# Experience with a Semantically Cognizant Internet White Pages Directory Tool[1]

Michael F. Schwartz

Panagiotis G. Tsirigotis

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

May 1990; Revised August 1990

Keywords: naming, resource discovery,
electronic mail address, interoperability

To appear, *Journal of Internetworking - Research and Experience*

## Abstract

As wide area networking technology and interconnection improve, an increasingly important problem is allowing users to navigate through the vast array of network accessible resources. In this paper we discuss experience with one technique we have developed in this regard, applied to a specific resource class. We have built 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 that user. We estimate that the scope of the directory is upwards of 1,147,000 users in 1,929 administrative domains, yet the tool does not require the type of global cooperation that many existing or proposed directory services require, namely, running special directory servers at many sites around the Internet. We accomplish this by building an understanding of the semantics of this particular resource discovery application into the algorithms that support searches, allowing the tool to make aggressive use of existing sources of relatively unstructured information. Being able to make use of such information is important in heterogeneous, administratively decentralized environments, where global agreement about highly structured information formats is difficult to achieve. At present, the tool utilizes information from USENET news messages, the Domain Naming System, the Simple Mail Transfer Protocol, and the "finger" protocol, as well as a variety of information about the meaning of and relationships between these information sources. 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 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. The project involves several goals and techniques [Schwartz 1989]. Our main concerns are flexible organization, scalability, and lack of global administrative authority. These constraints are difficult to satisfy simultaneously. Traditional directory services (such as the CCITT X.500 standard [CCITT 1988]) rely on hierarchical organization to achieve good scalability. Unfortunately, as a hierarchy is required to register an increasingly wide variety of resources, trying to search for resources becomes difficult, because the organization becomes convoluted and requires users to understand how its components are arranged. We are also concerned with providing mechanisms that allow resource discovery to take place without global administrative agreement over protocols and information formats. We focus on mechanisms that permit information sources to be incorporated incrementally, as they become available.

To date, the techniques we use fall into two categories. For supporting "yellow pages" (or attribute based) resource discovery, we have developed a set of probabilistic protocols that disseminate and search for resource information, augmented by caching mechanisms that remove pointers that are not actively in use. These techniques are designed to optimize the case where a small number of instances of a large class of objects is being sought, which we feel is a common scenario in yellow pages searches [Schwartz 1990].

For supporting "white pages" (or name based) resource discovery, we have developed a technique that we call "exploiting semantics". This technique involves building an understanding of the semantics of a particular application into the algorithms that support searches, to permit resource discovery given simply structured information. This capability is important in heterogeneous, administratively decentralized environments, where global agreement may not have been reached over highly structured information formats. While developing standards is clearly an important part of this process, standards often do not provide interim solutions in the short term. In the long term, existing standards need to evolve, to fit changing technology and user needs. Our approach is to accommodate existing systems, rather than to define a new standard.

In the current paper we discuss applying the technique of exploiting semantics to a single useful application, namely, providing an Internet "white pages" directory facility, to assist users in finding other users' electronic mailboxes, telephone numbers, and postal addresses. We have built a prototype white pages tool based on this technique, and have experimented extensively with it. We have also measured various aspects of the tool's performance, as used by a collection of geographically distributed users.

We chose Internet white pages as a test application 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 Internet directory services on the part of CCITT [CCITT 1988] and the U.S. Federal Research Internet Coordinating Committee [Sollins 1989].

While we find the tool to be quite useful, it has limitations. It only supports searches generated by Internet users looking for other Internet users, where TCP-based "SMTP" [Postel 1982] or "finger" [Harrenstien 1977] packets can pass between the initiating user's machine and a machine at the site where the user being sought works. Some sites disallow one or the other of these protocols for security reasons or privacy concerns. Nonetheless, to our knowledge the tool provides access to information about more Internet users than any other white pages facility.

Part of the inspiration for this tool was an experiment to see what level of functionality could be achieved if typical assumptions about Internet bandwidth limitations are relaxed. Early experience with the tool demonstrated the potential of the techniques we developed, but also indicated that Internet load generated by the tool was more wasteful than necessary [Schwartz & Tsirigotis 1990]. By incorporating more understanding of the application semantics into the tool, we were able to reduce Internet load to the point that it became feasible to distribute the tool for general use by users around the Internet. While the tool still generates more Internet load

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

than special purpose directory service mechanisms, we believe that the bandwidth requirements are reasonable, particularly as wide area networks increase in capacity. In the current paper we report our experiences with evolving the tool, as well as measurements of its more widespread use across the Internet.

# 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 potential 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 in a bibliographic database system.

The fact that these systems treat resource information as simple text without any understanding of the semantics of applications that use 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 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.

In the above examples, resource discovery is difficult essentially because the systems attempt to provide a general purpose facility with very simply structured information. To improve the situation, one could either make use of more highly structured information, or one could focus on a particular application, and build more understanding of that application into the resource discovery mechanism. The former strategy is the approach taken by most systems to date, including database management systems and highly structured services such as X.500. We chose to explore the latter approach because we believe it offers better hope of operating in a heterogeneous, administratively decentralized environment.

Building understanding of the semantics of the resources being sought into the algorithms that support searches can form the basis for an effective resource discovery mechanism, because the system can use information in ways that cannot be used in a more general purpose facility. To illustrate this point, we now briefly describe how the tool we have built works.

We begin by building a database of "seed" data, which describes machines to search for various possible requests. This database is built 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. Search requests use the format "*UserString InstString [InstString ...]*", where *UserString* identifies the user (typically by last name) and one or more *InstString*s identify the institution where the user works. For example, a search could be requested for "schwartz university colorado" or "schwartz boulder". When a search is requested, the seed database is consulted, using the supplied institutional information. Some of the machines found in the seed database plus others determined from the Domain Naming System

---

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

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

[Mockapetris 1987b] can then probed using the Simple Mail Transfer Protocol (SMTP) [Postel 1982] or "finger" protocol [Harrenstien 1977], to see if the user can be found at these machines.

SMTP is an Internet standard protocol that provides (among other things) a network interface to allow mail forwarding information to be retrieved about particular users. Finger is an Internet standard protocol that runs mostly on UNIX and TOPS-20 machines, although there is nothing about the protocol that limits it to these operating systems. 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 finger query can also be made without specifying a user's name, to retrieve a list of the users currently logged into those machines, as well as the machines from which they are remotely connected. Some of the machines in this secondary list can then be fingered, to see if the user being sought has an account on one of those machines.

In contrast to the information retrieval systems discussed earlier in this section, the use of simple keyword-based lookups from the seed database is quite effective, because this database contains a narrow class of information keyed a small number of different ways (primarily by city and organization name). Moreover, we use the information returned from the seed database only as a hint to find places to search for the information being sought.

It would probably not be feasible to use USENET, Domain, SMTP, and finger information in a resource discovery mechanism that was not tailored for use in a specific application, since 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 these simple information sources are used to provide an Internet white pages directory facility. In using this particular combination of information sources, 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 SMTP and finger protocols. 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. The tool also exploits the fact that workstation environments often provide a central machine to process USENET 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. These points contribute substantially to the tool's effectiveness.

# 3. Implementation

In this section we describe various aspects of the implementation of the tool. We begin by describing the basic algorithms used by the tool, and then discuss modifications we made to reduce the Internet load generated by the tool. We then discuss the part of the software that supports remote measurement collection. Finally, we discuss changes to the BIND name service "resolver" routines to make them compatible with lightweight processes, and the structure and size of prototype software.

## 3.1. Basic Algorithm

As mentioned earlier, the basic mechanism used by the 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 SMTP and finger protocols. The initial implementation only used finger. We will now describe that implementation.

A typical seed database entry looks as follows:

```
%H anchor.colorado.edu
%O university of colorado, boulder
```

This USENET data is gathered by a program that runs weekly, and traverses the USENET news spool directory hierarchy, checking all of the files that have been received since the program last ran. Messages containing host names that can be determined not to be on the Internet (such as BITNET and UUCP hosts) are ignored. Also, messages are ignored if they contain certain invalid formats, such as host names that are not fully qualified domain names, and host names containing invalid characters. Messages from moderated newsgroups are also ignored, since the "From" lines in those cases refer to the newsgroup moderator, while the "Organization" lines refer to the person who posted the message. The relevant data from the message headers is then extracted, and an inverted index is generated using the *bib* inverted index system [Budd & Levin 1982], modified to support larger databases, and to use "." as a field delimiter (in addition to blanks and tabs), to allow domain name components to act as additional keys.

