

A Decentralized and Self-Organizing Discovery Mechanism

Michael Moore (mikemo@ics.uci.edu) and Tatsuya Suda (suda@ics.uci.edu)
Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425

Abstract

Distributed peer-to-peer systems often require a discovery mechanism to locate specific information, applications, or users contained within the system. In this environment, these objects often dynamically enter, leave, and travel around the network. For discovery in this paper, these objects are represented as agents that contain a limited number of relationships to other agents (relationships also include information about the other agents). We then propose a discovery mechanism that structures relationships and forwards discovery queries based on keyword similarity, usage history, and small-world clustering. Simulations are presented that demonstrate how each part of the discovery mechanism contributes to the performance of the discovery algorithm. These simulations show that the mechanism is adaptive and that the mechanism improves performance of discovery.

Introduction

Dynamic or mobile networks with peer-to-peer functionality require addressing new networking challenges. In such environments peers manage data, and collectively, large amounts of data dynamically enter, leave and migrate around the network. Discovery of data is a basic functionality that enhances the accessibility of information in the distributed context. In a dynamic environment, network services such as discovery should be decentralized in order to avoid potential computation bottlenecks. Such bottlenecks result from the need for knowledge about a centralized location, and the need for updates to that location whenever local data properties may change. Also, as network systems become large or highly dynamic, self-organization provides a reasonable mechanism for the ease of configuration and administration of such a dynamic network

Many existing peer-to-peer discovery mechanisms focus on implementing a robust distributed hash

table where data reference hash values are stored and retrieved (E.G. Chord [Cla01], Freenet [Sto00]). These mechanisms focus on locating individual hash values, which may limit the performance of other types of queries. For example, objects that are not exactly similar will be hashed into relatively random locations on the overlay network. This may make it costly to locate other objects that are similar to a query since the number of overlay network hops between similar objects may be quite large. Some other existing mechanisms use history to adapt the discovery mechanism (E.G. Neurogrid [Jos01]). History represents performance of past discoveries and is used to improve the performance of frequent queries over time.

In this paper, we propose and describe a discovery algorithm in the following sections: (1) the definition of agents and their relationships, (2) how the overall network of relationships are to be organized, (3) how relationships are acquired and selected to meet this desired relationship network organization, (4) how the relationship organization is used for discovery forwarding, and (5) simulation results that evaluate the relationship network organization.

1. Agents and relationships

In this paper, a peer-to-peer system is modeled as a population of agents networked together by relationships. Agents represent data, applications, or users available on the network and contain keywords that describe attributes they have or services they might provide. These agents contain links (also called relationships) to, and information about, several other agents. The relationships as a whole form a relationship network on which discovery is performed. Relationships and information contained in relationships provide a highly decentralized mechanism to discover other agents while remaining adaptive to a dynamic network environment. Discovery messages are forwarded

along these relationships, attempting to reach agents that would satisfy a particular discovery query.

2. Organizing the Relationship Network

In order to remain decentralized, agents in the proposed discovery mechanism improve discovery performance based only on localized information. Relationship keyword similarity, relationship history and small-world clustering represent localized information used at each agent to determine which relationships should be selected. The cumulative results of all agents' localized decisions lead to emergence of certain relationship network characteristics. *Keyword similarity* is defined as the ratio of keywords that are in common between an agent and its relationship partner. By selecting relationships based on keyword similarity, agents with similar keywords will be located nearby one another on the relationship network, allowing quick location of other objects or keywords related to the query. This results in clustering of agents with similar keywords. Within the cluster itself, there are likely to be sub-clusters in which a second keyword is shared among a subset of the agents in the original cluster. Such clustering may lead to network partitioning if no agents have relationships between dissimilar clusters. In order to ensure connectivity between these clusters, *small-world clustering* [Wat99] based on similarity is included as an additional relationship network organization. This small-world organization involves forming clusters of agents containing similar keywords and also forming random links between clusters of different keywords. The random links add connectivity between dissimilar clusters, preventing the dissimilar clusters from becoming disconnected. *Relationship history* summarizes information on how a relationship partner performed in discoveries in the past. History of a relationship is defined as the relative ratio of successful discovery queries against all the discovery queries forwarded on the relationship. An agent that has performed well in the past at satisfying discoveries is more likely to perform well in future discoveries. By using history in this discovery mechanism, discovery also gains the ability to adjust the relationship network towards various user demands. For example, an agent may

have the choice of keeping one of two equally similar relationships; however, user demand may actually find one of those relationships more useful than the other. Accordingly, the agent can use history to distinguish which of the two relationships are more likely to be useful for satisfying user demand.

