Scalable Internet Resource Discovery: Research Problems and Approaches

C. Mic Bowman
Transarc Corp.
The Gulf Tower
707 Grant Street
Pittsburgh, PA 15219
+1 412 338 6752
mic@transarc.com

Peter B. Danzig
Computer Science Department
University of Southern California
941 W. 37th Place
Los Angeles, California 90089-0781
+1 213 740 4780
danzig@usc.edu

Udi Manber
Computer Science Department
University of Arizona
Tucson, Arizona 85721
+1 602 621 4317
udi@cs.arizona.edu

Michael F. Schwartz
Computer Science Department
University of Colorado
Boulder, Colorado 80309-0430
+1 303 492 3902
schwartz@cs.colorado.edu

February 23, 1994

Abstract

Over the past several years, a number of information discovery and access tools have been introduced in the Internet, including Archie, Gopher, Netfind, and WAIS. These tools have become quite popular, and are helping to redefine how people think about wide-area network applications. Yet, they are not well suited to supporting the future information infrastructure, which will be characterized by enormous data volume, rapid growth in the user base, and burgeoning data diversity. In this paper we indicate trends in these three dimensions and survey problems these trends will create for current approaches. We then suggest several promising directions of future resource discovery research, along with some initial results from projects carried out by members of the Internet Research Task Force Research Group on Resource Discovery and Directory Service.

1 Introduction

In its roots as the ARPANET, the Internet was conceived primarily as a means of remote login and experimentation with data communication protocols. However, the predominate usage quickly became electronic mail in support of collaboration. This trend continues into the present incarnation of the Internet, but with increasingly diverse support for collaborative data sharing activities. Electronic mail has been supplemented by a variety of wide-area filing, information retrieval, publishing and library access systems. At present, the Internet provides access to hundreds of gigabytes each 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.

To make effective use of this wealth of information, users need ways to locate information of interest. In the past few years, a number of such *resource discovery* tools have been created, and have gained wide popular acceptance in the Internet [4,17,18,27,31,35,44].[1] Our goal in the current paper is to examine the impact of scale on resource discovery tools, and place these problems into a coherent framework. We focus on three scalability dimensions: the burgeoning diversity of information systems, the growing user base, and the increasing volume of data available to users.

Table 1 summarizes these dimensions, suggests a set of corresponding conceptual layers, and indicates problems being explored by the authors, who comprise the Internet Research Task Force (IRTF) Research Group on Resource Discovery and Directory Service. Users perceive the available information at the *information interface* layer. This layer must support scalable means of organizing, browsing, and searching. The *information dispersion* layer is responsible for replicating, distributing, and caching information. This layer must support access to information by a large, widely distributed user populace. The *information gathering* layer is responsible for collecting and correlating the information from many incomplete, inconsistent, and heterogeneous repositories.

The remainder of this paper covers these layers from the bottom up. Section 2 discusses problems of information system diversity. Section 3 discusses the problems brought about by growth in the user base. Section 4 discusses problems caused by increasing information volume. Finally, in Section 5 we offer a summary.

2 Information System Diversity

An important goal for resource discovery systems is providing a consistent, organized view of information. Since information about a resource exists in many repositories—within the object

[1]The reader interested in an overview of resource discovery systems and their approaches is referred to [43].

Scalability Dimension	Conceptual Layer	Problems	Research Focus
Data Volume	Information Interface	Information Overload	Topic Specialization; Scalable Content-Indexing;
User Base	Information Dispersion	Insufficient Replication; Manual Distribution Topology	Massive Replication; Access Measurements; Object Caching
Data Diversity	Information Gathering	Data Extraction; Low Data Quality	Operation Mapping; Data Mapping

Table 1: Dimensions of Scalability and Associated Research Problems

itself and within other information systems—resource discovery systems need to identify a resource, collect information about it from several sources, and convert the representation to a format that can be indexed for efficient searching.

As an example, consider the problem of constructing a user directory. In a typical environment, several existing systems contain information about users. The Sun NIS database [50] usually has information about a user's account name, personal name, address, group memberships, password, and home directory. The `ruserd` [32] server has information about a user's workstation and its idle time. In addition, users often place information in a ".plan" file that might list the user's travel schedule, home address, office hours, and research interests.

As this example illustrates, information in existing Internet repositories has the following characteristics:

- It is heterogeneous.

  Each repository maintains the information it needs about resources. For example, the primary purpose of the NIS database is to maintain information about user accounts, while a user's ".plan" file often contains more personal information. In addition, the two repositories represent the information differently: records in an NIS database have a fixed format, but a ".plan" file contains unstructured text.

- It is inconsistent.

  Most information contained in Internet repositories is dynamic. Certain properties change frequently, such as which workstations a person is using. Other properties change more slowly, such as a user's mail address. Because information is maintained by several repositories that perform updates at different times using different algorithms, there will often be conflicts between information in the various repositories. For example, information about account name, address, and phone number may be maintained by both the NIS database and an X.500 [26] server. When a user's address or phone number changes, the X.500 service will probably be updated first. However, if the account changes, the NIS database will usually be the first to reflect the change.

- It is incomplete.

  Additional attributes of a resource can often be obtained by combining information from several repositories. For example, a bibliographic database does not contain explicit information about a person's research interests. However, keywords might be extracted from the person's research papers, to infer research interests for a user directory.

There are two common approaches to these information gathering layer problems. The first approach—*data mapping*—generates an aggregate repository from multiple information sources. The second approach—*operation mapping*—constructs a "gateway" between existing systems, which maps the functionality of one system into another without actually copying the data. Below we discuss these approaches, and our research efforts for each.

2.1 Data Mapping

The first approach for accommodating diversity is to collect data from a set of underlying repositories, and combine it into a homogeneous whole. Doing so involves two parts: mapping algorithms for collecting information, and agreement algorithms for correlating information [7].

A mapping algorithm is implemented as a function that collects information from a repository and reformats it. There may be several implementations of mapping algorithms, each customized for an existing repository. The most common mapping algorithms are implemented as clients of an existing service. For example, Netfind extracts user information from several common Internet services [46], including the Domain Naming System (DNS) [33], the finger service [53] and the Simple Mail Transfer Protocol [40].

The agreement algorithm defines a method for handling conflicts between different repositories. For example, Figure 1 illustrates data for the Enterprise [6] user directory system, which is built on top of the Univers name service [7]. This figure shows three mapping algorithms that gather information from the NIS database, the ruserd server, and the user's electronic mail, respectively. Several attributes can be generated by more than one mapping algorithm. For example, address information potentially exists in both the NIS database and the information supplied by the user. The agreement algorithm considers information gathered directly from the user as the most reliable. Depending on the attribute, the agreement algorithm may permit some properties to have several values, such as the two address attributes that describe the user in Figure 1.

