Applying an Information Gathering Architecture to Netfind: A White Pages Tool for a Changing and Growing Internet

Michael F. Schwartz, University of Colorado, Boulder (schwartz@cs.colorado.edu)

Calton Pu, Oregon Graduate Institute of Science and Technology, Portland (calton@cse.ogi.edu)

University of Colorado Technical Report

CU-CS-656-93 December, 1993
Revised July 1994

To appear, *IEEE/ACM Transactions on Networking*

The Internet is quickly becoming an indispensable means of communication and collaboration, based on applications such as electronic mail, remote information retrieval, and multimedia conferencing. A fundamental problem for such applications is supporting resource discovery in a fashion that keeps pace with the Internet's exponential growth in size and diversity. Netfind is a scalable tool that locates current electronic mail addresses and other information about Internet users. Since the time we first deployed Netfind in 1990, it has evolved considerably, making use of more types of information sources, as well as more sophisticated mechanisms to gather and cross-correlate information. In this paper we describe these techniques, and present a general framework for gathering and harnessing widely distributed information in a diverse and growing internet environment. At present Netfind gathers information from 17 different sources, providing a particularly thorough demonstration of an information gathering architecture.

1. Introduction

Since its inception in the late 1960's as the ARPANET, the Internet has grown substantially in size, speed, and diversity. At present, it interconnects 20,000 networks worldwide, supporting an increasingly diverse means of communication and collaboration among educational, commercial, government, military, and other types of users. Doubling in size every year [Lottor 1992, Schwartz & Quarterman 1993], the Internet is one the fastest growing human-constructed phenomena in history.

The explosive growth of the Internet has brought with it corresponding growth in the amount of information available to Internet users. An enormous amount of information is already publically accessible, in the form of software, documents, sounds, images, and other file system data; library catalog and user directory data; weather, geography, telemetry, and other physical science data; and many other types of information. This volume and diversity of information creates a great need for directory and resource discovery support [Schwartz et al. 1992].

Unfortunately, traditional software engineering does not tell us how to build software that can adapt to the explosive growth of the Internet. Systems that can scale up by one or two orders of magnitude have generally been considered very successful [Schroeder, Birrell & Needham 1984]. However, the Internet's host count has grown by four orders of magnitude in the past 10 years. Such growth presents difficult problems for Internet tools, as the volume of information, load on popular servers, and diversity of information all increase rapidly [Bowman et al. 1994b].

In this paper we describe Netfind, a tool that locates current electronic mail addresses and other information about Internet users among a rapidly growing, changing, heterogeneous base of information. Rather than relying on manual administration to update a directory of Internet users, Netfind gathers and cross-correlates information from many different sources. Because of this approach, Netfind can locate current information about over 12 million users worldwide, outstripping all other user directories, including X.500 [CCITT/ISO 1988]. Netfind's ability to keep pace with the Internet's explosive growth has made it quite popular. It is in constant use, supporting approximately 25 thousand queries per day, from users in 8,390 sites in 60 countries.

Netfind's information gathering algorithms have evolved considerably since our original paper about the system [Schwartz & Tsirigotis 1991], to meet the needs of the Internet's rapid growth and increasing diversity. In the current paper we describe how Netfind works and why it is scalable. We focus particularly on the techniques that allow Netfind to adapt to a changing, heterogeneous collection of information sources. We also propose a framework for more general resource discovery solutions based on our experiences to date.

We introduce the Internet white pages problem in Section 2. In Section 3 we present a framework for information gathering in a large, heterogeneous environment. We discuss the information gathering techniques Netfind uses in Section 4. We discuss related work in Section 5. Finally, in Section 6 we offer our conclusions and discuss future work.

2. The Internet White Pages Problem

The basic problem we address in this paper is how to locate an Internet user, given the user's name and the name or rough geographical location where the person works. This is called a *white pages* problem, borrowing a term from telephone directories to indicate the operation of searching for objects by name. This stands in contrast to a *yellow pages* problem, where one tries to locate a class of objects using some type of attribute-based description.

As an example of the problem, Figure 1 illustrates the use of Netfind to locate the person ''Schwartz'' at the University of Colorado – Boulder. After the user specifies the name and site information, Netfind asks the user to select a few domains[1] to search, among those that matched the specified site description keys. After this *search targetting* process, Netfind attempts to locate the requested person, producing the output illustrated in

[1] Throughout this paper we use the term ''administrative domains'' (or simply ''domains'' or ''sites'') to mean organizational groupings inferred by the Domain Naming System (DNS) tree [Mockapetris 1987]. For example, ''cs.colorado.edu'' is an administrative domain, because there are hosts one level below it in the tree (such as ''latour.cs.colorado.edu''). Hosts are identified as nodes having IP addresses registered in the DNS.

Figure 2. We discuss the techniques used to obtain this information in Section 4.

```
Enter person and keys (blank to exit) → schwartz colorado boulder university
Please select at most 3 of the following domains to search:
     0. acc.colorado.edu (academic computing center, university of colorado, boulder)
     1. astro.colorado.edu (astronomy department, university of colorado, boulder)
     2. ba.colorado.edu (business school, university of colorado, boulder)
     3. cats.colorado.edu (computing and technology services, university of colorado, boulder)
     4. cc.colorado.edu (computer center, university of colorado, boulder)
     5. civil.colorado.edu (civil engineering department, university of colorado, boulder)
     6. cs.colorado.edu (computer science department, university of colorado, boulder)
     7. earth.colorado.edu (earth science department, university of colorado, boulder)
     8. energy.colorado.edu (electrical engineering department, university of colorado, boulder)
     9. geology.colorado.edu (geology department, university of colorado, boulder)
     10. jila.colorado.edu (joint institute for laboratory astrophysics, university of colorado, boulder)
     11. library.colorado.edu (library, university of colorado, boulder)
     12. ocean.colorado.edu (oceanography department, university of colorado, boulder)
     13. psych.colorado.edu (psychology department, university of colorado, boulder)
     14. telecomm.colorado.edu (telecommunications department, university of colorado, boulder)
Enter selection (e.g., 2 0 1) → 6 3
```

Figure 1: Netfind Interactions During Search Targetting

```
MAIL IS FORWARDED TO schwartz@latour.cs.colorado.edu
NOTE: this is a domain mail forwarding arrangement, so mail should be addressed to "schwartz@cs.colorado.edu" rather
      than "schwartz@latour.cs.colorado.edu".

SYSTEM: latour.cs.colorado.edu
        Login name: schwartz                In real life: Mike Schwartz
        Directory: /home/latour/schwartz    Shell: /bin/tcsh
        On since Nov 13 10:31:10 on console 3 hours Idle Time
        No unread mail
        Project: Resource Discovery and Internet Information Systems
        Plan:
        Department of Computer Science, University of Colorado
        Boulder, CO  80309-0430
        Voice: +1 303 492 3902; Email: schwartz@cs.colorado.edu

        Login name: schwartz                In real life: Mike Schwartz
        Directory: /home/latour/schwartz    Shell: /bin/tcsh
        On since Nov 18 21:14:35 on ttyp5 from eve.cs.colorado.

SUMMARY:
- "schwartz" is currently logged in from eve.cs.colorado.edu, since Jul 18 21:14:35.
- The most promising email address for "schwartz" based on the above search is schwartz@cs.colorado.edu.

Continue the search ([n]/y) ? → n
```

Figure 2: Example Netfind Search Results for Search shown in Figure 1

Providing an Internet user location service is difficult because users expect ubiquitous coverage. If it cannot answer users' specific queries some reasonable proportion of the time, a user location service is perceived as offering little value to users. In turn, it is difficult to provide ubiquitous coverage in the Internet because of its exponential growth rate and the diversity of ways that sites can connect to and use the Internet [Carl-Mitchell &

Quarterman 1994, Schwartz & Quarterman 1993]. Some sites connect to the Internet, register with a central naming registry, publicize their network presence, and deploy network services that allow remote users to search for people there. Other sites do no more than enlist the services of a third-party mail forwarder, providing little visibility and essentially no mechanisms for locating users at their site. Many other variations are also possible.

To locate users, traditional directory services (such as X.500 [CCITT/ISO 1988] and WHOIS [Harrenstien, Stahl & Feinler 1985]) use a manual administrative process to build databases of users. There are two problems with this approach. First, it is difficult to populate a database with all users in an environment as decentralized as the Internet. Unlike the situation with telephone companies, Internet users can be added without informing a central directory maintenance organization. Even sites that do register with such an organization have no continuing impetus to provide individual updates. Second, it is difficult to keep a manually created database up-to-date as users move around.

As examples of these problems, consider the WHOIS and X.500 services. Originally, there was a single centralized WHOIS server run by the SRI Network Information Center (NIC). As the Internet grew it became infeasible to register all users with one central server, and many sites began operating their own servers. The lack of support for locating the appropriate server for a particular query motivated approaches such as X.500, which provides a global naming tree with support for searching and distributed operation. However, because X.500 (like WHOIS) depends on manual administration, at present its coverage is limited to approximately 2,500 domains – fewer than 5% of the existing Internet domains. Moreover, many of the X.500 servers hold information about only a handful of the users at a site.

To overcome these scaling problems, Netfind gathers and cross-correlates information from many different sources. Doing so allows people to be found at many sites without requiring significant administrative effort on the part of those sites to install new software, or to populate a database of user information. Clearly, the information that Netfind gathers had to be manually entered at some point, but Netfind does not require additional effort to build a database specifically for the purposes of providing a white pages service. Instead, it makes use of bits and pieces of information that were created for various other uses, which often have natural reasons for being kept up-to-date. For example, by examining mail forwarding information, Netfind is often able to locate current information about users, because mail forwarding information tends to be kept current for users who use electronic mail regularly. Of course, in some cases Netfind provides poor information, because it attempts to exploit heuristics that do not apply to some sites.

