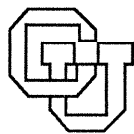


**Scalable Internet Resource Discovery
Among Diverse Information**

Darren R.Hardy

CU-CS-650-93 May 1993



University of Colorado at Boulder

DEPARTMENT OF COMPUTER SCIENCE

ANY OPINIONS, FINDINGS, AND CONCLUSIONS OR RECOMMENDATIONS EXPRESSED IN THIS PUBLICATION ARE THOSE OF THE AUTHOR(S) AND DO NOT NECESSARILY REFLECT THE VIEWS OF THE AGENCIES NAMED IN THE ACKNOWLEDGMENTS SECTION.

Scalable Internet Resource Discovery Among Diverse Information

Darren R. Hardy, University of Colorado, Boulder (hardy@cs.colorado.edu)

University of Colorado Technical Report

CU-CS-650-93 May 1993

M.S. Thesis directed by Assistant Professor Michael F. Schwartz

Abstract

Internet users need tools to help them discover resources that are available throughout the network. Unfortunately, current Internet resource discovery systems are inadequately prepared for the Internet's rapid growth. In particular, scalable techniques that adapt to growth in information diversity will provide a basis for future Internet resource discovery systems. To explore some techniques, we have built two resource discovery systems: *Dynamic WAIS*, which provides seamless information-level access to remote search systems through the Wide Area Information Servers (WAIS) interface; and *Essence*, which exploits file semantics to index a variety of types of data.

Acknowledgments

I would like to thank Sean Coleman, Jim O'Toole, Panos Tsirigotis, and David Wood for their helpful comments on earlier drafts of Chapter 4 (*Essence*). I would like to thank the many people who have offered comments and ideas during the development of *Essence* and *Dynamic WAIS*, including Tim Berners-Lee, Jonny Goldman, Jim Fulton, Simon Spero, Panos Tsirigotis, and Greg Whitehead. I would also like to thank Roger King and Ken Klingenstein for their support, comments, and interest in this thesis.

I would especially like to thank Mike Schwartz for his support, encouragement, and guidance throughout my years of Internet resource discovery research, and in the development of this thesis.

Finally, I would like to thank my family, friends, and Michelle for their support.

This material is based upon work supported in part by the National Science Foundation under grant NCR-9105372, and a grant from Sun Microsystems' Collaborative Research Program.

Table of Contents

Chapter 1: Introduction	1
1.1. Internet Resource Discovery	1
1.1.1. Why is Internet Resource Discovery Difficult?	1
1.1.2. Approaches to Internet Resource Discovery	2
1.2. Internet Growth	4
1.2.1. Information Volume	4
1.2.2. User Base	5
1.2.3. Information Diversity	5
Chapter 2: Information Diversity	6
2.1. Problems for Resource Discovery	6
2.1.1. Information System Diversity	6
2.1.2. Resource Representation Diversity	6
2.1.3. Heterogeneity	6
2.1.4. Inconsistency	7
2.1.5. Incompleteness	7
2.2. Approaches for Adapting to Information Diversity	7
2.2.1. Operation Mapping	7
2.2.2. Data Mapping	8
Chapter 3: Operation Mapping – <i>Dynamic WAIS</i>	9
3.1. Information System Gateways	9
3.1.1. Menu-Level Gateways	10
3.1.2. Information-Level Gateways	10
3.2. The Dynamic WAIS System	10
3.2.1. Supported Remote Search Systems	11
3.2.2. Dynamic WAIS Design	12
3.3. Prototype	12
3.3.1. WAIS-to-Netfind Information-Level Gateway	12
3.3.2. WAIS-to-archie Information-Level Gateway	13
3.3.3. Prototype Availability	13
3.4. Conclusions	14
Chapter 4: Data Mapping – <i>Essence</i>	15
4.1. Full Text vs. Filename vs. Semantic Indexing	16
4.2. The Essence System	16
4.3. Design Overview	17
4.4. Scalable Techniques	18
4.4.1. Determining File Types	18

