001), 46135–46143.

[2] ADAMIC, L. Personal communication, 2002.

[3] AXELROD, R. *The Evolution of Cooperation*. Basic Books, 1984.

[4] BRIAN F. COOPER, MAYANK BAWA, N. D., AND GARCIA-MOLINA, H. Protecting the PIPE from malicious peers. Technical report, Stanford University, 2002.

[5] CORNELLI, F., DAMIANI, E., AND CAPITANI, S. D. Choosing reputable servents in a p2p network. In *Proc. of the Eleventh International World Wide Web Conference* (2002).

[6] CRESPO, A., AND GARCIA-MOLINA, H. Routing Indices For Peer-to-Peer Systems. *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)* (July 2002).

[7] DAMIANI, E., DI VIMERCATI, D. C., PARABOSCHI, S., SAMARATI, P., AND VIOLANTE, F. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proceedings of the 9th ACM conference on Computer and communications security* (2002), ACM Press, pp. 207–216.

[8] DEVORE, J. L. *Probability and statistics for engineering and the sciences*, 3rd ed. Brooks/Cole Publishing Co., 1991.

[9] DINGLEDINE, R., FREEDMAN, M. J., HOPWOOD, D., AND MOLNAR, D. A reputation system to increase MIX-net reliability. *Lecture Notes in Computer Science 2137* (2001), 126+.

[10] DOUCEUR, J. R. The Sybil Attack. In *Proceedings of the International Workshop on Peer-to-Peer Systems* (2002).

[11] eBay - The World's Online Marketplace. http://www.ebay.com/.

[12] FALOUTSOS, M., FALOUTSOS, P., AND FALOUTSOS, C. On power-law relationships of the internet topology. In *SIGCOMM* (1999), pp. 251–262.

[13] FRIEDMAN, E., AND RESNICK, P. The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy 10*, 2 (1998), 173–199.

[14] GUPTA, M., JUDGE, P., AND AMMAR, M. A reputation system for peer-to-peer networks. In *ACM 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video* (2003).

[15] HUBERMAN, B. A., AND WU, F. The dynamics of reputations. www.hpl.hp.com/shl/papers/reputations/, 2002.

[16] JURCA, R., AND FALTINGS, B. Towards incentive-compatible reputation management. In *Proceedings of the AAMAS 2002 Workshop on Deception, Fraud and Trust in Agent Societies*.

[17] KAMVAR, S. D., SCHLOSSER, M. T., AND GARCIA-MOLINA, H. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the Twelfth International World Wide Web Conference* (2003).

[18] KNUTH, D. E. *Fundamental Algorithms*, 2nd ed., vol. 1 of *The Art of Computer Programming*. Addison-Wesley Publishing Co., 1973.

[19] LAI, K., FELDMAN, M., STOICA, I., AND CHUANG, J. Incentives for cooperation in peer-to-peer networks. In *Workshop on Economics of Peer-to-Peer Systems* (2003).

[20] LV, Q., CAO, P., COHEN, E., LI, K., AND SHENKER, S. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*.

[21] MARIMON, R., NICOLINI, J., AND TELES, P. Competition and reputation. In *Proceedings of the World Conference Econometric Society* (2000).

[22] PAGE, L., BRIN, S., MOTWANI, R., AND WINOGRAD, T. The PageRank citation ranking: Bringing order to the web. Tech. rep., Stanford Digital Library Technologies Project, 1998.

[23] PALMER, C., AND STEFFAN, J. Generating network topologies that obey power laws. In *Proceedings of GLOBECOM '2000*.

[24] RANGANATHAN, K., RIPEANU, M., SARIN, A., AND FOSTER, I. To share or not to share: An analysis of incentives to contribute in collaborative file sharing environments. In *Workshop on Economics of Peer-to-Peer Systems* (2003).

[25] REICH, V., AND ROSENTHAL, D. S. H. LOCKSS: A Permanent Web Publishing and Access System. *D-Lib Magazine 7*, 6 (June 2001). http://www.dlib.org/dlib/june01/reich/06reich.html.

[26] RESNICK, P., ZECKHAUSER, R., FRIEDMAN, E., AND KUWABARA, K. Reputation systems. Communications of the ACM, pages 45-48, December 2000.

[27] SAROIU, S., GUMMADI, P. K., AND GRIBBLE, S. D. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)* (San Jose, CA, USA, January 2002).

[28] SRIPANIDKULCHAI, K. The popularity of gnutella queries and its implications on scalability. Featured on O'Reilly's www.openp2p.com website, February 2001.

[29] YANG, B. Personal communication, 2002.

[30] YANG, B., AND GARCIA-MOLINA, H. Comparing hybrid peer-to-peer systems (extended). Tech. rep., 2000.