Peer-to-Peer Research at Stanford

Mayank Bawa, Brian F. Cooper, Arturo Crespo, Neil Daswani,
Prasanna Ganesan, Hector Garcia-Molina, Sepandar Kamvar, Sergio Marti,
Mario Schlosser, Qi Sun, Patrick Vinograd, Beverly Yang
Computer Science Department, Stanford University
Contact: smarti@db.stanford.edu

1 Introduction

Peer-to-peer (P2P) systems have become a popular medium through which to share huge amounts of data. P2P systems distribute the main costs of sharing data – disk space for storing files and bandwidth for transferring them - across the peers in the network, thus enabling applications to scale without the need for powerful, expensive servers. Their ability to build an extremely resource-rich system by aggregating the resources of a large number of independent nodes enables peer-to-peer systems to dwarf the capabilities of many centralized systems for relatively little cost. Examples include the massive computation power of systems such as SETI@Home, or the ability to aggregate data, storage and processing in a network of mobile, ubiquitous devices. The Kazaa file-sharing system alone reported, as of April 30th 2003, over 4.5 million users sharing a total of 7 petabytes of data.

There are, however, important challenges that must be overcome before the full potential of P2P systems can be realized. For example, the scale of the network and the autonomy of nodes make it difficult to identify and distribute the resources that are available. Furthermore, because some peers may be malicious, peers may receive inauthentic information or may be victims of denial-of-service attacks.

These issues, and others, have motivated substantial research on understanding and improving P2P networks. In this paper we present recent and ongoing research projects of the Peers research group at Stanford University. Section 2 studies the problems relating to locating resources in P2P systems. Section 3 discusses work on resource allocation and aggregation. Section 4 focuses on issues of resource availability and authenticity. Note, this paper should not be construed as an overview of all research problems pertaining to peer-to-peer networks. Only projects connected to our Peers group are described. Additional citations can be found in the papers referenced below.

2 Queries and Topologies

A key challenge to the usability of a data-sharing peer-to-peer system is implementing efficient techniques for search and retrieval of data. The best search techniques for a system depend on the needs of the application. For example, search techniques based on distributed hash tables (DHTs) are well-suited for web caches or archival systems focused on availability, because they guarantee location of content if it exists, within a bounded number of hops. In many scenarios, the increased search efficiency makes structured networks preferable to the widely deployed unstructured networks which rely on flooding. To achieve these properties, these techniques tightly control the data placement and topology within the network, and currently only support search by identifier.

In contrast, other work focuses on more flexible applications with rich queries such as regular expressions, meant for a wide range of users from autonomous organizations. We are interested in studying the search problem for these "flexible" applications because they reflect the characteristics of the most widely used P2P systems deployed today. Search techniques for such networks must operate under a different set of constraints than techniques developed for persistent-storage utilities, such as providing greater respect to the autonomy of individual peers.

We first discuss our work on unstructured systems, followed by a description of our work on structured ones.

2.1 Unstructured Systems

Three main themes have emerged from our work on unstructured systems. First, the search techniques should be simple and practical enough to be easily incorporated into existing systems. Current successfully deployed P2P data-sharing systems follow very simple protocols. Although these protocols are clearly suboptimal, they highlight how simplicity is the key to wide and rapid adoption. Second, we need to understand and characterize the behavior of existing P2P applications. Effective search techniques need to make provisions for the unreliable nature of peers, and take advantage of observed user behavior. Finally, any technique should be adaptive, and tune itself according to the current state of the system. Because P2P systems are by nature highly dynamic, a rigid search mechanism that is effective in one scenario or for one particular user is likely to become ineffective as the system evolves or users change.

2.1.1 Improving Existing Systems

One important aspect of search we have studied is the "message routing protocol," used to disseminate queries amongst peers. The routing protocols used in practice (e.g., Gnutella [15]) are based on flooding messages across the overlay network. The effectiveness of this technique depends on (i) the availability of the data that can satisfy the query, (ii) the position of the peer in the overlay, and (iii) the overlay structure itself. This technique can clearly be suboptimal in many cases. In [29, 11], we investigate simple but effective improvements over the existing flooding protocol. Reference [29] presents the Directed BFS technique, which relies on feedback mechanisms to intelligently choose which peer a message should be sent to. Neighbors that have provided quality results in the past will be chosen first, yet neighbors with high loads will be passed over, so that good peers do not become overloaded. Reference [29] also presents the Iterative Deepening technique, which allows search to proceed incrementally until the user is satisfied with the results. These two simple techniques allow search to be tuned on a per-query, per-user basis. Experiments over detailed query traces from the Gnutella network show that our techniques greatly reduce the cost of search, while maintaining good quality of results.