4.4.2. Nested File Structure	19
4.4.3. Summarizers	19
4.5. Prototype	20
4.5.1. Determining File Types	20
4.5.2. Summarizers	21
4.5.3. WAIS Interface	24
4.5.4. Performance	25
4.5.5. Analysis of Keyword Quality	28
4.5.6. Prototype Availability	28
4.5.7. Anonymous FTP Indexing	28
4.6. Conclusions	28
Chapter 5: Related Work	29
5.1. Identifying and Locating File Resources	29
5.2. Exploiting File Semantics	29
5.3. Semantic File Indexing	29
5.4. Hypertext	30
5.5. Semantic Data Modeling	32
5.6. Heterogeneous, Distributed Databases	33
Chapter 6: Future Directions	34
6.1. Scalable Content-Based Searching	34
6.2. General Techniques for Information-Level Gateways	35
6.3. Supporting Differing Query Semantics in WAIS	35
Chapter 7: Summary	36
Chapter 8: References	37

List of Tables

Table 1: Approaches to Information System Gateways	10
Table 2: Supported File Types in Common File System Environments	22
Table 3: Essence Summarizer Techniques	23
Table 4: Time and Space Measurements for WAIS and Essence	26
Table 5: Weighted Time and Space Averages for WAIS and Essence	26
Table 6: Percentage of Interpretable Data	27
Table 7: Nested File Structure Overhead	28

List of Figures

Figure 1: Menu-Level Gateway: Internet Gopher-to-Netfind	11
Figure 2: Information-Level Gateway: WAIS-to-Netfind	13
Figure 3: Information-Level Gateway: WAIS-to-Archie	14
Figure 4: Organization of the Essence System	18
Figure 5: How a Summarizer Works	20
Figure 6: WAIS Search Using Essence-based Index	25
Figure 7: NCSA Mosaic: <i>archie</i> Query	31
Figure 8: NCSA Mosaic: <i>archie</i> Response	32
Figure 9: Indexing Space Efficiency vs. Representativeness	34

Chapter 1

Introduction

The Internet is a collection of high-speed computer networks which interconnects nearly a million computers around the world [Lottor 1992]. Each day millions of people use the Internet to transfer data, communicate via electronic mail, access remote computers, and retrieve information from a wide variety of databases. The Internet has evolved into a unique medium that promotes global collaboration and communication [LaQuey 1993].

For many years, the Internet community has concentrated on building the world's largest and fastest computer network. Unfortunately, the rapid evolution of the Internet's physical infrastructure has left behind an immature *information infrastructure*. The Internet's resources, which include a wide variety of information, software, computers, and people, are loosely structured, and as a result, locating Internet resources is difficult. The need for Internet *resource discovery* tools has grown dramatically in the past several years. The Internet community has started to meet this growing demand by developing several resource discovery tools [Schwartz et al. 1992b]. Nevertheless, these tools are inadequately prepared for the Internet's rapid growth. This thesis focuses on overcoming these inadequacies by developing scalable Internet resource discovery techniques that will support the next generation of resource discovery systems.

This chapter introduces Internet resource discovery, and characterizes the growth of the Internet. Chapter 2 discusses the problem of growing information diversity in the Internet, and outlines two approaches to this problem: *operation mapping* and *data mapping*. Chapter 3 examines the operation mapping approach by analyzing techniques used in *Dynamic WAIS*. Chapter 4 examines the data mapping approach by analyzing techniques used in *Essence*. Chapter 5 discusses some related work to the techniques presented in chapters 3 and 4. Chapter 6 explores some future directions for improving these techniques. Finally, chapter 7 offers a summary.

1.1. Internet Resource Discovery

The Internet resource discovery field has gained much popularity recently. Users find that Internet resource discovery systems indeed help them use the Internet more effectively. Internet resource discovery techniques, however, need improvement to support the Internet's rapidly evolving environment. Understanding current techniques helps in the development of such improvements.