We actually used two different database packages for the seed database implementation. For collecting the news articles, we used *ndbm* [USENIX Association 1986a], because it allows for fast lookups and updates. On the machine where the tool runs, we used the *bib* inverted index system, because it allows multiple key lookups, and it supports larger data entries than are supported by *ndbm*.

At search time, the index is consulted to find matches for the keywords describing the organization name and/or city. The basic algorithm proceeds as follows. If fewer than a threshold number of matches (currently 200) are found for the specified keywords, a finger query is invoked for each match, within a Sun lightweight process [Kepecs 1985]. If the number of matches exceeds the threshold, the user is informed that this has happened, and is shown a partial list of the machines matched from the original query, to help in forming a more specific query.

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, under SunOS 4.0 this limit is 64, including 3 for standard input, standard output, and standard error. During searches, the tool maintains a queue of hosts to finger, accessed from within a monitor by each of the lightweight processes. 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 earlier, a finger query is also made on each host to request 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 to initiate further finger searches. Only one such level of search indirection is used. A check is also made to ensure that no host is queried more than once (which could happen, for example, if the results from a two level deep finger query list the same machine more than once).

## 3.2. Modifications to Reduce Internet Load

The basic algorithm described above provides the basis for a tool with usefully large search scope. However, it makes somewhat inefficient use of Internet bandwidth, because it searches many hosts for a particular user, based on the large number of machines that can be found in the seed database for typical search requests. These extra searches are not useful in most cases. Often, a user has accounts on many machines at an institution, and finding the user on any of the machines is sufficient.

One possibility for reducing Internet load would have been to perform the searches sequentially, since then if a correct response were received, the user could interrupt the tool without generating any more Internet load. We did not want to do this, partly because it does not not take advantage of the available parallelism. More importantly, it would introduce long delays into the process in cases of machine or network failure. The parallel search mechanism avoids these delays by accepting the responses of any probe without waiting for other probes to time out.

Instead, we introduced a number of limitations on the amount of searching the tool would perform on behalf of any search, as well as a number of semantically based load reduction techniques. The most significant reduction was achieved by interactively limiting the number of administrative domains (such as "colorado.edu") to be searched. If more than three are found in the seed database corresponding to the keys specified by the user, the user is asked to select up to three domains to search. Often, this *search scoping* reduces the amount of activity by an order of magnitude (e.g., from 20 domains to 2 domains). In addition, search scoping achieves the added benefit that the user need not use carefully chosen keys to limit Internet load. For example, instead of searching for "schwartz university colorado boulder", the user can simply search for "schwartz boulder", and then select among the somewhat larger collection of matching domains. By specifying fewer keys, the user is also more likely to match at least some data in the seed database. In addition to limiting the number of administrative domains the tool will search on behalf on any request, we also limit the number of machines that will be searched to 50.

The next most important Internet load reduction comes from exploiting the semantics of Internet naming. In particular, the Domain Naming System structures machine names hierarchically, with the first component being the machine name, and the remaining components being the institutional Domain name. For instance, "latour.colorado.edu" refers to the machine "latour" at the domain "colorado.edu". We observed that machines running Domain name servers are often good candidates for searches, because they often are run on administration-oriented machines containing mail forwarding information or accounts for many users at the site. Therefore, after search scoping is complete the tool attempts to locate two domain name servers that serve each of the domains isolated during search scoping, and request from them mail forwarding information about the person being sought, using SMTP "VRFY" commands [Postel 1982]. If the VRFY commands are successful, the tool displays the machine names returned from the SMTP queries, which will usually include the person's "home" machine. The tool then attempts to finger the person at these machines, to provide more information. If those finger requests fail, the tool tries to finger the name server machines.

We search two name servers rather than all servers for a domain as a compromise between higher replication/availability and reducing Internet load. Note also that the use of SMTP in this phase of operation causes searches to take slightly longer in some cases, because they involve an extra sequential step. However, this step typically directs searches to more appropriate machines, reducing Internet load and retrieving more up-to-date information. Moreover, SMTP improves the tool's scope, because people can be found in administrative domains that do not permit interdomain finger packets (such as many corporate sites).

Figure 1 illustrates the flow of data and events in the case where the Domain Naming System, SMTP, and finger searches described above are successful. The example shown is the optimal case where the Domain Naming System query returns the names of authoritative name server machines, and the SMTP servers on those machines return mail forwarding information that points to the user's home machine, which responds with current data when fingered. This figure only shows a single thread of execution, and does not show search scoping (which would force the user to select at most 3 of the 13 domains that match "boulder" using the version of the seed database at the time of this writing). It also does not illustrate the various fall-back procedures, such as fingering the name server machines, and fingering machines found from previous finger probes.

If the algorithm described above is successful, no more finger activity is generated. Otherwise, the tool falls back to the original algorithm of searching multiple hosts within a domain concurrently, with 3 additional restrictions. First, we modified the tool so that once *some* match has been found, no more finger searches are initiated until the currently executing searches complete, after which point the user is asked whether to continue the search. This step reduces the number of times probes are issued that will likely return duplicate information from multiple hosts. By pausing between search initiations in this fashion, we reduce Internet load further. Second, we reduced the number of concurrent threads from the original 30 to 10. We found that this reduction did not make response time noticeably worse (since parallelism mostly masks network failures, as opposed to carrying on slow tasks concurrently), yet many fewer fingers were initiated between user prompts. Finally, we modified the tool to ask the user whether to continue between the various search phases (domain search, direct host search, and search of hosts found during direct search), to give the user further opportunity to minimize wasted Internet load.

We utilize a simple heuristic to determine the domain associated with a particular host name. The basic idea is to strip off the machine name component of the host name. That does not always work, however, because some names are both domains and host names. For example, "cs.washington.edu" is both a domain and a host name. To handle this, we used a simple heuristic, where the first component is not stripped for two component long names (such as "sun.com"), since stripping the first component off of such names would never make sense. This heuristic works correctly for most cases. Using a heuristic that handles most cases simply is appropriate