3. Acquiring/Selecting Relationships:

An agent may establish a relationship with another agent through actively searching for other agents nearby, through introduction via other agents, and through discovery-related interactions. Actively searching for other agents involves broadcasting an inquiry message using the network's node level connectivity to learn about nearby agents. (Once a nearby agent is found, a relationship can be established.) An agent can introduce two agents that it has relationships with to each other by having one agent pass knowledge about itself to the other agent. Discovery related interactions occur when agents communicate during the discovery process, and can be used to introduce other agents. In this manner, a set of relationships is built and adjusted to form a network of relationships over all agents. As more relationships are acquired between the agents, these relationships form a network with various emergent properties. For example, this relationship network may contain many relationship pathways for reaching a target agent and thus, communication failure or agent loss is not likely to impact the reachability of agents during discovery. Also, if agents migrate (for example in a mobile/wireless environment), relationships are maintained, as agents are aware of which other agents they have relationships with, and the migrating agent can update the other agents' relationships when it changes location on the network. This provides a robust environment for the movement of agents. Since each agent only uses its own relationships to perform discoveries, this discovery method based on relationships is entirely decentralized.

4. Discovery Processing using the Relationship Network

The proposed discovery mechanism uses the relationship network in two phases: discovery query forwarding, discovery result backtracking. The discovery query forwarding process moves

discovery queries through relationships, seeking agents that would satisfy the discovery queries. Keyword similarity and relationship history are used at each agent to determine which relationships have priority in forwarding discovery queries. Queries are forwarded along relationships that are most similar to the current query. Since the agent of the relationship is similar to the query, that agent is likely to be part of a cluster of agents with similar properties, and should accordingly be able to reach most other agents that belong to that cluster. Relationship history is used as a secondary measure for priority during forwarding. History captures information about past discoveries, helping to distinguish between agents were not very useful in discovery vs. agents that were useful in discovery. The discovery result backtracking process gathers results from the agents visited during discovery query forwarding. Agents filter out lower quality results (i.e., results that are less related to a query) and adapt the relationships so that future discoveries can take advantage of the information gathered during the current discovery (i.e. which relationships performed well).

5. Evaluation:

The discovery mechanism in this paper was evaluated through simulations. Each agent in our simulations is generated with a set of keywords and contains a set of relationships to other agents. All agents use the same query forwarding and relationship selection policy in each simulation. In the simulation scenarios described in this paper, discovery proceeds in a depth-first fashion, and continue to propagate until the target agent is found, or until all reachable agents have been exhausted. These simulations focus on the organization of the relationships and abstract away considerations about networking issues such as processing or bandwidth constraints.

Initial Conditions: At the beginning of each simulation, 10,000 agents are generated with 5-10 random keywords (out of 680 global keywords). Each agent is assigned 10 random relationships to other agents.

Query Generation: Queries are initiated at some randomly selected agent and contain 4-5 keywords. All generated queries can be exactly satisfied by at least one agent in the network.

Search Time measurement: For each discovery, this represents the average number of simulator cycles used to fulfill a discovery. This includes the time to forward a query and to process a query. Since these simulations abstract away networking issues, the time to send a message is equal between all agent pairs.