The following subsections first describe why Internet resource discovery is difficult. They then discuss some common approaches used in Internet resource discovery systems.

1.1.1. Why is Internet Resource Discovery Difficult?

Discovering Internet resources is difficult primarily for two reasons: the Internet's decentralized administration, and the Internet's massive amount of diverse resources.

First, the Internet consists of many autonomous networks that span geopolitical boundaries, and are each managed by their own organization. In many of these networks, its management is divided further among several autonomous organizations. For example, Westnet manages a network spanning Arizona, Colorado, Idaho, New Mexico, Utah, and Wyoming. This network is connected to the Internet in Utah. Westnet's network itself consists of dozens of smaller networks that are managed by autonomous organizations such as universities, non-profit organizations, and businesses. Although these smaller networks connect to the Internet via Westnet's network, Westnet has little control over their administration. Therefore, no organization has administrative control over the Internet as a whole. In fact, the Internet is close to an anarchy, because the administration of the Internet's networks is so decentralized. The Internet is governed loosely by community consensus rather than a cohesive organization.

This lack of a controlling body creates difficulties for users who are trying to locate Internet resources. In particular, no organization controls which resources should be available on the Internet; and furthermore, no organization maintains a directory of the available Internet resources. Some organizations, however, maintain customized

directories that contain some of the available resources at their site. Each directory has its own semantics, organization, and scope. As a result, conglomerating these directories from different organizations into a meaningful, global resource directory is difficult. Therefore, Internet resource discovery tools need non-trivial approaches to adapt to diverse administrative techniques such as site-specific resource directories.

Second, the Internet contains a massive amount of diverse resources. Hundreds of gigabytes worth of resources are available from public Internet file archive sites alone [Emtage & Deutsch 1992]. For a system to create a single, global Internet resource directory, it needs to gather sufficient information about Internet resources from many different resource directories, or from the resources themselves if they are not contained within a directory. To gather this information, the system needs techniques to extract information from these heterogeneous repositories. It must also adapt to the sheer magnitude of building a global resource directory. As the Internet grows, creating a global resource directory is an infeasible task. Therefore, Internet resource discovery systems need to rely on novel techniques to discover resources without the aid of a global Internet resource directory.

In summary, discovering Internet resources is difficult due to the Internet's decentralized administration and massive volume of diverse Internet resources. Users need Internet resource discovery tools to help them use the Internet more effectively. These tools need to employ non-trivial resource discovery techniques.

1.1.2. Approaches to Internet Resource Discovery

In this section, we discuss several Internet resource discovery systems¹, and analyze two common resource discovery paradigms: browsing and searching.

The *whois* system, an Internet directory service, was one of the first tools to help users discover Internet resources [Harrenstien et al. 1985]. *whois* is a centralized database of registered Internet users, hosts, networks, and organizations. Users with *whois* client programs can query a *whois* server over the network to retrieve matching resource information.

The *Wide Area Information Servers* (WAIS) system is a distributed collection of databases from which users can search and retrieve documents [Kahle & Medlar 1991]. For efficiency, WAIS uses an *index* to represent the relationships between keywords and documents. To support fine-grained information access, WAIS builds full-text indexes, in which every keyword from a textual document appears in the index. As a result of using full-text indexing, WAIS has large space requirements: its indexes are comparable in size to the data files they represent. Because of these space requirements, WAIS distributes the indexes among the hosts that provide data. This approach is primarily useful for purely textual, widely popular data since it consumes a large amount of storage resources.

The *archie* system is an Internet file location service [Emtage & Deutsch 1992]. Periodically, *archie* retrieves file listings from known public Internet file archive sites, and then builds an index. *archie* avoids large space requirements by using only filenames as keywords in the index. Users can query *archie*'s index to locate files that are available from public Internet file archive sites, also known as *anonymous FTP sites*.