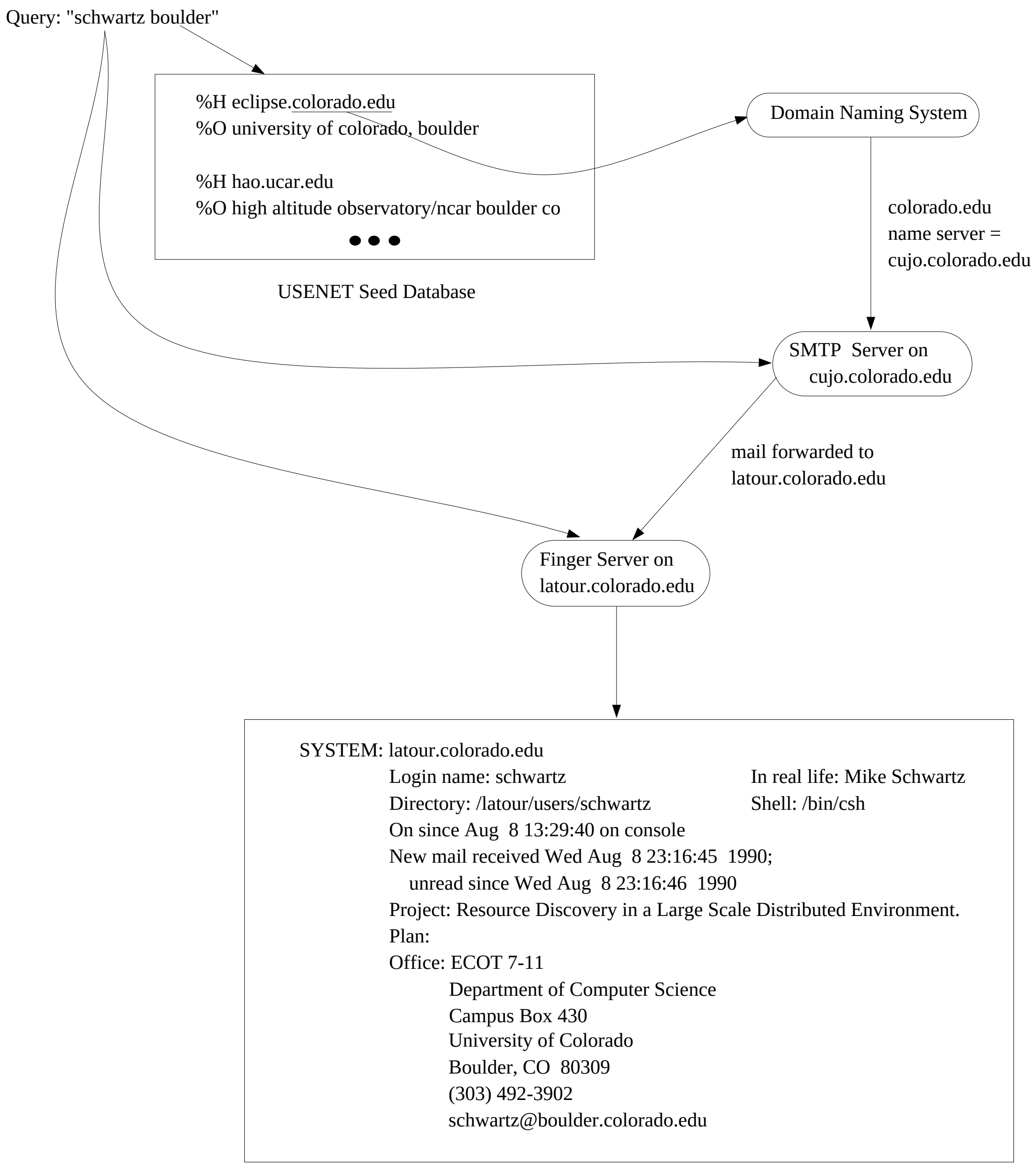


**Figure 1: Single Thread of Optimal Case of Load Reduction Search Algorithm**

because extracting the domain helps reduce Internet load but is not necessary for correctness.

## 3.3. Remote Measurement Collection

The white pages tool contains code to collect measurements and transmit them at the end of each search session in a single datagram packet to a statistics logging server on a machine at the University of Colorado. The measurements collected mostly concern Internet load, such as the number of name lookups, fingers, and bytes received. Other measurements indicate the time of day and an Internet-wide unique identification of who invoked the tool. Having such an identification allows us to determine the tool use frequency on a per-individual basis. These measurements allow us to estimate the frequency with which the tool would be used in a large scale deployment. Measurements will no longer be transmitted after December 31, 1990.

Privacy concerns that could be raised by the collection of these measurements should be allayed by the fact that the user's name, host name, name of the person being searched for, and search keys are not collected. Moreover, measurement collection can be disabled when the tool is compiled.

## 3.4. Changes to BIND Resolver For Light Weight Process Compatibility

To support concurrent searches, it was necessary to modify the BIND *resolver* routines, to make them reentrant. The resolver is the client-resident library code that communicates with a Domain name server to initiate the name lookups [Mockapetris & Dunlap 1988]. Making these changes involved modifying 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 1986b] transformations in the "makefile".

## 3.5. Structure and Size of Prototype

There are several parts of the software that comprises this prototype, all of which are written in C. In our modifications to make the BIND name service resolver reentrant, the original code (BIND version 4.7.3) was about 2,200 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. We also added a new routine to lookup a specified name and return information on its Internet address if it has one, and any authoritative name servers for it, if it has any. This interface was needed as part of the modifications to reduce Internet load, and contains approximately 330 lines of code, derived from software developed at the University of Washington. We later upgraded to BIND version 4.8.1, since it was a more stable version, and because a version of that software was available with a copyright that allowed the code to be redistributed.

The code that collects the USENET seed data is about 1,300 lines long. The code to handle inverting the seed data and looking up keywords in the inverted index is about 900 lines long, and was taken, with few changes, directly from the *bib* inverted index system [Budd & Levin 1982].

The code to do the parallel finger lookups (including instrumentation for the measurements described in Section 4) is about 3,400 lines long. The code to provide the top level functionality that brings the search mechanism and user interface together is about 720 lines long. The code to collect incoming statistics packets is about 350 lines long.

# 4. Measurements

In this section we give measurements concerning the effectiveness and Internet load generated by the tool, for both the original implementation and the version containing the Internet load reduction modifications. We then estimate the scope of the directory, and discuss the evolution of the seed database. Finally, we estimate the load on the Internet if the tool were to receive widespread use.

## 4.1. Basic Tool Effectiveness and Internet Load

To help assess the effectiveness of the techniques discussed here, we constructed a tool usage session that consisted of a series of searches to determine how well the tool works from a user's perspective, and how much Internet load the tool generates. We later explore tool usage measurements collected from real users around the Internet. The reason for exploring a synthetic usage session first is to show how the tool performs under controlled circumstances. In particular, it is not possible to determine whether the tool actually found the users that were being sought from the information collected from real usage measurements, because a search could match the wrong user.

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. We did not select searches for people known to be reachable by the tool. Rather, we chose a people based only on the knowledge that their sites resided on the Internet. 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 people
- 25 had 200-1,000 people
- 7 had 1,000+ people

During these tests we collected average data about the success rate, elapsed times, number of name server lookups, number of connect requests, number of finger probes for both direct fingers and fingers at a machines found via previous fingers, number of characters received, number of duplicate responses, and the effect of parallelism in queries. We ran the measurements once at night and once during the day, to assess the effect of having different numbers of users logged in, 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. The results are shown in Table 1.

| | Mid-Morning (Wed. 10 AM MDT) 30 Threads 2 Levels Deep | Evening (Wed. 7 PM MDT) 30 Threads 2 Levels Deep | Evening (Wed. 7 PM MDT) 1 Thread 2 Levels Deep | Morning (Thurs. 10 AM MDT) 30 Threads 1 Level Deep |
|---|---|---|---|---|
| Successful searches | 32 | 33 | 33 | 29 |
| Elapsed time, succ. [sec.] | 8.15 | 9.88 | 11.13 | 8.45 |
| Elapsed time, fail. [sec.] | 24.57 | 30.71 | 37.86 | 18.64 |
| Name Lookups, succ. | 15.85 | 17.79 | 18.53 | 16.45 |
| Name Lookups, fail. | 10.29 | 12.14 | 6.29 | 8.55 |
| Connects, succ. | 18.85 | 22.61 | 23.66 | 19.35 |
| Connects, fail. | 8.86 | 10.57 | 6.14 | 9.18 |
| Depth 0 Fingers, succ. | 17.49 | 20.03 | 19.94 | 18.41 |
| Depth 1 Fingers, succ. | 0.52 | 1.42 | 2.72 | -- |
| Depth 0 Fingers, fail. | 4.14 | 9.86 | 2.29 | 8.73 |
| Depth 1 Fingers, fail. | 1.71 | 0.43 | 3.43 | -- |
| Chars. Rcvd., succ. | 2,725.97 | 9,745.79 | 8,773.47 | 5,015.79 |
| Chars. Rcvd., fail. | 6,570.43 | 1,687.71 | 3,840.57 | 4,650.91 |
| Dupl. Resp., succ. | 0.70 | 0.85 | 1.00 | 0.55 |

**Table 1: Original Prototype Average Search Measurements**

These measurements (and all others in this section) were taken on an unloaded Sun 4/110 running SunOS 4.0.1, with 8 megabytes of primary memory and a local disk. This machine is connected by a 10 megabit/second 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 the Sun 4 is a fairly powerful machine, 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.

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. Of course, some users might let the tool run "in the background" for a long period of time without watching it run. This will not cause uncontrolled Internet load, since the tool limits the number of probes it will make on behalf on any one search request.

We also interrupted the tool when a query took a long 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 success is obtained.

Failures happened for a variety of reasons. Primarily they were because of name server lookup failures and lapses in Internet availability. Some failures were also due to search requests that could not complete because insufficient information was found in the seed data or finger information.

"Chars. Rcvd." refers to the number of characters received back from requests. Unlike the other measures, this measurement is probably somewhat under-reported in the table, since it is possible that responses currently en route across the Internet were not counted because the program was interrupted after a successful response.

