Exploiting Semantics to Provide Internet White Pages Without Global Cooperation[1]

Michael F. Schwartz
Panagiotis G. Tsirigotis

CU-CS-444-89 October 1989

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

Abstract

In this paper we discuss a prototype tool that provides a simple Internet "white pages" directory facility. Given the name of a user and a rough description of where the user works (e.g., the company name or city), the tool attempts to locate telephone and electronic mailbox information about the user. The scope of the directory is large enough to be useful, yet the tool does not require the type of global cooperation that many existing or proposed directory services require, namely, having special directory servers running at many sites around the Internet. We accomplish this by building an understanding of the semantics of this particular type of resource information into the algorithms that support searches, to allow the tool to be intelligent about utilizing existing sources of information in a variety of formats and locations. At present, the tool utilizes information from USENET news messages and the "finger" protocol. Other sources of resource information (such as the CCITT X.500 directory service) can easily be incorporated into the tool as they become available. The tool achieves good response time through the use of parallel queries.

[1] 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[2] environment, including retail products, network services, and people in various capacities. While the project involves several goals and techniques [Schwartz 1989], in the current paper we discuss one particular technique applied to a single useful application. The application is providing an Internet "white pages" directory facility, to assist users in finding other users' electronic mailboxes, telephone numbers, and postal addresses. The technique involves building an understanding of the semantics of this class of resource information into the algorithms that support searches. This stands in contrast to treating information simply as text to be managed and retrieved, as many directory and information retrieval systems do.

Unfortunately, using this technique alone could require special purpose software for every type of resource (to capture resource semantics) and every user (to capture user context) in the environment. Since this combination is prohibitively expensive, the general strategy we use in the Networked Resource Discovery Project involves techniques that are not resource-specific for most cases, and semantically cognizant techniques for a few particularly useful types of resources. We also provide "hooks" in the system to allow users to write their own semantically cognizant procedures for resource types of particular interest to them.

We chose Internet white pages as a test application for the technique of exploiting semantics because it is important: with all of the work that has gone into providing usable interorganizational services like mail and naming, it is still not possible, in general, to determine the electronic mailbox for any particular Internet user. This situation has prompted efforts to provide usable network directory services on the part of CCITT [CCITT 1987] and the U.S. Federal Research Internet Coordinating Committee [Sollins 1989]. A difficult problem is providing a service that it capable of interoperating with a rapidly changing collection of hardware, network infrastructure, software, and administrative concerns. While the development of standards is clearly an important part of this process, in the short term no standard directory service is available, and in the long term, standards that are currently being devised will need to be evolved, to fit the changing technology base. Our approach to the problem is to accommodate existing systems, rather than defining a new standard.

While we find the tool described in this paper to be quite useful, it has some limitations. First, its scope is limited. It only supports searches generated by Internet users looking for other Internet users, where TCP-based "finger" packets can pass between the initiating user's machine and a machine at the site where the user being sought works [Harrenstien 1977]. (Some sites disallow interdomain finger packets for security reasons or privacy concerns.) Second, the tool is somewhat wasteful of network bandwidth. If a large number of people were to make regular use of this tool, it could contribute a non-trivial amount of load to the Internet.

We address these points later in this paper. For now, the point to bear in mind is that the tool is not intended to be a final solution to the Internet white pages problem. Rather, it was built to experiment with several interesting resource discovery techniques, and to demonstrate the feasibility of the concepts involved. It provides a useful function in the short term, until sophisticated directory services (like CCITT's X.500 [CCITT 1987] or the longer term prototype from the Networked Resource Discovery Project) are available. Moreover, the techniques we have developed can be used to access new directory services as they become available, providing a larger directory service than any one standard.

The remainder of this paper is organized as follows. In Section 2 we outline the strategy of exploiting semantics to provide a resource discovery mechanism. In Section 3 we discuss the implementation of the prototype. In Section 4 we discuss various measurements of the prototype. In Section 5 we discuss related work. Finally, in Section 6, we offer some conclusions.

---

[2] Throughout this paper we use "Internet" to refer to general internetworks. We use "Internet" to refer specifically to the growing collection of government sponsored networks (including NSFNet, ARPANET, and CSNet) and regional networks (such as Westnet and NYSERNet) that interconnect academic, industrial, and government institutions, primarily in the U.S.A.

