A Scalable, Non-Hierarchical Resource Discovery Mechanism Based on Probabilistic Protocols†

Michael F. Schwartz

CU-CS-474-90 June 1990

Department of Computer Science
Campus Box 430
University of Colorado
Boulder, Colorado 80309-0430
(303) 492-7514

Computer network interconnection provides access to a bewildering array of resources, including databases, network services, and people in various capacities. We consider the problem of allowing users to discover the existence of such resources in a large scale, administratively decentralized environment. While hierarchically organized resource registries have good scalability properties, they provide poor support for resource discovery, because users must understand how the nested components are arranged. In this paper we present a probabilistic approach that supports non-hierarchical, attribute based "yellow pages" searches. The protocols support locating a small number of instances of moderately large classes of objects. The resource graph evolves over time in accordance with what resources exist and the types of searches that users make. Simulation results indicate that the approach can support scalable and flexible resource discovery for an environment roughly the size of a large country, with several thousand administrative domains participating in resource registration and searches. Moreover, the probabilistic search strategy naturally supports fair access among competing information providers.

†This material is based upon work supported in part by NSF cooperative agreement DCR-8420944, and by a grant from AT&T Bell Laboratories.

1. Introduction

The Networked Resource Discovery Project at the University of Colorado, Boulder, is investigating means by which users can discover the existence of a variety of resources in an internet environment, such as retail products, network services, and people in various capacities. A key problem is organizing the resource space flexibly. While approaches based on a hierarchical organization (such as the CCITT X.500 directory service [CCITT 1988]) have good scalability properties, they are ultimately of limited use for resource discovery. As a hierarchically organized service registers an increasingly wide variety of resources, searching for resources becomes difficult because users must understand how the nested components are arranged. Moreover, a hierarchically organized resource space tends to become convoluted and inconsistent as new types of resource information are encoded into the hierarchy. For example, the UNIX[1] file name */users/faculty/schwartz/pdp/monte/asynch/init.o* contains (from left to right) information about the file's disk location, creator's role, creator, research project, research subproject, algorithm variant, contents (*init* = "initialization routines"), and file type (*.o* = "object code").[2] Reorganizing such a hierarchy is time consuming and not easily accommodated, once a user base has been established. Finally, a hierarchy is inflexible. For example, in searching for people having technical expertise in three dimensional graphics algorithms, one person might prefer the resource space to be organized as "/Computers/Graphics/3D/Experts", while another might prefer an organization like "/People/Interests/Technical/Graphics/3D".

The current paper focuses on protocols to support non-hierarchical resource discovery in a large scale internet environment. We do not mean to imply that hierarchically structured resource management mechanisms are bad, or that they should not be used. Hierarchical organization is helpful for certain purposes, such as grouping related files together in a file system. Rather, we propose to *augment* any existing resource space structure with a mechanism to support non-hierarchical searches.

We focus on supporting attribute based "yellow pages" searches, for which we assume it is desirable to find a small number of instances of a moderately large class of objects. For example, in searching for a supplier of a particular piece of computer hardware, finding 5 out of 100 suppliers in a metropolitan area would often suffice. We also assume that it is acceptable to return different answers to the same query across search sessions. This assumption differentiates the resource discovery problem from traditional database and information retrieval problems [Fischer & Stevens 1990]. If consistent responses to queries were desired, one could build a front-end user interface that cached results, and provided identical responses across search sessions.

Based on these assumptions, we have designed and simulated a set of probabilistic protocols that support scalable, non-hierarchical resource discovery. The protocols are scalable in the sense that the number of network messages needed grows very slowly with the size of the network. This is significant because, to our knowledge, no scalable system has ever been built that did not depend upon a hierarchical organization [Lampson 1987]. Our approach involves building a resource graph that evolves over time in accordance with what resources exist, and the types of searches that users make. The graph structure allows nodes to be related to one another through multiple groupings that we call "specialization subgraphs", motivated in part by observations about the organization of human social networks [Schwartz & Wood 1990]. Each specialization subgraph corresponds to a grouping of nodes that are related according to some particular criterion. For example, one specialization subgraph could link databases containing information about automobile parts, while another subgraph could link databases according to geographic boundaries. The automobile parts subgraph could, in turn, have one subgraph organized according to function (engine parts, tires, etc.), another subgraph organized according to manufacturer, etc.

We use probabilistic protocols to disseminate resource information and to search the graph, and caching mechanisms to remove edges that are not actively being used. We chose probabilistic protocols for several reasons. First, because we do not require that all matching information be found, probabilistic protocols provide a less expensive means to support searches. Also, the probabilistic nature of the protocols provides a natural means to limit potentially large responses, avoiding flooding users with too much information, or overly loading the network. Moreover, providing high availability in a wide area network environment is difficult, particularly

[1] UNIX is a trademark of AT&T Bell Laboratories.

[2] This example is a modified version of one given in [Greenspan & Smolensky 1983].

if no widespread agreement has been reached on common replication protocols. Searching probabilistically rather than depending on high availability of the network links and information repositories provides a natural degree of resilience to failures.

Searching for information probabilistically also ensures fair access to competing information providers. This is an important consideration in commercial environments, in which the provider of an information service could bias sales to clients that use their service. This is currently an issue of legal contention in airline computer reservation systems. Also, in the 1988 triennial review of the Modified Final Judgement (which divested AT&T of the Regional Bell Operating Companys), Judge Greene stated this problem as the primary reason for restricting the RBOCs from providing information services, since the RBOCs control the vast majority of the communication networks that are used for information services [Greene 1988].

Because a prototype implementation would need to gain very wide usage before it could yield useful results concerning scalability, we chose to simulate the protocols. The simulation analyzes a number of parameters for an environment roughly the size of a large country, with several thousand administrative domains (such as universities or companies) participating in resource registration and searches. While some aspects of the protocols could have been explored through mathematical analysis, it is difficult to capture the behavior of caching mechanisms analytically. This was a particularly important consideration because, in an environment of the scale we consider, far more information exists than can fit within any single memory hierarchy, and hence cache management must be represented in the analysis.

The remainder of this paper is organized as follows. In Section 2 we introduce our model of the environment within which resource discovery takes place. In Section 3 we detail the graph construction and searching protocols, as well as the cache management protocols. In Section 4 we overview the simulation structure and parameters. In Section 5 we present the results of the simulation, showing how the protocols perform in response to interesting ranges of the input parameters. In Section 6 we discuss related work. Finally, in Section 7, we offer our conclusions.

2. Resource Discovery Environment

Our model of the environment within which resource discovery takes place involves four types of entities: a set of *Resource Information Repositories (RIRs)*, which will typically be databases or files; *brokers* that encapsulate the heterogeneity and access control concerns of the RIRs; *agents* that dynamically construct links between the brokers; and *clients* that initiate resource searches on behalf of users by communicating with agents. This architecture is illustrated in Figure 1.

When an RIR enters the network, its broker contacts any agent, and registers a high level description of the information it maintains. (We also refer to the registration process as "advertising".) For the sake of simplicity, in this paper we consider this description to be some list of *categories*. Category descriptions could be as simple as textual keywords, or as sophisticated as semantic database class names.

Since resource information is distributed among autonomous systems, sharing the information raises privacy and security issues. Our approach to this problem involves encapsulating each autonomous RIR by a broker responsible for accessing the RIR and deciding exactly what information may be released to the outside world. Brokers are analogous to human operators that currently allow the general public limited access to many existing RIRs. For example, telephone directory service operators will refuse to answer queries asking what person has a particular telephone number, but they will answer other types of queries.

A client initiates a resource search by contacting any agent. If the agent does not have any information about the specified categories, it tries to find some other agent that does, possibly following a chain of agents. The agents examine requests and decide, based on the named categories, how best to route searches. In some cases, agents initiate transactions with brokers to access online information maintained by various organizations around the network (e.g., a telephone directory or a company’s product line description) to discover what resources exist at these organizations. In other cases, searches are routed to other agents that know more about the resources being sought. The mechanism for deciding how to route searches is discussed in the next section.

There are a number of policy issues that we do not address in this paper. One problem involves restricting the use of aggressive advertising, which would tend to flood the network with repeated advertisements on behalf of database owners who wish to magnify their visibility on the network. Other issues include removing stale

Figure 1: Basic System Architecture

pointers to resources, and charging for advertisements and searches. While these issues are important, they are outside the scope of the current work, because we believe they can be resolved in a straightforward fashion.

3. Description of Protocols

The basic resource discovery protocol can be thought of as having two phases. In the *dissemination* phase, information about particular resources is replicated at uniformly distributed, randomly chosen nodes around the network. In the *search* phase, information is sought by sending search requests randomly around the network, in a parallel fashion. (Request routing and termination will be discussed shortly.) The point of the dissemination phase is to place information about any resource within a reasonably small neighborhood of any node in the network, so that during the search phase it is likely that the information can be found using simple random probes. Randomization provides a natural means to distribute and replicate the data.