The combination of scaling and heterogeneity requirements can create additional problems. For example, scaling concerns suggest that information should only be gathered from sources where it most naturally resides – e.g., using the Finger protocol [Zimmerman 1990] to gather user information from workstations where users do their daily work. But the addition of heterogeneity means information must be gathered from a variety of sources, including ones (such as WHOIS) that tend to become stale. In this case, there is some tension between the goals of scalability and heterogeneity, which can only be partially resolved.

The reader may note that Netfind cannot help users who wish to locate people about whom no site information is available. This limitation is a consequence of our belief that a white pages service should access individual user information from the end nodes where it naturally resides, rather than trying to build a global database of all users (which would be necessary to support search requests without specifying site information). Our approach avoids difficult problems of consistency and transfer of authority that arise when constructing a centralized database [Schwartz & Tsirigotis 1991]. For example, the "On since" time shown in Figure 2 was only a few hours old when this figure was captured, and often tends to be this current. Seeing this time indication often lets users know that they have found quite recent/valid information with Netfind.

By way of comparison, the Internet Engineering Task Force is developing a distributed directory service called Whois++ (described in Section 5) intended to support global searches for people and other network-accessible information. Whois++ uses an approach intermediate between a global database and the Netfind approach. In the Whois++ approach, servers gather indexes that map names of people and information objects to pointers to sites that hold the full records. At search time, the system can then retrieve the full records from the remote sites. However, we believe this approach will not scale for use as a global user directory, because most of the names for which users will search will be found at a large number of sites. (For example, there is probably a "Schwartz" in every moderately large city in the world.) This means either that a large number of servers must be contacted to answer each query, or that the user be asked to select among the sites – as Netfind requires.

In the next section we present a general framework for gathering information to support resource discovery in a large, heterogeneous internet.

3. Information Gathering Framework

Information gathering is a complex process, because it must function in an environment characterized by diverse, changing information sources, and typically requires application-specific processing. Even so, one goal of this paper is to demonstrate that information gathering can be implemented in an organized and extensible fashion. Towards this end, in the current section we outline a general framework for information gathering. The framework consists of an iterative composition of four basic steps, illustrated in Figure 3.

1. Information Source Selection
2. Information Collection
3. Information Filtering
4. Combining Heterogeneous Information

Figure 3: Basic Information Gathering Steps

Step 1 involves examining the information that has already been gathered, and applying some rules to determine what information sources should be selected for the next gathering iteration. As an initial condition, no information has been gathered, so this step simply selects all possible sources, one at a time. We discuss the case where information has already been gathered shortly.

Step 2 involves collecting information from a selected external source. For example, this step could gather domain registration records from a NIC database, or could collect logs of the hosts that used a network service (such as Gopher [McCahill 1992] or electronic mail) on a given day.

Step 3 involves applying some source-specific criteria to eliminate unusable information. For example, NIC domain registration records typically contain street addresses, which can be excluded for the purposes of an electronic mail directory service. This step might also filter invalid information, such as the names of non-existent domains.

Step 4 involves combining records gathered from multiple sources, and resolving heterogeneity problems among the various sources. Syntactic heterogeneity (such as differing log file formats collected from various network services) can usually be handled by simple mechanical translations. Semantic differences present more difficulties. For example, some NICs gather records for individual users, while other NICs gather records for entire administrative domains. Extracting domain information from a user record requires an understanding of how users are associated with domains. In some cases these relationships are not obvious. For example, users may be registered according to their community role (such as a liaison for a standards effort), but with an electronic mail address corresponding to their place of employment (such as a particular company). Correctly extracting domain information from such records requires an understanding of the semantics of the data. We consider this issue in more depth in Section 4.1.

Once some initial set of data has been gathered, filtered, and combined, the process iterates. For example, information combined in step 4 from network service logs might indicate that a new query should be made to a NIC, to retrieve more detailed information about some newly discovered domains. Step 1 makes the decision to generate this query based on an understanding of what type of information is available from each source, and of the circumstances in which a NIC is likely to have any additional information about the newly discovered domains.

At its core, an application like Netfind involves an iterative composition of these basic steps. Each information source is described in terms of the decision of when to tap that source (step 1), how to gather (step 2) and filter (step 3) the data, and how to combine the resulting data with other data that have been previously gathered (step 4). The application can then be implemented by constructing software that meets the specifications uncovered by this analysis, and piecing these parts together in an appropriate sequence.

Over time the implementation of any particular discovery application will naturally become more complex, because data must be gathered from an increasingly diverse and continually changing collection of sources.

Decomposing the application according to the framework improves scalability by keeping the complexity of the component pieces manageable. Because there are many sources of information, each source is first analyzed in isolation. The interactions among sources are then isolated into specific combinations. In this way, when new information sources are to be incorporated or existing sources change, the effects can be addressed by localized modifications.

Note that this framework does not address how to store the rules or how to manage the flow of data between gathering components. These are important but separate problems, which have been addressed elsewhere [Rajendra, Jagannathan & Baum 1989, Wood, Coleman & Schwartz 1993].

While the basic steps can be implemented for an application given only a conceptual description of each data source, a working system will typically involve a number of important optimizations. In the case of Netfind, for example, the information gathering process is broken into two phases, the first of which runs essentially continuously, building a cache of usable data for the second phase. Without this intermediate cache Netfind would require an enormous amount of activity to support each search.

We now discuss the specific information gathering mechanisms used by Netfind.

4. Information Gathering in Netfind

As illustrated in Figure 1, the user targets a Netfind search in two phases. In the first interaction, the user specifies the name of an person to locate and a set of keywords describing the site where that person works. In the second interaction the user selects among a list of possible sites to search.

As illustrated in Figure 4, information gathering in Netfind is divided into two phases corresponding to these two phases of user interaction. During the *site discovery* phase, names and descriptive information about Internet sites are collected, refined, and stored as records in a *site database*. This site database is consulted in response to the first user interaction, to map the specified keywords to a list of possible search sites. The second user interaction narrows the site list to a set of starting points for the *user search* phase. This phase then consults a number of Internet services to search for user information at the selected sites.

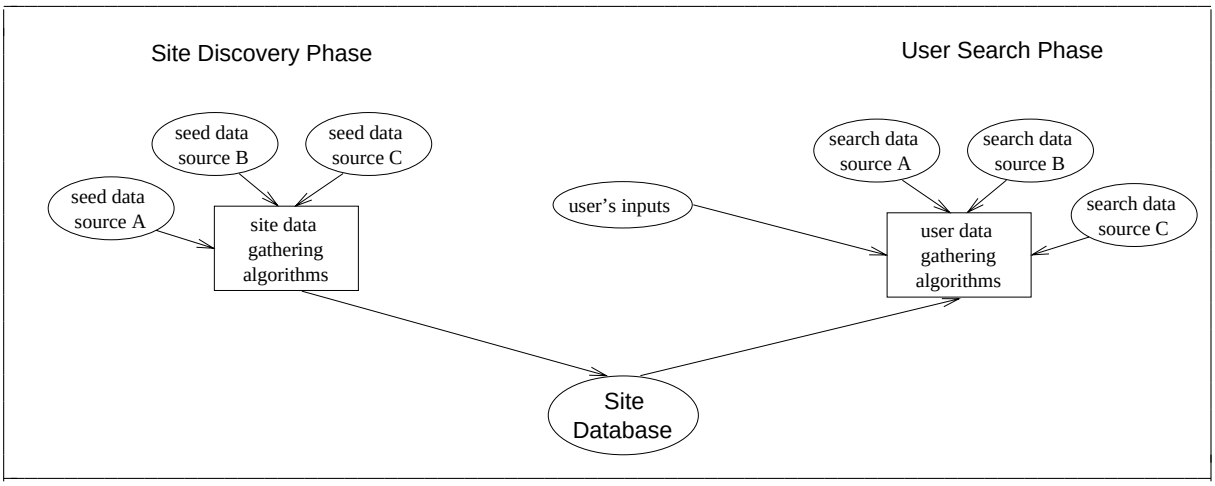

Figure 4: Overview of Netfind Information Gathering Phases

As illustrated in Figure 5, each site database record contains three fields: A domain name, an organizational description,[2] and a list of some hosts in the domain. During the first search targetting interaction users can specify any of the keywords listed in the organizational description, as well as any of the dot-delimited domain

[2] Ideally, organizational descriptions would be further divided into separate fields indicating city, state, country, etc. In the future we may modify the system to provide this additional degree of information structure.

name components. These keywords usually include site names and some geographical information.

```
cs.colorado.edu
computer science department, university of colorado, boulder
kinglear chainsaw anchor piper bruno eclipse ...

math.colostate.edu
mathematics department, colorado state university, fort collins
euclid cayley von-neumann erdos galois gauss ...

univnorthco.edu
unspecified
dijkstra slinky unc-gw talisman ...
```

Figure 5: Example Site Database Records

The Netfind site discovery and user search phases each involve a number of steps. Below we discuss the concrete workings of each phase, decomposing each into component steps using the framework presented in Section 3. Figure 6 illustrates Netfind's overall framework steps.

4.1. Site Discovery Phase

The algorithms that gather information for the site database provide critical support for Netfind's functionality, performance, and scalability. They support functionality by providing a descriptive, far-reaching database of sites to search. They support performance because they amortize the cost of collecting site information across many searches.[3] Finally, they support scalability because only the site database must grow as the Internet grows. The number of sites that are searched in response to each user query remains small.