"Duplicate responses" refers to how many responses came back for which the identical response had been seen 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, duplicate responses are not a large source of Internet load. Third, the tool achieved quite good success rates (approximately 80%), given that the people being sought were known only to be on the Internet, but not necessarily reachable by the tool. Finally, 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 do not vary much by time of day. It turns out that the USENET seed database provides such good coverage that indirect probes are not terribly important. As can be seen from the table, probes at depth greater than level 0 make a small but noticeable difference in the success rates, and also have little pronounced effect on any of the other measures.

Another set of measurements to note in Table 1 concern the effects of parallelism. The effect of parallelism on elapsed time was not as pronounced as we had originally expected, given that finger queries each take several seconds to complete. The main advantage of parallelism for this application turns out to be decreasing the response time for the case where a machine being probed is unavailable, since it takes a fairly long time for connection attempts to time out in the singly threaded case. In the multi-threaded case, the connection timeout is not perceived by the user. Parallelism also reduces the mean response time and, in most situations, the response time variance. The latter is helpful, because it is annoying to users to have to wait widely varying amounts of time for responses/failures.

## 4.2. Modifications to Reduce Internet Load

As mentioned earlier, we augmented the semantic understanding of the tool to reduce its Internet load. Table 2 gives measurements corresponding to the mid-morning, 30 thread, 2 level deep case from Table 1, with the load reduction modifications in place. For a fair comparison, it should be noted that the original implementation became much more expensive as the seed database grew in size, because that implementation fingered all matching machines from the seed database. In contrast, most of the machines are not fingered with the Internet load reduction mechanisms in place. Therefore, Table 1 shows the expense of the original implementation with a large seed database, and compares these costs with the costs of the Internet load reduction implementation on the same database.

The reduction in search success occurred because the tool searches fewer hosts. Also, the time for successful searches increased because the load reduction modifications introduce an extra sequential step (contacting SMTP servers), reduce the amount of search concurrency, and ask the user whether to continue at various stages. However, these changes are a reasonable price to pay, because the load measures reduced significantly, except for a large percentage (but small magnitude) increase in depth 0 fingers during unsuccessful searches. Note that the most significant costs (name lookups and fingers) reduced by an order of magnitude for successful searches.

In collecting these measurements, we noticed that in most cases the Internet load reduction modifications caused the standard deviations of the various measures to increase. This observation reflects the fact that the new algorithm has more possible variation in how searches proceed. Our overall conclusions from these

| | Original Implementation, Large Seed Database | Internet Load Reduction Implementation | % Change |
|---|---|---|---|
| Successful searches | 37 | 31 | -16.22 |
| Elapsed time, succ. [sec.] | 11.11 | 20.25 | +82.27 |
| Elapsed time, fail. [sec.] | 15.75 | 30.25 | +92.06 |
| Name Lookups, succ. | 35.44 | 4.59 | -86.94 |
| Name Lookups, fail. | 28.50 | 11.25 | -60.53 |
| Connects, succ. | 42.61 | 4.59 | -89.23 |
| Connects, fail. | 18.75 | 9.75 | -48.00 |
| NameSvr Fingers, succ. | -- | 0.44 | -- |
| Depth 0 Fingers, succ. | 30.03 | 2.50 | -91.67 |
| Depth 1 Fingers, succ. | 0.18 | 0.00 | -100.00 |
| NameSvr Fingers, fail. | -- | 0.38 | -- |
| Depth 0 Fingers, fail. | 0.25 | 5.88 | +2252.00 |
| Depth 1 Fingers, fail. | 0.00 | 0.00 | 0.00 |
| Chars. Rcvd., succ. | 5591.61 | 691.97 | -87.62 |
| Chars. Rcvd., fail. | 11.00 | 673.50 | -6022.73 |
| Dupl. Resp., succ. | 0.78 | 0.06 | -92.31 |

**Table 2: Internet Load Reduction Average Search Measurements**

measurements are that for most cases the searches are nearly as successful, slightly slower, and cause significantly lower Internet load, albeit with somewhat higher variance.

## 4.3. Scope of Directory

To help ascertain the scope of the directory provided by the tool, we ran a measurement experiment in which we sought to execute the search protocol (Domain lookup, SMTP probe, and finger probe) on a small number of machines in each domain found in the seed database. A success was counted as contacting either an SMTP or a finger server in a domain. Once one success or more than ten failures occurred for a domain, no more probe attempts were made on machines in that domain. The experiment was run over a number of sessions at varying times of the day. For each session, the measurement program read the log output of the previous session, and tried contacting machines that either had not been tried before (because they were newly added to the growing seed database), or that had failed due to time outs on previous tries, in domains for which fewer than 10 failures had occurred. Sessions were run until the results of a session differed by little from the results of a previous session, indicating that most of the reachable domains had been counted.

The results of this experiment were as follows. At the time of measurement, the seed database contained 21,070 host names in 5,138 apparent domains. The apparent domain for each host was determined according to the simple heuristic for finding domains within host names discussed in the "Modifications to Reduce Internet Load" Section, which does not always correctly determine the domain associated with a host. Of these domains, the program managed to connect successfully to servers in 1,929 domains. Table 3 lists the top-level domain breakdown for the reachable domains.

Most of the failures (5,490 hosts) from this experiment were caused by host names that could not be resolved to Internet addresses. This type of failure primarily indicates hosts that do not have direct Internet connectivity, registering only a "mail exchange" record in the Domain Naming System. These records allow electronic mail to be forwarded to the host through an intermediary host on the Internet. Since the Internet is growing rapidly, the number of domains reachable by the tool will increase accordingly. Failing to resolve a host name to an Internet address could also occur when the Domain Naming System failed. Earlier measurements indicated that such failures occur about 6% of the time [Schwartz & Tsirigotis 1990]. The remainder of the failures were from

| Top-Level Domain Name | Description | Reachable Sub-Domains |
|---|---|---|
| edu | U.S. Educational Institutions | 870 |
| arpa | Old-style ARPANET Node Names | 310 |
| com | Commercial Institutions | 238 |
| gov | U.S. Government Institutions | 80 |
| ca | Canadian Institutions | 80 |
| au | Australian Institutions | 71 |
| mil | U.S. Military Institutions | 68 |
| se | Swedish Institutions | 34 |
| nl | Netherlands Institutions | 23 |
| org | Non-profit Institutions | 22 |
| net | Institutions named by Network Connections | 19 |
| fi | Finnish Institutions | 19 |
| jp | Japanese Institutions | 16 |
| de | German Institutions | 16 |
| no | Norwegian Institutions | 15 |
| fr | French Institutions | 13 |
| dk | Danish Institutions | 11 |
| nz | New Zealand Institutions | 8 |
| it | Italian Institutions | 5 |
| us | U.S. Institutions | 4 |
| uk | British Institutions | 2 |
| mx | Mexican Institutions | 2 |
| ch | Swiss Institutions | 2 |
| pr | Puerto Rican Institutions | 1 |

**Table 3: Breakdown of Top-Level Reachable Domains**

connection timeouts (3,015 hosts) and connection refusals (335 hosts). The latter category means that the host was reached, but that no SMTP or finger servers were running.

One other point deserving comment about this experiment is the fact that the program tried to connect to an average of 3.31 hosts per domain for domains that could not be reached. This proportion represents probing far fewer hosts than the white pages tool itself attempts to contact. Hence, there might be cases where a domain could not be reached by the measurement program that would have been reached by the white pages tool itself.

To turn the above domain reachability figures into an estimate of the number of users reachable by the tool, we next ran a measurement experiment to determine the average number of hosts per administrative domain. To do this, we selected 105 domains at random from a list of domains in the Internet obtained from the SRI Network Information Center. We then constructed a script that invoked the BIND *nslookup* program, causing a "zone transfer" of all host information from the name servers at each of these domains. 30 domains did not respond, either because of Internet problems or because of security restrictions on transfering such information. Of the remaining domains, the average number of hosts found was 119. Finally, we can multiply together the number of reachable domains, the average number of hosts per domain, and a conservative estimate of 5 users per host to arrive at an estimated scope of approximately 1,147,000 users currently reachable by the tool.