Netfind is an Internet user directory service [Schwartz & Tsirigotis 1991]. Netfind uses heuristics to dynamically locate Internet users, and discover user information. It continually gathers information from the Internet about where to search for users. When a user queries Netfind, they supply information about where a user might be located (for example, within a specific city or university). With this information, Netfind executes a narrowly-focused search for user information. Currently, Netfind is able to reach more than 5 million Internet users – more than any other Internet user directory service.

X.500 is a distributed resource directory service in which resources are hierarchically organized [CCITT/ISO 1988]. Users then traverse the hierarchy to locate resource information. For example, some initial X.500 prototypes, such as Performance System International Inc.'s white pages project (WPP) [Rose 1991], contain Internet user information. WPP's hierarchy contains many levels starting with countries, then organizations, then departments

¹ For a more detailed discussion of Internet resource discovery system approaches, refer to [Schwartz et al. 1992b].

within organizations, and so on. Thousands of entries are possible on each level. Browsing through such a large hierarchy can be difficult, so X.500 supports subtree searches. Searching in WPP, however, is limited to a small number of levels, since the amount of information grows quickly. The X.500 standard does not specify how to support searching, so implementations vary in their support of searching.

Prospero provides an organizational model that allows users to create their own private hierarchical *views* of Internet file resources [Neuman 1992]. Once users have built their own views, access to file resources is transparent. Users traverse their views to locate information.

Internet Gopher organizes Internet resources into a hierarchical, menu-based user interface [McCahill 1992]. Users traverse menus to locate Internet resources, and then select menu items to access the corresponding Internet resource. Internet Gopher maintains information about where resources are located, so when a menu-item for a resource is selected, Internet Gopher can connect the user to the resource. The *Veronica* service supports searching the registered menu items [Foster 1992], but the information contained within the resources themselves is not visible to Veronica.

These Internet resource discovery systems have two common paradigms for discovering resources: *browsing* and *searching*.

Browsing is a user-guided activity where users discover resources by traversing a graph, typically a hierarchical, directed graph. Consider locating a user within the WPP X.500 Internet user directory. Users must first traverse several levels of the hierarchy, sifting through potentially thousands of entries at each level, to reach the department or organization with which the desired user is associated. Once users reach this level, browsing for the desired user information is less labor-intensive, since the scope of the resources left to be browsed is narrowly focused. For example, consider users who are looking for information about a University of Colorado student. Once they have found the level containing information about University of Colorado students, they only need to browse the user information at this level. Therefore, browsing is least labor-intensive during the final stages of resource discovery, when the scope of the resources to be browsed has been sufficiently narrowed, such that the remaining resources are closely related to the desired resource. *Fine-grained* resource discovery occurs when the desired resource lies within a narrowly focused set of resources. Browsing at the lowest levels of WPP's hierarchy is an example of fine-grained resource discovery.

Searching is an automated activity where a system discovers resources by locating those resources whose descriptions match user-provided keywords. Consider locating information within WAIS. Users supply WAIS with keywords that describe the contents of the desired information. WAIS then matches these keywords with the keywords in the full-text index. If WAIS finds any matches, then the documents associated with the matching keywords are returned to the user. Typically, keyword searching returns many matching resources, since it is difficult for users to provide enough specific keywords to exactly match the desired resource. Therefore, searching is most effective for narrowing the collection of resources to search, by locating resources that are closely related. *Rough-grained* resource discovery occurs when the desired resource lies within a large collection of resources whose contents are usually closely related. Searching all anonymous FTP sites for a file is an example of rough-grained resource discovery.

Most current Internet resource discovery systems use some combination of these two paradigms. To support future Internet resource discovery systems, we would like to find the most effective and efficient combination that will scale in the Internet's rapidly growing environment. Before defining such a combination, we first analyze the problems associated with these two paradigms.

Organization, the classification of resources based on attributes that typically describe a resource's contents, is essential to the effectiveness of these resource discovery paradigms. Without organization, these paradigms fail – like finding a needle in a haystack. The degree to which these paradigms fail differs, however.