The preceding description oversimplified the scenario in two ways. First, the two phases actually occur continuously and in an arbitrarily interleaved fashion, according to when resources are introduced into the network, and when searches are requested. Second, searches do not proceed completely at random. Over time, cache management protocols develop connections between agents that maintain related information, to form specialization subgraphs. A search initiated at a random agent may cause some random search behavior at the start of the search, until a member of an appropriate specialization subgraph is reached. If such a subgraph is reached, searches will proceed in a more directed fashion.

Below, we begin by describing a primitive to support probabilistic information dissemination and searching. We then describe the use of this primitive for information dissemination, followed by the cache management protocols that evolve specialization subgraphs.

3.1. Sparse Diffusion Multicast Primitive

To support probabilistic information dissemination and searches, we have proposed a primitive called a *Sparse Diffusion Multicast (SDM)*, defined as follows [Schwartz 1989]. Given a network containing N agents, a message is sent using an SDM to $n \ll N$ agents, selected at random. The parameter n determines the *SDM* *density*. In addition to SDM density, the SDM primitive is also parameterized by scope. For example, an SDM could be initiated within a city, or across a continent. A possible implementation of this primitive based on modifications to Deering's multicast groups mechanism [Deering 1988] is discussed in another paper [Angevine, Butt & Schwartz 1990]. This implementation does not assume that a complete list of agents exists.

As its name implies, the SDM primitive is intended to support information dissemination and searching in ways roughly analogous to the movement of molecules of an ideal gas within a container. The random diffusion mechanism of the SDM is analogous to the diffusion of a gas into a region of lower concentration. The concentration of information throughout the network is analogous to the pressure/volume relationship of an ideal gas. According to ideal gas laws, a gas in a container can be increased in presure by increasing the quantity of gas or decreasing the volume of the container. By analogy, the SDM primitive provides a means to disseminate and search for information with the property that the larger the class of object (quantity), the more dense the SDM (quantity), or the smaller the scope (volume), the "higher pressure" the information dissemination or search will be, and hence the more individual class members will be located. Hence, a broker advertising a resource within a metropolitan area will achieve more dense information diffusion than one that advertises nationally, and, similarly, a search will be successful in relation to the size of the class being sought, and to the geographical scope of the search.

3.2. Disseminating and Searching for Information Using SDMs

We begin by defining a quantity called *network diffusion density*, which determines how much effort will be devoted to disseminating and searching for information. Effort here is measured in network messages. In a network containing N agents, the network diffusion density is defined to be $k + \log_b(N)$. The log function was chosen to cause the expense of SDMs to grow very slowly in the size of the network. Note that, because this function grows so slowly, it is not necessary to have an accurate estimate of network size. k is a tunable constant that ensures that some minimum set of agents receives the transmission. b is a tunable constant that sets the scale of logarithmic growth.

When an agent is first contacted by a broker, the agent uses an SDM to announce the existence of the information available at that broker to a small number of other agents, using an SDM density equal to the network diffusion density. At randomly selected intervals, agents invoke SDMs to disseminate the set of categories in their caches to other agents in the network. Agents receiving these SDM announcements respond with the set of categories in their caches. In other words, when an SDM is used to disseminate information, all of the agents contacted perform a cache exchange with the agent that initiated the SDM. (Cache management will be considered shortly.) This protocol is illustrated in Figure 2. In this figure, hashed lines indicate SDM messages, and solid lines indicate cached pointers. Initially, node **A** issues an SDM, which reaches four other nodes. When an agent executes an SDM, it exchanges cache categories with each of the agents contacted. Over time, the information is diffused around the network as the agents exchange caches with other agents. Note that this figure indicates a much higher network diffusion density than would occur from the logarithmic function we chose.

When a client contacts an agent to initiate a search, the agent checks its cache for information about the requested resource. In addition to returning any information it has available locally about brokers that advertised the category, the agent routes the search to a number of agents equal to the network diffusion density. The choice of agents to which searches are routed depends on how much information the agent has cached about the category. If the agent has no knowledge of other agents that have information about the category, it uses an SDM to involve other agents in the search. These other agents execute the same protocol. An indication of maximum depth of recursion is passed along with the SDM to limit the cost of the search.

If, on the other hand, an agent has some information in its cache about other agents that know about the category in question, it sends queries to these agents rather than to randomly selected agents. More precisely, if the network diffusion density is d and the agent has n pointers to other agents that know about the category, it sends $min(n,d)$ searches to these other agents, and initiates the remaining searches by using an SDM with SDM density $max(0,d-n)$. We call the $min(n,d)$ searches *direct searches*, and the $max(0,d-n)$ searches *SDM searches*.

While searches from a particular agent are chosen in a fashion to guarantee that they are each routed to distinct sites, no attempt is made to avoid recursively initiated searches to agents that have previously been searched, other than avoiding having agents route searches back to the agent from which they received the

Figure 2: Information Diffusion Using the SDM Primitive

request. While it would be possible to pass a list of agents known to have been searched to agents when initiating searches at each agent, for large scale networks the probability of searching an agent multiple times is fairly small anyway. This point is discussed in Section 5.1.

Using the search protocols we have described, the cost of a search is directly proportional to the network diffusion density, and exponential in the search depth. One could modify the protocol to stop initiating new searches if some number of instances of the resource being sought have been located. We do not simulate this idea because the effect on search cost would be dependent on real system usage patterns.

3.3. Specialization Subgraph Evolution Through Cache Management Protocols

Different resources will be available in different parts of the network, and each user will probably search for certain types of resources more frequently than others depending on the user's interests. Because of these points, we have developed a mechanism to allow the graph organization to evolve in accordance with what resources exist, and the types of searches users make. We utilize a set of caching protocols for this purpose. When an agent receives an SDM from some other agent, it caches the announced categories, as well as an indication of other agents known to maintain information about those categories (which we refer to as "cognizant agents"). It also caches an indication of the brokers that originally announced instances of the categories (which we refer to as "originating brokers"). The category cache is the primary data structure involved. The cognizant agent cache and the cognizant broker cache exist as structures within each category cache. Each of the caches can be managed using a different caching policy.

Agents set cache timeouts for each category in proportion to the amount of information already cached for that category. The effect of this policy is that over time various agents tend to form specialized resource type "interests" about which they maintain many pointers. If a search is routed to one such agent, searches will be more successful for the categories with which that agent is concerned. Since agents cache pointers to other cognizant agents, groups of agents that share resource interests will form specialization subgraphs based on this caching protocol. Searches could be routed to such a subgraph because the agent is running at a site that is actively involved in using a particular type of resource, causing that focus to be reflected in the agent's cache.

Searches could also be routed to such a subgraph through the randomized search procedure described in Section 3.2.

This policy is implemented using Least Frequently Used (LFU) caching. Of course, one could choose a different cache maintenance policy (such as Least Recently Used) to bias caching in other fashions. We use LFU cache management only for the set of categories held by each agent. The set of cognizant agents and the set of originating brokers for each category are maintained using FIFO caching, since there is no reason to prefer more frequently used cognizant agents or originating brokers. In fact, it may be preferable not to prefer frequently used cognizant agents or originating brokers in this fashion, to avoid forming "cliques" of agents and brokers that use only each other for locating resources.

4. Simulation

The simulation does not attempt to capture all of the details of a real distributed implementation of the protocols described in 3. It does not differentiate between the cost of searching local versus remote agents, because we assume that as long distance network technology improves, this difference will become decreasingly relevant. Also, the simulation does not represent the possibility that agents might be added to the network incrementally. Instead, all agents are placed in the network at once. Finally, the simulation does not represent the fact that information dissemination and search activity can occur in an interleaved fashion. To realistically represent interleaved information dissemination and searching, one would need to represent real usage patterns.

4.1. Overview

When the simulation is initialized, a set of randomly chosen "categories" (represented by integer values) is generated at each agent, representing a set of broker announcements to that agent. We do not explicitly represent the existence of brokers in the simulation. Rather, we simulate the size of the network by the number of agents, with the idea that each agent is contacted by some set of brokers. The number of categories generated at each agent is a random number between 1 and a parameter that specifies an administrative limit on the number of categories any broker is allowed to advertise.

After initializing the category list for each agent, a set of *SDMAdvertise* simulation events is generated for each agent, which when executed will cause the agent to perform cache exchanges with a small number of other, randomly chosen agents, and then schedule another *SDMAdvertise* event for the agent. The simulation then executes continuously, performing *SDMAdvertise* events and cache exchanges until a specified number of simulation events have taken place. At this point, a set of randomly chosen SDM searches are initiated, based on categories that are known to have existed when the simulation started executing,[3] and various performance measurements are collected for the entire process. 500 searches are performed, to produce statistically averaged measurements. At the end of the simulation, a number of aggregate measurements are output:

- The total number of simulated categories generated by the random selection process that runs during simulation initialization.
- The total number of cache exchanges.
- The number of times cached category, cognizant agent, and originating broker records were dropped because of cache overflows.
- The average number of categories cached by each agent.
- The average number of cached cognizant agents over all categories known by an agent.
- The average number of cached originating brokers over all categories known by an agent.