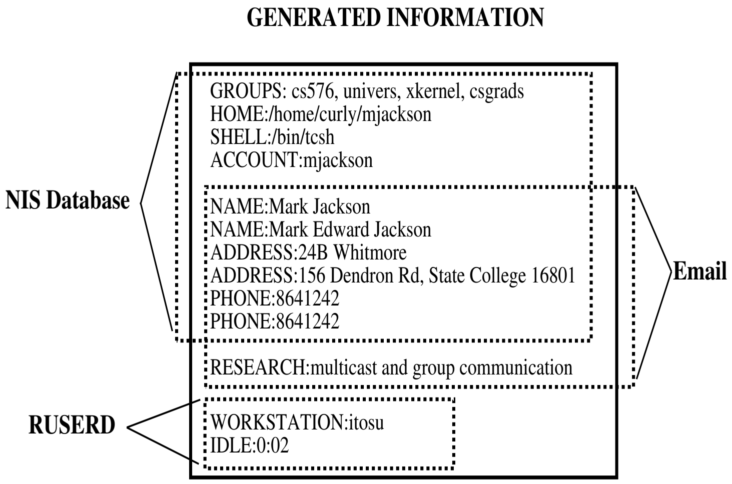

Figure 1: Example Mapping Algorithms for User Directory Information

Data mining represents a special form of agreement algorithm, which works by cross-correlating information available from multiple repositories. This can have two benefits. First, it can flag inconsistencies. For example, Enterprise could inform users if it detected conflicts between the electronic mail addresses listed in different repositories. Second, data mining can deduce implicit information by cross-correlating existing information. For example, Netfind continuously collects and cross-correlates data from a number of sources, to form a far-reaching database of Internet sites. One source might discover a new host called "astro.columbia.edu" from DNS traversals, and cross-correlate this information with the existing site database record for columbia.edu ("columbia university, new york, new york"), its understanding of the nesting relationships of the Domain name space, and a database of departmental abbreviations, to derive a new record for the Astronomy Department at Columbia University.

Mapping and agreement algorithms generally operate best when they exploit the semantics of specific resource discovery applications. In the above Netfind example this was possible because the data were gathered from particular services, each with semantics that Netfind understands.

More generally, data gathering depends on some data type structure to help select semantically-specific gathering methods. We are exploring a variety of data typing approaches in the context of gathering file system data. We now briefly consider the problems that arise in typing and gathering this data.

To gather information effectively from file system data, it is helpful to extract information in different ways depending on file type. For example, while it is possible to index every token in program source code, the index will be smaller and more useful if it distinguishes the variables and procedures defined in the file. In contrast, applying this data gathering algorithm to a program's associated textual documentation will not yield the most useful information, as it has a different structure. By typing data, information can be extracted in whichever way is most appropriate for each type of file.

File data can be typed explicitly or implicitly. Explicitly typing files has the advantage that it simplifies data gathering. An explicitly typed file conforms to a well known structure, which supports a set of conventions for what data should be extracted. Explicit typing is most naturally performed by the user when a file is exported into the resource discovery system. We are exploring this approach in the Nebula file system [16] and Indie discovery tool [12].

Many files exist without explicit type information, as in most current anonymous FTP[2] files. An implicit typing mechanism can help in this case. For example, Essence [24] uses a variety of heuristics to recognize files as fitting into certain common classes, such as word processor source text or binary executables. These heuristics include common naming conventions or identifying features in the data. The MIT Semantic File System uses similar techniques [22].

Given a typed file, the next step is to extract indexing information. This can most easily be accomplished through automatic content extraction, using a grammar that describes how to extract information from each type of file. For example, given a TeX word processing document, the grammar could describe where to extract author, title, and abstract information. For cases where more complex data extraction methods are needed, one can provide an "escape" mechanism that allows arbitrary data extraction programs to be run.

Automatic extraction methods have the advantage that they can provide an immediate base of usable information, but in general will generate some inappropriate keywords and miss generating other, desirable keywords. For this reason, it is prudent to augment these methods with means by which people can manually override the automatically extracted data.

In many cases file indexing information can be extracted from each file in isolation. In some cases, however, it is useful to apply extraction procedures based on the relationships between files. For example, binary executable files can sometimes be indexed by gathering keywords from their corresponding documentation. One can use heuristics to exploit such implicit inter-file relationships for common cases, augmented by means of specifying explicit relationships. For example, we are exploring an approach that allows users to create files in the file system tree that specify relationships among groups of files.

One final observation about data type structure is that the index should preserve type information to help identify context during searches. For example, keywords extracted from document

[2] FTP is an Internet standard protocol that supports transferring files between interconnected hosts [39]. 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.

titles can be tagged in the index so that a query will be able to specify that only data extracted from document titles should match the query. This stands in contrast to the common approach (used by WAIS [27], for example) of allowing a free association between query keywords and extracted data.

We discuss indexing schemes further in Section 4.2.

2.2 Operation Mapping

A gateway between two resource discovery systems translates operations from one system into operations in another system. Ideally, the systems interoperate seamlessly, without the need for users to learn the details of each system. Sometimes, however, users must learn how to use each system separately.

Building seamless gateways can be hindered if one system lacks operations needed by another system's user interface [43]. For example, if would be difficult to provide a seamless gateway from a system (like WAIS) that provides a search interface to users, to a system (like Prospero [35]) that only support browsing. Even if two systems support similar operations, building seamless gateways may be hindered by another problem: providing appropriate mappings between operations in the two systems. To illustrate the problem, consider the current interim gateway from Gopher [31] to Netfind, illustrated in Figure 2.[3] Because the gateway simply opens a telnet window to a UNIX program that provides the Netfind service, users perceive the boundaries between the two systems.

In contrast, we have built a system called Dynamic WAIS [25], which extends the WAIS paradigm to support information from remote search systems (as opposed to the usual collection of static documents). The prototype supports gateways from WAIS to Archie and to Netfind, using the Z39.50 information retrieval protocol [2] to seamlessly integrate the information spaces. The Dynamic WAIS interface to Netfind is shown in Figure 3.

The key behind the Dynamic WAIS gateways is the conceptual work of constructing the mappings between the WAIS search-and-retrieve operations, and the underlying Archie and Netfind operations. In the case of Netfind, for example, when the Dynamic WAIS user requests a search using the "dynamic-netfind.src" WAIS database, the search is translated to a lookup in the Netfind site database, to determine potential domains to search. The Netfind domain selection request is then mapped into a WAIS database selection request (the highlighted selection in the XWAIS Question window). Once the user selects one of the domains to search, the WAIS retrieval phase is mapped into an actual domain search in Netfind (the uppermost window).

We are developing these techniques further, to support gateways to complex forms of data, such as scientific databases.

3 User Base Scale

New constituencies of users will make the Internet grow significantly beyond its present size of 2 million nodes. This growth will overburden the network's resource discovery services unless we address four problems of scale. First, discovery services should monitor data access patterns to determine how best to replicate themselves, to determine whether to create specialized services that manage hot subsets of their data, and to diagnose accidentally looping clients. Second, discovery

[3] Efforts are under way to improve this gateway.