2. Exploiting Semantics for Resource Discovery

In this section we discuss some common resource discovery mechanisms. We then introduce the semantically-based techniques we have used, and discuss the advantages of using these techniques.

Various systems have been built that provide some means of navigating around an information space. File systems typically provide operations to browse through a hierarchical file name space by reading the contents of a single directory, plus tools on top of these operations to support capabilities such as regular expression based pattern matching, interactive file name completion, and exhaustive recursive searching. As a second example, information retrieval systems typically use precomputed indexes to support text retrieval based on a set of descriptive keywords, such as the author and title of a document. Example systems include bibliographic database systems and online information services.

The fact that these systems treat resource information as simple text without any understanding of the semantics of the information causes problems for users trying to locate information. In the case of a file system, the organization can become convoluted and inconsistent over time, because users are forced to encode a variety of different semantic information syntactically into the hierarchy. For example, the UNIX[3] 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").[4] Trying to find a particular piece of information is difficult, because users must browse through various parts of the hierarchy, looking at strings that have no consistent semantic structure imposed upon them.

While index-based systems usually do not require users to browse through a hierarchy, the fact that they use simple strings to describe a wide variety of resources makes effective resource discovery difficult. When searching for a particular type of resource (such as books about some topic), users of information retrieval systems typically find that their searches either match many unwanted resources (because of lack of *precision* of the specified keyword combination) or miss wanted resources (because of lack of *recall* of the specified keyword combination) [Salton 1986]. Choosing appropriate keyword combinations becomes increasingly difficult as the size of the information space increases.

Building understanding of the semantics of the resources being sought into the algorithms that support searches can form the basis for a more effective resource discovery mechanism. Such algorithms can use information in ways that cannot be used in a syntactically-based facility. To demonstrate why this is true, we now briefly overview how the tool we have built works.

We begin by building a database of "seed" data describing machines to search for various possible requests, by gathering information from the headers of USENET news messages [Nowitz 1979]. These headers typically list the user name, organization name, city, electronic mailbox, and telephone number for users who post messages. When a search is requested (using a request format like "schwartz university colorado" or "schwartz boulder"), the seed database is consulted, using a combination of information the user is able to supply, such as the name of the institution where the person being sought works.[5] Each of the machines found in the seed database is then probed using the finger protocol, to see if the user can be found at that host. With the finger protocol, a server running on a particular machine returns the name and electronic mailbox for any user that has an account on that machine when queried about that user, plus whatever other information individual users chose to make accessible (such as organization name, telephone number, and postal address). A second finger query is made on that host without specifying a user's name, to get a list of the users currently logged into that machine, as well as the machines from which they are remotely connected. Each of the machines in this secondary list is then fingered, in case the user being sought has an account on one of those machines.

It would probably not be feasible to use USENET message header and finger information in a resource discovery mechanism that was not tailored for use in a specific application (such as Internet white pages), since

---

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

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

[5] In contrast to the information retrieval systems discussed earlier in this section, this use of simple keyword-based lookups is quite effective, because the database only contains a narrow class of information, keyed a small number of different ways (primarily, by city and organization name).

these information sources only contain data applicable to a narrow range of searches. For most types of searches (e.g., searching for retail products or network services) this information would not be relevant. It is only because the tool understands the semantics of the particular application that this information can be effectively used.

The tool's use of semantic information is actually more involved than just the fact that USENET and finger information can be used in providing an Internet white pages directory facility. In using the combination of USENET and finger information, the tool is exploiting the fact that there is a sizable and growing overlap between those sites that post messages on USENET and those sites accessible via the Internet, and thus hosts on USENET can sometimes be queried using the the finger protocol. In addition, the tool exploits the fact that people from a large number of different sites post USENET news messages, and hence over time collecting the headers of these messages is likely to produce a substantial listing of hosts and organizations. These two points contribute substantially to the tool's effectiveness, as will be discussed in Section 4.

Another aspect of the semantics exploited by the tool concerns understanding the policies and usage patterns of the organizations that provide the information it uses. In particular, the tool takes advantage of the fact that each organization sets the various fields in the news message headers differently. The more variety that is found in these strings, the more different keywords can be used in searches, and hence the more flexible the tool becomes. For example, one department at the University of Colorado might use the string "University" as part of the organizational description, while another uses "Univ.", in which case either string would work for finding machines at this university. Also, the tool exploits the fact that workstation environments often provide a central machine to process network news, and that this machine often registers many of the users at the institution, either because they have accounts on the machine, or because they have mail aliases there. Again, this fact contributes substantially to the tool's effectiveness.