In addition to these aggregate measurements, the means and standard deviations for several measurements pertaining to the attempted resource searches are output:

- The number of brokers originating information about the resources being sought.

[3] Some categories could have been dropped because of cache overflows.

- The number of originating brokers that were found through searches.
- The number of searches conducted based on cached cognizant agent pointers, representing an agent's direct knowledge of a resource category.
- The number of searches conducted through SDMs, representing a lack of direct information about the resource category being sought.
- The number of distinct agents reached during searches. This provides an indication of wasted search effort, when compared with the total number of agents searched.

We do not attempt to model the effect of cached searches on later searches directly, because achieving meaningful results would require a representation of real usage patterns. Instead, cache management is simulated during the dissemination phase, and searches are performed on existing caches without modifying their contents. The reason we do not simulate caching during searches is that the effects of caching during searches can be modeled by the effects of caching during the dissemination phase. A reasonable caching policy during the dissemination phase will tend to form specialization subgraphs in response to agents seeing announcements for certain types of categories more frequently than for other categories. Similarly, a reasonable caching policy during the search phase will tend to form specialization subgraphs in response to agents seeing searches for certain types of categories more frequently than for other categories. Therefore, the effects of caching policy on search effectiveness for non-uniformly distributed search requests can be modeled by observing the effects of caching policy on search effectiveness for non-uniformly distributed resource advertisements.

To test the caching policy under such non-uniform situations, we provide a means to instantiate a non-uniform distribution of announced categories, rather than the default uniform random distribution. This mechanism is described at the end of Section 4.2.

4.2. Parameters

The simulation supports the following parameters. Parameter names are given in italics, to ease referring back to their descriptions as simulation results are presented. We also give the range of values studied for each parameter. The parameters are as follows:

- The number of agents (*NumAgents*). Because an agent supports resource registrations and searches for an administrative domain, this parameter represents the overall scale of the network. We simulated networks of between 500 and 5000 agents.
- The constants k and b, used to set the network diffusion density. In the simulation we represented the additive constant k as *AddK*, and the base of the logarithm b as *TimesK*, since logarithms of different bases are related multiplicatively. In the simulations we fixed *TimesK* to be 2, and simulated *AddK* values between 1 and 50.
- The number of resource categories in the simulated universe (*NumCategs*). We simulated *NumCategs* values between 1000 and 9000. These small values forced individual classes to be fairly large, in accordance with the assumptions of these protocols.
- The maximum number of resource categories that any particular broker is allowed to announce (*MaxInitCategs*). This parameter represents an administrative limit on advertisements. All simulation runs used *MaxInitCategs* = 5.
- The size of each agent's category cache (*CategsPerAgent*). Because we simulated large networks, we had to impose an artificially small limit on this parameter. While a realistic system might allow thousands or more categories to be cached, for most simulation runs we set cache sizes to allow 50 categories and their associated agent pointers to be held per agent. Even so, the simulation runs typically needed 50 megabytes of physical memory to proceed without page thrashing. We simulated *fICategsPerAgent* values between 50 and 250. The sizes of the cognizant agent and originating broker caches were fixed as compile-time constants, at 45 entries per category each.
- The cache maintenance policy to use (*CacheMech*). This could be either FIFO or LFU. The latter is used for building specialization subgraphs, as discussed in Section 3.3.
- The range of delays to be used when inserting new events into the event queue (*EventDelayRange* = *MinEventDelay* to *MaxEventDelay*). These parameters represent the uncertainty in message delivery times

when using wide-area communication networks. We set *EventDelayRange* to be *1* to *NumAgents + 10*, to ensure that events representing message transmissions occur in a non-deterministic order.

- The number of events to simulate during the information dissemination phase (*MaxEvents*). Typically, we set this value to be equal to the number of agents in the simulation, allowing a subset of the agents to do SDM advertisements. Many agents do not initiate any SDM advertisements with this setting because each SDM advertisement generates a number of cache exchange events equal to the network diffusion density. However, even if an agent does not itself initiate a cache exchange, it is likely to participate in some cache exchange initiated by another agent. Note that setting *MaxEvents* equal to *NumAgents* is a pessimistic assumption. In a real implementation in a distributed environment, allowing each agent to do many cache exchanges would be quite feasible. However, since the simulation run time is dominated by cache exchanges, we needed to keep this value reasonably low. Also, note that the search phase does not execute as simulation events. Rather, once *MaxEvents* events have executed, the dissemination phase completes, and a set of searches are initiated by simply searching the data structures that contain the simulated caches. We simulated *MaxEvents* values between 1000 and 10000.
- The maximum depth of recursion in using SDM searches (*MaxSearchDepth*). The depth is counted starting at 1, with the meaning that searches are forwarded until depth equals *MaxSearchDepth* - 1. For example, *MaxSearchDepth* = 3 (the value we used) will cause searches to be forwarded from the starting agent to a set of agents, and then no deeper. We simulated *MaxSearchDepth* values between 1 and 6.
- The number of searches to initiate after the information dissemination phase completes, for statistical averaging purposes (*NumSearches*). We ran most simulation runs with *NumSearches* equal to 500, except for one particularly expensive set of runs, for which we set *NumSearches* equal to 500.
- The number of instances of a single distinguished category (*NumDistingInsts*) to generate during the part of agent initialization that corresponds to broker announcements. This parameter is used to simulate differential concentrations of resource categories, as well as differential concentrations of searches made for various categories, as discussed at the end of Section 4.1. If *NumDistingInsts* is set to zero, all categories and searches are chosen at random. If *NumDistingInsts* is greater than zero, it specifies the number of instances of a particular distinguished category to generate, and all searches are initiated for that distinguished category. This parameter allows experimenting with the abilities of different caching policies to form specialization subgraphs. We simulated *NumDistingInsts* values between 0 and 5000.

4.3. Implementation

The simulation is implemented in the C programming language, and is approximately 2800 lines long. The primary data structure is an array of structures indexed by agent number. Each entry of this array contains the set of categories cached by an agent, the set of cached cognizant agents for each category, and the set of cached originating brokers for each category. (Since brokers are not explicitly represented in the simulation, each broker in the originating broker cache is just listed as the index of the agent that originally generated the particular category instance during initialization.) All cache memory is allocated at the beginning of the simulation. Overflowed caches do not require freeing dynamically allocated memory.

We initially used a binary tree implementation for the three caches, but because the ability to simulate large networks is bound by memory limitations, we sought to eliminate the wasted memory required by pointers. Therefore, we reimplemented the cache mechanism using hash tables. However, we found that hash tables still had too much memory overhead, because of the need to maintain overflow chains. Therefore, we eventually settled on unordered linear lists. While linear lists are inefficient in terms of the number of operations needed to implement searches and inserts (which are necessary to support cache exchanges), they are memory efficient.

We ran the simulation on an 8 processor Alliant FX80 with 64 megabytes of primary memory and 117 megabytes of swap space, and on a 3 processor Solbourne Series 4/600 with 49 megabytes of primary memory and 131 megabytes of swap space. While the Solbourne has much faster processors (SPARCs, compared with the MC68020 processors on the Alliant), the larger simulations had working sets too large to fit in the primary memory of the Solbourne. These simulations were run on the Alliant. Although both of these are multiprocessor machines, the simulation is implemented on a single processor. We considered building a distributed simulation in order to have more memory available for the simulation, but decided against it, because SDMs have poor locality, and hence a distributed simulation would quickly become limited by network message transmission costs.

The simulation runs reported in Section 5 took approximately three days of computing time to complete. We ran many trials before the final runs reported here, to arrive at reasonable parameter ranges experimentally.

5. Results

In this section we present the results of our simulations. We begin in Section 5.1 by exploring the effectiveness of probabilistic information dissemination and searching without the specialization subgraph caching protocol, to isolate the effects of these two protocols. For this purpose we used a simple FIFO cache management policy. Some cache management policy was needed, because the simulation quickly filled available memory. We could also have used a random cache replacement policy for this purpose. Since category advertisements and searches are generated uniformly at random, these policies are equivalent. In Section 5.2 we explore the effect of forming specialization subgraphs on simulated network load, using an LFU cache management policy.

Throughout this section, we vary individual parameters of the simulation, and keep the other parameters fixed. Unless explicitly stated otherwise, the fixed parameters have the following values:

NumAgents = 5000
TimesK = 2, AddK = 1
NumCategs = 1000
MaxInitCategs = 5
CategsPerAgent = 50
EventDelayRange = 1 to *NumAgents + 10*
MaxEvents = NumAgents
MaxSearchDepth = 3
NumSearches = 50
NumDistingInsts = 0

These values were arrived at experimentally. For example, the maximum number of agents simulated was arrived at by testing how big a simulation could be run before page thrashing occurred on the machine running the simulation. As a different example, the values for the SDM density parameters k and b were arrived at as a compromise to reach a reasonable level of search effectiveness without causing too much search expense.