In reference [11], message routing is further improved with "routing indices", compact summaries of the content that can be reached via a link. With routing indices, nodes can quickly route queries to the peers that can respond, without wasting the resources of many peers who cannot. Interesting research challenges arise as to how indices are updated simply and efficiently as links are created and destroyed. Simulations show that the techniques developed in [11] are effective, and that the cost tradeoff between maintaining the indices and querying is significantly positive in many scenarios.

We have also studied "role differentiation," another important aspect of an efficient search. For example, super-peer networks differentiate between "super-peers" and "clients," where super-peers act as mini-index servers to a number of clients, but interact with each other as peers in a regular P2P system. Super-peers are used in currently deployed systems and have already proven to be effective in improving search performance. In [30] we conduct an in-depth study on the design of super-peer networks and show how a straightforward implementation can be orders of magnitude less effective than one that is tuned to the particular requirements and workload of a system. From our investigation we present several design principles for an effective super-peer network, and a global design procedure that takes as input the workload and constraints on a system, and produces an efficient super-peer topology. Because workload and requirements evolve over time within a single system, it is important also to be able to evolve the design of the super-peer network to meet these changing needs. To this end, our global design procedure may be applied incrementally, such that peers can be directed to make runtime changes that tune the network. We also present local decision-making guidelines by which peers can make individual, runtime decisions that result in a globally efficient topology.

The results of our studies in [29, 11] show that incremental forwarding of query messages and intelligent server selection greatly improves search performance without affecting quality of results, while [30] shows that an improperly organized network topology and role differentiation can result in high overhead in message forwarding and processing. These conclusions lead us to consider a new type of search architecture, in which messages are not forwarded, and a peer has complete control over who receives its queries and when. We are currently studying this architecture in the context of the GUESS protocol [16], an under-construction specification that is meant to become the successor of the widely-used but inefficient Gnutella protocol. Under the GUESS protocol, peers directly probe each other with their own query messages, rather than relying on other peers to forward the message. However, beyond this simple concept, there are many issues to be addressed before the protocol can be successful. For example, when processing a query, in what order should peers be probed? The solution to this "server selection" problem must balance efficiency of the query with load-balancing among the peers. Also, practical problems not directly related to search performance must also be addressed; for example, since peers no longer rely on other peers to forward their queries, it is much easier for peers to abuse the system for personal gain. How can we detect and prevent selfish behavior? We are currently investigating solutions to these and other issues to make GUESS a viable alternative to other proven P2P search protocols.

2.1.2 New Directions

In addition to studying ways to improve existing systems, our group is exploring novel ways to organize and use unstructured P2P systems.

In particular, we have explored the possibility of a completely decentralized search network built in an ad hoc way [8]. Unlike structured topologies, hosts here are not restricted to certain neighbors. Instead, the protocol is devoted to incrementally improving the established network through *self-supervision*. Using two simple operations (`connect()` and `break()`) to maintain the network, we show that ad hoc networks can be optimized for both homogeneous and heterogeneous networks and can adapt to varying search profiles. The results indicate that in several situations, hosts make local decisions that are both beneficial to themselves and good for the network as a whole.

The design of efficient search networks is complicated by the vast space of possible design choices: neighbor selection, query routing, query evaluation, content replication, etc. To help make exploration of the design space manageable, we proposed separation of design into two phases [9]: (a) Architectural and (b) Operational. In the Architectural phase, designers concentrate on neighbor selection, query routing and content replication. In the Operational phase, designers study alternatives for maintaining neighbors, network exploration, etc. We developed the *Search/Index Link (SIL)* model for representing and visualizing search networks at the Architectural level. We demonstrated use of the model to design and evaluate novel architectures that are more robust and efficient.

We have also worked jointly with IBM on the development and implementation of a new P2P search infrastructure called YouSearch [2]. YouSearch provides a simple hybrid architecture in which the P2P network is augmented with a light-weight centralized component. Peers maintain compact site summaries (in the form of a Bloom filter [4]) which are aggregated at a centralized registrar. These summaries are queried so that searches target only the relevant machines. Peers help reduce query load on the system by caching and sharing query results. Peers also cooperate to maintain the freshness of the summary aggregation at the registrar. This minimizes the role of registrar for low cost and graceful scaling while ensuring fast, fresh and complete searches. YouSearch was deployed within the IBM corporate intranet in September 2002. Within 2 months, it was adopted by nearly $1,500$ users.