Browsing depends on strict resource organization, since its performing fine-grained resource discovery to locate a specific resource. If resources are not sufficiently organized, browsing them quickly becomes too difficult. Searching, however, does not rely on such strict resource organization, since it is performing rough-grained resource discovery and is only responsible for locating a collection of related resources. Effective searching is possible with only loosely organized resources. For example, *archie* users can locate file resources using only filename-based descriptions. Filename-based descriptions only loosely organize resources since they represent the brief, subjective

perceptions of resource contents of many people, rather than a cohesive organization like call numbers for published books.

For Internet resource discovery, a major distinguishing characteristic of organization is whether or not resource organization relies on human guidance. Organizing resources by filename, for example, relies on *manual* organization, since filenames are manually written by humans to reflect a resource's contents. Similarly, organizing books by call number relies on manual organization. A published book is assigned a call number by a human to reflect the book's contents. In Internet resource discovery, manual organization does not scale well since it relies on human guidance. *Automated* organization, where a program organizes resources without human guidance, scales better since it relies on a program to make classification decisions.

Indexing improves the efficiency and scalability of searches. Indexes are only computed once to represent the organization of resources by associating a resource to a keyword that describes the resource's content. Searching for resources is reduced to finding matching keywords in an index and then returning the associated resources. Since indexes are precomputed, resources need only to be scanned once; thus, saving future searches the computational and communication costs for scanning them.

Programs can generate indexes automatically, although some people do generate indexes by hand (a personal address book that uses last names as keywords, for example). *Automated index generation* is a more scalable technique than manual index generation, since an automated program can generate a large number of index entries relatively quickly and inexpensively. A major problem with automated index generation is obtaining resource descriptions, without relying on users to manually write them; thus, automated *resource classification* techniques are needed to support automated index generation. Automated index generation involves two distinct steps which are performed by a program: first, identifying a resource's representation and interpreting it to generate a resource description (*automated resource interpretation*), and second, deciding how to classify a resource based on its description (*automated resource classification*).

WAIS's full-text indexing is an example of an automated resource classification technique. WAIS classifies files depending on each word they contain. This technique, however, is too space inefficient to scale well. *archie*'s filename indexing is a resource classification technique that is more space efficient. It relies, however, on filenames which users must manually write to describe a resource's contents, so it is not as fully automated as WAIS's technique. Typically, filenames do not sufficiently describe a resource's contents. An automated resource classification technique is needed that sufficiently describes resource contents, yet is space efficient.

A promising combination of searching and browsing for Internet resource discovery systems is where users first search for a collection of narrowly focused, closely related resources (rough-grained resource discovery), then browse these resources for the desired resource (fine-grained resource discovery). This approach saves users from browsing through too much information, and doesn't rely on searching to produce exact resource matches. Also, to support scalable, efficient searching, an Internet resource discovery systems can use automated resource classification and automated index generation techniques. This thesis, in particular, focuses on developing such techniques.

1.2. Internet Growth

Now that we have examined both Internet resource discovery problems and approaches, we want to develop some requirements for the next generation of Internet resource discovery systems. To develop these requirements, we first look at how the Internet is growing, and how current Internet resource discovery approaches fail to adapt to this growth. Understanding the dimensions of the Internet's growth [Lottor 1992, Klingenstein 1992, Lynch & Rose 1993] will help develop useful approaches for the next generation of Internet resource discovery systems

The following subsections focus on defining the dimensions of the Internet's growth, and describing how current Internet resource discovery approaches are inadequately prepared for this growth. This thesis focuses on one of these dimensions, information diversity, since developing techniques to adapt to this growth provides a basis for future Internet resource discovery systems.

1.2.1. Information Volume

As the number of users, hosts, and applications continues to grow, so does the amount of information available in the network. An *information base* is the information resources that an Internet resource discovery system includes in its scope. Future Internet resource discovery systems will need to accommodate much larger information bases. Doing so, however, is difficult.