5.1. Probabilistic Information Dissemination and Searching

Basic Effectiveness and Cost as a Function of Network and Category Space Size

The basic effectiveness of using SDMs for building and searching a resource graph can be measured by plotting the proportion of brokers found that advertised a particular category being sought, as a function of the number of agents in the network. This plot is shown in Figure 3. This figure shows that even as the size of the network grows fairly large, the proportion of brokers found remains above 3%.

While finding 3% of the resources may seem small, there are several points to keep in mind. First, the protocols are designed to support yellow pages searches, for which we assume that finding a small number of instances of moderately large resource classes is sufficient. Second, the basic search effectiveness can be improved by several means, including narrowing the scope of the search to a smaller area, using a larger network diffusion density and/or search depth, and using a more intelligent caching policy. Each of these issues will be explored below.

Figure 3 also plots the network diffusion density. This value indicates the number of agents searched (and hence the search cost), since the default *MaxSearchDepth* value (3) caused all searches to be initiated from one agent.

Note that search effectiveness is not a monotonically decreasing function of network size. The non-monotonic behavior is caused by the fact that the network diffusion density function is logarithmically proportional to the number of agents, and hence increasing the network size can cause the network diffusion density to increase in discrete jumps. This is the case for the increase in search effectiveness when the network size

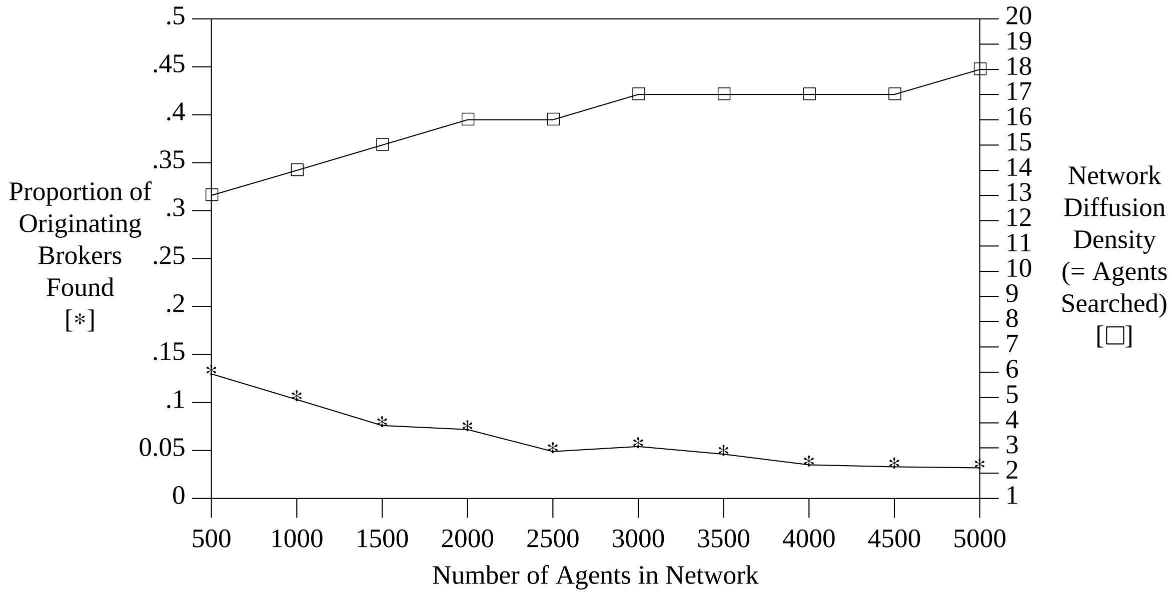

Figure 3: Basic Search Effectiveness and Cost as a Function of Network Size

increases from 2500 to 3000 agents.

Table 1 gives a more detailed view of the basic simulation results. In this table, "Actual Categories" is the sum of all unique categories randomly generated by all agents during initialization. Because these values are generated independently at random at each agent, the probability that all of the categories in the simulated universe will be used approaches 1 as the network size increases. Moments are given for values computed from averaging 500 searches on the established graph. From this table we can see that the number of originating brokers per category is actually fairly small, with a correspondingly small number of brokers being found. The effect of this small size can be seen in the fact that the standard deviations of the "Originating Brokers Found" column exceeded the means in magnitude, indicating that search success ranged fairly widely, from failed searches to very successful searches. Therefore, the search effectiveness curve in Figure 3 should be interpreted to mean that on average a reasonable proportion of categories can be found using the probabilistic dissemination and search protocol, but that if the resources being sought belong to a very small, broadly distributed class, the success in locating individual instances has high variance.

Referring to our ideal gas analogy, the next question that arises is how much more predictably searches will be successful when the "pressure" is increased. This can happen two ways: either searches are scoped to smaller areas (e.g., within a city instead of the entire network), or the resources being sought belong to a larger class. (The formation of specialization subgraphs can also aid "low pressure" searches, as will be considered in Section 5.2.) To explore this question, we restricted the size of the simulated category universe, so that each category being sought belonged to a large class. The effects of doing this are plotted in Figure 4. As can be seen, search effectiveness is significantly higher when class size is increased.

We next explored the effect of varying the number of categories on search effectiveness. The results are shown in Figure 5. Search effectiveness drops off slowly in the size of the category space. The occasional increases were caused by minor statistical inaccuracies resulting from doing 500 searches. These effects were more pronounced when only 50 searches were done per simulation run. We did not rerun the simulations using more than 500 searches because doing so would increase simulation costs, but would not provide conceptually more meaningful results.

Agents	Network Diffusion Density	Actual Categories	Cache Exchanges	Originating Brokers		Originating Brokers Found		Proportion Originating Brokers Found
				μ	σ	μ	σ	
500	13	648	461	2.520	1.230	0.330	0.560	0.130
1000	14	862	930	4.088	1.642	0.400	0.590	0.103
1500	15	952	1405	5.638	2.123	0.410	0.628	0.076
2000	16	970	1878	6.968	2.409	0.500	0.694	0.072
2500	16	993	2351	8.358	2.684	0.414	0.625	0.049
3000	17	998	2830	9.564	3.023	0.518	0.671	0.054
3500	17	998	3302	11.526	3.073	0.508	0.711	0.046
4000	17	999	3776	13.130	3.591	0.472	0.685	0.035
4500	17	1000	4244	14.694	3.879	0.470	0.699	0.033
5000	18	1000	4735	15.998	3.847	0.506	0.700	0.032

Table 1: Detailed Measurements from Basic Simulation Series of Figure 3

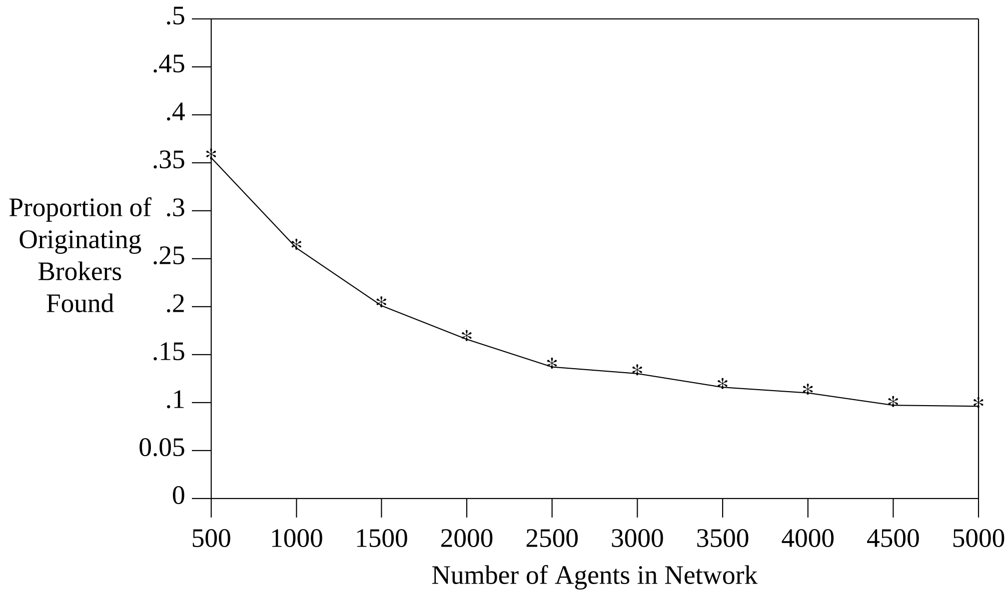

Figure 4: Search Effectiveness For Large Classes
NumCategs = 5

Effect of Network Diffusion Density and Search Depth

The next parameter combinations we explored involved the protocol parameters that control search expense and effectiveness. We began with the network diffusion density. By fixing *TimesK* and varying *AddK*, we achieved a range of network diffusion densities, against which we plotted search effectiveness in Figure 6. This figure indicates that search effectiveness grows slowly and approximately linearly as a function of the network diffusion density up to a point. Increasing the network diffusion density beyond that value does not improve effectiveness significantly, because larger network diffusion density values cause caches to overflow more quickly. This effect is exaggerated in our simulation, since the default category cache size was set to only 50 categories per agent.

