An Agent-Based Approach for Trustworthy Service Location*

Pınar Yolum Munindar P. Singh

Department of Computer Science
North Carolina State University
Raleigh, NC 27695-7535, USA
{pyolum, mpsingh}@eos.ncsu.edu

Abstract. We view the Internet as supporting a peer-to-peer information system whose components provide services to one another. The services could involve serving static pages, processing queries, or carrying out transactions. We model service providers and consumers as autonomous agents. Centralized indexes of the web are replaced by individual indexes kept by the agents. The agents can cooperate with one another. An agent may provide a service to another agent or give a referral that leads it in the right direction. Importantly, the agents can judge the quality of a service obtained and adaptively select their neighbors in order to improve their local performance.

Our approach enables us to address two important challenges. One, in contrast with traditional systems, finding trustworthy parties is nontrivial in open systems. Through referrals, agents can help one another find trustworthy parties. Two, recent work has studied the structure of the web as it happens to have emerged mostly through links on human-generated, static pages. Whereas existing work takes an after-the-fact look at web structure, we can study the emerging structure of an adaptive P2P system as it relates to the policies of the members.

1 Introduction

Peer-to-peer (P2P) systems can provide a natural basis for large-scale, decentralized information systems architectures. The two functions of information systems, querying and modifying information, are broadened in their scopes when we move to open environments. For querying, instead of looking for correct or relevant results, we look for authoritative (more generally, trustworthy) resources who can provide correct and relevant results, even though a unique correct result may not be defined. For transactions, instead of precise or relaxed consistency, we look for trustworthy resources who can deliver consistent performance with respect to suitable (e.g., economic or contractual) criteria. In both cases, there is an increased emphasis on *locating* trustworthy resources, who are willing and able to provide the *services* needed.

* This research was supported by the National Science Foundation under grant ITR-0081742. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the National Science Foundation. We thank the anonymous reviewers for helpful comments.

Traditional mechanisms for locating services are based on search engines and registries. However, many niche providers will be invisible to traditional search engines, thereby yielding low recall. Because user needs are personalized, identifying the right services from a registry is nontrivial, thereby yielding low precision. Lastly, a registry or certificate authority cannot determine trustworthiness, especially for specific tasks.

By contrast, a P2P approach is natural. Some peers would be service providers, possibly catering to a niche clientele. Other peers would learn about and use the above peers and help others find them. When peers mutually help each other, they can potentially develop into communities of interest and practice where the reputations of different providers can be made and broken. Some peers may take on specialized functions similar to service registries, but others will still have to establish that these specialized peers make trustworthy service recommendations.

Referrals are essential for locating services in decentralized systems. Referrals have been used in specific applications (see Section 4). However, we propose that referrals form the key organizing principle for large-scale systems. Links over which parties request or give referrals and the referrals they give induce a natural structure on a system, leading to two important consequences. One, major application classes can be modeled via different structures. Two, the structure evolves in interesting ways based on the policies followed by the different parties during the referral process.

Organization. Section 2 introduces our model of adaptive agent-based P2P systems. Section 3 describes our experimental setup, key hypotheses, and results. Section 4 discusses the relevant literature and motivates directions for further work.

2 Technical Framework

We now introduce our basic model. We model a system as consisting of *principals*, who provide and consume *services*. The principals could be people or businesses. They offer varying levels of trustworthiness and are potentially interested in knowing if other principals are trustworthy. Our notion of services is broad, but we discuss two main kinds of services below. These correspond to knowledge management and e-commerce, respectively.

The principals can track each other's trustworthiness and can give and receive *referrals* to services. Referrals are common in distributed systems, e.g., in the domain name system (DNS), but are usually given and followed in a rigid manner. By contrast, our referrals are flexible—reminiscent of referrals in human dealings. Importantly, by giving and taking referrals, principals can help one another find trustworthy parties with whom to interact. Notice that trust applies both to the ultimate service provider and to the principals who contribute to referrals to that provider.

The principals are autonomous. That is, we do not require that a principal respond to another principal by providing a service or a referral. When they do respond, there are no guarantees about the quality of the service or the suitability of a referral. However, constraints on autonomy, e.g., due to dependencies and obligations for reciprocity, are easily incorporated. Likewise, we do not assume that any principal should necessarily be trusted by others: a principal unilaterally decides how to rate another principal.