Two conclusions can be drawn from these measurements. First, the tool has a usefully large scope. It can locate information about users in 1,929 of the 5,138 apparent domains found in the seed database. This amounts to approximately 40% of the estimated 4,800 currently existing administrative domains in the Internet [Lottor 1990].[5] Second, name service and Internet availability are good enough that this technique of searching for

[5] The seed database contains more apparent domains than the estimated total Internet domains because of the simple heuristic we used for finding domains within host names, and because the estimated total was based on Domain Naming System data retrieval transfers that in

information is likely to work much of the time. We expect that availability will improve as the Internet becomes more mature. Moreover, since any query might be satisfied by fingering several different hosts, one host's being down may not be critical to success in using this mechanism.

## 4.4. Seed Database Evolution

It is interesting to note that the coverage from the seed database used in Table 1 was obtained after collecting USENET data for only five weeks. Clearly, monitoring USENET transmissions for only a short period of time yields a very effective set of seed data, both in terms of Internet coverage, and in terms of alternative keywords. The growth rate of this input data is charted in Table 4. In this table, "Records" refers to the number of distinct organization name/host name entries.

| Date | File Size [bytes] | Records |
|---|---|---|
| Sep. 8, 1989 | 842,109 | 10,001 |
| Oct. 6, 1989 | 1,066,318 | 12,627 |
| Nov. 3, 1989 | 1,372,821 | 16,233 |
| Dec. 1, 1989 | 1,626,255 | 19,205 |
| Feb. 16, 1990 | 252,907 | 4,404 |
| Mar. 10, 1990 | 649,878 | 11,561 |
| Apr. 20, 1990 | 781,629 | 13,925 |
| May. 18, 1990 | 587,364 | 10,207 |
| Jun. 15, 1990 | 665,679 | 11,587 |
| Jul. 13, 1990 | 739,325 | 12,871 |
| Aug. 10, 1990 | 1,056,462 | 21,193 |

**Table 4: Growth of Seed Database**

After December 1 we temporarily discontinued collecting seed data, because there was too much redundancy in the information (i.e., many hosts per administrative domain). Before introducing the Internet load reduction modifications, this redundancy caused the tool to finger many more sites than necessary. The Internet load reduction modifications removed this problem, because in many cases only a few hosts are searched using these modifications. We therefore began collecting seed data again in February. At this time we also modified the seed database collection mechanism so that it would only maintain records consisting of organization names and host names. In the original version of the tool, login and user names were stored in the seed database (in addition to organization name / address and host name) to allow non-Internet users who post USENET messages to be located by the tool. Storing these extra fields caused a good deal of redundant information to be stored, as records were considered unique based on all four of these fields, rather than just organization name and host name. Yet, we found that in only 4 of the searches discussed in the "Basic Tool Effectiveness and Internet Load" section, the response being sought was available directly in the seed data, without the need to use the finger protocol. Moreover, the information in the seed database should really only be used as a hint to find the more up-to-date information available via the SMTP and finger protocols, since the seed data can become dated over time. Therefore, we modified the seed database collection mechanism to stop collecting the user name and login name fields in the seed database. As can be seen from the table, doing this reduced the the seed database size by approximately 85%.

After April 20 we modified the seed database collection mechanism to avoid collecting records for hosts that are not on the Internet, such as records containing BITNET and UUCP hosts, and to treat the data in a case

some cases could not reach all domains.

insensitive fashion. This modification reduced the seed database size by approximately 25%.

After July 13 we manually augmented the seed database with entries from the SRI International Network Information Center "HOSTS.TXT" file [Feinler et al. 1982] that were not already present in the seed database. The primary reason for this was to capture host names for sites that do not typically post messages on USENET, such as sites on Milnet. Doing this increased the seed database by approximately 40%, but also improved the scope of the tool significantly. Note that USENET postings indicate many hosts that are not registered in HOSTS.TXT, and also provide valuable additional keys in the "Organization" lines.

An immediate observation from Table 4 is that, considering the large scope of white pages searches it supports, the seed database is relatively small. After nearly one year of data collection, the seed database size is slightly over 1 megabyte. The combined size of the seed database and its inverted index is approximately 2.5 megabytes. This amounts to approximately 2.5 bytes of seed data needed per user that can be located by the tool. (See Section 4.4.)

# 4.5. Estimating Total Internet Load Under Widespread Use

We now estimate of the Internet load that would result if people at many sites around the Internet were to use the tool regularly. We are concerned here with Internet packet load, although CPU cycles consumed by finger server processes running on machines at various sites might also be an issue.

In the following discussion, we assume the reader is familiar with the basic organization of the Domain Naming System. If not, we refer the reader to References [Mockapetris 1987a, Mockapetris 1987b].

The parameters of the tool's Internet load behavior are as follows:

$F$ = Number of machines fingered
$N$ = Number of name lookups
$D$ = Maximum depth of Domain name server tree (e.g., colorado.edu. is at level 3)
$C$ = Number of TCP connection requests
$B$ = Size of a single Internet packet small enough for all networks involved
$S$ = Number of name servers running in local organization
$R$ = Total packets generated due to finger responses
$P$ = Total packets generated on the Internet

A search involves $N$ name lookups, $C$ connection requests, and $F$ finger requests. $N$ would equal $2F$ if there were no name lookup failures, and if the user never interrupted the program between the two fingers applied to any machine. Similarly, $C$ cannot be directly calculated from $F$. Also, name lookups use UDP, and hence do not cause connection requests.

In this analysis, we make several simplifying assumptions. In computing the Internet load from the name lookups, we ignore the effects of caching intermediary nodes in the Domain tree (such as the server for the colorado.edu domain), rendering somewhat pessimistic estimates of Internet load. We also consider all domain servers to lie at the maximum depth of the domain tree, again pessimistically expanding the load estimates. We also do not distinguish between local and long-haul message transmissions. This is a pessimistic assumption, because local messages are less expensive. Finally, we ignore the costs of retransmissions due to link failures and down hosts. Doing this yields somewhat overly optimistic results, although as long-haul networks become less lossy with the advent of fiber optics, this assumption will probably become fairly reasonable.

The first name lookup request sent to any particular name server will go from the tool to a local name server, and then to a root server, and then down the $D-1$ servers to the name server that holds the needed information for the remote domain. The total cost for this lookup is $D+1$ message send/receive pairs. From that time on, the local name server will have cached the location of the name server for the remote domain, and thus all subsequent requests will cause just 2 message pairs. Since the LWP compatible version of the BIND resolver chooses among the local name servers at random to distribute the load on them, each local name server will have to acquire the cached information independently. Hence, the total number of message pairs for doing $N$ name lookups using $S$ local name servers is

$$(D+1) * min(N,S) + 2 * max(N-S,0).$$

Since each of these messages fits within one UDP packet, each message pair corresponds to 2 Internet packets.

Each finger request involves one pair of packets for connecting to the remote server, one packet to send the finger request, and $R$ packets to send back the response. Therefore, ignoring retransmissions due to link failures and unavailable hosts, we have:

$$P = 2((D+1)*min(N,S) + 2*max(N-S,0) + C) + F + R$$

$R$ is calculated by summing the number of packets required for each finger response, where an individual finger response of $b$ bytes requires $\left\lceil \frac{b}{B} \right\rceil$ packets.

We built the above computation into the tool, so that each invocation of the tool calculates the estimated number of Internet packets it generated, and transmits the results along with the other measurements described in Section 3.3 to a logging server at the University of Colorado. The results for users other than the authors (whose tests and measurements represent atypical use) are plotted in Figure 2. We used a histogram rather than an exact graph to eliminate the "spikes" that would occur from showing the frequency of individual packet counts. The large number of requests satisfied in the 0-75 packet range correspond primarily to search failures and successes where the person being sought was located using the optimal load reduction algorithm case of a two SMTP probes followed by a finger probe on the person’s home machine. The mean costs are similar to the costs of a small file transfer or a short remote login session, except that the packets are generated in closer succession, because of the use of parallel searches.