In our original implementation [Schwartz & Tsirigotis 1991], site discovery was limited to gathering USENET [Quarterman 1990] electronic bulletin board article headers. However, that data turned out to be quite incomplete, and to contain many errors. In response to these problems we began collecting information from a number of other sources. We also developed techniques for improving data quality and completeness, based on application-specific semantics.

Gathering site data is complicated by the enormous decentralization of the Internet. While there are a variety of NICs and other focal registration points in the network, no NIC carries anything approaching complete information about the Internet. Instead, we must gather the site database from a variety of sources, to resolve the inherent incompleteness and data quality problems this causes.

Figure 7 illustrates the current site discovery process. Ovals represent information sources, while rectangles represent processing blocks. The blocks are numbered according to their basic sequence, so that we may refer to them in the text below.[4] In some cases the order was chosen to ease description, as the blocks are not necessarily executed in a single predetermined order.

The initial state of the system is an empty site database, from which the system is ''bootstrapped'' (step 1 of the information gathering framework) with only the rules that tell it which sources to include for the information collection step. The decision at this point is simple: Each of the information sources is marked eligible for collection (step 2 of the framework). The set of information sources is summarized in the Appendix.

The first observation is that some information sources provide more detailed information than others do. At a minimum, each source provides a set of host or domain names. Given only this information, the names can be

[3] In fact this is more than just a performance issue. Some information sources (such as electronic bulletin board postings) are regularly deleted after brief periods of time. It is necessary to gather such information regularly, in order to capture all of it.

[4] We refer to block numbers with braces, such as ''{4b}''.

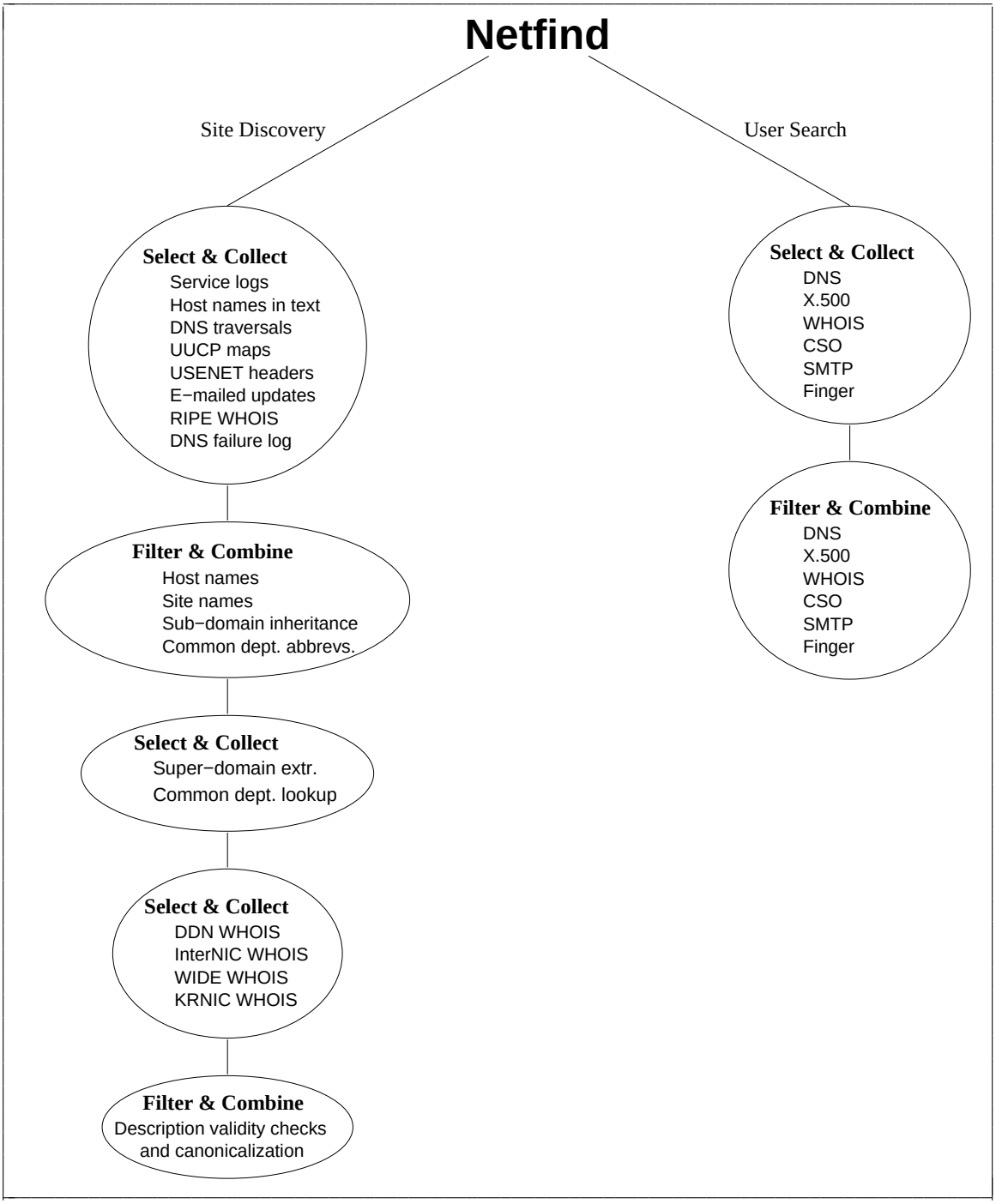

Figure 6: Overall Netfind Information Gathering Framework

validated and included in the site database, with ‘‘unspecified’’ listed for the organizational description. Entries like this can be used in searches, but lack some of the keywords users might expect, and provide less useful clues to users when presented at search targetting time. Some of the sources (those under the right-hand horizontal line at the top of Figure 7) also provide site organizational descriptions.

Newly discovered hosts are filtered (step 3 of the framework) and combined into a standard ‘‘host & site names’’ table. Invalid hosts could be filtered simply by consulting the DNS for each name. However, this

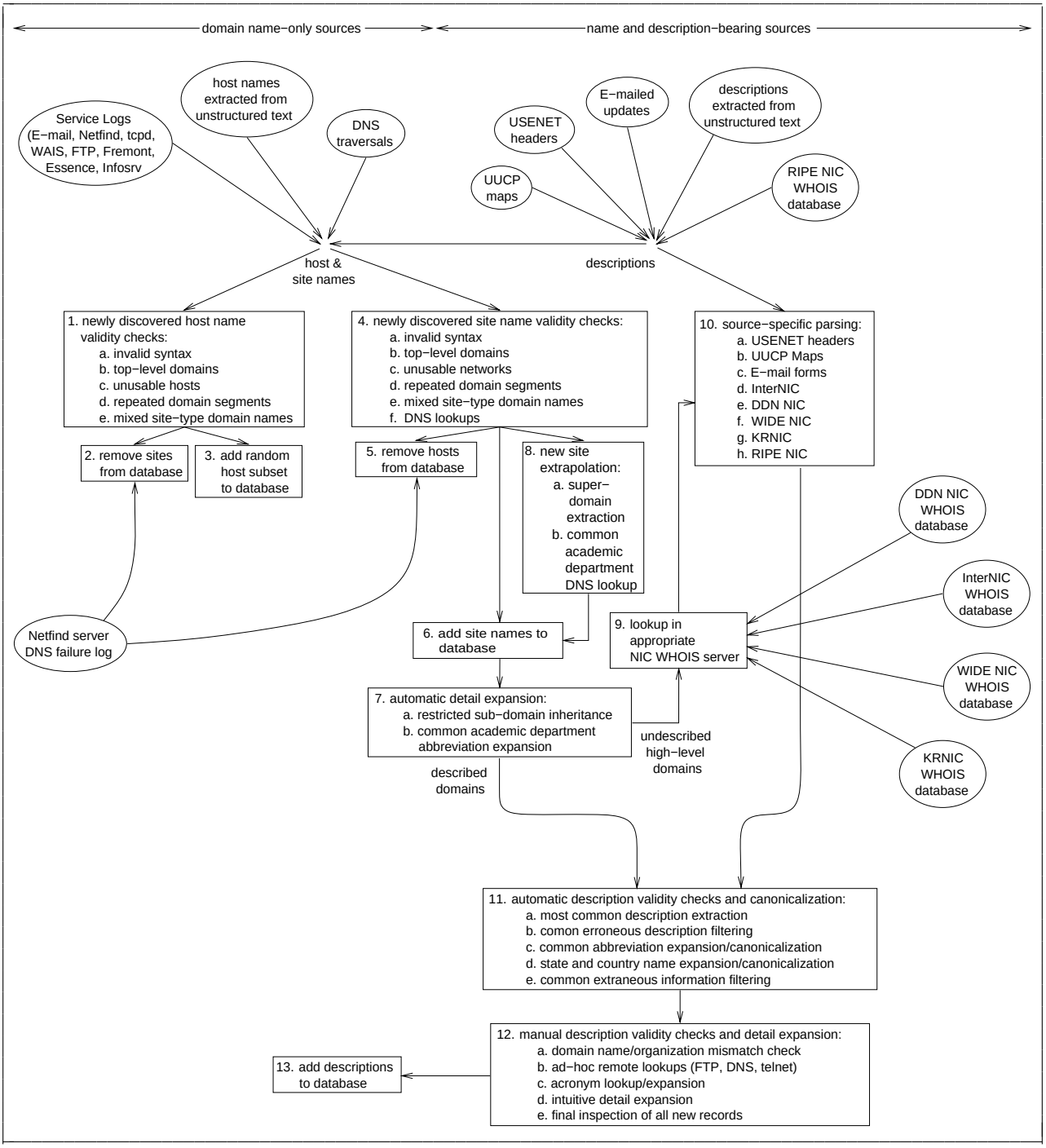

Figure 7: Detailed Data Flow for Site Discovery Process

would cause unnecessary load, because many hosts in the database may never be used during searches. Instead, some local filtering is done using application-specific techniques at gathering time, and the remaining filtering is done at search time by logging DNS lookup failures (a ‘‘lazy evaluation’’ cache-coherence strategy; about 0.5% of the lookups result in failures). Newly discovered sites are checked using both local filtering and gathering-time DNS lookups, because bad sites show up during search targetting, and hence can confuse users. The use of application-specific local filtering reduces gathering-time DNS lookups by an order of magnitude.