The above properties of principals match them ideally with the notion of *agents*: persistent computations that can perceive, reason, act, and communicate. Agents can represent different principals and mediate in their interactions. That is, principals are seen in the computational environment only through their agents. The agents can be thought of carrying out the book-keeping necessary for a principal to track its ratings of other principals. Moreover, the agents can interact with one another to help their principal find trustworthy peers.

In abstract terms, the principals and agents act in accordance with the following protocol. Either when a principal desires a service or when its agent anticipates the need for a service, the agent begins to look for a trustworthy provider for the specified service. The agent queries some other agents from among its *neighbors*. A queried agent may offer its principal to perform the specified service or, based on its *referral policy*, may give referrals to agents of other principals. The querying agent may accept a service offer, if any, and may pursue referrals, if any.

Each agent maintains models of its acquaintances, which describe their *expertise* (i.e., quality of the services they provide), and *sociability* (i.e., quality of the referrals they provide). Both of these elements are adapted based on service ratings from its principal. Using these models, an agent applies its *neighbor selection policy* to decide on which of its acquaintances to keep as neighbors. Key factors include the quality of the service received from a given provider, and the resulting value that can be placed on a series of referrals that led to that provider. In other words, the referring agents are rated as well. An agent's own requests go to some of its neighbors. Likewise, an agent's referrals in response to requests by others are also given to some of its neighbors, if any match. This, in a nutshell, is our basic social mechanism for locating services.

The above framework accommodates the following important properties of open information systems. One, the peers can be *heterogeneous*. The peers can offer services or follow policies distinct from all others. Two, each peer operates *autonomously* based on its local policies. Three, the peers can *adapt*. Each peer can arbitrarily modify its offerings and their quality, its policies, and its neighbors.

Together, the neighborhood relations among the agents induce the structure of the given society. In general, as described above, the structure is adapted through the decisions of the different agents. Although the decisions are autonomous, they are influenced by various policies.

2.1 Applicable Domains

The above framework enables us to represent different application domains naturally. In a typical commerce setting, the service providers are distinct from the service consumers. The service consumers lack the expertise in the services that they consume and their expertise doesn't get any better over time. However, the consumers are able to judge the quality of the services provided by others. For instance, you might be a consumer for auto repair services and never learn enough to provide such a service yourself, yet you would be competent to judge if an auto mechanic did his job well. Similarly, the consumers can generate difficult queries without having high expertise. For example, a consumer can request a complicated auto-repair service without having knowledge of the domain.

The commerce setting contrasts with the knowledge management setting where the idea for "consuming" knowledge services might be to acquire expertise in the given domain. Yet the consumer might lack the ability to evaluate the knowledge provided by someone who has greater expertise. However, agents would improve their knowledge by asking questions; thus their expertise would increase over time. Following the same intuition, the questions an agent generates would also depend on its expertise to ensure that the agent doesn't ask a question whose answer it already knows.

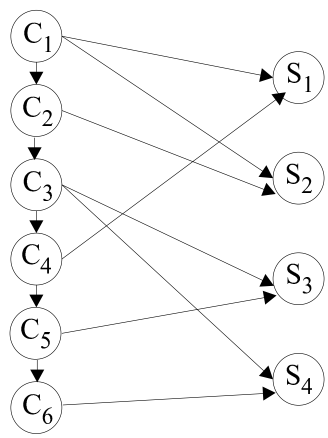

Fig. 1. A schematic configuration for e-commerce

Figure 1 is an example configuration of service consumers and providers that corresponds to a commerce setting. The nodes labeled C denote consumers and the nodes labeled S denote service providers. Consumers are connected to each other as well as to the service providers. These links are essentially paths that lead to service providers with different expertise. In this model, the service providers are dead ends: they don't have outgoing edges because they don't initiate queries or give referrals. Thus their sociability stays low. Their true and modeled expertise may of course be high. Section 3.3 considers some other characterizations of structure that influence and are influenced by different elements of our approach.

2.2 Evaluation Metrics

The relevant global properties of the system that we study here are formally characterized by some metrics, usually involving vector operations.