Understanding semantics also helps reduce the set of unwanted responses that are presented to the user, for those cases where inappropriate matches come from similar looking keywords for the wrong type of resource. For example, in a resource discovery mechanism not tailored to the semantics of a particular class of resources, searching for "wood" might yield information about the building material when in fact the user was trying to find a person by their last name. Because our tool is aware that users are specifically being sought, it can avoid many instances of this problem.

3. Implementation

In this section we describe various aspects of the implementation of the tool, including the algorithms used, changes to the BIND name service "resolver" to make it compatible with lightweight processes, and the structure and size of prototype software.

Basic Algorithm

As mentioned in Section 2, the basic mechanism used by our tool is to gather information from USENET news messages, and use parts of the headers of this information as initial "seed" data describing sites to search using the finger protocol. The USENET data is gathered by a program that runs weekly, and traverses the USENET news spool directory hierarchy for a set of specified news groups, checking all of the files that have been created since this program last ran. The relevant data from the message headers is then extracted, and an inverted index is generated using the UNIX invert tool [Budd & Levin 1982].

At search time, this index is consulted to find matches for the keywords describing the organization name and/or city. If fewer than a threshold number of matches (currently 100) are found for a given set of keywords, a finger query is invoked for each match, within a Sun lightweight process [Kepecs 1985].[6] If the number of matches exceeds the threshold by more than 10%, the user is informed that this has happened, and is shown a partial list of the machines matched from the original query, to help the user form a more specific query.

---

[6] In building this tool, we used Sun's Lightweight Process (LWP) library package to support inexpensive concurrency within a single UNIX process. We chose this package over other packages (such as the LWP package used with the CMU/ITC Andrew RPC system [Morris et al. 1986] or AT&T's C++ tasking package [Stroustrup & Shopiro 1985]) because we wanted a package that was widely available.

Since each lightweight process uses a UNIX file descriptor for its network communication, the number of concurrent fingers that can simultaneously be active is limited. For instance, on the Sun 4/110 on which this prototype was developed this limit is 64, including 3 for standard input, standard output, and standard error. This limit is reasonable, since it allows a fairly large amount of concurrency, while limiting how much network activity a single search can generate. Since the threshold number of matches the tool is willing to honor is greater than the number of finger queries that can be active simultaneously (due to the the file descriptor limitation), the tool maintains a queue of outstanding queries, accessed from within a monitor by the lightweight processes that are doing the lookups. Usually, not all of the machines in this queue will be probed, because a response will be received from earlier machines, at which time the user can interrupt the tool.

As mentioned in Section 2, a finger query is also made on each host requesting a list of all people logged in. While the information that is returned is not in a completely standardized format, we use some simple heuristics to parse the information and extract a list of the machines from which the users are logged in. This list can then be used in further finger queries. Each time such a list is received, the individual machines in it are added to the queue of outstanding queries. A check is also made to ensure that no host is queried more than once (as could happen, for example, if the results from a two level deep finger query list the same machine more than once).

Many of the host names encountered in USENET messages are UUCP mailbox addresses. Since UUCP does not support finger queries, these addresses are not useful for this step in the process. However, in some cases the user being sought can be found directly in this seed data (because that user at one time posted a USENET message).

Changes to BIND Resolver For LWP Compatibility

Each finger query involves three network queries (and hence three places for the lightweight processes to block): the request of the Domain name service to find the Internet address of the host name on which the query is to take place, and the two finger lookups. These delays can be substantial, ranging from a few tenths of a second for a name lookup and finger query on a host in the local network, to a minute for a timed out response from a down host. It is important to allow concurrent activity to proceed during such delays. Therefore, we needed to modify the BIND *resolver*, to make it compatible with lightweight processes. The resolver is the client-resident library code that communicates with a Domain name server to initiate the various queries [Mockapetris & Dunlap 1988].

The changes involved making the routines reentrant, so that several threads could be doing Domain naming queries concurrently. This meant changing any code that accessed global or "static" variables to access a global state array indexed by a thread identifier instead. Since the BIND code is continually evolving, we sought to make these changes by modifying as little code as possible. We accomplished this by using a combination of C macro definitions and some "sed" [USENIX Association 1986] transformations in the "makefile".