5.1 Keyword similarity simulations

These simulations demonstrate the impact of using the keyword similarity measure in query forwarding and relationship selection. Forming these similarity-based relationships involves agents regularly acquiring random relationships and keeping the 9 relationships that are most similar to themselves along with 1 random relationship. Figure 1 shows the affect of this similarity policy. At the beginning of the simulation, relationships are random, and discovery performs poorly. As more simulator cycles elapse, agents gradually acquire many relationships similar to themselves, leading to improved performance in discovery. This improvement results from clustering of agents based on similarity: with enough hops, agents are likely to encounter a member of the cluster being searched for, after which, discovery explores that cluster to locate the other keywords of the query. The result of this similarity-based relationship simulation is compared to the results of simulations in which agents keep random relationships and forward queries based on similarity. Figure 2 shows the results of the simulation with random relationships. In this simulation, clusters are not formed, and finding a keyword of the query does not help locate other agents with that keyword. Accordingly, discovery wanders randomly until it happens to encounter the correct agent. Comparison of these results demonstrates that the clustering based on similarity is useful for improving discovery performance.

5.2 Evaluation of random relationship

Small-world clustering in this discovery mechanism includes many similar relationships and a couple random relationships. With this clustering based on similarity, the random relationship helps ensure that clusters of keywords remain connected to other clusters of keywords. Figure 3 shows simulation results of what may

occur without the random relationship. In this simulation, specific sets of keywords are much more likely to co-occur causing clustering to be much stronger (given that an agent has a particular keyword, the other keywords of that agent are taken from a strongly biased distribution). Initially, there is enough randomness in the relationships to greatly improve the performance of the algorithm; however, as the agents acquire relationships that are more similar to themselves, randomness is lost, and performance greatly degrades. This demonstrates that the random relationship of small-world clustering allows the system to perform well even when the global-level keyword sets are tightly clustered.

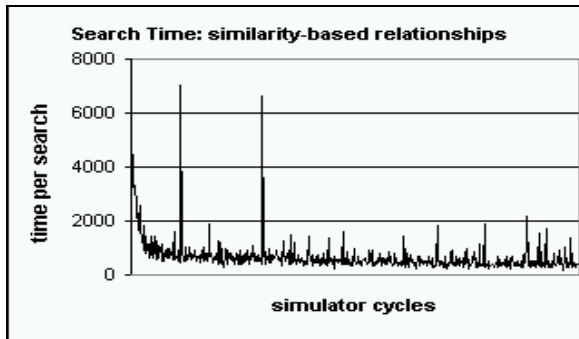


Figure 1: Similarity-based relationships

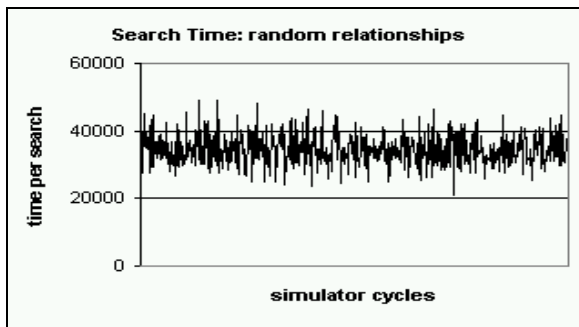


Figure 2: Random relationships

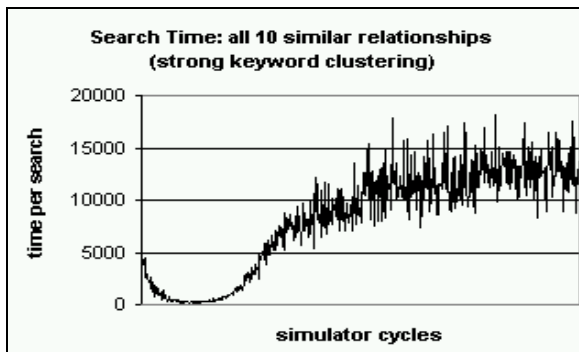


Figure 3: No random relationships

5.3a Evaluation of history

In this scenario, search forwarding and relationship selection use not only similarity as a primary measure, but also history as a secondary measure. History is calculated when search results return: history is incremented on success and decremented on failure. Failure occurs when an agent does not return a result that exactly matches the query. Table 4 compares the average search time for simulations where relationship selection and forwarding are performed based only on keyword similarity to simulations where these are based on both similarity and history. These simulations show that the addition of the history measure improved performance of discovery.