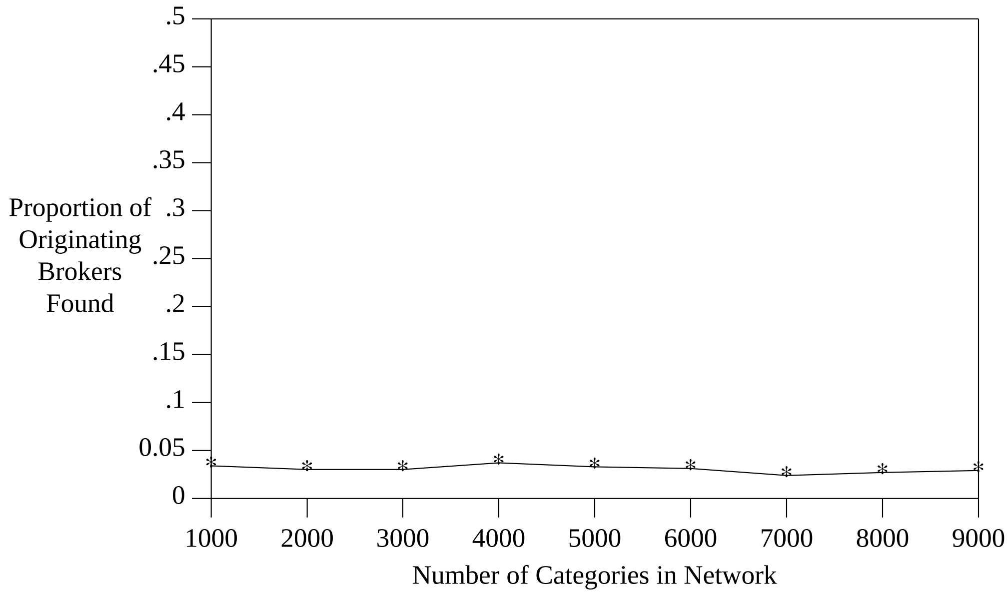

Figure 5: Search Effectiveness as Category Space Increases

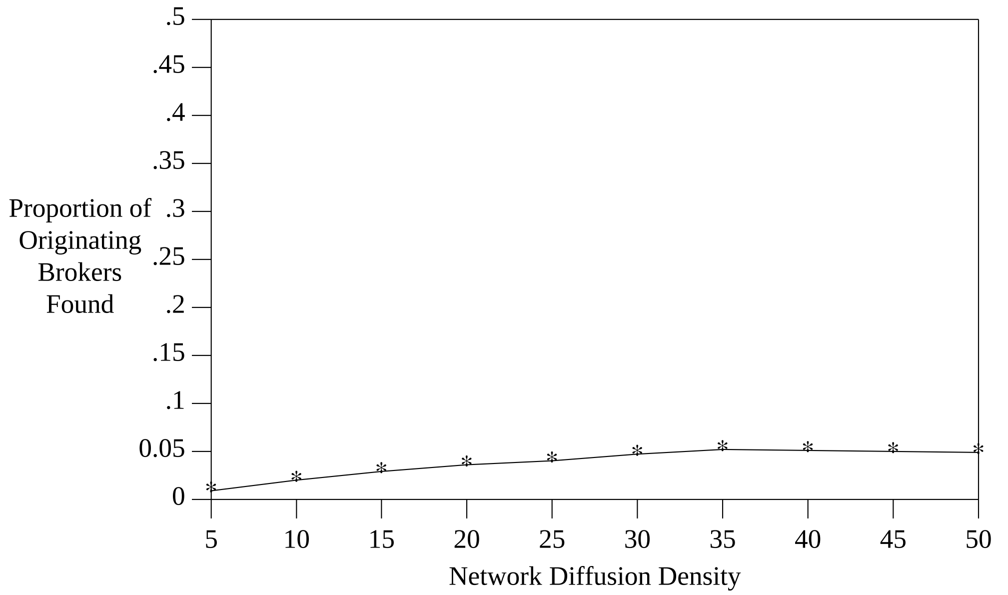

Figure 6: Effect of Network Diffusion Density on Search Effectiveness

In Figure 7 we explore the effect of search depth on search effectiveness and wasted search effort. Wasted effort is defined as the proportion of searches routed to agents that have already been searched. (See Section 3.2 for a description of how agents could be searched multiple times.) As expected, the expense of a search increases very quickly with search depth, as does wasted search effort. Note that even when all agents have been reached by a search, not all originating brokers are found. This is because of cache size limitations, which caused advertised categories to be dropped from full caches. Searches with depth 3 or less do not cause any wasted effort because searched agents are all contacted by one agent that chooses distinct agents to search. (See the definition of the *MaxSearchDepth* parameter in Section 4.2.) Also note that for this particular set of simulations, we only did 50 searches per run (rather than the standard 500), because of the expense of doing each search. At 50 searches per run the series took 10 hours to complete.

Clearly, one must attempt to choose a value below the "knee" of this curve, to avoid infeasible expense in a large scale environment. One possibility to allow fine tuning would be to support a notion of non-integral search depth. A simple algorithm to implement this idea would be to generate SDMs up to the largest integral depth less than the search depth, and then at the final level, choose to generate an SDM at each agent with a probability equal to the fractional part of the search depth. Alternatively, one could use a search depth of 3, so that all searches are guaranteed to be unique, and improve search effectiveness by varying the *AddK* and *TimesK* network diffusion density parameters. The problem with doing this is it reduces the probability of finding a specialization subgraph along which to route searches, because the specialization subgraph would only be found if the original agent contacted happened to be a member of a specialization subgraph for the resources being sought.

The network diffusion density and search depth parameters could be used as a simple mechanism to allow users to pay more for more thorough searches, particularly if non-integral values of search depth were supported.

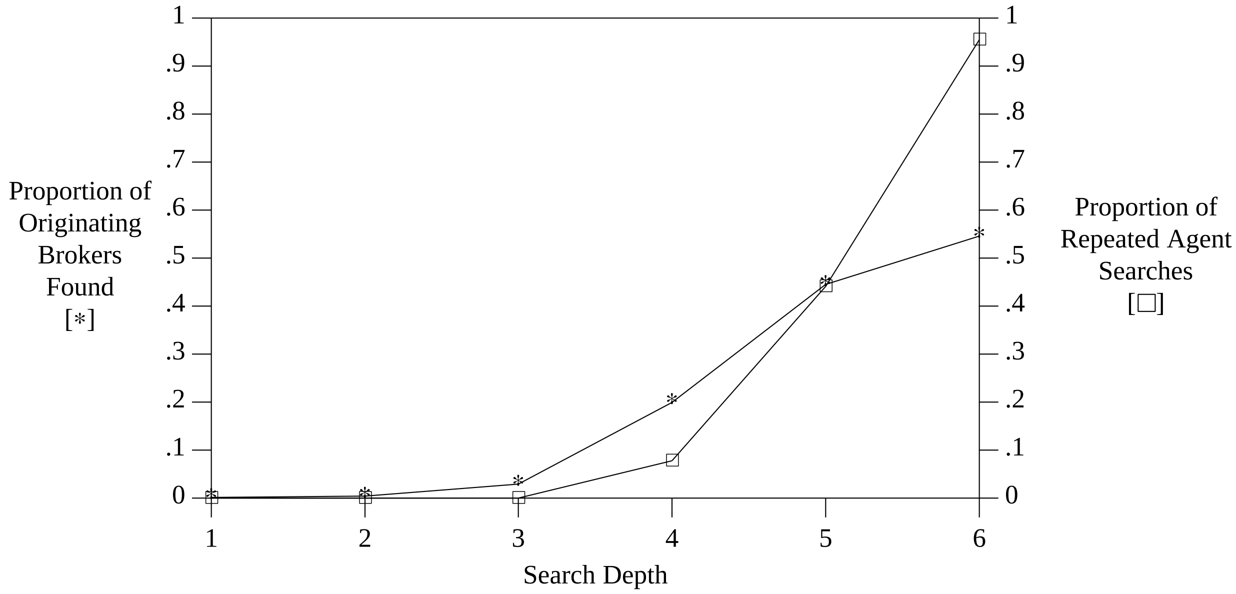

Figure 7: Effect of Search Depth on Search Effectiveness and Wasted Effort
NumSearches = 50

5.2. Specialization Subgraph Formation Through Caching Protocols

Section 5.1 indicated that the SDM primitive provides a simple and effective means to disseminate and search for information without undue expense. The caching policy used (FIFO) was purposefully simplistic, to allow the effects of the SDM to be measured with minimal interference from the effects of caching. We now consider the effect of caching policy on searches.

Throughout this section, we compare the effect of using FIFO versus LFU cache management during the dissemination phase, as measured during the search phase. This comparison is intended to indicate the potential formation of specialization subgraphs through use of a LFU caching policy. We do not directly measure the formation of specialization subgraphs, because that would require real system usage patterns to model the non-uniform nature of resource advertisements and searches. Rather, we measure the potential to support such non-uniform behavior.