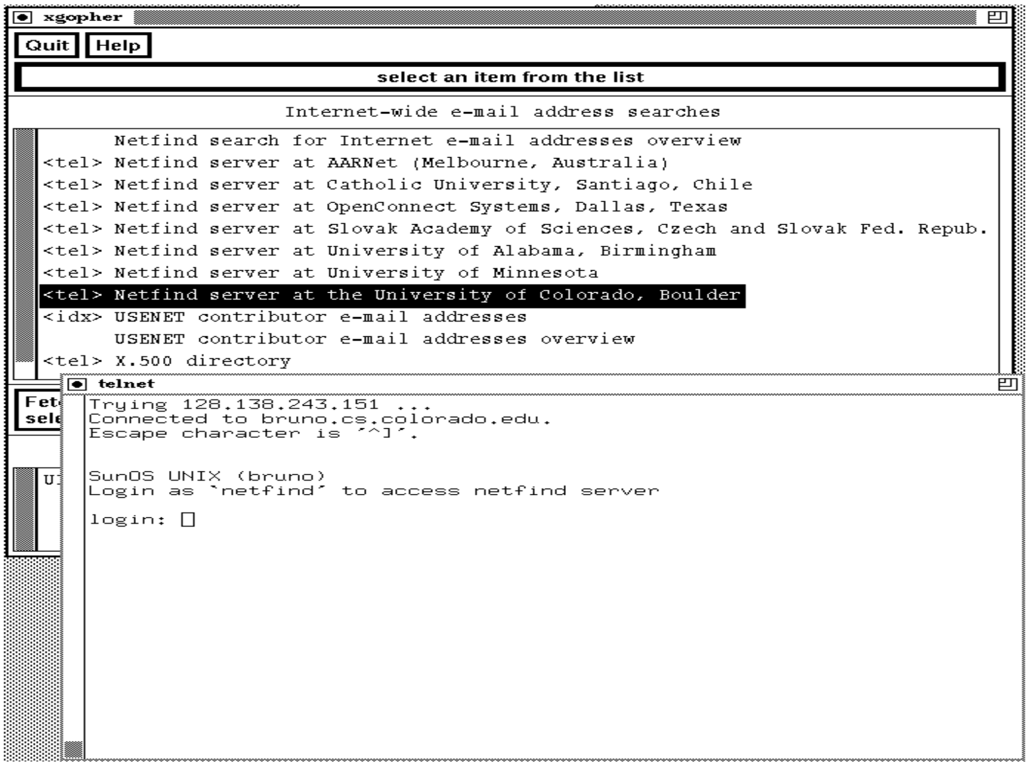

Figure 2: Gopher Menu-Level Gateway to Netfind

services should support significantly higher levels of replication, using algorithms specifically designed to function in the Internet's dynamic patchwork of autonomous systems. Third, the range of user expertise and needs require that user interfaces and search strategies be highly customizable. Fourth, the Internet will need a hierarchically structured, extensible, object caching service through which clients can retrieve data objects, once they're discovered. We consider these issues below.

3.1 Server Instrumentation

The designers of new information systems can never fully anticipate how their systems will be used. For this reason, we believe that new Internet services should instrument their query streams.

Self-instrumented servers could help determine where to place additional replicas of an entire service: If some X.500 server in Europe finds that half of its clients are in the United States, the server itself could suggest that a strategically located replica be created.

Self-instrumented servers could also identify the access rate of items in their databases for use by more specialized services. For example, an instrumented Archie server would note that properly formed user queries only touch about 16% of Archie's database. Such instrumentation would enable the creation of a complementary service that reported only popular, duplicate free, or nearby objects. We discuss these ideas more in Section 4.2.

Self-instrumented servers could also identify server-client or server-server communication run amok. Large distributed systems frequently suffer from undiagnosed, endless cycle of requests. For

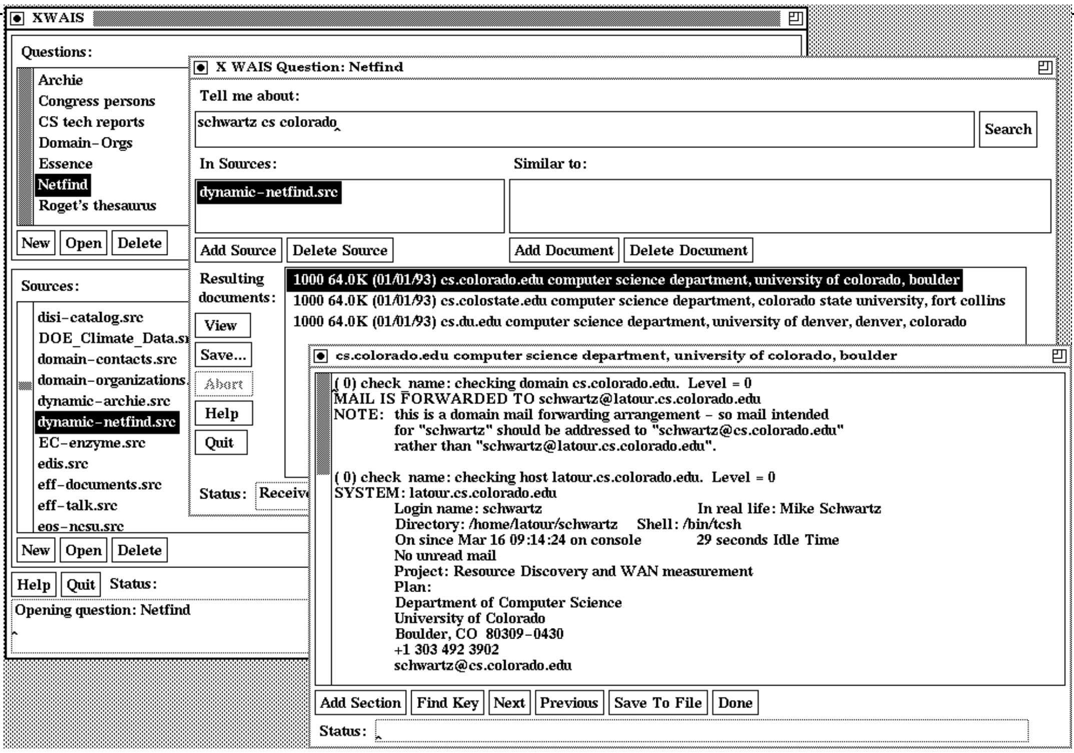

Figure 3: Dynamic WAIS Information-Level Gateway to Netfind

example, self-instrumented Internet name servers show that DNS traffic consumes 20 times more bandwidth than it should, because of unanticipated interactions between clients and servers [13]. Self-instrumentation could identify problem specifications and implementations before they become widespread and difficult to correct.

3.2 Server Replication

Name servers scale well because their data are typically partitioned hierarchically. Because resource discovery tools search flat, rather than hierarchical or otherwise partitionable views of data, the only way to make these tools scale is by replicating them. To gain an appreciation for the degree of replication required, consider the Archie file location service.

The global collection of Archie servers currently process approximately 100,000 queries per day, generated by a few thousand users worldwide. Every month or two of Internet growth requires yet another replica of Archie. Thirty Archie servers now replicate a continuously evolving 150 MB database of 2.1 million records. While a query posed on a Saturday night receives a response in seconds, it can take several hours to answer the same query on a Thursday afternoon. Even with no new Internet growth, for the current implementation of Archie to yield five second response times during peak hours we would need at least sixty times more Archie servers - i.e., 1800 servers. Because of its success and the continual rapid growth of the Internet, in time Archie will require thousands of replicas. Other successful tools that cannot easily partition their data will also require massive replication.