Structure and Size of Prototype Software

There are several parts to the software that comprises this prototype; all of the software is written in C. In our modifications to make the BIND name service resolver compatible with lightweight processes, the original code (BIND version 4.7.3) was about 2200 lines long, including comments. Of this, we changed about 60 lines, replacing them with about 480 lines, of which about half were declarations and macro definitions.

The code that collects the USENET seed data is about 1050 lines long. The code to invert the seed data and look up keywords in the inverted index is about 850 lines long, and was taken, with few changes, directly from the bib code [Budd & Levin 1982].

The code to do the parallel finger lookups (including instrumentation for the measurements described in Section 4) is about 1800 lines long. The code to provide the top level functionality that brings the search mechanism and user interface together is about 500 lines long.

4. Measurements

To help assess the effectiveness of the techniques discussed here, we constructed a tool usage session that measured how well the tool works from a user's perspective, and how much overhead the tool imposes on the Internet. We wanted this session to represent how a typical user would use the tool. In particular, the average

user usually would not know the Domain name of the institution being sought (e.g., "colorado.edu"). A user who knows this information can generate a search request that causes fewer hosts to be probed, reducing response time and Internet loading. For example, searching for "schwartz boulder colorado edu" would probe at most 8 machines based on our current seed database, whereas the search "schwartz boulder" would probe up to 26 machines. (Fewer machines would be probed if a user got a successful response and interrupted the tool.)

The tool usage session consisted of searching for white pages information concerning 40 different specific individuals, at a variety of institutions around the U.S.A. The institutions had the following characteristics:

- 12 were on the west coast
- 4 were in the mountain region
- 6 were in the central region
- 18 were on the east coast
- 34 were universities
- 3 were government laboratories
- 3 were industrial research laboratories
- 8 had 50-200 persons
- 25 had 200-1000 persons
- 7 had 1000+ persons

In all but 4 of the searches, the response being sought was not directly available in the seed data. In the 4 cases where it was, we let the tool go so far as to finger the individuals, to count these searches in the statistics being measured.

Within these tests we collected data about the success rate, elapsed times, number of name server lookups, number of connect requests, number of finger probes at both levels of depth (direct finger and finger at a machine found via a previous finger), number of characters received, number of duplicate responses, and the effect of parallelism in queries. We ran the measurements twice: once at night, and once during the day. The purpose of doing this was to assess the effect of having more users logged in (during the day), which helps the tool to find out about other machines in the environment being searched. Also, at night the Internet is usually less loaded. During the daytime there will tend to be more congestion from other sources. We wanted to see the extent to which this affected the tool. Our results are shown in Table 1.

	Mid-Morning (Wed. 10 AM MDT) 30 Threads		Evening (Wed. 7 PM MDT) 30 Threads		Evening (Wed. 7 PM MDT) 1 Thread	
Successful searches	32		33		33	
	mean	standard deviation	mean	standard deviation	mean	standard deviation
Elapsed time, succ. [sec.]	8.15	4.18	9.88	8.68	11.13	8.07
Elapsed time, fail. [sec.]	24.57	19.45	30.71	27.79	37.86	37.28
Name Lookups, succ.	15.85	11.10	17.79	11.63	18.53	13.55
Name Lookups, fail.	10.29	12.01	12.14	12.92	6.29	8.33
Connects, succ.	18.85	12.36	22.61	15.97	23.66	17.74
Connects, fail.	8.86	15.40	10.57	17.06	6.14	10.72
Depth 0 Fingers, succ.	17.49	11.70	20.03	15.74	19.94	16.06
Depth 1 Fingers, succ.	0.52	1.13	1.42	3.34	2.72	4.86
Depth 0 Fingers, fail.	4.14	6.98	9.86	16.50	2.29	3.77
Depth 1 Fingers, fail.	1.71	4.20	0.43	0.73	3.43	7.61
Chars. Rcvd., succ.	2725.97	3172.70	9745.79	19087.34	8773.47	12209.45
Chars. Rcvd., fail.	6570.43	16094.20	1687.71	2802.64	3840.57	8331.59
Dupl. Resp., succ.	0.70	1.17	0.85	1.46	1.00	1.46

Table 1: Measured Prototype Search Statistics

In the table, "succ." in a row means that measurement was taken for successful searches, and "fail." in a row means that measurement was taken for searches that failed. We interrupted the tool when a successful response was received, as a real user of the tool would likely do.[7] We also interrupted the tool when a query took a long

[7] Some users might let this tool run "in the background" for a long period of time without watching it run. This will not cause uncon-

time (20-40 seconds) with no response. Elapsed times for failures were not much longer than for successes because users are typically willing to wait for one or more responses to print, once some evidence of the query's success is obtained.