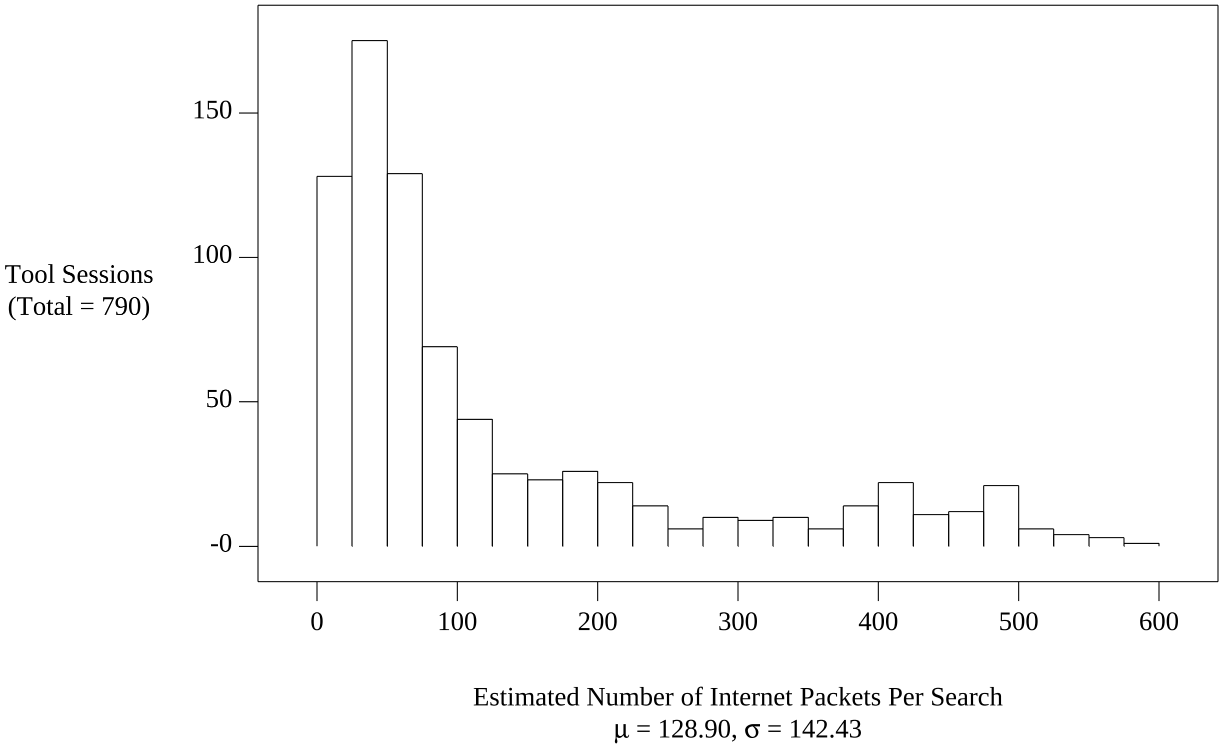


**Figure 2: Estimated Packet Cost Histogram from User Measurements**

In addition to collecting packet costs for tool usage sessions, we also recorded the times when each user invoked the tool. Figure 3 plots the distribution of usage times. During 69 days of measured use, 70 users on 54 machines in 19 administrative domains used the tool 790 times (excluding the authors). As can be seen, users typically experimented with the tool a number of times on the first day of use, and then settled into a period of much less frequent usage. After day 1, the mean number of tool uses per user per day was 0.061, with a standard deviation of 0.107. This measured mean is markedly less than the projection made by a recent Internet white pages workshop, which estimated that an Internet directory facility would be used for an average of ten searches per user per week [Sollins 1989].

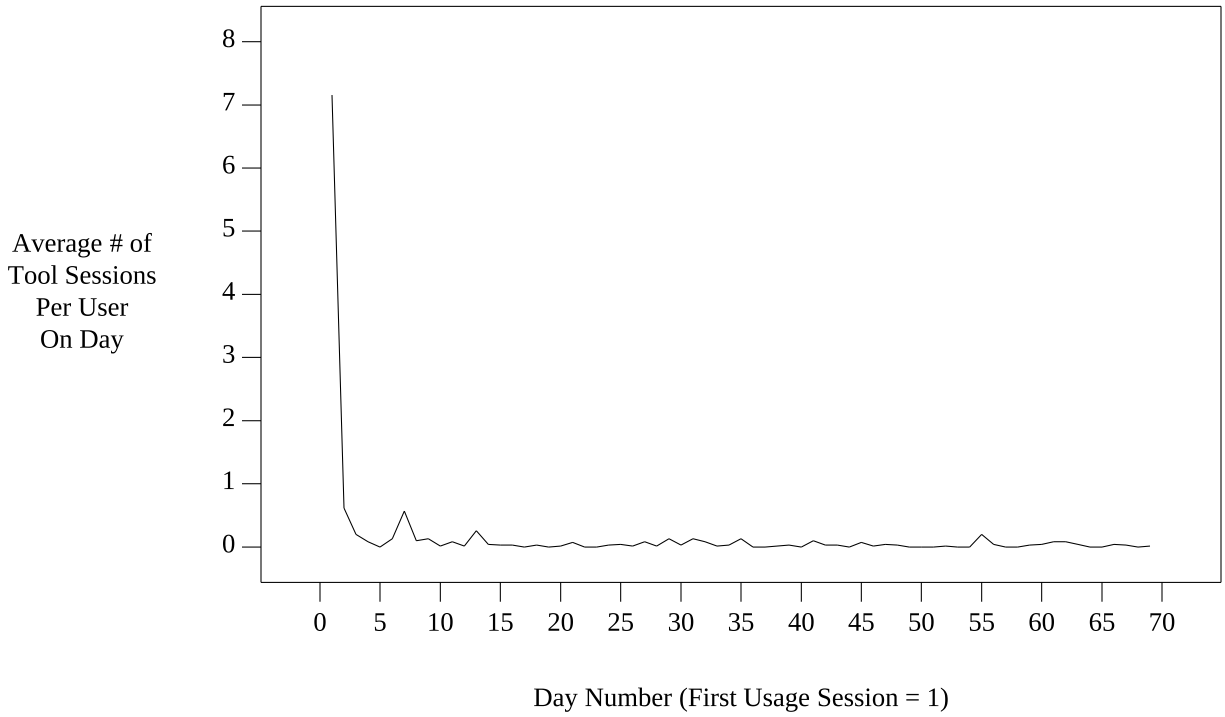


**Figure 3: Tool Usage Time Distribution from User Measurements**

Using the mean values of 0.061 searches per user per day and 128.90 packets per search, we now estimate the total Internet load that would result from widespread use of the tool. Assuming that each of the 1,929 domains reached in the Internet (see Section 4.3) have an average of 100 people who use the tool, this amounts to $4.55 \times 10^8$ additional packets on the Internet per month, ignoring improvements that will be realized as the minimum packet size *B* increases in future networks. As a comparison, all protocols combined used $10^9$ packets per month on the fiber hubbed NSFNet backbone as of the Summer of 1989, and the backbone was running at 10% capacity at that time [NSF Network Service Center 1989]. Note that NSFNet is only one part of the entire Internet. The Internet includes many regional and other types of networks in addition to the NSFNet backbone. We mention these figures only as a basis for comparison, and do not mean to imply that the load on NSFNet would increase by $4.55 \times 10^8$ packets per month.

While this tool is more expensive than a special purpose directory service would be, we believe that the Internet could handle a fairly substantial level of use of this tool. It should also be noted that the estimated number of added Internet packets really represents the cost of adding *directory service* to the Internet, as opposed to the cost of adding our tool. A directory service that generated one quarter as many packets as our tool would still add $1.14 \times 10^8$ packets per month to the Internet. (We compare the load generated by our tool to that generated by other white pages facilities in Section 5.6.)

# 5. Related Work

In this section we compare and contrast several projects and systems related to the tool we have developed. We limit the discussion here to tools that provide service across administrative boundaries. This eliminates from consideration such facilities as MIT's university-wide white pages facility, the University of Illinois' "411" service, and Bellcore's "dq" service. At the end of this section we compare various aspects of each of the compared systems.

## 5.1. 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 of consistency and transfer of authority.

## 5.2. X.500

CCITT is approaching the white pages problem with its X.500 directory service standard, a prototype implementation of which has been the basis for NYSERNet's White Pages Pilot Project [Rose & Schoffstall 1989], now being managed by Performance Systems International. This facility is seen as the eventual replacement for the WHOIS service. X.500 involves a hierarchical collection of servers running at participating sites, each of which maintains directory information about that site. Browsing and searching operations are supported. X.500 also provides a mechanism to "flatten" parts the directory tree, using a client capable of locating organizational domains to search, given only string names. For example, "-org col" could match the University of Colorado and Columbia University if used within the United States. This facility is helpful, but only supports flat institution names below the country level of the tree. In contrast, the tool described in the current paper supports global flat institution names. 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 existing sources of simply structured information 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 the current paper.