We believe massive replication requires additional research. On the one hand, without doubt we know how to replicate and cache data that partitions easily, as in the case of name servers [5, 33]. Primary copy replication works well because name servers do not perform associative access, and organizational boundaries limit the size of a domain, allowing a handful of servers to meet performance demands.[4] We have learned many lessons to arrive at this understanding [13, 34, 42]. On the other hand, we have little experience deploying replication and caching algorithms to support massively replicated, flat, yet autonomously managed databases.

What do existing replication schemes for wide-area services lack? First, existing replication systems ignore network topology. They do not route their updates in a manner that makes efficient use of the Internet. One day, some nascent, streaming reliable multicast protocol might serve this purpose. Today, we believe, it is necessary to calculate the topology over which updates traverse and to manage replication groups that exploit the Internet's partitioning into autonomous domains.

Second, existing schemes do not guarantee timely and efficient updates in the face of frequent changes in physical topology, network partition, and temporary or permanent node failure. In essence, they treat all physical links as having equal bandwidth, delay, and reliability, and do not recognize administrative domains.

We believe that flooding-based replication algorithms can be extended to work well in the Internet environment. Both Archie and network news [41] replicate using flooding algorithms. However, for lack of good tools, administrators of both Archie and network news manually configure the flooding topology over which updates travel, and manually reconfigure this topology when the physical network changes. This is not an easy task because Internet topology changes often, and manually composed maps are never current. While we are developing tools to map the Internet [51], even full network maps will not automate topology-update calculation.

Avoiding topology knowledge by using today's multicast protocols [3, 15] for data distribution fails for other reasons. First, these protocols are limited to single routing domains or require manually placed tunnels between such domains. Second, Internet multicast attempts to minimize message delay, which is the wrong metric for bulk transport. At the very least, we see the need for different routing metrics. Third, more research is needed into *reliable*, bulk transport multicast that efficiently deals with site failure, network partition, and changes in the replication group.

We are exploring an approach to providing massively replicated, loosely consistent services [12, 37]. This approach extends ideas presented in Lampson's Global name service [28] to operate in the Internet. Briefly, our approach organizes the replicas of a service into groups, imitating the idea behind the Internet's autonomous routing domains. Group replicas estimate the physical topology, and then create an update topology between the group members. The left hand side of Figure 4 shows three replication domains. Inside each replication domain, a logical update topology is established that best uses the physical topology connecting the replicas. Whenever the algorithm detects a meaningful change in physical topology between members of a replication domain, it modifies the logical update topology accordingly. Because it does not require a separate recovery algorithm, it is simpler than solutions based on Internet multicast.

[4] Because it is both a name service and a discovery tool, X.500 could benefit from massive replication.

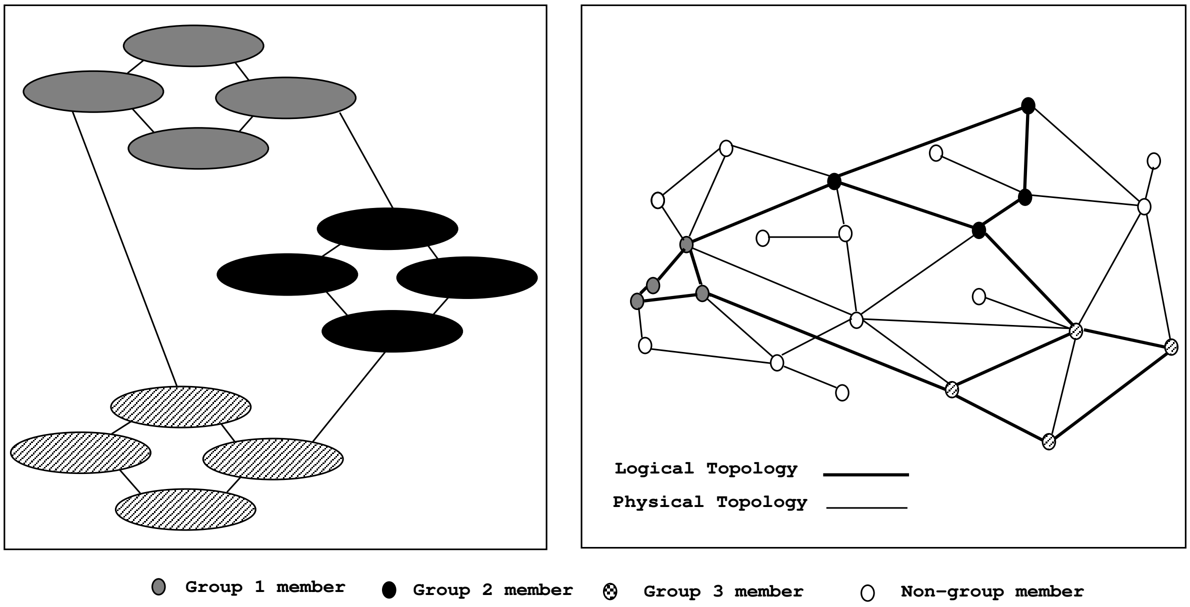

Figure 4: Replication domains and physical versus logical update topology

3.3 Data Object Caching

We believe that the Internet needs an object caching service, through which search clients can retrieve the data objects they discover. We say this for two reasons.

First, people currently use FTP and electronic mail as a cacheless, distributed file system [14, 38]. Since the quantity and size of read-mostly data objects grows as we add new information services, an object cache would improve user response times and decrease network load (e.g., we found that one fifth of all NSFNET backbone traffic could be avoided by caching FTP data). Second, caches protect the network from looping client programs that repeatedly retrieve the same object. While the cache does not fix the faulty components, it does isolate the fault. A caching service would obviate the need for every new client and Internet service to implement its own data cache.

We believe the object cache should be hierarchically organized, as illustrated in Figure 5. The dark ovals in this figure represent file caches residing on secure machines placed near the entry points to regional networks and near the core of backbone networks. The organization of these caches could be similar to the organization of the Domain Name System. Clients would send their requests to one of their default cache servers. If the request missed the cache, the cache would recursively resolve the request with one of its parent caches, or directly from the FTP archive.

3.4 Client Customization

As the number of Internet information systems users gets larger, it naturally becomes more diverse. Allowing users flexible customization can be a great help, because people have different search styles and needs. To see how this can be done, consider the analogy of a newcomer to a town. One first establishes general acquaintance with the town layout and major services. One then learns about services close to one's heart - for example, clubs, specialized stores, and recreation facilities - usually

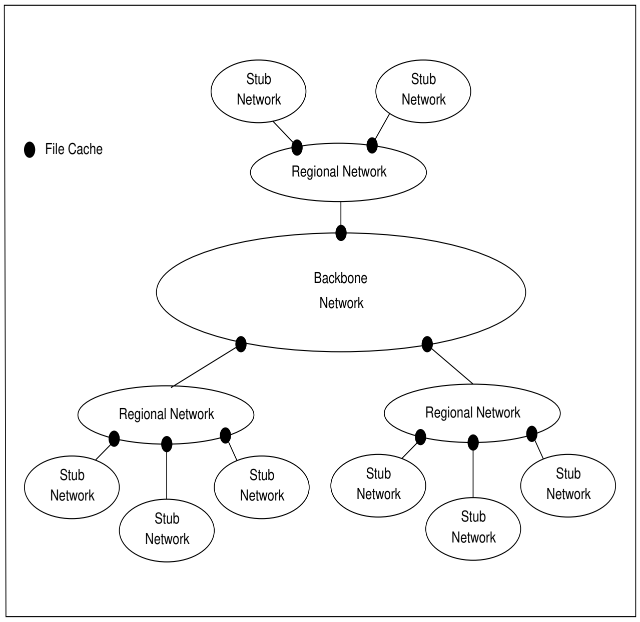

Figure 5: File caches organized hierarchically by network topology

by word of mouth, the media, or advertizing. After a while, one develops networks of friends, better knowledge of different services, and experience based on habits and interests. This is a continuing process, because new facilities appear, and one's interests evolve. The same process occurs on a much larger scale in the Internet, and suggests ways that interactions between users and discovery services can be made flexible. Below we discuss three types of customization that can be addressed in discovery tools, based on some of our experimental systems.