Note that failures happened for a variety of reasons. Primarily they were because of name server lookup failures, lapses in network availability, and differing data available at remote sites indicating other machines to try, based on where users of the machines being checked were logged in from at the time of the search. Some failures were also due to query requests that could not complete because insufficient information was found in the seed data and/or finger information.

"Duplicate responses" refers to how many responses came back for which we had seen the identical response before (indicating that a user was found who has the same information stored on several machines). This is a measure of wasted effort, although this redundancy also provides added robustness to network and host failures.

There are several immediate conclusions to be drawn from these measurements. First, the elapsed times are quite reasonable for this application, ranging from a few seconds to about 20 seconds for successes, and somewhat longer for failures. Second, even though one might expect to have more successful searches in the daytime (when more people are logged in, and hence more machines will be found that can be used for a second level of finger lookups, the success rates are quite high (around 80%) regardless of when the searches are performed.[8] The use of indirect finger queries is helpful, but not absolutely necessary for the success of the tool, as can be seen by the relatively small number of indirect queries performed.[9] Third, while Internet use is somewhat wasteful, it is within acceptable limits for the time being. (See Section 6 for a discussion of future work on this issue.). Duplicate responses are not a large source of Internet load. These numbers are probably somewhat underreported, since it is possible that responses currently en route across the Internet were not counted because the program was interrupted after a successful response. Similarly, the characters received measurements are probably somewhat underreported.

The final set of measurements to note in Table 1 concern the effects of parallelism. The main advantage of parallelism here is to decrease the response time for the case where a machine being fingered is unavailable, since it takes a fairly long time for the connect request to that machine to time out in the singly threaded case. In the multi-threaded case, the connect timeout is not perceived by the user. Parallelism also reduces the mean response time and, in most situations, the standard deviation. The latter is helpful, because it is annoying to users to have to wait widely varying amounts of time for responses/failures.

The effect of parallelism is not nearly as pronounced as we had originally expected (given that finger queries each take several seconds to complete). This is because the USENET seed data often points to machines that are good candidates for searches (such as large time shared machines on which many users within the domain have accounts). Nonetheless, the response time improvements from multiple threads are worthwhile, and do not cause significantly different amounts of Internet loading than the single thread case.

These measurements were taken on an unloaded Sun 4/110 running SunOS 4.0.1. This machine is connected by an Ethernet-based local area internet to Westnet, an NSF-funded regional network. Westnet is connected by a T1 (1.544 megabit per second) transmission link to the National Center for Atmospheric Research, an NSFNet hub site. While these measurements were done on a fairly powerful machine (a Sun 4), we noticed very little difference in overall tool performance on the less powerful Sun 3/60. Most of the delay comes from waiting for responses from the finger queries, so the difference in speed of the machines initiating the queries is not very important.

5. Related Work

Throughout this paper we have referred to several projects and systems that support some type of resource discovery. We now discuss a few other such systems.

---

trolled network loading, since the tool limits the number of probes it will make on behalf on any one search request.

[8] Similar results were obtained when we ran the measurements at 2 AM MDT.

[9] Slightly more direct and indirect queries were completed at night on average. We believe this is because there is less traffic on the Internet at night, and hence more finger queries were able to complete before the measurement was interrupted.

WHOIS Service

The SRI Network Information Center provides a centralized TCP-based Internet directory facility called the WHOIS service [Harrenstien, Stahl & Feinler 1985]. While the directory provided by this service is helpful, by itself it is not sufficient to handle the Internet white pages problem. First, the database is incomplete, containing only the small fraction of Internet users who have registered with the NIC. Second, the information is often out of date, since people who register often forget to update the NIC when their information changes (e.g., when they change work addresses). Since the prototype tool we have built uses information where it "natively" resides (i.e., on users' workstations), it avoids these problems.