## 5.3. 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 tool described in the current paper, 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 supporting resource discovery in the absence of global cooperation. Moreover, Profile utilizes syntactic techniques in searching for resource information.

## 5.4. Knowbot Information Service

Droms has built a tool called a *Knowbot Information Service (KIS)* [Droms 1990]. Like the tool discussed in the current paper, KIS provides an Internet white pages facility by utilizing existing sources of information. However, the focus of KIS is quite different from the focus of our tool. KIS is oriented towards providing a front end to a variety of different information sources, supporting a uniform query and response syntax by translating into and out of the various formats. KIS can access several information sources, including Profile, WHOIS, the CSNet name server [Solomon, Landweber & Neuhengen 1982], finger, MCI Mail, the MIT white pages, and the X.500 Pilot. In contrast, the prototype discussed in the current paper utilizes information from only four types of sources. However, our tool accesses information from a much larger collection of administrative domains, via the SMTP and finger protocols. By accessing white pages information from the sites that naturally maintain that information (rather than from administratively centralized repositories like WHOIS), we avoid difficult problems concerning consistency and transfer of authority. While KIS understands the finger protocol, it only fingers explicitly specified machines.

## 5.5. 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 HNS 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, to provide a naming mechanism that accessed information from its native sources, rather than requiring a "reregistered" global database [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.

## 5.6. Comparison of Internet White Pages Facilities

Table 5 compares the above Internet white pages facilities along various metrics. In constructing this table, we chose metrics to highlight the advantages and disadvantages of each facility, rather than just the advantages of the tool discussed in this paper. "Name Space Structure" considers the structure of names presented to the user. As discussed in section 1, hierarchical name spaces scale well, but tend to be difficult to use for resource discovery. "Information Source Types" refers to the number of different types of information sources, which reflects the degree to which the facility accommodates heterogeneity. "Administrative Domains Holding Information" indicates the degree of administrative decentralization supported by the facility. Note that this does not reflect the number of administrative domains about which information is registered. "Information Timliness Support" indicates the mechanisms that affect how likely information is to be up-to-date. "Individuals Reached" measures the scope of the facility. "Query Structure" measures the support for making meaningful queries. "Structure of Result Information" indicates the ease with which the result information can be used as input to a program. "Response Time Range" measures user-oriented performance. "Internet Load" measures network-oriented performance. "Replicated Information" indicates availability.

The clearest advantage of the Networked Resource Discovery Project's white pages tool is its ability to reach a large amount of highly decentralized information without global administrative cooperation. Also, the fact that users need not be concerned with an artificially imposed hierarchical structure or a specific set of keywords in specifying an organization to search makes the tool quite easy to use. Users can choose any set of keywords of which they are aware concerning the organization's name, geographic location, function, etc., and based on the information that has been gathered by the USENET monitoring mechanism, are reasonably likely to find the person they desire. Also, the fact that information is obtained directly from sources that are needed for daily functioning of users' accounts (rather than requiring updates to a separate information source) makes the tool more likely to provide up-to-date information than the other white pages facilities. In particular, if the tool finds a person, it is because that person's login accounts currently exist, and it can be seen when the user last logged into those accounts. In contrast, the X.500 service, for example, requires users or their systems administrators to keep a secondary source of the information up-to-date.

The most apparent disadvantage of our tool form this table is its Internet load. However, while the tool generates more Internet load than many white pages tools, we believe its load is acceptable, particularly as Internet capacity increases. Another disadvantage of our tool is that it does not structure the result information, making it more difficult to use the information from within a program; rather, it is left in its original textual format. That functionality could be added to the tool without undue difficulty, just as KIS has added the ability to convert from the output format of finger into its standard output format.

The fact that our tool uses a very large, administratively decentralized collection of information sources has both advantages and disadvantages. The clear advantage is that acquired information is quite timely, and does not suffer the type of consistency problems that white pages services based on a separate registration paradigm do. On the other hand, such wide distribution increases the number of different failure modes the tool can experience, because of fluctuations in availability of the Internet and the sites being searched.

The figures in Table 5 were obtained through a combination of our experimental measurements and personal communications. The response time measurement ranges represented various circumstances. For the

| Metric | NRD WP Tool | WHOIS | X.500 Pilot | Profile | KIS |
|---|---|---|---|---|---|
| Sponsoring Organization | Univ. Colorado | SRI NIC | NYSERNet/ PSI | Univ. Arizona | NRI |
| Name Space Structure | Flat user names + flat organization names | Flat user names, no organization names | Hierarchical organization names "flattened" below country level | Flat organization names with search paths | Flat user names + optional information source specifications |
| Information Source Types | 4 | 1 | 1 | 1 | 7 |
| Administrative Domains Holding Information | 1,929 | 1 | 100 | 20 | 106 |
| Information Timliness Support | Information obtained from sources needed for daily work | Updates through central authority | Users or administrators update registered information | Updates through administrators of information sources | Updates through administrators of information sources |
| Individuals Reached | 1,147,000 | 70,000 | 100,000 | 5,000 | 390,000 |
| Query Structure | User name near matches; wide variety of organizational keywords | User name substrings | Regular expressions over typed fields | Regular expressions + matching operators over typed fields; path preferences | Regular expressions over typed fields |
| Structure of Result Information | Text with some format conventions | Uniformly formatted text | Structured fields | Structured fields | Structured fields |
| Response Time Range [seconds] | 3-45 | 8-45 | 14-182 | 1-2 | 10-102 |
| Internet Load Per Query [packets] | 129 | 12 | 40 | 25 | 140 |
| Replicated Information | Usually | No | Yes | No | Yes |

**Table 5: Comparison of Internet White Pages Facilities**

NYSERNet White Pages Pilot, the low end represents a local query directed to a specific existing individual, while the high end represents an "organizationally flattened" (see Section 5.2) query to a non-existent individual. The low end measurement for KIS represents querying a single fast information source, and the high end represents querying a set of 9 different information sources. The main bottleneck is that queries are performed sequentially in the current prototype.

For the Networked Resource Discovery Project white pages tool, the "Internet Load" measurement is a weighted average of the successful and unsuccessful search costs. For the other facilities, this measurement was estimated for average query situations in each of the respective environments.

All name lookups were approximated as costing 3 connections and 3 request/response pairs, for traversing the local Domain server, the (assumed cached) second level server (e.g., "edu"), and the server holding the naming information for the organization in question. The costs of acknowledgement packets and retransmissions due to errors were ignored.

For the WHOIS service, the Internet load value includes the cost of a name lookup, connecting to the server, a query message, and three response packets (for the common case where a response matched several people, and the output filled a single screen). For the X.500 Pilot, the Internet load value includes the cost of name lookups and connections to Directory System Agents at the local, country, and organizational levels, querying and displaying the information available at each of these levels, and finally displaying the information needed for the particular user, filling a single screen with output at each point. For Profile, the Internet load value includes the cost of name lookups and contacting 3 servers (an average case), and then retrieving a single packet response. For KIS, the Internet load value includes the costs of querying the WHOIS service, the CSNet name server, three Profile servers, and an MCI Mail server (the configuration at the time of this writing). Contacting the CSNet name server was estimated to cost the same as contacting the WHOIS service. The MCI Mail server was approximated as costing a single name lookup, a query message, and 2 screens worth of response text. The added cost of transmitting all of the information obtained by the KIS server back across the Internet was not included, since in principle one could simply run a copy of the KIS server locally.

# 6. Conclusions

The ability to make use of a variety of simply structured information is useful for providing resource discovery mechanisms in a heterogeneous, administratively decentralized environment. This capability can be supported by exploiting the semantics of particular applications. As a test application of this technique, we developed and experimented extensively with an Internet white pages directory tool. This tool uses four large classes of existing Internet accessible information, providing a usefully large scope for the directory without the type of global cooperation that many directory services require. This does not imply that no existing infrastructure is needed. Rather, by taking the meaning of the application into consideration we have been able to make use of some primitive existing infrastructure to provide a facility whose functionality is, in a sense, greater than the sum of its parts. Essentially, exploiting semantics provides a means to "glue" together a very large number of administratively decentralized information spaces.