The first type of customization involves tracking a person's search history. For example, recently one of this paper's authors discovered that the New Republic magazine was available online while browsing Gopherspace via Mosaic [1]. But only two days later it took him 15 minutes to navigate back to the same place. To address this problem, the user interface can keep track of previous successful and unsuccessful queries, comments a user made on past queries, and browsing paths. Existing systems record some of this information (e.g., the ability to set "bookmarks" in Gopher and "Hotlists" in Mosaic); saving the whole history can help further. For example, we built a system that records the paths users traverse when browsing FTP directories, and allows this information to be searched [45]. It is also helpful to record search history, and allow the searches themselves to be searched. For example, the X-windows interface to agrep [52] translates every command into the corresponding agrep statement, and allows flexible retrieval of this information. This type of history can support queries such as "What was the name of the service that allowed me to search for XYZ that I used about a year ago?"

The second type of customization is the ability to choose not only according to topic but also according to context and other attributes. If the search is for papers on induction, for example, knowledge of whether the searcher is a mathematician or an electrical engineer can be very useful. When one looks for a program named ZYX using Archie, for example, it would be useful to specify a preference for, say, a UNIX implementation of the program.

The third type of customization is ranking. WAIS provides one form of ranking, in which matched documents are ordered by frequency of occurrence of the specified keywords. However, it would be useful to allow users to customize their notion of ranking, so that they could choose to rank information by quality or reliability. Clearly, these notions are very subjective. If we are naive users, we may want information from someone who knows how to build easy-to-use systems; if we are academics, we may want information from someone with deep understanding; if we absolutely positively need it by tomorrow, we want someone who can deliver, and so on. We believe that in time the Internet will support many types of commercial and non-profit review services, and people will subscribe and follow recommendations from reviewers they trust (just like they go to movies or buy refrigerators based on reviews).

Customizations like the ones discussed here are missing from most current resource discovery client programs because they were not designed to be extensible (like the Emacs text-editor[47]).

4 Data Volume

The amount of information freely available in the Internet is huge and growing rapidly. New usage modes will contribute additional scale. For example, as multimedia applications begin to proliferate, users will create and share voluminous audio, image, and video data, adding several orders of magnitude of data. Many users already store voluminous data, but do not share it over the Internet because of limited wide-area network bandwidths. For example, earth and space scientists collect sensor data at rates as high as gigabytes per day [21]. To share data with colleagues, they send magnetic tapes through the postal mail. As Internet link capacities increase, more scientists will use the Internet to share data, adding several more orders of magnitude to the information space.

While resource discovery tools must deal with this growth, the scaling problems are not quite as bad as they may seem, because the number and size of "searchable items" need not grow as fast. Resource discovery tools may be needed to find the existence and location of gigabytes or terabytes of raw sensor data, but probably they will not search or otherwise process all this data. A reasonably small-sized descriptor object will be sufficient to point anyone to this data. Only this descriptor object will need to be searched and indexed. The same holds for sound, video, and many other types of non-textual data. Searching image files is desirable, but current pattern matching techniques are still too slow to allow large-scale image processing on-the-fly. Again, a descriptor object can be associated with every image, describing it in words.

Below we discuss three aspects of data volume scalability: user interaction paradigms, indexing schemes, and services specialized to support particular topics and user communities.

4.1 User Interaction Paradigms: Browsing vs. Searching

Loosely speaking, there are two resource discovery paradigms in common use in the Internet: organizing/browsing, and searching. *Organizing* refers to the human-guided process of deciding how to interrelate information, usually by placing it into some sort of a hierarchy (e.g., the hierarchy of directories in an FTP file system). *Browsing* refers to the corresponding human-guided activity of exploring the organization and contents of a resource space. *Searching* is a process where the user provides some description of the resources being sought, and a discovery system locates information that matches the description.

We believe that a general discovery system will have to employ a combination of both paradigms. Let's analyze the strengths and weaknesses of each paradigm. The main weakness of organizing is that it is typically done by "someone else" and it is not easy to change. For example the Library of Congress Classification System has *Cavalry* and *Minor Services of Navies* as two second level topics (under "Military Science" and "Naval Science" respectively), each equal to all of Mathematical Sciences (a second level topic under "Science"), of which computer science is but a small part.[5] Ironically, this is also the main strength of organizing, because people prefer a fixed system that they can get used to, even if it is not the most efficient.

Browsing also suffers from this problem because it typically depends heavily on the quality and relevance of the organization. Keeping a large amount of data well organized is difficult. In fact, the notion of "well organized" is highly subjective and personal. What one user finds clear and easy to browse may be difficult for users who have different needs or backgrounds. Browsing can also lead to navigation problems, and users can get disoriented [10, 23]. To some extent this problem can be alleviated by systems that support multiple views of information [9, 35]. Yet, doing so really pushes the problem "up" a level—users must locate appropriate views, which in itself is another discovery problem. Moreover, because there are few barriers to "publishing" information in the Internet (and we strongly believe there should not be any), there is a great deal of information that is useful to only very few users, and often for only a short period of time. To other users, this information clutters the "information highway", making browsing difficult.

Searching is much more flexible and general than organizing/ browsing, but it is also harder for the user. Forming good queries can be a difficult task, especially in an information space unfamiliar to the user. On the other hand, users are less prone to disorientation, the searching paradigm can handle change much better, and different services can be connected by searching more easily than by interfacing their organizations.

Many current systems, such as WAIS, Gopher, and WorldWideWeb [4], employ an organization that is typically based on the location of the data, with limited per-item or per-location searching facilities. Browsing is the first paradigm that users see[6], but once a server or an archive is located, some type of searching is also provided. Searching can be comprehensive throughout the archive (for example, WAIS servers provide full-text indexes), or limited to titles.

4.2 Indexing Schemes

The importance of searching can be seen by the recent emergence of file system search tools [22, 24, 30, 48] and the Veronica system [20] (a search facility for Gopher). Moreover, it is interesting to note that while the Prospero model [36] focuses on organizing and browsing, Prospero has found its most successful application as an interface to the Archie search system.