Qualifications. Two variants of a provider's expertise for a desired service are introduced. To capture the *similarity* between an agent and a query, we seek a formula that is commutative, i.e., a vector i is as similar to j as is j to i. A common similarity measure is the cosine of the angle between two vectors, but measuring the similarity of two vectors using the cosine of the angle between them does not capture the effect of the length of the vectors. Since the two vectors will always be in the first quadrant, we choose a formula that does not consider the angle between the two vectors explicitly. The following formula captures the Euclidean distance between two vectors and normalizes it to get a result between 0 and 1. It also applies in measuring the similarity of the members in a group based on their interests and expertise. (n is the number of dimensions I and J have.)

$$I \oplus J = \frac{e^{-\|I-J\|^2} - e^{-n}}{1 - e^{-n}} \tag{1}$$

The *capability* of an agent for a query measures how similar and how strong the expertise of the agent is for a given query [11]. Capability resembles cosine similarity but also takes into account the magnitude of the expertise vector. What this means is that expertise vectors with greater magnitude turn out to be more capable for the query vector. In (2), Q refers to a query vector and E refers to an expertise vector.

$$Q \otimes E = \frac{\sum_{t=1}^{n} (q_t e_t)}{\sqrt{n \sum_{t=1}^{n} {q_t}^2}} \tag{2}$$

Quality. The quality of a system measures how easily agents find useful providers. Quality is the basis upon which different policies are evaluated. We define quality as obtained by an agent and then average it over all agents.

The *direct quality* viewed by an agent reflects, via (2), the usefulness of the neighbors of the agent, given its interest and their expertise. That is, we estimate the likelihood of the neighbors themselves giving good answers to the questions and ignoring the other agents.

Next, we take into account an agent's neighbors and other agents. Here, we measure how well the agent's interest matches the expertise of all other agents in the system, scaled down with the number of agents it has to pass to get to the agent. That is, the farther away the good agents from the agent, the less their contribution to the quality seen by the agent. The contribution of j to i's quality is given by:

$$\frac{I_i \otimes E_j}{path(i,j)} \tag{3}$$

where the shortest path length is used in the denominator. This metric is optimistic, since a provider may not respond and peers may not produce helpful referrals.

n**th Best.** For a small population, it is reasonable to assume that each agent can potentially reach all other agents to which it is connected. But in large populations, an agent will be able to reach only a small fraction of the population. For this reason, instead of averaging over all agents, we take the nth best measure. That is, we measure the quality obtained by a peer by its nth best connection in the network. The choice for n is tricky. If n is too big, each peer's quality is equally bad. On the other hand, if n is too small, the quality will reflect the neighbors quality as in the direct quality metric. For the results reported below, we take n to be twice the number of neighbors of an agent.

2.3 Evaluation Methodology

We have implemented a distributed platform using which adaptive P2P systems can be built. However, since large-scale systems of services don't yet exist, we investigate the properties of interest over a simulation, which gives us the necessary controls to adjust various policies and parameters. The simulation involves n agents, a large fraction of whom are service consumers looking for providers. Consumers have high *interest* in getting different types of services, but they have low expertise, since they don't offer services themselves. Providers have high expertise but low sociability. The interests and expertise of the agents are represented as term vectors from the vector space model (VSM) [10], each term corresponding to a different domain.

Each agent is initialized with the same model for each neighbor; this model being rigged to encourage the agents to both query and generate referrals to their neighbors. Since we do not have actual principals (i.e., humans) in the evaluation, the queries and the answers are generated by the system. More precisely, an agent generates a query by slightly perturbing its interest vector, which denotes that the agent asks a question similar to its interests. An agent answers a question if its expertise matches a question. If the expertise matches the question, then the answer is the perturbed expertise vector of the agent. When an agent gets an answer to its question, it evaluates it by again comparing the answer to the question. When an agent does not answer a question, it uses its *referral policy* to choose some of its neighbors to refer. After an agent receives an answer, it evaluates the answer by computing how much the answer matches the query. Thus, implicitly, the agents with high expertise end up giving the correct answers. After the answers are evaluated, the agent uses its *learning policy* to update the models of its neighbors. In the default learning policy, when a good answer comes in, the modeled expertise of the answering agent and the sociability of the agents that helped locate the answerer (through referrals) are increased. Similarly, when a bad answer comes in, these values are decreased. At certain intervals during the simulation, each agent has a chance to choose new neighbors from among its acquaintances based on its *neighbor selection policy*. Usually the number of neighbors is limited, so if an agent adds some neighbors it might have to drop some neighbors as well. Section 3 studies the referral policies and the neighbor selection policies in more detail.