Gathering-time host filtering includes checks for a variety of common errors: invalid syntax (for example, host names containing invalid punctuation characters) {1a}; invalid top-level domains (such as ‘‘.eud’’ where ‘‘.edu’’ was intended) {1b}; hosts that are not useful in searches (such as ‘‘localhost.anything’’, since this translates into an address that always refers to the local host, independent of what ‘‘anything’’ is) {1c}; repeated domain segments (such as ‘‘latour.cs.colorado.edu.colorado.edu’’) {1d}; and names that contain some common mixed top-level domains (e.g., ‘‘cs.colorado.edu.com’’) {1e}. Hosts that pass these tests are added to the site database {3}, up to a maximum of 20 hosts per domain.[5] If they had previously been listed in the site database as domains, they are removed from the domain list {2}.

Information sources that provide both names and descriptions are filtered for accuracy and conformity to a simple source-specific data structuring convention, and then combined into a standard ‘‘description table’’ format for the site database. The filtering block for these sources uses validity checks similar to those for hosts, except checks are made for networks (such as BITNET) that do not support remote operations in block {4c} in place of the corresponding block {2c}; and each new site is checked against the DNS {4f}, as mentioned above.

Note that the validity checks do not include checking if a site is connected to the Internet. This check is made at search time (Section 4.2) instead of excluding the site from the site database, so that sites not connected to the Internet at site discovery time will be immediately searchable if they later connect to the Internet.

In terms of our framework, the above blocks each have some understanding of the structuring convention for the data being parsed, and use this convention to filter unneeded or unusable data. The information loss is usually acceptable, if the structuring convention is of reasonable quality.

After the newly discovered site names have been added to the site database {6}, a number of blocks are executed to try to fill in organizational descriptions for sites discovered from name-only sources, first based on other data that have been collected (corresponding to step 4 of our framework), and then by consulting external data sources (corresponding to steps 1 and 2 of the framework). In block {7a} a check is made to see if definitions exist for higher level domains. For example, if the domain ‘‘astro.columbia.edu’’ is discovered and there is a record for ‘‘columbia.edu’’ in the site database with the description ‘‘columbia university, new york, new york’’, the new domain can inherit this description. The module that does these collection operations must take care not to inherit descriptions from generic parts of the domain tree. For example, it is not possible to infer much useful information about the meaning of ‘‘colorado.edu’’ from the ‘‘edu’’ record. Block {7b} attempts to add department names to records that inherited organizational descriptions from higher level domains, based on a list of approximately 450 common abbreviations. For example, this abbreviation list would allow ‘‘astronomy department’’ to be inferred for the ‘‘astro.columbia.edu’’ record. The results of this step are sometimes incorrect (e.g., "cs" might refer either to "computer science" or "careers service"). However, the abbreviations database consists of abbreviations that are most commonly used one way, and we allow sites to send us updates (discussed below). On balance we feel the added detail merits the occasional errors.

Note that the above information combining blocks may also perform some filtering, because the combination of heterogeneous data is often imperfect.

Once a new site has been discovered, it is often possible to discover related sites. This represents step 1 of our information gathering framework, and is embodied in blocks {8} and {9}. Super-domain extraction checks if each of the ancestor domains is in the database {8a}. For example, discovering the site ‘‘camelot.cs.cmu.edu’’ indicates there should also be the domains ‘‘cs.cmu.edu’’ and ‘cmu.edu’’. Common academic department lookups involve checking a list of approximately 50 common department abbreviations (such as ‘‘bio’’, ‘‘math’’, etc.) for newly discovered domains {8b}. For example, given ‘‘cmu.edu’’, block {8b} would discover the domain ‘‘bio.cmu.edu’’.[6]

If no description could be produced by block {7}, sometimes information about a site can be found in a NIC WHOIS database.[7] The main limitation in this regard is that most NICs maintain information only about high-

---

[5] A random host subset is added to the database because there is no obvious way to decide that some hosts are better than others for search hints (the use of these hosts are explained in Section 4.2). However, we allow users to provide updates, to improve the listed set. Moreover, this list of hosts is not used for the most preferred cases, as described in Section 4.2.

[6] It turns out that some sites set up mail forwarding arrangements such that any string may be prepended to their domain name, and their DNS servers will simply respond with a mail exchange (MX) record for that string. Because this arrangement would defeat block {8b}, we test for it by looking up an unlikely name, and seeing if an MX record is returned.

[7] Many individual sites run WHOIS servers populated with data about individual people. Block {7} only uses WHOIS servers that pro-

level domains. For example, the U.S. InterNIC maintains information only about first and second level domains (such as ‘‘edu’’ or ‘‘colorado.edu’’). If information can be found in a WHOIS server {9}, it is parsed in block {10}. The RIPE WHOIS server is listed at the top of the figure rather than in connection with block {9} because RIPE provides a means to retrieve their database en masse. This allows us to use it as a primary discovery source. None of the other NICs provide such database access, so we are forced to perform individual lookups after discovering sites from other primary sources.

Block {10} involves source-specific parsing of incoming site description records, whether from primary discovery sources (listed at the top right of the figure) or from information derived from NIC lookups in block {9}. After parsing, a number of validity checks are performed {11}, and attempts are made to transform the records into a canonical format. In some cases (primarily for USENET header data), multiple records may be found from external sources. Block {11a} chooses the most commonly occurring description. This simple procedure eliminates a good deal of erroneous data, for example caused by individual users posting USENET messages with forged organization lines in the headers.[8] Block {11b} filters certain common erroneous descriptions, such as ‘‘no organization’’. In block {11c} we expand descriptions based on a list of common abbreviations, to increase the uniformity of keywords that describe sites. For example, this block expands ‘‘dept.’’ to ‘‘department’’. In block {11d} we apply a set of modules to expand geographic names. Unlike block {11c}, in block {11d} we limit the expansion based on position: We expand U.S. two-letter state abbreviations at the end of a description line (e.g., expanding ‘‘university of houston, tx’’ to ‘‘university of houston, texas’’), and we expand country names based on the ISO country codes embedded in site names (e.g., expanding the description for ‘‘infn.it’’ to include ‘‘italy’’). Finally, we filter common forms of unneeded information, using a set of pattern matching strings to remove information such as post office box numbers and zip codes. We remove this information because it usually does not provide useful search keys. At one time (as discussed in [Schwartz & Tsirigotis 1991]) we retained such additional information in the site database to augment the information found at the remote site, but we ceased doing that because it made the site database much larger and often provided stale information relative to what can be found at the remote site.

Many of the parts of blocks {1}-{11} were developed specifically to limit the amount of manual effort that must be applied when harvesting newly discovered information. The less manual processing that must be done, the better the process scales – an important issue, given the exponential growth rate of the Internet. Block {12} contains the processes we have not been able to automate, primarily because they rely on experience and intuition. These types of manual checks are needed to maintain a high quality site database, because of the mixed quality of the gathered information sources. Originally (before this block existed), we simply gathered data from USENET headers, and placed them in the site database without any validity checks. The amount of spurious information quickly grew to the point that it was a noticeable distraction to users at search targetting time. At that time we painstakingly checked over 4,000 records manually. While we are unable to eliminate such manual effort completely, we have automated many of the checks (placing them into the steps outlined in blocks {1}-{11}), and isolated the remaining checks to a single manual validation block ({12}). To make the process less burdensome, we now perform manual validation in small increments, by periodically validating updates for a few dozen updates at a time. While manual validation impacts scalability, we believe such effort is an inherent requirement when gathering and integrating imperfect information. Manual validation is particularly important in the case of seed data discovery because the inheritance operations performed in block {7} would propagate bad descriptions to many subsidiary records.

Block {12a} is the most basic check, looking for incorrect organization lines associated with site names. For example, this block catches problems where a site administrator copied configuration files from another site and neglected to localize them (resulting in USENET postings that list the wrong organization for a site, for example). Some of these checks might be automated (e.g., ‘‘university of wisconsin’’ shows up particularly often in such erroneous organization lines), although to date we have not done so.

Block {12b} involves manually searching for descriptive information using some common Internet services. For example, if a site supports anonymous FTP,[9] they often provide an announcement banner containing the

---

vide information about domains.

[8] USENET does not authenticate postings, and users sometimes forge message headers for the sake of anonymity. In fact, news groups concerning controversial subject matter often exhibit high rates of forged messages. We do not collect USENET headers from certain news groups for this reason.

[9] FTP is an Internet standard protocol that supports transferring files between interconnected hosts [Postel & Reynolds 1985].

name of the site when users login. There are similar (but less commonly successful) methods of locating information using the DNS and telnet.

Block {12c} involves looking up acronyms contained in discovered organizational descriptions, in an external database of acronyms we have amassed from a variety of sources. For example, this database allowed us to determine that ‘‘gsfc.nasa.gov’’ was NASA’s Goddard Space Flight Center. Block {12d} involves expanding details based on intuition or other personal knowledge. For example, if there are a list of domain names for company with first components corresponding to city names (‘‘sandiego.company.com’’, ‘‘boston.company.com’’, etc.), we could expand the descriptions to include the city names. Finally, block {12e} involves a last pass inspection of all new records, after all other manual checks and modifications have been made, to try to catch any problems that might have been overlooked.