Finally, we have also worked on queries that aggregate information across an unstructured P2P network. For example, an administrator who supports an application on a P2P network needs information about usage trends to tune their particular application. Specifically, they may want to compute an aggregate function (e.g., the average lifetime of hosts) over data residing at hosts in the network. The P2P networks of today lack mechanisms to compute even such basic aggregates as minima, maxima, sum, count or average. In [1], we define and study the above "node aggregation" problem. We study its computability for P2P networks and present generic schemes that can be used to compute any of the basic aggregation functions. The schemes can be chosen to balance accuracy and efficiency concerns for a particular task.

2.2 Structured Systems

In a structured P2P system, the location of an object (resource) is determined by a globally agreed-upon scheme, e.g., hashing the resource's key. Then, given the key for a desired object, one can easily find the location where that object should be. In some cases, there may be multiple objects associated with a given key. For instance, the network may hold many copies of a given song. In such cases, users often just want one (or a few) of the objects associated with a key, not all of the objects. In [28] we formalize this notion of a partial lookup query and present schemes for building such a key-lookup service. We study a variety of ways to distribute objects (not just hashing-based), and quantify the differences in performance, reliability, fairness, and other metrics.

In the area of hash-based schemes for full lookups, we proposed Symphony [20], inspired by Kleinberg's Small World construction [19]. Peers form *short distance* links with their neighbors on a ring. Additionally, each peer is equipped with a few *long distance* links that connects it with peers farther away along the ring. We showed that with $k = O(1)$ long distance links per peer in an n-peer network, it is possible to route lookup queries with an average latency of $O(\frac{1}{k}\log^2 n)$ hops. Among the advantages Symphony offers over existing DHT protocols [23, 27, 24] are (a) low state maintenance, (b) fault tolerance and (c) degree vs. latency tradeoffs that allows support for heterogeneous nodes, incremental scalability and flexibility.

In [3], we build on Symphony to provide an efficient *search* service called SETS, for Search Enhanced by Topic Segmentation. The key idea is to arrange peers in a *topic-segmented* network such that a search query probes only a small subset of hosts where most of the matching documents reside. In particular, SETS arranges peers in a topology where most of the links are short distance joining pairs of sites with similar content. The resulting topically focused segments are joined together into a single network by long-distance links. Queries are then matched and routed to only the topically closest regions.

Finally, we also have explored a new search protocol that can be viewed as a hybrid of structured and unstructured systems, providing flexibility and advantages from both. Our protocol, YAPPERS [14] provides a lookup service over arbitrary network topologies. The scheme involves each host participating in a distributed hashing protocol with nearby hosts, enabling efficient partial lookups. A separate protocol (flooding-based) is then used to combine partial lookups for complete results.

3 Resource Management

Aggregating and allocating peer-to-peer resources is much more difficult than in a centralized system. One reason is the autonomous nature of peers: rational, essentially selfish peers must be given an incentive to contribute resources. In addition, the scale of the system, with perhaps very many nodes, makes it hard to get a complete picture of what resources are available. This is especially true in a dynamic system, with nodes constantly joining and leaving, where resources and resource demands are constantly changing. Our approach to dealing with these issues is to use concepts from economics to construct a resource marketplace, where peers can buy and sell or trade resources as necessary. Economic incentives are used to encourage resource sharing, while the problem of system-wide resource allocation is broken down into numerous exchanges between pairs of nodes to enhance scalability.

For example, the RTR protocol uses an economic model to allocate query processing resources. In the RTR protocol, peers buy and sell the right-to-respond (RTR) to each query in the system. This gives peers an economic incentive to forward queries, which they otherwise would not do in a competitive P2P network. Furthermore, peers in this framework will connect to peers who are likely to vend them queries to which they can respond. Therefore, clusters of peers with similar interests are likely to form in the topology of a network implementing the RTR protocol, reducing network overhead and making search more efficient. Our work, described in [31], shows how peers can be given a direct incentive to pool resources for the benefit of others.