3 Locating Service Providers

The neighborhood relations among the agents define the structure of the society. More precisely, a directed graph $G(V, E)$ is constructed, in which each node $v \in V$ in the graph represents an agent and each edge $(u, v) \in E$ between two nodes u and v denotes that v is a neighbor of u. Since the whole society can be viewed as a graph, the search for a service provider is essentially a search starting from a consumer node, which may terminate at a provider node. In this respect, the search might look trivial and could be performed with any standard search algorithm. However, there are two major challenges. One, each agent in the system has a partial view of the graph. For example, in Figure 1, C_2 knows that C_3 and S_2 are its neighbors, but may not know that S_4 is C_3's neighbor. Two, each agent in the graph is autonomous and can well have different policies to take care of different operations like answering a question or referring a neighbor. Thus, getting at a node closer to a target provider does not guarantee that the search is progressing. For example, C_2 may ask C_3 but if C_3 is not responsive, then the search path becomes a dead-end.

With only incomplete information and possible non-cooperative peers, what is a good strategy to follow in order to find the desired service providers? We approach this question from several angles. In Sections 3.1 and 3.2, we study referral and neighbor selection policies that can be used in different populations. We evaluate the performance of these policies and suggest when they can be used. In Section 3.3, we study particular topologies of networks and show why some topologies are undesirable.

3.1 Referral Policies

A referral policy specifies to whom to refer. We consider some important referral policies. We tune the simulation so that an agent answers a query only when it is sure of the answer. This ensures that only the providers answer any questions, and the consumers generate referrals to find the providers.

1. *Refer all matching neighbors*. The referring agent calculates how capable each neighbor will be in answering the given query (based on the neighbor's modeled expertise). Only neighbors scoring above a given capability threshold are referred.
2. *Refer all neighbors*. Agents refer all of their neighbors. This is a special case of the matching policy with the capability threshold set to zero. This resembles Gnutella's search process where each servent forwards an incoming query to all of its neighbors if it doesn't already have the requested file [4].
3. *Refer the best neighbor:* Refer the best matching neighbor. This is similar to Freenet's routing of request messages, where each Freenet client forwards the request to a peer that it thinks is likeliest to have the requested information [6].

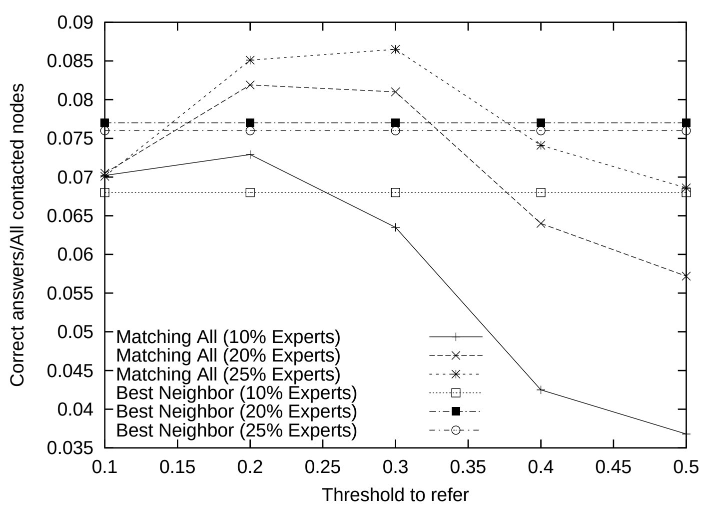

Fig. 2. Performance of referral policies

We test the performance of different policies by varying the capability threshold. Figure 2 plots this threshold versus the ratio of number of good answers received to the number of peers contacted for different policies. We plot different populations on this graph varying the percentage of experts in the population. There are three populations, each with 400 agents but with 10%, 20%, and 25% experts in them. Each agent generates eight queries during a simulation run, resulting in 3200 queries all together. Each agent is neighbors with two percent of the population, which in this case is eight agents. Each agent sends its query to its neighbors. The neighbors then apply the selected referral policy. Thus, based on the referral policy, each query results in different number of agents being contacted. We limit the length of the referral chains to five—similar to Gnutella's time-to-live value. In Figure 2, the lines marked *Matching All* show *Refer all matching* policy for varying thresholds on the x axis. The lines marked *BestNeighbor* plot the *Best Neighbor* policy, which is independent of the threshold.