Search Type	Search Time
Similarity Only	490
Similarity with History	300

Table 4: Including history in discovery

5.3b Biased user demand:

In this scenario, users are interested in a particular subset of the keywords (about 5% of the total keywords available.) Queries are generated as before with 4-5 keywords, but at least 1 of those keywords are from the chosen subset. Table 5 compares simulations in this biased user demand: discovery using only keyword similarity (without using history) compared to discovery using both history and similarity. Usage of history shows additional performance improvement when comparing the random user demand scenario to the biased user demand scenario. This indicates that history was able to adapt the relationship network to better support this specific distribution of user demand. (Discovery with similarity only organizes relationships independent of user demand and therefore performs identically in both the random user demand and the biased user demands.)

Search Type	Search Time
Similarity Only (Biased User Demand)	490
Similarity with History (Biased User Demand)	210

Table 5: Ability of history to fine-tune network to biased user demand

5.3c Additional evaluation of biased user demand: Adapting the relationships to a particular user demand distribution may impact the ability of the system to adapt to other user demand or the ability to discover agents that are not frequently searched for by the users. Additional simulations demonstrated that the system was capable of adapting the history to improve performance towards a new distribution of user demand (users demanded a different 5% of available keywords). Other simulations demonstrated that the keywords not favored by user demand did become more difficult to find when history was used.

5.4 Evaluation of scalability

Discovery by this algorithm can generally be broken into two phases: finding clusters with a keyword matching one of the search keywords, and searching through the found cluster. Finding clusters involves following the random relationships until one of the query keywords are found, so this is expected to be proportional to the frequency of the sought after keyword. Searching through a cluster is at worst the size of the cluster. In the discovery mechanism in this paper, the resulting clusters often lack structure due to the limit on the number of relationships, so searching through clusters is generally linear to the size of the cluster.

6. Discussion:

The usefulness of the discovery mechanisms proposed in this paper depends on the distributions of object data and the distribution of query types in the system. It is likely that these distributions will be application dependent: E.G. peer-to-peer file sharing demands are very different from sensor network applications. Organization based only on small-world concepts may make infrequently occurring keywords difficult to locate. A distributed hash table provides structure in which infrequently occurring keywords may be located, but may restrict what other types of queries can be performed. History improves searchability of frequent user queries, but may make infrequently used queries difficult to satisfy. Agent discovery organization should consider the expected demands of the system, and/or discovery agents should be capable of adapting organization to match those demands.

Evaluation of this discovery algorithm should be done with respect to existing algorithms. This evaluation should consider various data distributions, data properties, types of queries users generate, the distribution of those queries, and network dynamics. A test suite representing various reasonable applications should be generated for evaluation of performance and scalability in different application domains.

The method of acquiring relationships in these simulations is very simple. Other mechanisms can be introduced to increase the rate at which the agents acquire the relationships that meet the desired relationship organization.

These simulations were done independent of network considerations. Since peer-to-peer and dynamic environments often include resource heterogeneity, the discovery mechanism should take into consideration various constraints of storage and communication bandwidth. For example, resource optimization could be done using the bio-networking architecture [WS01]. Agents are more likely to interact more often with other similar agents, so resource optimization should cause the relationship network over time to more closely correlate to the underlying physical network.

[Cla00] I. Clarke et al., "Freenet: A Distributed Anonymous Information Storage and Retrieval System", <http://freenetproject.org>.

[Jos01] S. Joseph, "NeuroGrid White Paper," http://www.neurogrid.net/WhitePaper0_3.html, 2001.

[Sto01] I. Stoica et al., "Chord: a scalable peer-to-peer lookup service for Internet applications," ACMSIGCOMM 2001 Conference. ACM, Oct. 2001. p 149-60.

[Wat99] D. J. Watts, "Small Worlds: The Dynamics of Networks between Order and Randomness," Princeton University Press, 1999.

[WS01] M. Wang and T. Suda, "The Bio-Networking Architecture: A Biologically Inspired Approach to the Design of Scalable, Adaptive, and Survivable/Available Network Applications," *Proc. of the IEEE SAINT Symposium*, 2001