X.500

CCITT is approaching the white pages problem with its X.500 directory service standard [CCITT 1987]. This standard involves a hierarchical collection of servers running at participating sites, each of which maintains directory information about that site. While this is an important first step, we believe X.500's reliance on hierarchical organization will ultimately limit its usefulness. As the service is required to register an increasingly wide variety of resources (beyond just Internet white pages information), searching for resources will become difficult, because doing so requires that users understand how the hierarchy is organized in order to make effective use of it. While browsing operations are supported, browsing through a hierarchy becomes very difficult when the information space becomes large. In contrast, the Networked Resource Discovery Project focuses on techniques that do not restrict the resource space organization to be a simple hierarchy. Another difficulty with standards such as X.500 is that they require that a large number of standard-conformant servers be deployed in order to achieve reasonable network penetration. We avoid this problem by exploiting information sources in a variety of formats and locations. Finally, X.500 utilizes only syntactic techniques for naming and searching for resource information, unlike the semantically-based techniques discussed in this paper.

Profile

Peterson et al. have developed a system called Profile that supports queries over general types of objects, based on their Universal Naming Protocol [Peterson 1988]. Like the Networked Resource Discovery Project, Profile supports a non-hierarchical name space. However, Profile focuses more effort on the structure of query mechanisms for supporting directory services, whereas our work focuses on allowing resource information repositories to enter the network and be accommodated into the resource space organization dynamically [Schwartz 1989]. Furthermore, our work focuses attention on supporting directory mechanisms in the absence of global cooperation. Finally, while Profile utilizes syntactic techniques in searching for resource information, we use a combination of syntactically and semantically-based techniques.

HNS

The approach taken in this paper is related to that used by one of the authors in a previous research project, where a Heterogeneous Name Service (HNS) was built to support naming in a heterogeneous computing environment. The service used individual procedures called *Naming Semantics Managers*, each of which understood the semantics of naming for a particular class of naming lookups and a particular name service [Schwartz, Zahorjan & Notkin 1987]. In the current project, rather than providing a name service (which allows users to use high-level names that map to system-specific low-level identifiers in accessing resources), we consider the more basic problem of discovering the high-level names in the first place.

6. Conclusions

In this paper we have reported experience with the technique of exploiting semantics to provide a particular resource discovery facility, namely, an Internet white pages directory. We have found this technique to be quite effective. By building an understanding of the semantics of resource information into the algorithms that supports searches, we were able to make use of a large class of existing network accessible information, providing a usefully large scope for the directory without the type of global cooperation that many directory services require.

Because USENET and finger information were not originally intended for use in providing an Internet white pages facility, they do not provide a perfect base on top of which to build such a facility. In particular, our tool cannot provide answers to all queries, and is somewhat wasteful of network resources. Nonetheless, the tool

provides an immediately useful function, without waiting for agreement to be reached on a global service interface, or for servers to be built and deployed. Furthermore, when global resource discovery services such as X.500 become available, our strategy will be able to make use of the information they provide. In this sense, our technique provides superior information infrastructure penetration than any system based on a single standard protocol.

There are several possibilities for future work on this tool. The most important involves making more efficient use of Internet bandwidth, by using a more intelligent technique to manage the seed data. As it currently stands, the seed data coverage is becoming too large and redundant. Presently, it takes up 1.27 megabytes of space, and contains over 12000 entries (where an entry corresponds to a user who at one time posted a USENET message). While having some redundancy in the data is important for availability's sake, our measurements show that typical queries will yield 20-30 machines at a site that can be fingered. We believe this wastes more network bandwidth and remote CPU cycles than is necessary. Our plan now is to revise the tool to keep track of hosts at each institution that get good responses to most queries, and then have the tool finger those machines first for future queries, before initiating parallel searches on the other machines found in the seed database. We expect that this will yield results quickly in most cases, and substantially lower the average network load generated by the tool.

We are also considering extending the tool to use various other sources of information (such as the WHOIS service [Harrenstien, Stahl & Feinler 1985], the Sun Yellow Pages [Weiss 1985], and the CSNet name service [Solomon, Landweber & Neuhengen 1982]). However, since the tool already works quite well, doing so is not a high priority at this time.