Result 1 Among these referral policies *Refer all matching* finds providers with the highest ratio, where the best threshold increases with the percentage of experts in the society.

3.2 Neighbor Selection Policies

At certain intervals during the simulation, each agent has a chance to choose new neighbors from among its acquaintances. Usually the number of neighbors is limited so if an agent adds some neighbors it might have to drop some neighbors as well. A neighbor selection policy governs how neighbors are added and dropped. Such policies can strongly influence the structure of the resulting graph.

What would happen if each agent chose the best service providers as neighbors? Or is it better to choose agents with higher sociability rather than higher expertise? At one extreme, if each agent chooses the best providers it knows as neighbors, then the graph would acquire several stars each centered on an agent who is the best provider for the agents whose neighbor it is. On the other hand, if everybody becomes neighbors with agents that have slightly more expertise than themselves the structure will tend to be a tree, similar to an organizational hierarchy. To evaluate how the neighbor selection policies affect the structure, we compare three policies using which an agent selects the best m of its acquaintances to become its neighbors.

- *Providers.* Sort acquaintances by how their expertise matches the agent's interests.
- *Sociables.* Sort acquaintances in terms of sociability.
- *Weighted average.* Sort acquaintances in terms of a weighted average of sociability and how their expertise matches the agent's interests.

We measure the performance of different neighbor selection policies. Figure 3 plots direct quality metric versus the quality metric for different neighbor selection policies. W denotes the weight of the sociability in choosing a neighbor. When W is set to 0, the Providers policy, and when W is set to 1, the Sociables policy is in effect. Other values of W measure weighted averages of the sociability and expertise. In our simulation, each agent selects neighbors after every two queries. Thus, each policy is executed four times during the simulation run. The four points on the plot lines correspond to these.

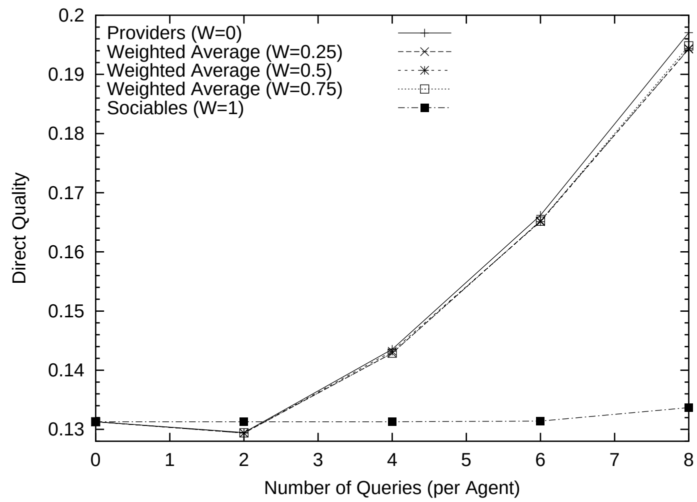

Fig. 3. Performance of neighbor selection policies

Result 2 When all agents apply the same neighbor selection policy, *Providers* yields the highest direct quality.

Notice that *Providers* might not perform as well if each agent can exercise a different policy. The benefit of this policy is that by trying to get close to the providers, each consumer maximizes the probability that it will be neighbors with at least one service provider.

While *Providers* ensures the proximity of consumers and providers, *Sociables* makes it impossible for the consumers to find the providers. Sociability corresponds to the likelihood of referring agents with high expertise. Since none of the agents have experts as neighbors, they cannot refer the experts. Interestingly, this policy results in the minimum number of neighbor changes. The sociability of the agents only increases during the first few questions, when there are still a few consumers who are still neighbors with providers. After that, those consumers who have providers as neighbors prefer more sociable consumers as neighbors.