To support efficient searching, various means of indexing data are required. As illustrated in Figure 6, Internet indexing tools can be placed on a spectrum of indexing space vs. representativeness. The upper left corner of this figure is occupied by systems that achieve very space efficient indexes, but only represent the names of the files or menus that they index. Archie and Veronica, for example, index the file and menu names of FTP and Gopher servers, respectively. The compact nature of these indexes mean that a single index can support far-reaching searches.

[5] There was a time, of course, when the study of cavalry was much more important than the study of computers.

[6] WAIS supports searching at the top level via the directory of servers, but many users simply browse a local copy of this server list.

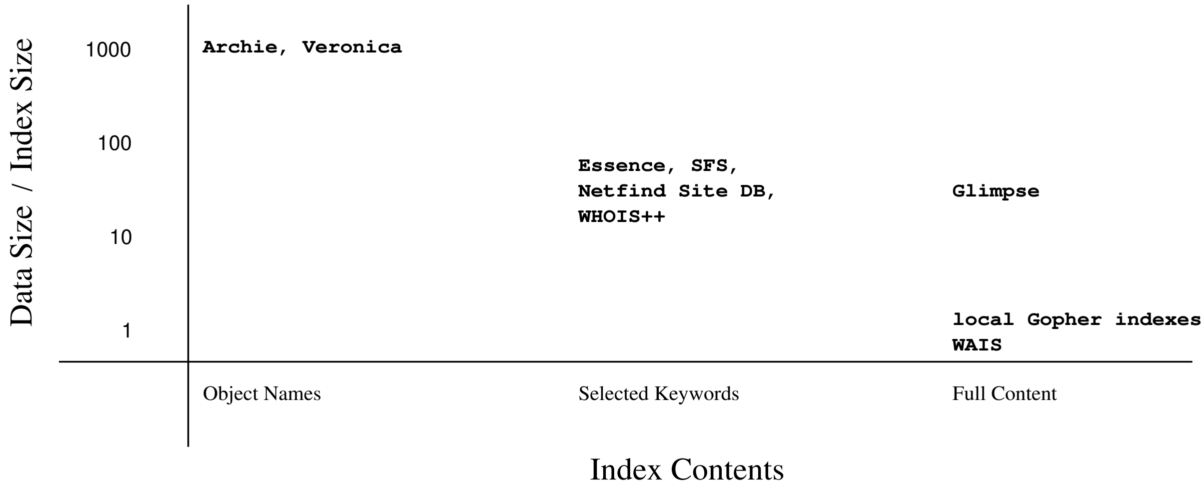

Figure 6: Indexing Space vs. Representativeness Spectrum

Yet, these indexes support very limited queries. For example, Archie supports only name-based searches; searches based on content are possible only when the file names happen to reflect some of the contents.

The lower right corner Figure 6, is occupied by systems that provide full-text indexing of the data found at individual sites. For example, a WAIS index represents every keyword in a set of documents located at a single site. Similar indexes are available for individual Gopher and WWW servers. Some recent advances have lowered the space requirements for full-text indexing, with as low as 2-4% for the index used in Glimpse [30]. There is usually (but not always) a time-space tradeoff; systems that use less space require more time for searching.

The middle region of Figure 6 is occupied by systems that represent some of the contents of the objects they index, based on selection procedures for including important keywords or excluding less meaningful keywords. For example, Essence and MIT's Semantic File System (SFS) select keys based on application-specific semantics of individual documents (e.g., locating the author lines within TeX documents). The Netfind site database includes keywords for each component of a site's Domain name, plus each keyword included in an organizational description that was constructed based on understanding the semantics of many information sources used to gather that information [46]. Whois++ [49] indexes templates that were manually constructed by site administrators wishing to describe the resources at their site.

By selecting relatively small but important information such as file names, Archie quickly became a popular search-based Internet resource discovery tool. But as we discussed in Section 3.2, Archie has scale problems. One way to overcome some of these problems is to introduce some hierarchy into the indexing mechanism in Archie (and similar tools). In addition to one flat database of all file names, it is possible to maintain much smaller slightly limited databases that will be replicated widely. For example, we can detect duplicates (exact and/or similar, see [29]), and keep only one copy (e.g., based on locality) in the smaller databases. Most queries will be satisfied with the smaller databases, and only few of them will have to go further.

The scale problems for full texts are much more difficult. Full-text indexes are almost always inverted indexes, which store pointers to every occurrence of every word (possibly excluding some very common words). The main problems with inverted indexes are their size – usually 50-150% of the original text[19] – and the time it takes to build and/or update them. Therefore, maintaining an inverted index of the whole Internet FTP space is probably out of the question. But separate local indexes, which is what we have now, do not present a general enough view of much of the available information. Users often need to perform a lengthy browsing session to find the right indexes, and they often miss. This problem will only get worse with scale.

We envision a search system that connects local indexes and other pieces of information via a multi-level indexing scheme. beyond the scope of this paper. The main principle behind such a scheme is that the number of search terms is quite limited no matter how much data exists. Search terms are mostly words, names, technical terms, etc, and the number of those is on the order of 10^6 to 10^7, but more importantly, this number grows more slowly than the general growth of information. Inverted indexes require enormous space because they catalog all occurrences of all terms. But if we index, for each term, only pointers to places that may be relevant, we end up with a much smaller index. Searching will be similar to browsing in the sense that the result of each query may be a list of suggestions for further exploration.

A big advantage of such a scheme is that the index can be partitioned in several ways, which makes it scalable. Specialized archives will, of course, keep their local indexes. Several local indexes can be combined to form a two-level index, in which the top level can only filter queries to a subset of the local indexes. For example, separate collections of technical reports can form a combined index. Indexes can also be combined according to topics or other shared attributes. The directory of services could be another index (but more widely replicated), which contains pointers to information about common terms and local information. There could also be a more detailed directory maintained at some servers with knowledge about more terms. Users could navigate by a combination of browsing and searching. In contrast with fixed browsing, this type of navigation will allow users to skip many levels, to better customize their searches, and to more easily combine information from different sources. Of course, many issues need to be resolved, including ranking, classification, replication, consistency, data extraction, privacy, and more.

An example (at a smaller scale) of the multi-level approach is *Glimpse* [30], an indexing and searching scheme designed for file systems. Glimpse divides the entire file system into blocks, and in contrast with inverted indexes, it stores for each word only the block numbers containing it. The index size is typically only 2-4% of the text size (hence its position in Figure 6). The search is done first on the index and then on the blocks that match the query. Glimpse supports approximate matching, regular expression matching, and many other options. Other examples of similar approaches include the scatter/gather browsing approach [11], the Alex file system's "archia" tool [8], and Veronica.

Essence and the MIT Semantic File System do selective indexing of documents, by selecting keywords based on knowledge of the structure of the documents being indexed. For example, Essence understands the structure of several common word processing systems (as well as most other common file types in UNIX environments), and uses this understanding to extract authors, titles, and abstracts from text documents. In this way, it is able to select fewer keywords for the index, yet retain many of the keywords that users would likely use to locate documents. Because these types of systems exploit knowledge of the structure of the documents being indexed, it would be possible to include document structure information in the index. (This idea was discussed in Section 2.)