The experiments we performed involved comparing FIFO versus LFU caching only in the category cache. The cognizant agent and originating broker caches are always maintained using FIFO caching, to avoid forming "cliques" of agents and brokers that use only each other for locating resources.

Figure 8 compares search effectiveness using a LFU cache maintenance policy with that from using FIFO caching for a uniform random distribution of categories in advertisements and searches. LFU caching is usually slightly more effective than FIFO caching, but since all category generations and searches are selected uniformly at random, neither policy is always superior, and neither policy works very well.

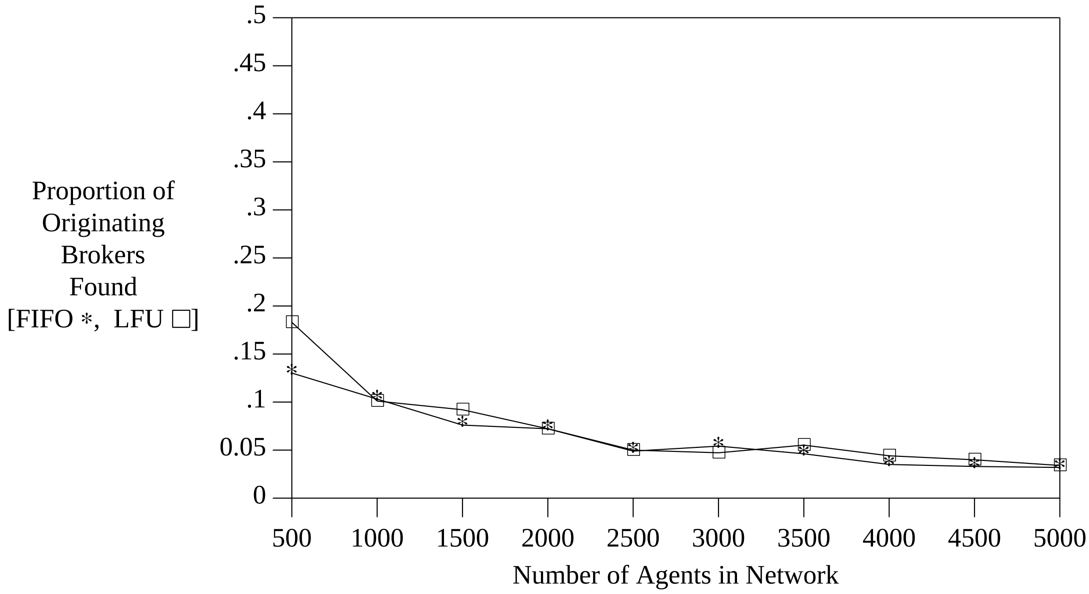

Figure 8: Basic Search Effectiveness Using Different Category Caching Policies

In a real environment, categories would not be generated or searched uniformly. Therefore, in Figure 9 (and all subsequent experiments in this section) we examine the differential effectiveness of these two caching policies for the case where categories are not generated or searched uniformly, by plotting search effectiveness against the simulation parameter *NumDistingInsts*. (See Section 4.2 for a description of this parameter.) As can be seen, when the category distribution is not uniform, LFU cache management outperforms FIFO cache management by a significant margin. Moreover, while the size of the distinguished category class increases tenfold, the search effectiveness only decreases by a factor of 3.18. This indicates that the less uniform the category space distribution, the better LFU performs for popular categories. In particular, it indicates that a search scoped for a part of the network that has a high level of interest in the category being sought will tend to be very effective.

Note that the method of generating non-uniform category spaces places an instance of the distinguished category into *NumDistingInsts* different agents' category caches during initialization. Hence, while the category space distribution becomes decreasingly uniform, the information is spread among many agents. This fact accounts for the decrease in absolute search effectiveness as the category space becomes decreasingly uniform.

The suitability of LFU caching for non-uniform category spaces can also be seen by examining what proportion of each search is satisfied through direct searches versus SDM searches. The higher this proportion, the more effective a search will tend to be, because it indicates that an agent has relatively more pointers to other agents that know about the category being sought. Figure 10 plots the proportion of direct searches to SDM searches as a function of network size, for both LFU and FIFO caching with *NumDistingInsts* = 500.

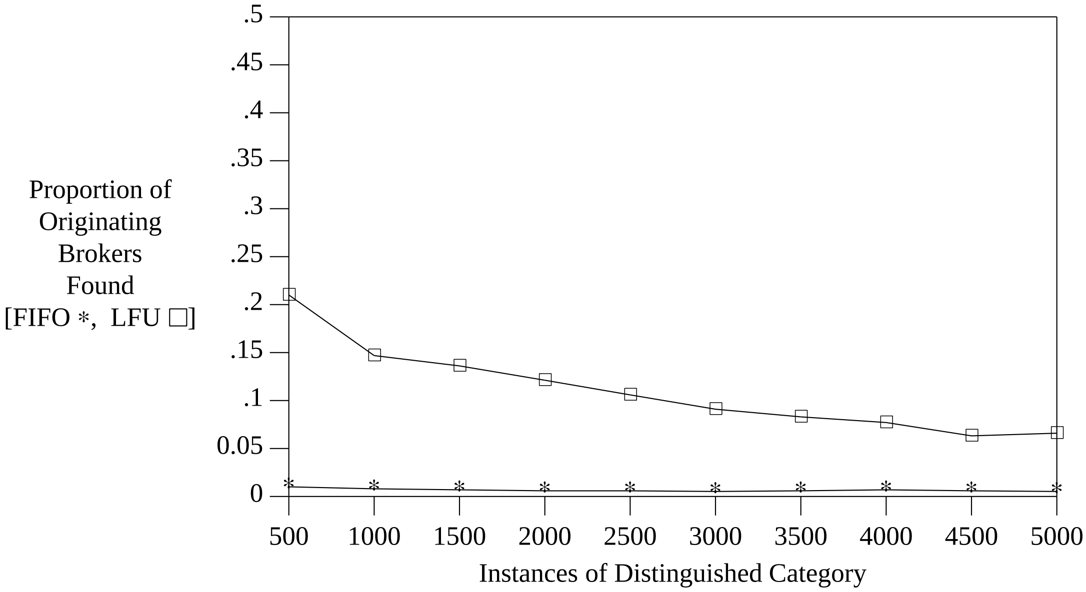

Figure 9: Search Effectiveness as a Function of Category Distribution Non-Uniformity

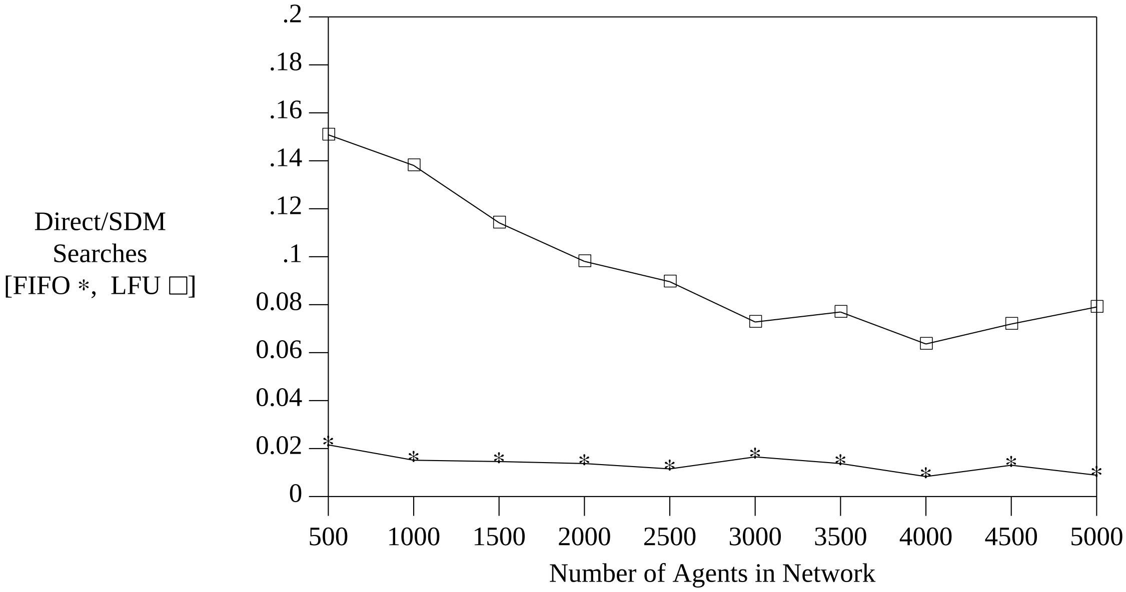

Figure 10: Direct/SDM Searches vs. Network Size for LFU vs. FIFO Caching
NumDistingInsts = 500

Cache Size and Cache Exchange Count

The next effect we measured concerned the effect of cache size. Figure 11 plots search effectiveness against category cache size, for both FIFO and LFU caching policies. We do not simulate caches larger than 250 entries because larger caches required too much memory. We also limited the number of agents simulated to 1000 to allow for larger caches. The effectiveness of FIFO caching increases approximately linearly with cache size. LFU caching effectiveness starts high, and stays approximately level. Further improvements would require more cache exchanges.