3.3 Structure

Recall that each agent chooses its neighbors based on local information only, without knowing which neighbors other agents are choosing. Even though each agent is doing the best for itself, the resulting graph may be undesirable. Consider a bipartite graph. A graph G is bipartite if it consists of two independent sets, i.e., two sets of pairwise nonadjacent vertices. When the simulation is started, we know that there is one independent set, the group of service providers. Since these do not have outgoing edges, no two service providers can have an edge between them. Thus the providers form an independent set. Now, if the consumers also form an independent set, then the graph will be bipartite. Essentially, the consumers' forming an independent set means that all the neighbors of all the consumers are service providers. Notice that if this is the case, then the consumers will not be able exchange referrals. If the graph becomes bipartite, the system loses all the power of referrals and all consumers begin operating on the sole basis of their local knowledge. We observe that the quality of a bipartite graph is stable and non-optimal. Since the service providers do not have outgoing edges, they will not refer any new agents. Thus, the consumers will not get to know new agents, and will not be able to change their neighbors, making the graph stable. However, for each agent there will be many other agents that it cannot reach. Configurations that allow reachability to these agents will have better quality. Thus, the quality of the bipartite graph is not optimal.

Even if the graph is not bipartite, the structure could be very close to a bipartite graph. Let's say that the graph would be bipartite if we took out a few edges from the graph. This is still dangerous, since the graph might quickly evolve into a bipartite graph. The number of edges needed to be removed is a metric for determining the structural quality of the graph.

Obviously, we need to prevent the graph from turning into a bipartite graph. The only way to do so is if the agents choose their neighbors in a certain manner so as to ensure that these structures are not realized. Accordingly, we study the neighbor selection policies to see if they can cause the graph to turn into a bipartite graph.

Result 3 In a population where each agent exercises the *Providers* policy, if there are more providers than the number of neighbors an agent can have, then the graph converges to a bipartite graph.

Convergence to a bipartite graph is unavoidable when each agent discovers the service providers in the society. A partial solution is to try to obstruct this discovery by keeping the length of the referral graph short. With a short referral graph, each agent can discover only a small number of new agents. Thus, it is likely for a consumer to find a couple of service providers but unlikely that it will find all of them.

A weakly connected component of a graph is a maximal subgraph that would be connected when the edges are treated as undirected [14]. Thus different components have disjoint vertices and are mutually disconnected. Consequently, consumers can at best find service providers in their own components: We observer that if there is more than one weakly connected component in a graph, then there is at least one consumer that will not be able to find at least one service provider.

Result 4 In a population where each agent exercises the *Sociables* policy, the graph ends up a with a number of weakly connected components. Since the consumers are the only sociable agents, consumers link up with other consumers only. This results in the providers being totally isolated from the consumers.

3.4 Clustering

We define a clustering coefficient that measures how similar the neighbors of an agent are. Our coefficient is similar in motivation to Watts' coefficient [13]. However, we also take into account how similar the agent itself is to its neighbors. The average of all the agents' clustering coefficients constitutes the clustering coefficient of the graph. The reflexive interest clustering $\gamma(i)$ measures how similar the interest vectors of an agent i's neighbors (including i itself) are to each other. The reflexive interest clustering of graph G is the average of $\gamma(i)$ for all nodes in G. Below, N_i denotes the set consisting of node i and all its neighbors. E_i denotes all the edges between the nodes in N_i.

$$\gamma(i) = \frac{\sum_{(i,j)\in E_i} I_i \oplus I_j}{|N_i|(|N_i| - 1)} \tag{4}$$

Result 5 Reflexive interest clustering decreases with an increase in quality. An increase in quality shows that some consumers are getting closer to the qualified service providers. This decreases the reflexive interest clustering since now all those clustered consumers can get to the service provider through referrals and no longer need to be neighbors with other similar consumers. Consider a group of travelers who are not aware of a qualifier travel agent. As soon as one of them discovers it, the quality of the network will increase. Further, it will refer this new travel agent to its neighbors when asked for, affecting the neighbors to eventually point to the travel agent. This will decrease the interest clustering of that particular group of travelers.

4 Discussion

Our approach takes an adaptive, agent-based stance on peer-to-peer computing. This enables us to study the emergent structure of peer-to-peer networks as they are employed to help participants jointly discover and evaluate services. Below, we discuss some related approaches and then consider the greater goals of our work and the directions in which it might expand.