Another example of our work is storage resource allocation using "data trading." Consider a data archive that is trying to make copies of its data collections at remote sites to give them a better chance of surviving local failures. A remote site will not be willing to donate storage without getting something in return. Under a data trade, the local archive trades away some of its local storage in order to get storage at the remote site. Then, both sites can make remote copies of their collections. A series of such trades between pairs of sites builds up a peer-to-peer replication network. In this way, the basic primitive of a "data trade" is used by sites as needed to allocate storage resources. In [7, 5], we examine how sites can best use that primitive to achieve high reliability. Such a trading marketplace can use techniques from economic models. For example we have studied how sites can use auctions to negotiate how much storage space is exchanged [6]. The techniques we have developed show how a trading-based economy can be an effective resource allocation mechanism in a peer-to-peer system.

4 Security

P2P data sharing systems are highly susceptible to many forms of malicious attacks. Nodes in a P2P system operate in an autonomous fashion, and any node that speaks the system protocol may participate in the system. However, just because a node can speak the protocol does not mean that it will do so with good intentions. As a result, nodes cannot necessarily assume that other nodes will respond to their queries, limit the number of queries they generate, produce authentic results, or keep the contents of their queries private. In this section, we will describe our work that is targeted at mitigating attacks by nodes that abuse the P2P network by exploiting the implicit trust peers place on them. Specifically we discuss research meant to address the security issues around availability, authenticity and trust.

4.1 Availability

Attacks against a system's availability are often called *denial-of-service* (DoS) attacks, and are targeted at degrading system performance, or shutting down a system completely by having malicious clients use up resources (CPU cycles, disk space, network bandwidth, etc.) such that these resources cannot be used by legitimate clients in the system. In addition, a common characteristic of such attacks is that it is often hard to distinguish nodes that are malicious from those that are simply under a high load. As a result, a common theme in the research we describe here is to balance the generated load so that malicious nodes can use a portion of the system's resources, but not a disproportionate amount of the resources.

In [13], we studied denial-of-service attacks against the Gnutella P2P system [15]. Nodes in a Gnutella system search for documents by flooding. That is, nodes broadcast searches to all of their neighbors, and each of these neighbors do the same. While this effectively distributes a client's search to a large number of nodes quickly, it also serves as a natural amplifier for malicious nodes that are interested in attacking the system by simply generating many, many queries. To deal with this problem, we developed a traffic model that can be used to understand the effects of query flooding in the Gnutella network. We ran simulations based on the model on small network topologies (14 to 16 nodes) to fundamentally analyze how different choices of network topology and application-level load-balancing policies minimized the effect of these types of DoS attacks. We found that complete and grid network topologies, when used together with "fractional" and "prefer-high-ttl" traffic management policies, are able to cut the amount of query processing induced by malicious nodes by a factor of 2 to 4. In [21], we expand on this work by experimenting with larger networks of thousands of nodes arranged in hypercube-like topologies that we designed based on our findings in [13].

In [12], we studied the new GUESS protocol, noted in Section 2.1.1. In this protocol, nodes do not arrange themselves into an explicit software overlay topology. Instead, each node keeps track of a list of other nodes that they interacted with in the past in a data structure called a "pong cache." Since nodes in the system may leave at any time without giving notice to nodes that include them in their pong cache, some entries in pong caches may become invalid. In GUESS, nodes continuously exchange information about which nodes are available to process queries through a series of ping and pong messages, in the hopes of keeping their pong caches populated with nodes that are available.

Malicious nodes may collude in an attempt to attack a GUESS system in many ways. For example, they may work to propagate their node ids into the pong caches of many other nodes, and then all leave the system at the same time, leaving the pong caches of nodes in the system filled with invalid entries. The resulting network is likely to be fragmented or partitioned, and good nodes will have trouble finding a critical mass of other good nodes to which to send their queries. In [12], we study how to mitigate such denial-of-service attacks that can be carried out by malicious nodes "poisoning" the pong caches of good nodes in the system. We find that damage can be minimized by using cache management strategies that balance node ids equally across pong caches.

4.2 Authenticity and Trust

It has been suggested that the future development of P2P systems will depend largely on the availability of novel methods for ensuring that peers obtain reliable information on the quality of resources they are receiving [10]. In this context, attempting to identify malicious peers that provide inauthentic files or bogus content is more effective than attempting to identify inauthentic resources themselves, since malicious peers can easily generate a virtually unlimited number of inauthentic resources if they are not banned from participating in the network. The process of tracking the apparent behavior of peers and selecting resource providers based on such information is the work of a reputation system.