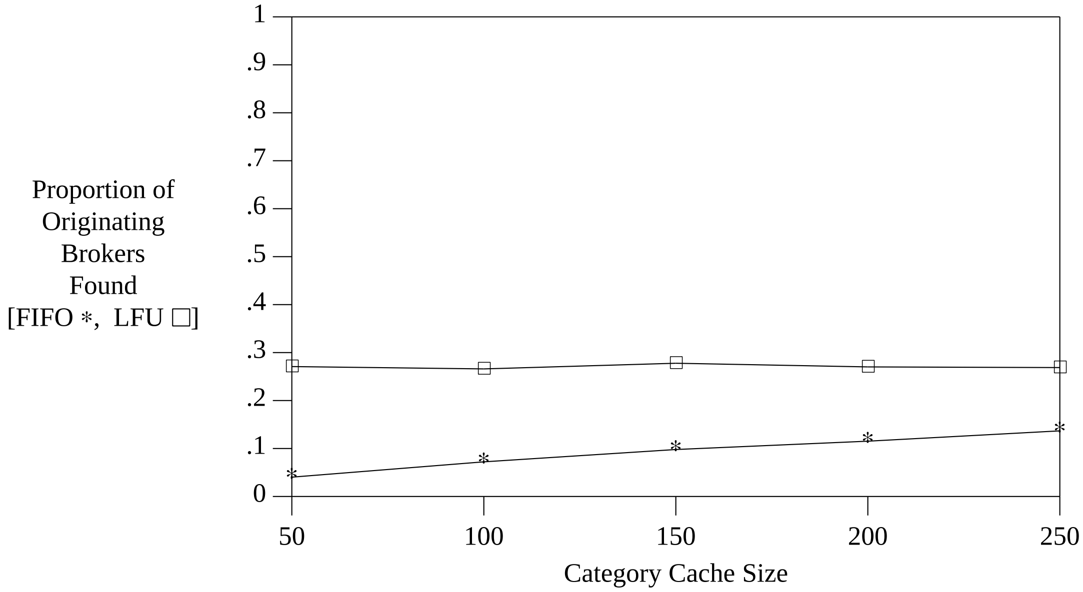

Figure 11: Search Effectiveness as a Function of Category Cache Size
NumAgents = 1000
NumDistingInsts = 500

Next, we measured the effect of the number of cache exchanges on search effectiveness. Figure 12 shows that search effectiveness grows approximately linearly until all caches fill under LFU caching, at which point caches begin to overflow and decrease search effectiveness. FIFO caching performs quite poorly here because it has no basis for making good use of an increasing number of cache exchanges.

These two sets of measurements also underscore the pessimism of the parameter ranges we needed to chose because of memory and processing constraints. A real implementation could support much larger caches and many more cache exchanges, further boosting search effectiveness.

Cache Contents and Cache Overflows

We next measured cache contents as a function of network size. As discussed in Section 3.3, we use LFU cache management only for the set of categories held by each agent. The set of cognizant agents and the set of originating brokers for each category are maintained using FIFO caching, since there is no reason to prefer more frequently used cognizant agents or originating brokers. This difference is reflected in the differences between the contents of the various caches using the two cache management policies, as shown in Table 2. This table reports the average number of entries in the various caches for the smallest and largest networks simulated. The fact that the averages fall below the maximum cache size reflects the fact that some agents received less activity than others, and hence did not fill their caches. Also, note that the primary cache is the category cache. The originating broker cache entry counts represent the total number of originating brokers for all categories in the category cache. Similarly, the cognizant agent cache entry counts represent the total number of cognizant agents for all categories in the category cache.

From the table we can see that that average cache had information about slightly more than one originating broker per category, for both FIFO and LFU caching. LFU caching tended to keep track of more brokers per category as the network increased in size. This point is related to the difference between FIFO and LFU caching for the category cache. LFU caching in the category cache caused agents to focus on fewer categories, and hence to retain significantly more pointers to other cognizant agents about those more focused categories than FIFO caching did. This increase indicates the tendency for agents under LFU cache management to form specialization subgraphs. Because agents maintain more pointers to other cognizant agents under LFU, more

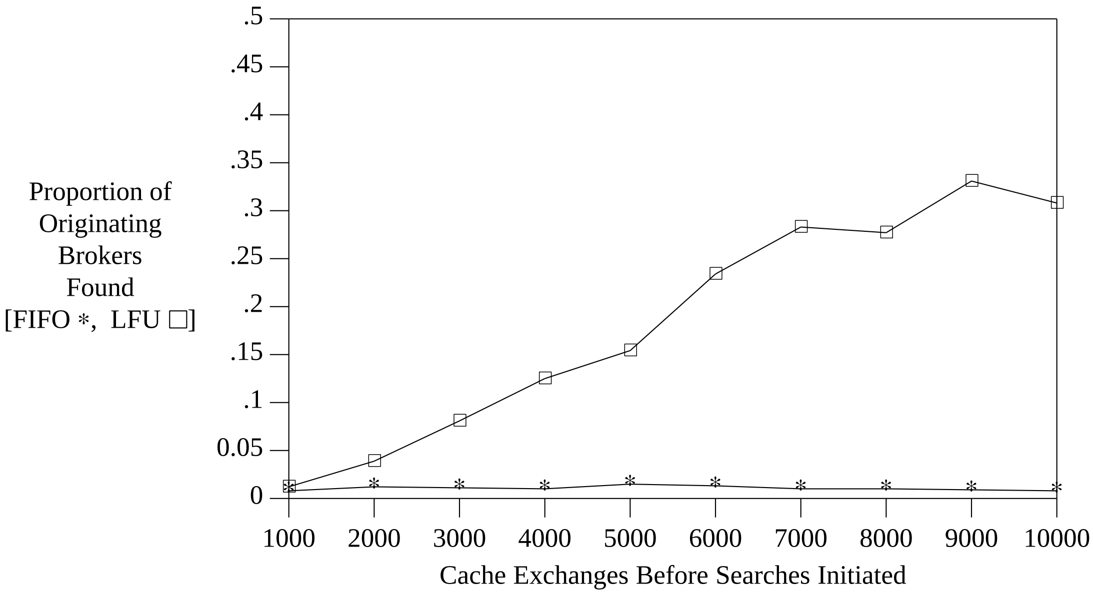

Figure 12: Search Effectiveness as a Function of Cache Exchanges
NumDistingInsts = 500

Number of Agents	Category Cache		Originating Broker Cache		Cognizant Agent Cache	
	FIFO	LFU	FIFO	LFU	FIFO	LFU
500	29.3	28.0	30.0	31.4	41.0	62.8
5000	30.3	30.2	31.4	37.6	38.1	70.4

Table 2: Average Number of Cache Entries as a Function of Network Size and Caching Policy

originating brokers can also be found, which increases search effectiveness.

As another indication of caching behavior, we measured cache overflows. The results are given in Table 3. To understand the magnitudes of the numbers in this table, note that in the worst case, for a particular cache (category, cognizant agent, or originating broker), a cache exchange will cause one agent to drop all of its cached entries when it receives the categories from the other agent. The number of cache exchanges will be slightly lower than the number of agents in the network (as discussed in Section 4.2 and Table 1). Hence, an upper bound on the number of cache drops that could occur for a particular cache is *MaxEvents* * *CacheSize*. The category cache size was set to 50 categories per agent, while the originating broker and cognizant agent caches were each set to 45 entries per category. Hence, the maximum number of category cache overflows ranges from 25,000 to 250,000, while the maximum number of originating broker and cognizant cache overflows each range from 1.125 million to 11.25 million.

The number of cache overflows in this table underscores the small simulated cache sizes. Also, the fact that an order of magnitude fewer cache overflows happened under LFU indicates the relative effectiveness of LFU caching for the category cache. This effect ripples through to the cognizant agent and originating broker caches, since they are substructures of the category cache.

Number of Agents	Category Cache		Originating Broker Cache		Cognizant Agent Cache	
	FIFO	LFU	FIFO	LFU	FIFO	LFU
500	14380	11504	14812	11189	25946	10524
5000	199666	158764	208544	175570	345900	150607

Table 3: Cache Overflows as a Function of Network Size and Caching Policy

6. Related Work

Most network directory mechanisms built to date take a deterministic approach to resource discovery. This includes the CCITT X.500 directory service standard [CCITT 1988], Peterson's Profile service [Peterson 1988], and the Internet WHOIS service [Harrenstien, Stahl & Feinler 1985]. Each of these services is tailored to handle situations requiring repeatable responses to searches for specific individuals.