Directory services. WHOIS++ uses a centroid-based indexing scheme which resembles an inverted index. Each WHOIS server maintains a centroid for itself. It is free to pass its centroid to other servers. Since the servers do not model each other, they send their centroids to arbitrary servers. Lightweight Directory Access Protocol (LDAP) allows clients to access directories on different servers, which are arranged in a hierarchy. Some LDAP servers can give referrals to other servers, but as for DNS, the referrals in LDAP are given rigidly.

Referral networks. These are a natural way for people to go about seeking information [7]. One reason to believe that referral systems would be useful is that referrals capture the manner in which people normally help each other find trustworthy authorities. MINDS, based on the documents used by each user, was the earliest agent-based referral system [2]. Kautz *et al.* model social networks statically as graphs and study some properties of these graphs, e.g., how the accuracy of a referral to a specified individual relates to the distance of the referrer from that individual [5]. Yu presents a more extensive literature survey [15].

Service location. Gibbins and Hall study the techniques for query routing in mediator-based resource discovery systems [3]. They represent the queries and resources through a description logic and determine the relevance of a query to a resource by subsumption or unification. Our approach is closer to what Gibbins and Hall term the disordered mediator networks, where the mediators are not forced into a particular network topology but choose who to contact as they see fit.

Recently, several peer-to-peer network architectures have been proposed, e.g., [12, 9, 1]. Essentially, these systems model the network as a distributed hash table where a deterministic protocol maps keys to peers. The peers in these systems are not autonomous: the peers don't choose the keys that are assigned to them. Each peer has a table that aids the search when the item being searched does not reside at this peer. This is similar to our neighbors concept. However, in our approach, each peer can change its neighbors as it sees fit. Current systems lack this adaptability. First, the peers in the tables are defined deterministically. Second, peers cannot change their neighbors, unless the neighbors get off-line.

Directions. Our framework provides additional opportunities for research. One, probe deeper into the characteristics of the application domain, such as the services being offered, the demand for them, payment mechanisms in place, and so on. Two, explore the relationships between various policies and performance further, especially in the context of the structural assumptions of different applications. Three, model richer properties underlying the connectivity among the peers, e.g., communication cost and available bandwidth. This work will bring us closer to our long-term research goal of developing principles that can be cast in practical algorithms for producing robust, efficient, and trustworthy adaptive peer-to-peer information systems.

References

1. K. Aberer. P-Grid: A self-organizing access structure for P2P information systems. *Proc. Coop. Info. Sys. (CoopIS)*, pp. 179–194, 2001.
2. R. Bonnell, M. Huhns, L. Stephens, & U. Mukhopadhyay. MINDS: Multiple intelligent node document servers. *Proc. IEEE Intl. Conf. Office Automation*, pp. 125–136, 1984.
3. N. Gibbins and W. Hall. Scalability issues for query routing service discovery. In *Proc. of the Second Workshop on Infrastructure for Agents, MAS and Scalable MAS*, May 2001.
4. G. Kan. Gnutella. In *[8]*, chapter 8, pp. 94–122. 2001.
5. H. Kautz, B. Selman, & M. Shah. ReferralWeb: Combining social networks and collaborative filtering. *CACM*, 40(3):63–65, Mar. 1997.
6. A. Langley. Freenet. In *[8]*, ch. 9, pp. 123–132. 2001.
7. B. A. Nardi, S. Whittaker, & H. Schwarz. It's not what you know, it's who you know: work in the information age. *First Monday*, 5, 2000.
8. A. Oram, ed. *Peer-to-Peer*. O'Reilly, 2001.
9. S. Ratnasamy, P. Francis, M. Handley, R. Karp, & S. Shenker. A scalable content-addressable network. *Proc. SIGCOMM*, pp. 161–172, 2001.
10. G. Salton & M. J. McGill. *An Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.
11. M. P. Singh, B. Yu, & M. Venkatraman. Community-based service location. *CACM*, 44(4):49–54, Apr. 2001.
12. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, & H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. *Proc. SIGCOMM*, pp. 149–160. 2001.
13. D. J. Watts. *Small Worlds*. Princeton Univ. Press, 1999.
14. D. B. West. *Introduction to Graph Theory*. Prentice Hall, 2nd ed., 2001.
15. B. Yu. *Emergence and Evolution of Agent-based Referral Networks*. PhD thesis, Computer Science, NCSU, 2001.