One weakness of reputation systems is their reliance on persistent identity in order to maintain a behavioral history of nodes in the network. Due to the open and anonymous nature of P2P networks, it may be infeasible to enforce the usage of persistent non-repudiable identities by all nodes. Thus, a malicious node's ability to change identities would require that new nodes in the network be treated with equal suspicion as overtly misbehaving nodes. But malicious nodes could not prevent well-behaved nodes from accruing a positive reputation, associated with some form of unforgeable identity. Tying a node's ability to access resources to their perceived reputation would encourage nodes to participate fairly and provide incentive to share resources.

Many reputation systems have been proposed to deal with authenticity attacks in P2P networks, but little work has gone into evaluating and comparing them. For this purpose, we are developing an extensible simulation model as well as several metrics for analyzing P2P reputation algorithms and techniques.

The first questions our model addresses are, what does it mean for a file or document to be authentic, and how is the authenticity verified? For simplicity we maintain a strict definition of authenticity, appropriate for document preservation and retrieval systems: a file must contain sufficient metadata to uniquely describe its content, and the metadata must be consistent with itself and the content. When a document is fetched from a peer its authenticity is checked by the receiver. This may be accomplished programmatically, but most often may involve the human user or a third-party. This authenticity check is usually the most expensive part of the process from the user's perspective. Therefore a key function of a reputation system would be to reduce the number of authenticity checks performed on bogus files while maintaining the effectiveness of the system at answering queries. This constitutes one of the metrics used by our comparative model.

In addition to developing a model and associated metrics for evaluating reputation systems, several projects have designed new and innovative reputation algorithms targeted at the authenticity attacks existing in deployed P2P networks. One such system is known as "EigenTrust".

The EigenTrust algorithm [25] is a method for assigning each peer i a unique *global trust value* that reflects the experiences of all peers in the network with peer i. At the same time, the EigenTrust algorithm is applicable in entirely decentralized P2P systems, not requiring any centralized, globally-trusted servers.

The basic concept behind EigenTrust is that each peer i is assigned a *global trust value*, or *EigenTrust score*, that is given by the sum of local trust values assigned to peer i by the peers who have interacted with it, weighted by the global trust values of those assigning peers. Thus, authenticity evaluations of peer i's resources by many different other peers in the network are aggregated into a fair and globally known trust value for peer i. The algorithm has been shown to resist attacks, even when collectives of malicious peers cooperate to boost the global trust values of selected malicious peers.

The recursive weighting leads to a large eigenvector computation, much like the PageRank algorithm for web search [22]. In the EigenTrust algorithm, all peers in the network participate in computing the global EigenTrust scores in a distributed and node-symmetric manner with minimal overhead on the network. The scores are stored in a content-addressable overlay network formed by the participating peers themselves and are thus globally accessible.

Global EigenTrust scores of peers can be used in a variety of ways. First, these values can effectively isolate malicious peers from a P2P network. Peers that provide material deemed inappropriate by the users of the network are not chosen as download source any more if peers bias the selection of their sources of downloads based on EigenTrust scores.

Second, EigenTrust scores may be interpreted as an evaluation of a peer's active contributions to a P2P network [17]. P2P networks tend to suffer from a large percentage of freeloaders, peers which do not contribute any resources to the network, yet consume bandwidth. EigenTrust scores can be used to drive quality of service for peers in a P2P network. For example, peers with high EigenTrust scores can be granted superior access and superior view of the network by reserving them higher download bandwidths and increasing the hop count horizon of their queries. Such networks – networks in which high EigenTrust scores are used as incentives – effectively foster active participation of all peers and may serve to reduce the number of freeloaders and to improve the overall performance of the network.

5 Conclusion

In this paper, we have presented an overview of the research relating to P2P systems proceeding within the Peers group at Stanford University. For more information on the projects discussed here, as well as more recent work, refer to the group's website at `http://www-db.stanford.edu/peers/`.

References

[1] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani. Estimating aggregates on a peer-to-peer network. Technical report, Computer Science Dept., Stanford University, 2003.

[2] M. Bawa, R. J. Bayardo Jr., S. Rajagopalan, and E. J. Shekita. Make it fresh, make it quick — searching a network of personal webservers. In *Proc. of the 12th Intl. Conf. on World Wide Web (WWW)*, 2003.