We also plan to distribute the tool to a small number of sites around the Internet, and gather measurements like those shown in Table 1 for real users, to get a more realistic appraisal of the effectiveness of the tool and the techniques it uses.

Since the Networked Resource Discovery Project is investigating means by which users can discover the existence of a variety of different resources (not just Internet white pages information), the natural question to ask at this point is under what circumstances can semantics be exploited to aid searches. While we do not yet have enough experience with this technique to outline any general principles, there are some obvious candidates for using the technique. As an example, one could build semantic understanding into a tool concerned with locating network services, making the tool use information in various existing forms, such as the UNIX "/etc/services" service listing file, and the keywords listed in the UNIX "man page" inverted index. One could also check the "Subject: " lines of various USENET newsgroups, such as comp.unix.questions or comp.sources. Further, one could glean some potentially useful information by monitoring network traffic, e.g., to detect the addresses of service-specific packets passing along one's Ethernet.

Acknowledgements

We would like to express our appreciation for the helpful comments received on this paper from Dirk Grunwald, Roger King, Alan Krantz, Michael Main, Evi Nemeth, Curt Stevens, David Wagner, and David Wood.

7. References

[Budd & Levin 1982]
T. A. Budd and G. M. Levin. A UNIX Bibliographic Database Facility. Tech. Rep. 82-1, Dept. Comput. Sci., Univ. Arizona, Tucson, AZ, 1982.

[CCITT 1987]
CCITT. The Directory - Overview of Concepts, Models and Services. ISO DIS 9594-1, CCITT, Gloucester, England, Nov. 1987. Draft Recommendation X.500. ISO/CCITT directory convergence document #1. Version 7.

[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 1977]
K. Harrenstien. Name/Finger. Req. For Com. 742, SRI Int., Dec. 1977.

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

[Kepecs 1985]
J. Kepecs. Lightweight Processes for UNIX Implementation and Applications. *Proc. USENIX Summer Conf.*, pp. 299-308, June 1985.

[Mockapetris & Dunlap 1988]
P. Mockapetris and K. J. Dunlap. Development of the Domain Name System. *Proc. ACM SIGCOMM Symp.*, pp. 123-133, Stanford, CA, Aug. 1988.

[Morris et al. 1986]
J. H. Morris, M. Satyanarayanan, M. H. Conner, J. H. Howard, D. S. H. Rosenthal and F. D. Smith. Andrew: A Distributed Personal Computing Environment. *Commun. ACM*, 29(3), pp. 184-201, Mar. 1986.

[Nowitz 1979]
D. A. Nowitz. UUCP Implementation Description. In *UNIX Programmer's Manual*, Vol 2, Jan. 1979.

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

[Salton 1986]
G. Salton. Another Look at Automatic Text-Retrieval Systems. *Commun. ACM*, 29(7), pp. 648-656, July 1986.

[Schwartz, Zahorjan & Notkin 1987]
M. F. Schwartz, J. Zahorjan and D. Notkin. A Name Service for Evolving, Heterogeneous Systems. *Proc. 11th ACM Symp. Operating Syst. Prin.*, pp. 52-62, Nov. 1987.

[Schwartz 1988]
M. F. Schwartz. Autonomy vs. Interdependence in the Networked Resource Discovery Project. Position paper, ACM SIGOPS European Workshop, Cambridge, England, Sep. 1988.

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

[Sollins 1989]
K. Sollins. A Plan for Internet Directory Services. Req. For Com. 1107, MIT Lab. for Comput. Sci., July 1989.

[Solomon, Landweber & Neuhengen 1982]
M. Solomon, L. H. Landweber and D. Neuhengen. The CSNET Name Server. *Computer Networks*, 6(3), pp. 161-172, July 1982. Presented at the Seventh Berkeley Workshop on Distributed Data Management and Computer Networks, Lawrence Berkeley Laboratory, Pacific Grove, CA, Feb. 1982.

[Stroustrup & Shopiro 1985]
B. Stroustrup and J. E. Shopiro. A Set of C++ Classes for Co-Routine Style Programming. In *AT&T C++ Translator 2.0 Release Notes*, 1985.

[USENIX Association 1986]
USENIX Association. UNIX Supplementary Documents. 4.3 Berkeley Software Distribution, Nov. 1986.

[Weiss 1985]
P. Weiss. *Yellow Pages Protocol Specification.* Sun Microsystems, Inc., 1985.