4.3 Topic Specialization, Classification and Information Quality

It is safe to say that at least 99% of the available data is of no interest to at least 99% of the users. The obvious solution to this problem is to construct specialized archives for particular domains of interest. We believe that new techniques are needed to simplify the process of managing specialized archives.

Currently, specialized archives rely on 1) direct contributions from their communities, and 2) administrators who contribute time and expertise to keep the collections well structured. Except for replication (or *mirrors* as they are usually called in the Internet) of FTP files, archives do not cooperate among themselves.

We see the need for a discovery architecture and set of tools that easily let people create specialized services and that automatically discover information from other services, summarize it, keep it consistent, and present it to the archive administrator for his/her editorial decision. An important component of any complete solution is a directory of services in which archives describe their interest specialization and keep this definition current.

This approach essentially amounts to defining archives in terms of queries [6, 12]. A server periodically uses its query to hunt throughout the Internet for relevant data objects. One type of server could, for example, be specialized to scan FTP archives, summarizing files and making these summaries available to yet other services.

Such an architecture could greatly reduce the manual steps that archive administrators currently perform to incorporate users' contributions. This architecture would also help users discover smaller, highly specialized archives.

To help support topic-specialized servers, we believe information must also be classified according to topic- or community-specific taxonomies. For example, an archive of Computer Science technical reports might be classified using the ACM Computing Reviews taxonomy. Taxonomies allow a more uniform search space than possible solely by content-indexing documents. Because a particular document may be of value to several communities, it should be possible to classify documents according to multiple taxonomies, and register classification information for the document in several specialized archives.

Classification should also include some description of information quality. For example, scientists often need to understand the methods by which data were gathered, what error controls were used, etc. At this point it is not clear how to specify quality, because there are many associated issues. At a minimum it would be useful to record the author, data collection process, and something about the review process to which the data were subjected before being published. Other possibilities might include pointers to conflicting points of view, support for "voting" by users who have perused the data, etc.

We believe tools should be developed that allow authors to mark up their documents with classification terms from some selected set of taxonomies. In turn these classification data should be included in topic indexes, with attributes indicating that the source of the information was a classification process, rather than just a content-indexing process (since the latter tends to generate many less well-focused or well-defined terms).

5 Summary

The Internet's rapidly growing data volume, user base, and data diversity will create difficult problems for the current set of resource discovery tools. Future tools must scale with the diversity of information systems, number of users, and size of the information space.

With growing information diversity, techniques are needed to gather data from heterogeneous sources and sort through the inherent inconsistency and incompleteness. Internet Research Task Force efforts in this realm focus on application-specific means of extracting and cross-correlating information, based on both explicit and implicit data typing schemes.

With a growing user base, significantly more load will be placed on Internet links and servers. This load will require much more heavily replicated servers and more significant use of data caching; highly customizable client interfaces; and self-instrumenting servers to help guide replication and specialization. IRTF efforts in this realm focus on flooding-based replication algorithms that adapt to topology changes, and on customized clients.

As the volume of information continues to grow, organizing and browsing data break down as primary means for supporting resource discovery. At this scale, discovery systems will need to support scalable content-based search mechanisms. Current systems tend to strike a compromise between index representativeness and space efficiency. Future systems will need to support indexes that are both representative and space efficient. IRTF efforts in this realm focus on scalable content-based searching algorithms, and on servers specialized to support particular user communities.

Acknowledgements

Bowman is supported in part by the National Science Foundation under grants CDA-8914587 and CDA-8914587AO2, the Advanced Research Projects Agency under contract number DABT63-93-C-0052, and an equipment grant from Sun Microsystems, Inc.

Danzig is supported in part by the Air Force Office of Scientific Research under Award Number F49620-93-1-0082, and by the Advanced Research Projects Agency under contract number DABT63-93-C-0052.

Manber is supported in part by the National Science Foundation under grant numbers CCR-9002351 and CCR-9301129, and by the Advanced Research Projects Agency under contract number DABT63-93-C-0052.

Schwartz is supported in part by the National Science Foundation under grant numbers NCR-9105372 and NCR-9204853, the Advanced Research Projects Agency under contract number DABT63-93-C-0052, and an equipment grant from Sun Microsystems' Collaborative Research Program.

We thank Alan Emtage for providing us with the Archie logs that led to some of the results in Section 3.1. Panos Tsirigotis implemented the software needed for analyzing this data.

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.

Author Information

C. Mic Bowman is a Member of Technical Staff at Transarc Corporation. He received his Ph.D in Computer Science from the University of Arizona in 1990. From 1990 to 1994 he was an assistant professor at the Pennsylvania State University. His research interests include descriptive naming systems for local and wide-area file systems, structured file systems, resource discovery, and protocols for remote computing. Bowman can be reached at mic@transarc.com.

Peter B. Danzig is an Assistant Professor of Computer Science at the University of Southern California. He received his Ph.D in Computer Science from the University of California, Berkeley in 1990. His current research addresses on the measurement and performance debugging of Internet services, distributed system architectures for resource discovery, and mathematical modeling of communication networks. Danzig can be reached at danzig@usc.edu.

Udi Manber is a Professor of Computer Science at the University of Arizona. He received his Ph.D in Computer Science from the University of Washington in 1982. His research interests include design of algorithms, pattern matching, computer networks, and software tools. Manber can be reached at udi@cs.arizona.edu.

Michael F. Schwartz is an Assistant Professor of Computer Science at the University of Colorado. He received his Ph.D in Computer Science from the University of Washington in 1987. His research focuses on issues raised by international networks and distributed systems, with particular focus on resource discovery and wide-area network measurement. Schwartz can be reached at schwartz@cs.colorado.edu.

References

[1] Marc Andreessen. NCSA Mosaic technical summary. Technical report, National Center for Supercomputing Applications, MAY 1993.

[2] ANSI. *ANSI Z39.50.* American National Standards Institute, May 1991. Version 2, Third Draft.

[3] S. Armstrong, A. Freier, and K. Marzullo. RFC 1301: Multicast transport protocol. *Internet Request for Comments*, February 1992.

[4] T. Berners-Lee, R. Cailliau, J-F. Groff, and B. Pollermann. World-Wide Web: The information universe. *Electronic Networking: Research, Applications and Policy*, 1(2):52–58, Spring 1992.

[5] A. Birrell, R. Levin, R. M. Needham, and M. D. Schroeder. Grapevine: An exercise in distributed computing. *Communications of the ACM*, 25(4):260–274, April 1982.

[6] C. M. Bowman and C. Dharap. The Enterprise distributed white-pages service. *Proceedings of the USENIX Winter Conference*, pages 349–360, January 1993.

[7] M. Bowman, L. L. Peterson, and A. Yeatts. Univers: An attribute-based name server. *Software - Practice & Experience*, 20(4):403–424, April 1990.

[8] V. Cate. Alex - A global filesystem. *Proceedings of the Usenix File Systems Workshop*, pages 1–11, May 1992.

[9] D. Comer and T. P. Murtaugh. The Tilde file naming scheme. *Proceedings of the Sixth International Conference on Distributed Computing Systems*, pages 509–514, May 1986.

[10] J.E. Conklin. Hypertext: An introduction and survey. *Computer*, 20(9):17–41, September 1987.