[3] M. Bawa, G. S. Manku, and P. Raghavan. SETS: Search Enhanced by Topic-Segmentation. In *Proc. of the 26th Intl. ACM Conf. on Research and Development in Information Retrieval (SIGIR)*, 2003.

[4] B. Bloom. Space/time Trade-offs in Hash Coding with Allowable Errors. In *Communications of ACM*, volume 13(7), pages 422–426, 1970.

[5] B. F. Cooper and H. Garcia-Molina. Creating trading networks of digital archives. In *Proc. 1st Joint ACM/IEEE Conference on Digital Libraries (JCDL)*, June 2001.

[6] B. F. Cooper and H. Garcia-Molina. Bidding for storage space in a peer-to-peer data preservation system. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, 2002.

[7] B. F. Cooper and H. Garcia-Molina. Peer-to-peer data trading to preserve information. *ACM Transactions on Information Systems (TOIS)*, 20(2), April 2002.

[8] B. F. Cooper and H. Garcia-Molina. Ad hoc, self-supervising peer-to-peer search networks. Technical report, Computer Science Dept., Stanford University, 2003.

[9] B. F. Cooper and H. Garcia-Molina. SIL: Modeling and measuring scalable peer-to-peer search networks. Technical report, Computer Science Dept., Stanford University, 2003.

[10] F. Cornelli, E. Damiani, S. De Capitani Di Vimercati, S. Paraboschi, and S. Samarati. Choosing reputable servents in a P2P network. In *Proceedings of the 11th World Wide Web Conference*, May 2002.

[11] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *Proc. of the 28th International Conference on Distributed Computing Systems*, July 2002.

[12] N. Daswani and H. Garcia-Molina. Pong-cache poisoning in GUESS. preprint.

[13] N. Daswani and H. Garcia-Molina. Query-flood DoS attacks in Gnutella networks. In *ACM Conference on Computer and Communications Security*, 2002.

[14] P. Ganesan, Q. Sun, and H. Garcia-Molina. YAPPERS: A peer-to-peer lookup service over arbitrary topology. In *Proc. of the 22nd Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM)*, 2003.

[15] Gnutella specification. www9.limewire.com/developer/gnutella_protocol_0.4.pdf.

[16] GUESS specification. groups.yahoo.com/group/the_gdf/files/Proposals/GUESS/guess_01.txt.

[17] S. Kamvar, M. Schlosser, and H. Garcia-Molina. Incentives for combatting freeriding on P2P networks. Technical report, Stanford University, 2003.

[18] Kazaa. www.kazaa.com.

[19] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proc. of the ACM Symposium on Theory of Computing (STOC)*, 2000.

[20] G. S. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed hashing in a small world. In *Proc. of the 4th USENIX Symp. on Internet Technologies and Systems (USITS)*, 2003.

[21] Q. Sun N. Daswani, M. Gulati and H. Garcia-Molina. On the flood-tolerance of large Gnutella topologies. preprint.

[22] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[23] S. Ratnasamy, P. Francis, M. Handley, and R. M. Karp. A Scalable Content-Addressable Network (CAN). In *Proc. of ACM SIGCOMM*, 2001.

[24] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of the Intl. Conf. on Distributed Systems Platforms (Middleware)*, pages 329–350. IFIP/ACM, 2001.

[25] M. Schlosser S. Kamvar and H. Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *WWW 2003*, 2003.

[26] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. A scalable and ontology-based P2P infrastructure for semantic web services. In *Proceedings of the 2nd International IEEE Conference on P2P Computing*, Linkoping, Sweden, September 2002.

[27] I. Stoica, R. Morris, D. Karger, M. Frans Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of ACM SIGCOMM*, pages 149–160, 2001.

[28] Q. Sun and H. Garcia-Molina. Partial lookup services. In *Proc. of the 23rd Intl. Conf. on Distributed Computing Systems (ICDCS)*, 2003.

[29] B. Yang and H. Garcia-Molina. Improving efficiency of peer-to-peer search. In *Proc. of the 28th International Conference on Distributed Computing Systems*, July 2002.

[30] B. Yang and H. Garcia-Molina. Designing a super-peer network. In *Proc. of the 19th International Conference on Data Engineering*, March 2003.

[31] B. Yang, S. Kamvar, and H. Garcia-Molina. Addressing the non-cooperation problem in competitive P2P systems. Stanford University Database Group Technical Report, 2003.