One would expect that using pre-existing information sources would be less effective than building a service and an associated set of information bases designed specifically for the purpose. Nonetheless, to our knowledge our tool provides a more complete, easy to use, and up-to-date white pages directory than any other existing Internet white pages facility. Furthermore, 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.

Building this tool was partly an experiment to see what level of functionality could be achieved if typical assumptions made about Internet bandwidth limitations were relaxed. An initial prototype demonstrated that a directory facility could be built using simply structured information. By incorporating a number of limitations and pieces of application-specific semantics into the tool, we were able to reduce the tool's Internet load to an acceptable level, particularly when compared with the bandwidth requirements of future scenarios involving visualization of graphical output from remote supercomputers [Office of Technology Assessment 1989].

Oue experiences with this tool provide some perspectives on the minimal support needed to provide a directory facility. The main comment we offer in this regard is that a carefully coordinated set of servers, each of which supports highly structured information (as is the case with X.500), is not necessary. It is more important that information sources exist for a sizable proportion of sites that might be searched, and that information be obtained directly from its native sources, rather than from a "reregistered" database. There should also be multiple paths to each source of information, to enhance availability. The tool we have built can tolerate the absence of some of the protocols it uses and still achieve success. This capability enhances its ability to function in an environment where global agreement over supported protocols is difficult to attain.

Information sources should also support a range of easily specified search requests. A shortcoming of the tool described in the current paper regarding this requirement is the fact that SMTP does not support as flexible a user naming scheme for inquiring about mail forwarding information as finger does. Finally, the information sources should support appropriate abstractions of the resources. A shortcoming of the tool regarding this requirement is the fact that finger was conceived prior to the existence of workstation environments. A more appropriate mechanism would support a notion of a user per administrative domain, rather than per machine.

Given the number of different types of resources a user might wish to locate in the general resource discovery problem, a natural question to ask at this point is under what circumstances can semantics be exploited to aid searches. One clear candidate is a tool concerned with locating network services. To build this tool one could use information available in the UNIX "/etc/services" service listing file and the keywords listed in the UNIX "man page" inverted index. One could also glean potentially useful information by monitoring network traffic, e.g., to detect the addresses of service-specific packets passing along an Ethernet.

Another candidate for exploiting application semantics involves supporting resource discovery among "anonymous FTP" public information sites around the Internet. Anonymous FTP provides public access to software, documents, and other information placed in particular file system directories on thousands of computers around the Internet [Postel & Reynolds 1985]. As it currently stands, a large amount of information is available through this mechanism, yet finding which hosts hold information of interest is difficult. Ad-hoc lists of the contents of the various anonymous FTP sites exist, yet such lists are by necessity incomplete. From time to time there are also postings on various USENET bulletin boards announcing the availability of information on different hosts, but this information is short-lived (since old news files are regularly deleted out of news spool directories), difficult to find (because of the high volume of news) and irregular. Based on these observations, we are currently building a prototype that will incorporate an understanding of anonymous FTP and the formats of various anonymous FTP site lists and news postings, and provide a caching mechanism that builds a site list based on this information. In addition, the tool will cache information about sites that have been contacted (such as directory listings and "README" files) to improve the quality of information it maintains on each site. We will also make the information cache available by anonymous FTP, so that other sites may run this tool and exchange cached information, quickly building a wide-area list of discovered information.

An interesting notion that arises from the current prototype and these other pieces of work in progress is the idea that by exploiting semantics it becomes possible to make use of public information gleaned by monitoring network traffic, to provide hints for use in aggressive resource discovery. Clearly, there are privacy considerations raised by this technique. Nonetheless, the technique merits further investigation, since it offers the possibility of very quickly "gluing" together a wide range of large information spaces.

**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, Paul Mockapetris, Evi Nemeth, Curt Stevens, David Wagner, David Wood, and the journal referees. We would also like to thank Ralph Droms, Larry Peterson, Colin Robbins, and Marshall Rose for their assistance in providing some of the information for Table 5, and Mark Lottor, for his Internet size estimates. We would also like to thank the users around the Internet (particularly David Goldstein) for their many suggestions for improving the tool and its documentation, during early trial runs using the tool. Finally, we would like to thank Jan Sanislo for suggesting that we use SMTP to improve searches.

# 7. References

[Budd & Levin 1982]
T. A. Budd and G. M. Levin. A UNIX Bibliographic Database Facility. Technical Report 82-1, Department of Computer Science, University of Arizona, Tucson, Arizona, 1982.

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

[Droms 1990]
R. E. Droms. Access to Heterogeneous Directory Services. Proceedings of the InfoCOM Conference, June 1990.

[Feinler et al. 1982]
E. Feinler, K. Harrenstien, Z. Su and V. White. DoD Internet Host Table Specification. Request For Comments 810, Network Information Center, SRI International, March 1982.

[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, University of California, San Diego, December 1983.

[Harrenstien 1977]
K. Harrenstien. Name/Finger. Request For Comments 742, SRI International, December 1977.

[Harrenstien, Stahl & Feinler 1985]
K. Harrenstien, M. Stahl and E. Feinler. NICName/Whois. Request For Comments 954, October 1985.

[Kepecs 1985]
J. Kepecs. Lightweight Processes for UNIX Implementation and Applications. *Proceedings of the USENIX Summer Conference*, pp. 299-308, Portland, Oregon, June 1985.

[Lottor 1990]
M. Lottor. Personal Communication. Discussion of measured number of domains and machines in the Internet. April 1990.

[Mockapetris 1987a]
P. Mockapetris. Domain Names - Implementation and Specification. Request For Comments 1035, USC Information Sciences Institute, November 1987.

[Mockapetris 1987b]
P. Mockapetris. Domain Names - Concepts and Facilities. Request For Comments 1034, USC Information Sciences Institute, November 1987.

[Mockapetris & Dunlap 1988]
P. Mockapetris and K. J. Dunlap. Development of the Domain Name System. *Proceedings of the ACM SIGCOMM Symposium*, pp. 123-133, Stanford, California, August 1988.

[NSF Network Service Center 1989]
NSF Network Service Center. NSF Network News. July 1989.

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

[Office of Technology Assessment 1989]
Office of Technology Assessment. High Performance Computing and Networking for Science. U.S. Government Printing Office stock number 052-003-01164-6, October 1989. Background paper.

[Peterson 1988]
L. L. Peterson. The Profile Naming Service. *ACM Transactions on Computer Systems*, 6(4), pp. 341-364, November 1988.

[Postel 1982]
J. B. Postel. Simple Mail Transfer Protocol. Request For Comments 821, USC Information Sciences Institute, August 1982.

[Postel & Reynolds 1985]
J. Postel and J. Reynolds. File Transfer Protocol (FTP). Request For Comments 959, USC Information Sciences Institute, October 1985.

[Rose & Schoffstall 1989]
M. T. Rose and M. L. Schoffstall. An Introduction to a NYSERNet White Pages Pilot Project. Technical Report, NYSERNet Inc., December 1989.

[Salton 1986]
G. Salton. Another Look at Automatic Text-Retrieval Systems. *Communications of the 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. *Proceedings of the Eleventh ACM Symposium on Operating Systems Principles*, pp. 52-62, Austin, Texas,

November 1987. Published as Operating Systems Review 21(5).

[Schwartz 1989]
M. F. Schwartz. The Networked Resource Discovery Project. *Proceedings of the IFIP XI World Congress*, pp. 827-832, San Francisco, California, August 1989.

[Schwartz 1990]
M. F. Schwartz. A Scalable, Non-Hierarchical Resource Discovery Mechanism Based on Probabilistic Protocols. Technical Report CU-CS-474-90, Department of Computer Science, University of Colorado, Boulder, Colorado, June 1990. Submitted for publication.

[Schwartz & Tsirigotis 1990]
M. F. Schwartz and P. G. Tsirigotis. Exploiting Semantics to Provide Internet White Pages Without Global Cooperation. Technical Report CU-CS-444-89, Department of Computer Science, University of Colorado, Boulder, Colorado, October 1989; revised February 1990.

[Sollins 1989]
K. Sollins. A Plan for Internet Directory Services. Request For Comments 1107, MIT Laboratory for Computer Science, Cambridge, Massachusetts, 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, California, February 1982.

[USENIX Association 1986a]
USENIX Association. UNIX System Manager's Manual. 4.3 Berkeley Software Distribution, November 1986.

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