[11] D. R. Cutting, D. R. Karger, and J. O. Pederson. Constant interaction-time scatter/gather browsing of very large document collections. *Proceedings of the ACM-SIGIR conf. on Information Retrieval*, pages 126–134, June 1993.

[12] P. B. Danzig, S.-H. Li, and K. Obraczka. Distributed indexing of autonomous Internet services. *Computing Systems*, 5(4):433–459, Fall 1992.

[13] P. B. Danzig, K. Obraczka, and A. Kumar. An analysis of wide-area name server traffic: A study of the Domain Name System. *ACM SIGCOMM '92 Conference*, pages 281–292, August 1992.

[14] Peter B. Danzig, Michael Schwartz, and Richard Hall. A case for caching file objects inside internetworks. *ACM SIGCOMM '93 Conference*, pages 239–248, March 1993.

[15] S. Deering and D. Cheriton. Multicast routing in datagram internetworks and extended lans. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

[16] Chanda Dharap, Rajini Balay, and Mic Bowman. Type structured file systems. *Proceedings of the International Workshop on Object-Orientation in Operating Systems*, December 1993.

[17] R. E. Droms. Access to heterogeneous directory services. *Proceedings of the IEEE INFOCOM '90*, June 1990.

[18] A. Emtage and P. Deutsch. Archie: An electronic directory service for the Internet. *Proceedings of the Winter 1992 Usenix Conference*, pages 93–110, January 1992.

[19] C. Faloutsos. Access methods for text. *ACM Computing Surveys*, 17:49–74, MAR 1985.

[20] S. Foster. About the Veronica service, November, 1992. Electronic bulletin board posting on the comp.infosystems.gopher newsgroup.

[21] James C. French, Anita K. Jones, and John L. Pfaltz. Summary of the final report of the NSF workshop on scientific database management. *SIGMOD Record*, 19(4):32–40, Dec, 1990.

[22] D. K. Gifford, P. Jouvelot, M. A. Sheldon, and J. W. O'Toole Jr. Semantic file systems. In *Proceedings of 13th ACM Symposium on Operating Systems Principles*, pages 16–25, October 1991.

[23] F.G. Halasz. Reflections on notecards: Seven issues for the next generation of hypermedia. *Communications of the ACM*, 31(7):836–852, July 1988.

[24] D. Hardy and M. F. Schwartz. Essence: A resource discovery system based on semantic file indexing. *Proceedings of the USENIX Winter Conference*, pages 361–374, January 1993.

[25] Darren R. Hardy. Scalable Internet resource discovery among diverse information. Technical Report CU-CS-650-93, Department of Computer Science, University of Colorado, Boulder, Colorado, May 1993. M.S. Thesis.

[26] International Organization for Standardization. Information Processing Systems – Open Systems Interconnection – The Directory – Overview of Concepts, Models, and Service. Technical report, International Organization for Standardization and International Electrotechnical Committee, December 1988. International Standard 9594-1.

[27] B. Khale and A. Medlar. An information system for corporate users: Wide Area Information Servers. *ConneXions - The Interoperability Report*, 5(11):2–9, November 1991.

[28] B. Lampson. Designing a global name service. *ACM Principles of Distributed Computing*, pages 1–10, August 1986.

[29] U. Manber. Finding similar files in a large file system. *Proceedings of the USENIX Winter Conference*, pages 1–10, January 1994.

[30] U. Manber and S. Wu. Glimpse: A tool to search through entire file systems. *Proceedings of the USENIX Winter Conference*, pages 23–32, January 1994.

[31] M. McCahill. The Internet Gopher: A distributed server information system. *ConneXions - The Interoperability Report*, 6(7):10–14, July 1992.

[32] Sun Microsystems. SunOS reference manual, Volume II, March 1990.

[33] P. Mockapetris. RFC 1034: Domain names - concepts and facilities. *Internet Request for Comments*, November 1987.

[34] P. Mockapetris and K. Dunlap. Development of the Domain Name System. *ACM SIGCOMM '88 Conference, Stanford, California*, pages 11–21, August 1988.

[35] B. C. Neuman. Prospero: A tool for organizing Internet resources. *Electronic Networking: Research, Applications, and Policy*, 2(1):30–37, Spring 1992.

[36] B. Clifford Neuman. The virtual system model: A scalable approach to organizing large systems. Technical Report 90-05-01, Department of Computer Science and Engineering, University of Washington, Seattle, Washington, May 1990.

[37] K. Obraczka, P. Danzig, D. DeLucia, and N. Alaam. Massive replication of data in autonomous internetworks. *Technical Report, University of Southern California*, March 1993.

[38] Vern Paxson. Empirically-derived analytic models of wide-area tcp connections: Extended report. Technical Report TR LBL-34986, Lawrence Berkeley Labs, June 15, 1993.

[39] J. Postel and J. Reynolds. RFC 959: File transfer protocol (FTP). Technical report, University of Southern California Information Sciences Institute, October 1985.

[40] J. B. Postel. RFC 821: Simple Mail Transfer Protocol. *Internet Request for Comments*, August 1982.

[41] J. Quarterman. *The Matrix - Computer Networks and Conferencing Systems Worldwide.* Digital Press, 1990.

[42] M. Schroeder, A. Birrell, and R. Needham. Experience with Grapevine: The growth of a distributed system. *ACM Trans. on Computer Systems*, 2(1):3–23, February 1984.

[43] M. F. Schwartz, A. Emtage, B. Kahle, and B. C. Neuman. A comparison of Internet resource discovery approaches. *Computing Systems*, 5(4):461–493, Fall 1992.

[44] M. F. Schwartz and P. G. Tsirigotis. Experience with a semantically cognizant internet white pages directory tool. *Journal of Internetworking: Research and Experience*, 2(1):23–50, March 1991.

[45] Michael F. Schwartz, Darren R. Hardy, William K. Heinzman, and Glenn Hirschowitz. Supporting resource discovery among public internet archives using a spectrum of information quality. *Proceedings of the Eleventh International Conference on Distributed Computing Systems*, pages 82–89, May 1991.

[46] Michael F. Schwartz and Calton Pu. Applying an information gathering architecture to Netfind: A white pages tool for a changing and growing internet. Technical Report CU-CS-656-93, Department of Computer Science, University of Colorado, Boulder, Colorado, December 1993. Submitted for publication.

[47] Richard Stallman. *GNU Emacs Manual*, sixth edition, March 1987.

[48] SunSoft. *SearchIt 1.0.* SunSoft, Inc., 1992. User's Guide.

[49] Chris Weider, Jim Fullton, and Simon Spero. Architecture of the Whois++ index service. Technical report, November 1992. Available by anonymous FTP from nri.reston.va.us, in internet-drafts/draft-ietf-wnils-whois-00.txt.

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

[51] D. C. M. Wood, S. S. Coleman, and M. F. Schwartz. Fremont: A system for discovering network characteristics and problems. *Proceedings of the USENIX Winter Conference*, pages 335–348, January 1993.

[52] Sun Wu and Udi Manber. Fast text searching allowing errors. *Communications of the ACM*, pages 83–91, October 1992.

[53] D. Zimmerman. RFC 1288: The finger user information protocol. *Internet Request for Comments*, November 1990.