Ahamad et al. developed a multicast scheme for locating specific objects in a distributed operating system [Ahamad et al. 1987], as opposed to instances of a class of objects, as in the current paper.

Gordon developed a probabilistic scheme for supporting document retrieval, which focused on allowing the system to adaptively relate keywords over time, to improve document recall as the system was used [Gordon 1988].

Deering has worked on the problem of supporting multicast in an Internet environment [Deering 1988]. Deering's work attempts to provide full delivery. In another paper we report simulation experiments for an implementation of SDMs based on modifications to Deering's multicast protocol [Angevine, Butt & Schwartz 1990].

There have also been a number of applied and theoretical results pertaining to probabilistic information dissemination. Demers et al. made use of probabilistic algorithms for circulating updates in the Clearinghouse name service, and analyzed the performance of this mechanism using techniques from epidemiology [Demers et al. 1987]. Drezner and Barak developed a probabilistic algorithm for which they showed it is possible to scatter information to a set of n nodes in $(1 + \ln 2)\log_2 n$ steps with probability approaching 1 for large values of n [Drezner & Barak 1984]. Alon et al. developed an algorithm for disseminating information reliably without broadcasting, based on a cyclic permutation matrix that describes how each node routes messages at each time unit, achieving robust broadcast without hardware support in $O(logN)$ steps for N nodes [Alon, Barak & Manber 1987]. Megiddo et al. consider the complexity of searching a graph for a fugitive with full knowledge of the captors, focusing on the complexity of determining the minimum number of searchers needed to guarantee success [Megiddo et al. 1988]. In all of these research projects, the goal was to reach a large (or complete) set of nodes through information dissemination, in contrast to our focus on finding a small number of instances of a moderately large class of objects. The protocols we have developed are much less costly because they do not attempt full replication, nor do they guarantee the ability to locate individual instances of data.

7. Conclusions

In this paper we explored a set of protocols that support attribute based "yellow pages" searches, for which we assume it is desirable to find a small number of instances of moderately large classes of objects, and that it is acceptable to return different answers to the same query across search sessions. These assumptions allowed us to take a probabilistic approach to resource discovery, resulting in a scheme that is scalable yet organizationally flexible, unlike the hierarchical organizations upon which virtually all other systems intended for use in a large scale environment depend. Moreover, locating information probabilistically ensures fair access to competing information providers.

The basic approach we have taken develops a resource graph structure dynamically, through the use of probabilistic protocols that support information dissemination and searches in ways roughly analogous to the movement of gas within a container. The protocols are scalable in the sense that their expense grows very slowly with the size of the network. Intuitively, the key to our ability to achieve scalability without hierarchical organization is that by diffusing information sparsely around a network, a small number of instances of a moderately large classes information can be found within a reasonably small neighborhood of any node in the network, so that during the search phase it is likely that the information can be found using simple random probes. This mechanism is based on the Sparse Diffusion Multicast primitive, which provides a simple and effective means to achieve a natural level of replication and distribution.

The protocols we have investigated are not solely probabilistic. The caching protocols are designed to allow the resource graph to evolve over time in accordance with what resources exist, and the types of searches that users make, based on an LFU cache management policy that links related resource information repositories into specialization subgraphs.

Simulating the dissemination and caching protocols for a realistically large scale environment is difficult because the simulation is memory intensive. Because of this, we simulated various parameters of environment size independently, including network size, cache size, and number and diversity of categories. The simulation also showed the effect of varying parameters that affect the way information and searches are disseminated. These parameters can be used to achieve desired levels of search effectiveness and per-search costs. Through a number of experimental runs, we chose an example set of such values. Our choices represent a useful part of the parameter space, where a network roughly the size of a large country can support resource discovery searches with reasonable success rates and search costs.

The measurements we have discussed show that SDM-based information dissemination and searches, combined with LFU caching, can provide a reasonably effective resource discovery mechanism for networks containing up to 5000 agents. Clearly, as the network increases in size, the effectiveness of these mechanisms will continue to decrease *for searches scoped over the entire network.* The mechanisms are only effective in relation to the number of instances of a resource being sought, the scope of the search, and the cached contents of the contacted agents. Therefore, in a global network containing, say, 1,000,000 agents, searches would only be successful if they were scoped to smaller regions, or if the class being sought had more instances, or if the agents contacted belonged to a specialization subgraph concerning the resources being sought. Note also that scoping is not limited to geographical designations. The SDM primitive can be implemented based on a notion of multicast group, the membership of which could be decided by arbitrary criteria.

Acknowledgements

We would like to thank Paul Tsuchiya for suggesting that we use simulation to experiment with these protocols. We would also like to express our appreciation to Panagiotis Tsirigotis for helping implement parts of the simulation, and for his comment on this paper. We would like to thank David Wood for discussions about this work and the paper. Finally, we would like to thank David Wagner for many useful comments about the paper and the modeling methodology.

8. References

[Ahamad et al. 1987]
M. Ahamad, M. H. Ammar, J. Bernabeu and M. Y. Khalidi. A Multicast Scheme for Locating Objects in a Distributed Operating System. Tech. Rep. GIT-ICS-87/01, School Information and Comput. Sci., Georgia Institute of Technology, Jan. 1987.

[Alon, Barak & Manber 1987]
N. Alon, A. Barak and U. Manber. On Disseminating Information Reliably Without Broadcasting. *Proc. 7th IEEE Int. Conf. Distrib. Comput. Syst.*, pp. 74-81, Sep. 1987.

[Angevine, Butt & Schwartz 1990]
S. M. Angevine, W. U. Butt and M. F. Schwartz. A Simulation Study of a Network Layer Sparse Diffusion Multicast Primitive. Tech. Rep., Dept. Comput. Sci., Univ. Colorado, Boulder, CO, July 1990. In preparation.

[CCITT 1988]
CCITT. The Directory, Part 1: Overview of Concepts, Models and Services. ISO DIS 9594-1, CCITT, Gloucester, England, Dec. 1988. Draft Recommendation X.500.

[Deering 1988]
S. E. Deering. Multicast Routing in Internetworks and Extended LANs. *Proc. ACM SIGCOMM Symp.*, pp. 55-64, Stanford, CA, Aug. 1988.

[Demers et al. 1987]
A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart and D. Terry. Epidemic Algorithms for Replicated Database Maintenance. *Proc. 6th ACM Symp. Principles Distr. Comput.*, pp. 1-12, Aug. 1987.

[Drezner & Barak 1984]
Z. Drezner and A. Barak. A Probabilistic Algorithm for Scattering Information in a Multicomputer System. Tech. Rep. CRL-TR-15-84, Univ. of Michigan, Computing Research Lab., Mar. 1984.

[Fischer & Stevens 1990]
G. Fischer and C. Stevens. Information Access in Complex, Poorly Structured Information Spaces. Tech. Rep. CU-CS-461-90, Dept. Comput. Sci., Univ. Colorado, Boulder, CO, Feb. 1990.

[Gordon 1988]
M. Gordon. Probabilistic and Genetic Algorithms in Document Retrieval. *Commun. ACM*, 31(10), pp. 1208-1218, Oct. 1988.

[Greene 1988]
H. H. Greene. United States of America, Plaintiff, v. Western Electric Company, Inc., et al., Defendants. Civil Action No. 82-0192, U.S. District Court, District of Columbia, Mar. 1988. Triennial review of Modified Final Judgement divesting AT&T of the Regional Bell Operating Companies.

[Greenspan & Smolensky 1983]
S. Greenspan and P. Smolensky. DESCRIBE: Environments for Specifying Commands and Retrieving Information by Elaboration. In *User Centered System Design, Part II: Collected Papers from the UCSD HMI Project*, Institute for Cognitive Science, Univ. of California, San Diego, Dec. 1983.

[Harrenstien, Stahl & Feinler 1985]
K. Harrenstien, M. Stahl and E. Feinler. NICName/Whois. Req. For Com. 954, Oct. 1985.

[Lampson 1987]
B. Lampson. *A Naming Service for a World-Wide Computer Network.* Digital Equipment Corporation, Syst. Research Center, Nov. 1987. Videotape Lecture from the University Video Communications Distinguished Lecture Series on Industry Leaders in Computer Science.

[Megiddo et al. 1988]
N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson and C. H. Papadimitriou. The Complexity of Searching a Graph. *J. ACM*, 35(1), pp. 18-44, Jan. 1988.

[Peterson 1988]
L. L. Peterson. The Profile Naming Service. *ACM Trans. Comput. Syst.*, 6(4), pp. 341-364, Nov. 1988.

[Schwartz 1989]
M. F. Schwartz. The Networked Resource Discovery Project. *Proc. IFIP XI World Congress*, pp. 827-832, San Francisco, CA, Aug. 1989.

[Schwartz & Wood 1990]
M. F. Schwartz and D. C. M. Wood. A Measurement Study of Organizational Properties in the Global Electronic Mail Community. Tech. Rep. CU-CS-482-90, Dept. Comput. Sci., Univ. Colorado, Boulder, CO, Aug. 1990. Submitted for publication.