

# 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].<sup>1</sup> 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

---

<sup>1</sup>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.

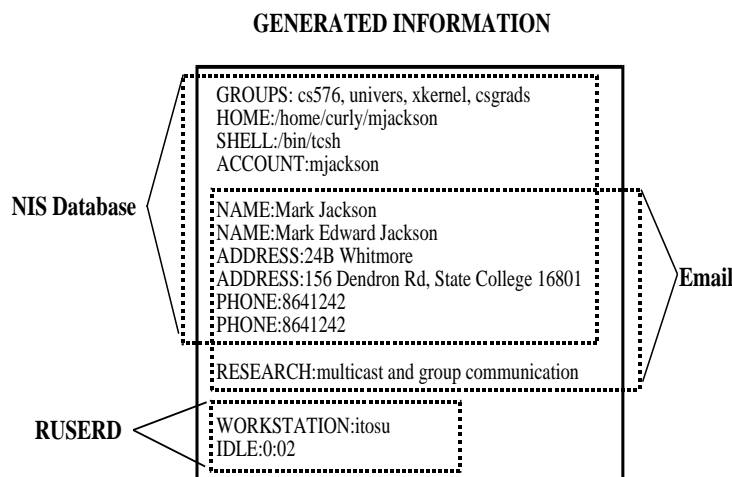


Figure 1: Example Mapping Algorithms for User Directory Information

*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<sup>2</sup> 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

---

<sup>2</sup>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, it 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.<sup>3</sup> 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

---

<sup>3</sup>Efforts are under way to improve this gateway.