As noted earlier, the site database essentially acts as a disk-based cache,[10] since the above discovery process could in principle be repeated every time Netfind is run. The purpose of the site database is to cache results that will be valid for a reasonably long time (usually on the order of months), and distribute them to all servers. This approach amortizes the cost of collecting site information and keeps the user search phase scalable, because only the site database must grow as the Internet grows – the number of sites that are searched in response to user queries remains small.

4.2. User Search Phase

Figure 8 illustrates the user search algorithm. It begins by querying the cached information present in the site database, using the user-specified search keys. This query results in a list of potential domains to search, from which the user is asked to select at most three (illustrated in Figure 1). Doing so reduces the scope of the search and implicitly selects (step 1 of the framework) a small set of remote information sources to be queried when collecting (step 2) the information.

After search targetting is complete, Netfind queries the DNS servers at the selected domains, to locate information that can be used to decide how best to perform a user search at each domain. In the most preferred arrangement, the domain will have registered a set of Uniform Resource Locators (URLs) [Berners-Lee 1993] describing the set of white pages services it runs.[11] Netfind sorts the returned URLs (e.g., preferring to retrieve structured data from X.500 when it is available), and then tries them in order. This approach corresponds to the recommendation put forward by RFC 1588 [Postel & Anderson 1994]. At present, Netfind’s user search phase is able to access information from X.500, WHOIS, CSO [Dorner & Pomes 1992], the Simple Mail Transfer Protocol (SMTP) [Postel 1982], and the Finger protocol.

Because SMTP and Finger are much more widely deployed than the other services in the current UNIX-[12] dominated Internet, Netfind attempts to use these services if no URLs are registered at a domain. Originally, Netfind tried only these common services (SMTP and Finger) [Schwartz & Tsirigotis 1991]. The use of URLs is intended to move away from informally deployed services, and towards an interoperable user directory service framework.

Because X.500, WHOIS, and CSO each register a domain’s accumulated user information in a logically centralized fashion, querying one of these services requires no further data gathering operations. In contrast, SMTP and Finger each provide only a fraction of a domain’s information at any particular machine. Therefore, Netfind must perform further gathering operations to use these information sources. Because they currently provide most of the answers to users’ searches, and because the gathering operations needed to use them provide good

---

Anonymous FTP is a convention for allowing Internet users to transfer files to and from machines on which they do not have accounts, for example to support distribution of public domain software.

[10] Cache consistency is managed through a combination of invalidating data when it is detected as being stale, periodically checking the data against the primary sources, and allowing manual updates (in the form of users requesting changes to the database; we receive approximately 180 such updates per year).

[11] URLs provide a uniform syntax for specifying addressing information needed to access an information resource, as well as a list of parameters that can vary according to the access method. At present the DNS does not support an explicit URL record type, so Netfind’s URLs are encoded using DNS text records, using a syntax described in [Schwartz 1994]. We are currently attempting to standardize URL support in the DNS.

[12] UNIX is a registered trademark of UNIX System Laboratories.

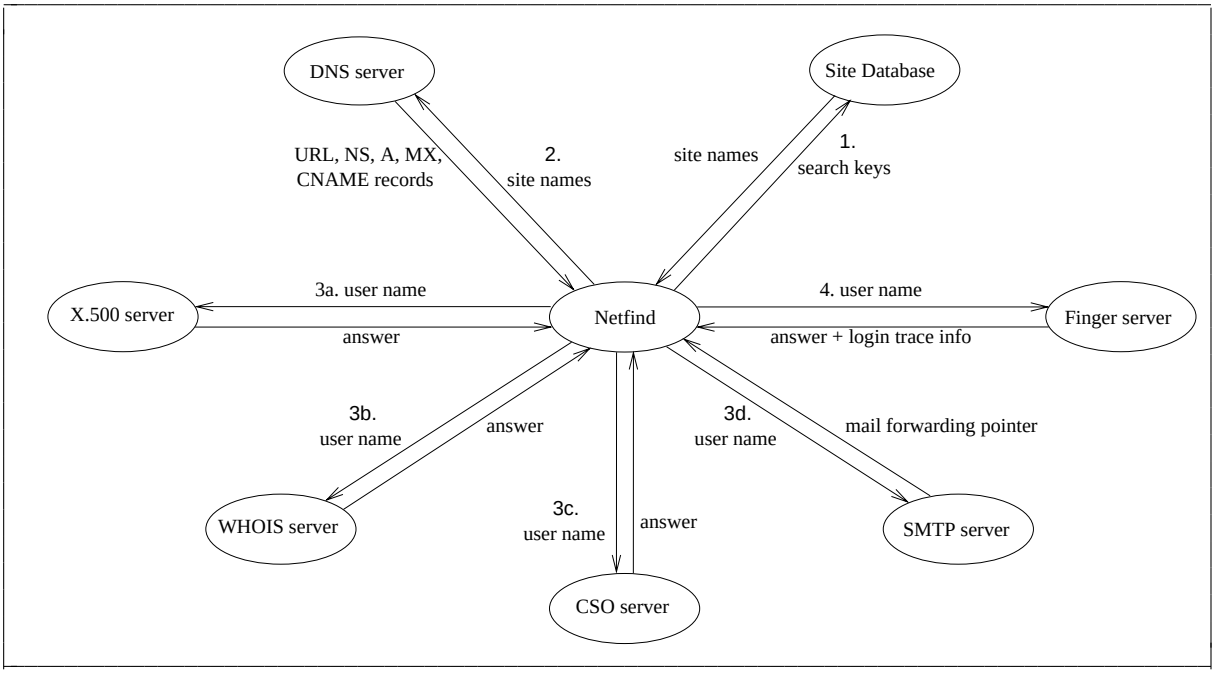

Figure 8: User Search Algorithm

examples of our gathering framework, in the remainder of this section we focus on gathering data using SMTP and Finger.

To locate information using SMTP and Finger, Netfind initiates a two stage search process. Stage 1 uses the DNS, SMTP, and Finger to try to locate users based on a set of heuristics to direct searches to promising hosts. Stage 2 involves a less focused (but still heuristically-based) search, for domains where the stage 1 heuristics did not provide satisfactory results. Both of these stages fall under step 2 of our information gathering framework. The results of the entire search process are filtered for clarity (framework step 3) and combined and summarized for the user (framework step 4). Each stage proceeds in a parallel fashion, using five ‘‘lightweight processes’’ [Kepecs 1985] to allow sets of DNS / SMTP / Finger lookup sequences to proceed in parallel during stage 1, and Fingers to proceed in parallel in stage 2. Doing so increases resilience to host and network failures, and reduces response time variance [Schwartz & Tsirigotis 1991]. Below we discuss the specific rules used to guide searches in each stage.

In stage 1, Netfind requests information from the DNS, to try to locate IP addresses associated with each domain (e.g., the domain ‘‘cs.colorado.edu’’ has an IP address corresponding to the machine ‘‘bruno.cs.colorado.edu’’), and up to two name servers for each of the domains isolated during search targetting. The located IP address and DNS server hosts are often central administrative machines, with accounts and mail forwarding information for many users at a domain. Hence, they are promising places to begin searching. If Netfind encounters a canonical name (CNAME) or mail exchange (MX) record, it follows up to one level of indirection, and uses this as the IP address for the domain. This is not the most thorough possible approach for MX records (it would be possible to check for mail forwarding information at each MX host), but it provides a reasonable approximation for many domains.

If the domain has an associated IP address, that address is used to check for mail forwarding information about the specified person. If there are no IP addresses but the domain is running its own name servers (as opposed to having another domain provide name service for that domain), the name server machines are used to check for mail forwarding information. If the domain has no address records registered in the DNS, the user is informed that the domain is probably not directly connected to the Internet, and hence is not searchable by Netfind.

Mail forwarding information is checked using SMTP servers on the located hosts, by issuing an ‘‘EXPN’’ command. If EXPN queries are not supported, Netfind also tries ‘‘VRFY’’.[13]

Netfind keeps track of which hosts have been listed for mail forwarding, and displays each to the user only once. Netfind also checks the results of each SMTP mail forwarding query, to determine if the contacted server responds incorrectly to queries (by returning the query itself as the result, for all queries). While doing so violates the SMTP specification, some domains set up their servers to respond incorrectly to queries for privacy or security reasons. Netfind also checks if the forwarding information contains extraneous details, such as the delivery path of a local mail agent through which mail is forwarded. In this case, the user is informed that mail is sent through a local delivery agent.

If valid mail forwarding information is found, Netfind displays the mail forwarding information, which usually includes the person’s ‘‘home’’ machine. If only a user name was found (rather than a personal name plus user name) in the mail forwarding information, Netfind also attempts to do an SMTP ‘‘EXPN’’ on the host to which mail is forwarded, since that will sometimes provide a personal name.

If mail forwarding information is located at the IP / CNAME / MX address for the domain, the user is informed that this is a domain mail forwarding arrangement, and hence that mail intended for that user should be sent to user@domain rather than user@<hostname>.domain. For example, the domain ‘‘cs.colorado.edu’’ has an associated IP address corresponding to the machine ‘‘bruno.cs.colorado.edu’’. This host is checked using SMTP for mail forwarding information. This check is useful because many domains attempt to hide individual machine names in outgoing electronic mail headers, so that users may send mail from any machine in the domain without having to inform an administrator or database that they want response messages forwarded back to their home workstation.

An exception to the mail forwarding check is if forwarding is found to cross domain bounds. If this is the case, it is likely that the person being sought has moved to another domain. Unlike the usual domain forwarding arrangement case, in this case the user is told that mail should probably be sent to the forwarded-to location.

Once a domain mail forwarding arrangement is found, no more mail forwarding information lookups are attempted.

After the mail forwarding search, Netfind attempts to Finger the user on the machines to which mail is forwarded, to acquire more information about the user.

If no mail forwarding information was found, Netfind attempts to Finger the user at the domain’s IP address host and on each name server host. Also, if there are fewer than 20 hosts listed in the site database record, Netfind attempts to locate other possible search hosts as follows. First, it Fingers any domain IP address and/or name server hosts uncovered above, without specifying a user (the result that would occur from the command ‘‘finger @host’’). This retrieves a list of of users logged in to these machines. Netfind then parses this list to find the machines from which the users are logged in, to locate other machines to search.[14]

Originally [Schwartz & Tsirigotis 1991], Netfind used this ‘‘finger @host’’ technique for each searched host (possibly recursively, but only one level deep). Since we found that this rarely produces useful information, we limited ‘‘Finger @host’’ probes as described above.

At this point, stage 1 is complete, and the user is asked whether to proceed. This is necessary because Netfind cannot determine whether a name match corresponds to the person actually being sought.

In stage 2, Netfind Fingers individual machines listed in the site database that fall within the domains specified during search targetting. (In the ‘‘optimal’’ case where users are located in stage 1, the list of machines in the site database is not used.) These hosts therefore act as ‘‘fallback’’ search targets if the URL / X.500 / WHOIS / CSO / DNS / STMP / Finger sequence fails. This stage is more expensive than stage 1, because dozens of Finger searches per domain can be initiated (as compared with the one to three searches per domain that execute in stage 1). Netfind places a limit of 50 machines that can be Fingered for any particular search. On average, however, far fewer machines are Fingered [Schwartz & Tsirigotis 1991]. Also, Netfind

---

[13] Netfind does not attempt ‘‘VRFY’’ queries unless ‘‘EXPN’’ failed, as ‘‘VRFY’’ works in fewer domains, and is considered by some domains to be more ‘‘invasive’’.

[14] In some cases the results of the ‘‘finger @host’’ output contain truncated machine names, due to limitations in the UNIX logging software. Netfind attempts to fill in such names by matching up the strings with the name of the domain being searched.

keeps track of which hosts have been Fingered, and does not Finger any host more than once (which could happen otherwise, for example, because mail forwarding information or information from the ‘‘finger @host’’ output could duplicate hosts in the site database).

At the end of each search stage, Netfind attempts to trace the person being sought back to his/her home workstation, by following the ‘‘Last login <date> from <host>’’ or ‘‘On since <date> on <tty> from <host>’’ information in the Finger output, as follows. For each Fingered host, the last login dates are compared. The <host> listed with the most recent login is then Fingered (in some cases first extending truncated host names to full names, as described above), to see when the person logged in from that host. This continues iteratively until no newer logins are located. ‘‘On since’’ information is processed similarly, except that it is preferred over ‘‘Last login’’ information (because it is usually more current), and the time preference is for oldest login sessions. This search process causes the site database hosts to be more useful, because starting a search with one of them will often lead back to the user’s home machine. This information is particularly useful if no mail forwarding information could be located during stage 1.

Last login/most recent login information is not traced if more than one person is located during the search. Also, information is not traced past domain boundaries, since, for example, if a person last logged in from an outside domain this often indicates they are traveling and remotely logged in from a colleague’s machine to read their mail.

Because Netfind attempts to locate people using directed queries before it ‘‘falls back’’ to stage 2 searches, it generates only a modest amount of network traffic. Our measurements indicate that the average search requires 136 packets, including those required for DNS lookups. For extensive load measurements, see [Schwartz & Tsirigotis 1991].

4.3. Distribution of Information Gathering

The implementation described thus far focuses on scaling problems associated with adapting to diverse, changing information sources that require application-specific processing. However, another important scaling issue concerns distributing the gathering process itself. A system that performs all gathering in a centralized fashion clearly will not scale for use in a global internet.

At present, Netfind uses simple techniques for distributing the gathering processes. The site database itself is constructed by a set of processes that run on several machines at the University of Colorado, which update a database at that site. This database is made available by anonymous FTP, allowing it to be replicated globally. At present, approximately 700 sites around the Internet replicate this database, in many cases using software that periodically checks for updates [McLoughlin 1993].

Even with this relatively centralized gathering approach, site discovery proceeds at a surprisingly rapid rate – and, to the best of our knowledge, provides the world’s most thorough database of sites. As an example, Figure 9 plots the rate of discovery of new sites from the service log data source (the upper leftmost oval in Figure 7). As this plot shows, after only one month of monitoring the service log gatherer discovered nearly 5,000 sites, distributed among 84 countries.

The user search phase of Netfind can run at any site that has a copy of the site database, in response to individual search requests. The user must pick a site to initiate a request; there is no automated load sharing. Most of these sites are running private copies of Netfind, in use by people at that site only. However, most of the searches performed by Netfind are handled by the collection of twenty publically accessible Netfind server sites, located in 11 countries.[15] For example, the University of Colorado Netfind server handles approximately 2,500 queries per day at present.

Netfind is basically a client of the many remote information services with which it interacts, and not a server in the usual sense of that term (i.e., a piece of software to which a remote client can connect and interact via a structured protocol). Rather, Netfind interacts with each user via a Network Virtual Terminal telnet interface. We call it a "server" because many users gain access to Netfind through this interface, and we feel it is less confusing to label it a server for the majority of users who simply perceive it as an information service, without

[15] Australia, Brazil, Canada, Chile, the Czech Republic, Poland, Singapore, the Slovak Republic, the United Kingdom, the United States, and Venezuela.

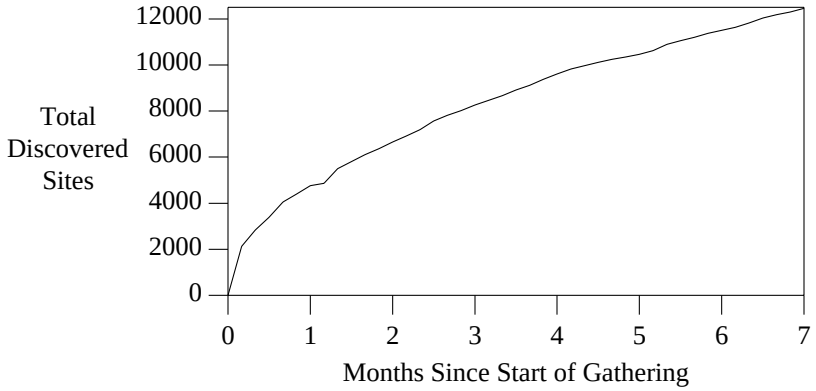

Figure 9: Service Log Site Discovery Rate

concern for architectural distinctions. However, we are currently developing a true client/server interface to Netfind. Once this protocol exists, it will be possible to provide a graphical, window-based interface to Netfind.

The Internet Research Task Force Research Group on Resource Discovery[16] is currently experimenting with more scalable approaches to distributed gathering, which incorporate incremental updates; a hierarchical, topology-aware flooding update scheme to circulate index updates around the network; and schemes for choosing nearby lightly loaded servers [Bowman et al. 1994a]. We believe these features will significantly enhance the scalability of information gathering. Note, however, that we are carrying out that work in the context of information accessible via standard Internet publishing tools (HTTP, FTP, and Gopher), rather than for Netfind specifically.

5. Related Research

At present, the predominant means of offering information services in the Internet is through services that require information to be collected through manual administrative methods. This is the model used by X.500, the Wide Area Information Servers (WAIS) system [Kahle & Medlar 1991], the Internet Gopher, and the World Wide Web [Berners-Lee et al. 1992]. However, there are some exceptions, several of which we discuss below.

Archie provides an Internet file white pages service, by gathering file system directory listings from a set approximately 1,100 UNIX anonymous FTP archives world-wide, and exporting this information via a client/server architecture [Emtage & Deutsch 1992]. These sites currently contain a total of approximately 150 gigabytes of information, in approximately 2.6 million files. In terms of our information gathering framework, Archie selects sources by cycling through a site list and checking each site once per month; Archie collects its database by using the FTP directory listing operation or by retrieving a listing file if the remote site makes such a file available; Archie filters its database by removing invalid names and making various other error checks; finally, Archie combines records in its database by replacing the old versions of the retrieved records and updating the site gathering timestamps.

Archie-like services have also been developed for other Internet information access methods, including Vernoica (which gathers and indexes Gopher menus) [Foster 1992] and the World Wide Web Worm (which gathers and indexes HTTP documents) [McBryan 1994].

Essence uses data gathering to extract summaries from file data [Hardy & Schwartz 1993, Hardy & Schwartz 1994]. In terms of the framework, selection is done by a site-customizable process that chooses files primarily based on file type and name. There is also a facility to run an arbitrary program to select files, to allow more complex decision making. Extraction is done by running a set of programs (called *summarizers*), each of which encapsulates application-specific understanding of a particular common type of file. For example, one summarizer extracts the author, title, and abstract from documents, based on its understanding of the structure of

[16] Schwartz chairs this IRTF Research Group.

the TEX word processing system. Filtering is done as part of the extraction process: Each summarizer applies its own transformations to extract desirable keywords. Finally, combining is done outside of Essence. Essence simply extracts keywords, and leaves their use to an outside system. In the prototype we used WAIS to combine and export the data. We are currently extending Essence to support other types of gathering and indexing applications, in the context of the Internet Research Task Force Research Group on Resource Discovery [Bowman et al. 1994b].

MIT's Semantic File System (SFS) uses similar techniques to Essence for selection, collection, and filtering [Gifford et al. 1991]. However, SFS does not support nested file processing, and SFS provides a specific model for information combining (the file system abstraction); Essence provides a general model for customized information extraction, independent of how the information is combined/exported.

Nebula also gathers keywords from file system data, but uses an explicit typing mechanism rather than heuristic techniques for recognizing and selecting common file types [Dharap, Balay & Bowman 1993]. Moreover, Nebula uses a set of grammars (rather than explicit programs) to collect and filter the data.

Gifford's Content Routing (CR) service gathers indexing information (called ''headlines'') from 500 WAIS databases, to provide a distributed query service [Sheldon et al. 1994]. In terms of the framework, selection is done by cycling through each entry in a list of WAIS servers. Collection is done using a program that retrieves each document's headline from each server. At present no filtering is done. Finally, the CR service combines records in its database, keeping track of which server held each headline.

The WHOIS and Network Information Look Up Service Working Group in the Internet Engineering Task Force is defining an Internet standard called ''Whois++'' for gathering directory information, by creating concise descriptions (called ''centroids'') of each indexed database [Weider, Fullton & Spero 1992]. In terms of the framework, selection is done in a decentralized fashion, where each participating site generates its centroids at whatever frequency it chooses. Collection is presently a manual operation, requiring site administrators to fill in templates describing site aspects such as information content and administrative staffing, and send these templates to a server. At present no automated filtering is done, relying instead on site administrators' creating templates manually. Finally, WHOIS++ combines records in its database by replacing the old versions of the retrieved records and updating the site gathering timestamps.

Fremont gathers data from a number of network protocols and information sources to construct a picture of key network characteristics, such as hosts, gateways, and topology [Wood, Coleman & Schwartz 1993]. Sources include routing table update traffic (RIP), naming tables (DNS), IP error reporting traffic (ICMP), plus a half dozen other sources. Fremont selects among its sources based on a configuration file that specifies the frequency of gathering from each source. Collection is done by running a set of ''explorer modules'', each of which understands how to gather data from a particular source. Explorer modules deposit the data they gather in a network accessible database called a *journal server*. Information filtering and combining are done by a set of application programs that query the journal server and process the outputs in a variety of ways, depending on what aspects need to be highlighted.

JS McBride & Co. collects electronic mail addresses, user names, and search keywords to support searches for Internet software products. They do this by gathering information from electronic news feeds, BITNET list servers, and other sources [McBride 1993]. While limited information was available about this system, it apparently operates by cycling through and selecting each source periodically, gathering data using a set of source-specific programs, and then running a filtering/combining step on the results.

6. Conclusions and Future Work

Providing a scalable Internet white pages service is difficult, because of the rapid growth and increasing diversity of the environment. For such a service to be perceived as valuable, it must provide reasonably ubiquitous coverage. A data gathering architecture provides a powerful means to achieve this goal.

In this paper we presented the Netfind Internet user white pages service, which gathers information from 17 different sources, including USENET news, UUCP maps, service logs, the DNS, WHOIS, CSO, X.500, SMTP, and Finger. In addition to providing the world's most extensive and dynamic directory of Internet users, the extent of this gathering suite serves as a particularly thorough demonstration of an information gathering architecture.

A key notion we wish to advance in this paper is that, while information gathering is a complex and application-specific process, it can be implemented in an organized and extensible fashion. We presented a general framework for data gathering that involves an iterative composition of four basic steps, which describe when to tap a particular data source, how to gather data from that source, how to filter these data, and how to combine the filtered data with other data gathered from other sources and earlier iterations. This decomposition improves scalability by allowing the gathering components to be analyzed first in isolation, and then as interactions among sources in specific combinations.

To go from a gathering specification to a working system, the framework must be augmented by specific performance optimizations according to the way the system is used. In the case of Netfind, gathered site information is cached, but user information is collected dynamically for each search, because of differences in aging rates and sharing patterns of these types of information. Netfind uses other performance optimizations as well, such as local filtering techniques to reduce gathering-time DNS load, and different cache invalidation strategies for hosts and domains.

While introduced in the context of a specific system, the techniques discussed in this paper can be used in any application where the needed data are widely distributed, heterogeneous, minimally coordinated, and continually changing. For example, our framework might be used to help automate a network publishing/archival mechanism, where individual participants maintained their own environment of documents using local naming and organizing conventions. These conventions could be codified and exploited by the gathering tools. As a second example, the framework could help independent sites join forces to provide a new service that harnesses each of their information systems. As a final example, data gathering techniques could be used to automate the process of tracking down security violations in a networked environment, based on audit trails collected on the individual machines.

Of course, there are limits to these techniques. They cannot compensate for poor quality information sources, or cases where resolving inconsistencies requires data-specific human knowledge [Walker et al. 1983]. Nor would it be wise to apply these techniques in situations where erroneous filtering and combining results might lead to catastrophic consequences.

The gathering framework presented in this paper focuses primarily on integrating information sources that were originally intended for uses other than providing a directory service. This approach allows one to construct a far-reaching index without widespread agreement about the many evolving sources of information. As a negative consequence, this approach is limited by the quality and form of available data. To provide more powerful services (e.g., capable of locating complex types of data), more structured input data are required. The Internet Research Task Force on Resource Discovery is exploring a hybrid solution, which will allow sites to define structured data formats and gathering algorithms if they like, but will also apply heuristics to exploit existing information when sites do not put in this extra level of effort [Bowman et al. 1994a]. We feel that the combination of these two approaches will allow a great deal of information to be indexed, yet will also allow sites willing to expend extra effort to reap the benefits of doing so.

Because it eases the task of accessing data that may not have been deployed explicitly for use as a directory (e.g., via the Finger protocol), Netfind raises some privacy concerns. While we are sensitive to this issue, we believe the more fundamental issue is when and how users have control over the information that is made publically available about them. This same issue arises in any user directory service, independent of its architecture (including X.500 and WHOIS), because directories are often populated without explicit prior approval of the people in the database. For a more detailed discussion of these issues, the reader is referred to [Schwartz 1993].

Internet users can use Netfind by telnet'ing to bruno.cs.colorado.edu, and logging in as "netfind" with no password.

Future Work

We are currently experimenting with four extended examples of the framework presented in this paper. First, we are migrating the implementation of the Netfind site database to use Fremont, because Fremont provides better database management and a more flexible architecture for building gathering modules. Doing this will enhance the scalability of site database collection, and provide a basis for building tools for mapping portions of the Internet. Second, we are implementing a set of data gathering modules to support discovery of Network Time Protocol servers [Mills 1985]. Doing this will allow us to explore the problems of discovering

network services, and will provide helpful support to NTP users and implementors. Third, we are investigating ways that service providers could describe properties of their exported data to ease incorporation into data gathering applications. Finally, we are extending Essence to support gathering and indexing of widely distributed data, in the context of the Internet Research Task Force Research Group on Resource Discovery [Bowman et al. 1994a, Bowman et al. 1994b]. This project involves a large effort to develop an architecture and a set of customizable tools for gathering information from diverse repositories in a distributed fashion, building topic-specific content indexes, flexibly searching the indexes, widely replicating them, and caching objects as they are retrieved across the Internet.

7. Acknowledgements

Schwartz was supported for this work in part by the National Science Foundation under grants DCR-8420944, NCR-9105372, and NCR-9204853, the Advanced Research Projects Agency under contract number DABT63-93-C-0052, a grant from AT&T Bell Laboratories, and a grant from Sun Microsystems' Collaborative Research Program. Pu was supported for this work in part by the National Science Foundation under grants IRI-9116798 and BIR-9310154, Hewlett-Packard, Oki Electric, Pacific Power & Light, and the Oregon Advanced Computing Institute.

The information contained in this paper does not necessarily reflect the position or the policy of the U.S. Government or other sponsors of this research. No official endorsement should be inferred.

We thank Darren Hardy, David Wood, and the journal referees for their helpful comments on this paper.

We thank Carl Malamud and Marshall Rose for design discussions about Netfind's URL-based interface to other white pages services.

8. Appendix: Site Database Information Sources

The set of information sources gathered by Netfind is summarized in in Table 1. The example records in this table are abbreviated.

Source	Originally Intended Use	Example Record
Netfind DNS log	logging DNS lookup failures, to remove from site database	oldname.arpa NO_SUCH_DOMAIN badhost.colorado.edu NO_SUCH_HOST
Service logs (WAIS, mail, etc.)	logging service usage	Jul 17 15:51 latour.cs.colorado.edu
Host names from text	textual discussions	To use Netfind, telnet to bruno.cs.colorado.edu and login as ''netfind''
DNS traversals	discovering extent of Internet host names	host : 128.138.204.19 : latour.cs.colorado.edu ::::
UUCP maps	supporting UUCP dialup network	#N boulder #S SUN IPX; SunOS 4.1.1 #O University of Colorado, Boulder #C Jason Ornstein #E postmaster@boulder.Colorado.EDU #T +1 303 492 6096 #P Univ. of Colorado, Boulder, CO 80309 #L 40 00 30 N /105 17 00 W city #R News and mail gateway for CU/Boulder. #W ornstein@boulder.Colorado.EDU Oct 3 17:32 MDT 1992
USENET headers	circulating USENET news messages	Path: boulder!darwin.sura.net!samba.oit.unc.edu!pswecker From: pswecker@med.unc.edu (Peter St.Wecker) Newsgroups: comp.infosystems.wais Subject: WinWais source file Date: 9 Jul 1993 23:44:33 GMT Organization: UNC School of Medicine
E-mailed updates	correcting Netfind site database problems	med.unc.edu medical school, university of north carolina, chapel hill
Descriptions from text	holding textual discussions	Sincerely, Michael Schwartz (schwartz@cs.colorado.edu) Computer Science Department, University of Colorado, Boulder
RIPE NIC WHOIS database	listing European IP networking organization member site points of contact	*an: AS1126 *de: SARA Amsterdam AS *ac: Willem van der Scheun *tc: Henk Steenman *ch: Erik–Jan.Bos@surfnet.nl 930504 *so: RIPE
DDN NIC WHOIS database	listing U.S. military IP networking points of contact	Naval Research Laboratory 4555 Overlook Avenue Washington, DC 20375 Nicknames: NRL.NAVY.MIL Coordinator: Kaczmarek, Edward kaz@GRIZZLY.NRL.NAVY.MIL
InterNIC WHOIS database	listing U.S. civilian IP networking points of contact	University of Colorado Domain Name: COLORADO.EDU Admin. Contact: Klingenstein, Ken KJK@SPOT.COLORADO.EDU Tech. Contact, Zone Contact: Wood, David DCMWOOD@SPOT.COLORADO.EDU
WIDE NIC WHOIS database	listing Japanese IP networking points of contact	Domain Information: a. [Domain Name] SONY.CO.JP g. [Organization] Sony Corporation j. [Address] Shinagawa–ku, Tokyo, Japan
KRNIC WHOIS database	listing Korean IP networking points of contact	address: Korea Advanced Institute of Science and Technology address: Center for Information Research Computing address: Taejon, Republic of Korea e-mail: kyw@kaist.ac.kr

Table 1: Netfind Site Database Information Sources

9. References

[Ardo & Koch 1993]
A. Ardo and T. Koch. *Experiment with Semi-Automatic Classification of WAIS Sources.* Lund Univ., Sweden, July 1993. Posting in Forum #72 on Wide Area Information Servers and Electronic Publishing.

[Berners-Lee et al. 1992]
T. Berners-Lee, R. Cailliau, J. Groff and B. Pollermann. World-Wide Web: The Information Universe. *Electronic Networking: Research, Applications and Policy*, 2(1), pp. 52-58, Meckler Publications, Westport, CT, Spr. 1992.

[Berners-Lee 1993]
T. Berners-Lee. *Uniform Resource Locators.* CERN, July 1993. Internet Draft, IETF URL Working Group.

[Bowman et al. 1994a]
C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber and M. F. Schwartz. Harvest: A Scalable, Customizable Discovery and Access System. Tech. Rep., Dept. Comput. Sci., Univ. Colorado, Boulder, July 1994. In preparation.

[Bowman et al. 1994b]
C. M. Bowman, P. B. Danzig, U. Manber and M. F. Schwartz. Scalable Internet Resource Discovery: Research Problems and Approaches. To appear, Commun. ACM, 1994.

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

[Carl-Mitchell & Quarterman 1994]
S. Carl-Mitchell and J. S. Quarterman. *The Internet Connection: System Connectivity and Configuration.* Addison Wesley, Reading, MA, 1994. ISBN 0-201-54237-4.

[Dharap, Balay & Bowman 1993]
C. Dharap, R. Balay and M. Bowman. Type Structured File Systems. *Proc. Int. Workshop on Object-Orientation in Operating Systems*, IEEE Computer Society Press, Dec. 1993.

[Dorner & Pomes 1992]
S. Dorner and P. Pomes. *The CCSO Nameserver - A Description.* Computer and Communications Services Office, Univ. of Illinois at Urbana, Aug. 1992.

[Emtage & Deutsch 1992]
A. Emtage and P. Deutsch. Archie - An Electronic Directory Service for the Internet. *Proc. USENIX Wint. Conf.*, pp. 93-110, Jan. 1992.

[Foster 1992]
S. Foster. About the Veronica Service. Electronic bulletin board posting on the comp.infosystems.gopher newsgroup, Nov. 1992.

[Gifford et al. 1991]
D. K. Gifford, P. Jouvelot, M. A. Sheldon and J. W. O'Toole, Jr. Semantic File Systems. *Proc. 13th ACM Symp. Operating Syst. Prin.*, pp. 16-25, Oct. 1991.

[Hardy & Schwartz 1993]
D. Hardy and M. F. Schwartz. Essence: A Resource Discovery System Based on Semantic File Indexing. *Proc. USENIX Wint. Conf.*, pp. 361-374, Jan. 1993.

[Hardy & Schwartz 1994]
D. R. Hardy and M. F. Schwartz. Customized Information Extraction as a Basis for Resource Discovery. Tech. Rep. CU-CS-707-94, Dept. Comput. Sci., Univ. Colorado, Boulder, Mar. 1994. Submitted for publication.

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

[Kahle & Medlar 1991]
B. Kahle and A. Medlar. An Information System for Corporate Users: Wide Area Information Servers. *ConneXions - The Interoperability Report*, 5(11), pp. 2-9, Interop, Inc., Nov. 1991.

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

[Lottor 1992]
M. Lottor. Internet Growth (1981-1991). Req. For Com. 1296, Network Information Systems Center, SRI Int., Jan. 1992.

[McBride 1993]
J. McBride. *Update: Internet "Whitepages" Database of Addresses.* JS McBride & Co., Nov. 1993. Posting on Internet Commercialization/Privitization (com-priv) mailing list.

[McBryan 1994]
O. McBryan. GENVL and WWWW: Tools for Taming the Web. *Proc. 1st Int. World Wide Web Conf.*, CERN, Geneva, Switzerland, May 1994.

[McCahill 1992]
M. McCahill. The Internet Gopher: A Distributed Server Information System. *ConneXions - The Interoperability Report*, 6(7), pp. 10-14, Interop, Inc., July 1992.

[McLoughlin 1993]
L. McLoughlin. Mirroring Software. Available from ftp://src.doc.ic.ac.uk/packages/mirror, June 1993.

[Mills 1985] D. L. Mills. Network Time Protocol (NTP). Req. For Com. 958, M/A-COM Linkabit, Sep. 1985.

[Mockapetris 1987]
P. Mockapetris. Domain Names - Concepts and Facilities. Req. For Com. 1034, USC Information Sci. Institute, Nov. 1987.

[Postel 1982]
J. B. Postel. Simple Mail Transfer Protocol. Req. For Com. 821, USC Information Sci. Institute, Aug. 1982.

[Postel & Reynolds 1985]
J. Postel and J. Reynolds. File Transfer Protocol (FTP). Req. For Com. 959, USC Information Sci. Institute, Oct. 1985.

[Postel & Anderson 1994]
J. Postel and C. Anderson. White Pages Meeting Report. Req. For Com. 1588, USC Information Sci. Institute, Feb. 1994.

[Quarterman 1990]
J. S. Quarterman. *The Matrix: Computer Networks and Conferencing Systems Worldwide.* Digital Press, Bedford, MA, 1990.

[Rajendra, Jagannathan & Baum 1989]
V. Rajendra, D. Jagannathan and L. S. Baum, editors. *Blackboard Architectures and Applications.* Academic Press, Boston, 1989.

[Schroeder, Birrell & Needham 1984]
M. D. Schroeder, A. D. Birrell and R. M. Needham. Experience with Grapevine: The Growth of a Distributed System. *ACM Trans. Comput. Syst.*, 2(1), pp. 3-23, Feb. 1984.

[Schwartz & Tsirigotis 1991]
M. F. Schwartz and P. G. Tsirigotis. Experience with a Semantically Cognizant Internet White Pages Directory Tool. *J. Internetworking: Research and Experience*, 2(1), pp. 23-50, Mar. 1991.

[Schwartz et al. 1992]
M. F. Schwartz, A. Emtage, B. Kahle and B. C. Neuman. A Comparison of Internet Resource Discovery Approaches. *Computing Systems*, 5(4), pp. 461-493, Fall 1992.

[Schwartz & Quarterman 1993]
M. F. Schwartz and J. S. Quarterman. The Changing Global Internet Service Infrastructure. *Internet Research: Electronic Networking Applications and Policy*, 3(3), pp. 8-25, Fall 1993.

[Schwartz 1993]
M. F. Schwartz. Resource Discovery and Privacy. *Internet Society News*, 2(1), pp. 16-18, Spr. 1993.

[Schwartz 1994]
M. Schwartz. Netfind Support for URL-Based Search Customization. Technical Note, Dept. Comput. Sci., Univ. Colorado, Boulder. Available from ftp://ftp.cs.colorado.edu/pub/cs/distribs/netfind/Netfind.WP.URLs, Jan., 1994.

[Sheldon et al. 1994]
M. A. Sheldon, A. Duda, R. Weiss, J. W. O'Toole, Jr. and D. K. Gifford. Content Routing for Distributed Information Servers. *Proc. 4th Int. Conf. on Extending Database Technology*, Cambridge, England, Mar. 1994.

[Walker et al. 1983]

B. Walker, G. Popek, R. English, C. Kline and G. Thiel. The LOCUS Distributed Operating System. *Proc. 9th ACM Symp. Operating Syst. Prin.*, pp. 49-70, Oct. 1983.

[Weider, Fullton & Spero 1992]

C. Weider, J. Fullton and S. Spero. Architecture of the Whois++ Index Service. Internet Draft, WNILS Working Group, Nov. 1992. Available from ftp://nri.reston.va.us/internet-drafts/draft-ietf-wnils-whois-00.txt.

[Wood, Coleman & Schwartz 1993]

D. C. M. Wood, S. S. Coleman and M. F. Schwartz. Fremont: A System for Discovering Network Characteristics and Problems. *Proc. USENIX Wint. Conf.*, pp. 335-348, Jan. 1993.

[Zimmerman 1990]

D. Zimmerman. The Finger User Information Protocol. Req. For Com. 1288, Center for Discrete Mathematics and Theoretical Computer Science, Nov. 1990.