Although *archie* covers a large information base, it restricts its functionality by only supporting filename-based searches. WAIS also covers a large information base, yet it supports content-based searches. WAIS, however, restricts users to somewhat arbitrarily focused searches. Users only receive a brief, ad-hoc description of a database's content, so focusing searches is difficult. Internet Gopher also covers a large information base, but it restricts its functionality by only supporting limited searches and forcing users to learn many different user interfaces to access information. *archie*, WAIS, and Internet Gopher demonstrate an unbalanced trade-off between functionality for the user and the information base supported by the system.

Current Internet resource discovery systems either sacrifice functionality to support a large information base, or focus on a small set of information to retain a high degree of functionality. Internet resource discovery systems will need to ease this trade-off, so that systems can support enormous information bases without sacrificing too much functionality in the process.

1.2.2. User Base

The number of Internet hosts is growing at 10-15% per month [Lynch & Rose 1993]. As the number of hosts increases, so does the number of Internet users. Centralized information repositories will become obsolete as users distributed throughout the Internet will demand efficient access to information. Sophisticated replication and caching techniques is a promising approach to providing users with efficient information access, and at the same time, minimizing Internet backbone traffic. Internet resource discovery systems will need to employ these techniques to efficiently handle information access requests from users distributed throughout the Internet.

Current Internet resource discovery systems have limited support for efficient information access. WAIS, *whois*, and Internet Gopher do not support replication or caching of information. As these systems increase in popularity, demands on the Internet backbone will continue to increase. The Netfind and *archie* databases are manually replicated to provide users more efficient information access. As these systems increase in popularity, however, manual replication is infeasible, especially with systems like *archie* whose databases are hundreds of megabytes. Although X.500 supports replication, the brief practical experience with X.500 prototypes in the Internet offers little insight to efficient replication techniques.

1.2.3. Information Diversity

Information in the Internet is typically available either through an information system interface (as with WAIS), or directly available as an autonomous resource (as with files available via anonymous FTP). In the Internet, the diversity of information is increasing as both information systems and autonomous resource representations diversify. For Internet resource discovery systems to include as much information as possible, they must adapt to this information diversity. For example, automated resource classification techniques need to interpret and classify a wide variety of diverse resources.

Users benefit when Internet resource discovery systems can include as many resources as possible throughout the Internet. Developing scalable techniques for adapting to information diversity provides a basis for these Internet resource discovery systems. Techniques that adapt to the Internet's growing user base and information volume, however, only address problems with resources that the Internet resource discovery system already includes. Therefore, this thesis focuses only on developing scalable resource discovery techniques to adapt to growing information diversity, both in information system diversity and in resource representation diversity.

Chapter 2

Information Diversity

In this chapter, we examine the problems for Internet resource discovery systems that strive to adapt to information diversity. Then, we outline two promising approaches to adapting to information diversity on which this thesis focuses.

2.1. Problems for Resource Discovery

Information diversity surfaces in two different ways in the Internet, diverse resource representations and diverse information systems, and it is primarily problematic because of its heterogeneity, inconsistency, and incompleteness. In the following subsections, we further discuss information system diversity and resource representation diversity, and analyze its characteristics of heterogeneity, inconsistency, and incompleteness.

2.1.1. Information System Diversity

The number of information systems in the Internet is growing as communities develop *information spaces*, each of which serve a community's specific interests. The *Usenet* news system is a good example of a collection of these information spaces [Quarterman 1990]. Usenet is divided into thousands of information spaces, or *newsgroups*, each of which is a forum for discussion on a specific topic. Usenet, however, has a general interface since it services a diverse user base. On the other hand, *Telesophy*, an information system for molecular biologists who study a species of nematode worm, has a custom interface [Schatz 1990]. This interface better services unique queries that molecular biologists require, such as DNA sequence matching. In general, information spaces have interfaces that best suits its user community.

2.1.2. Resource Representation Diversity

Applications are shifting the face of information from simple text to multimedia. This revolution brings with it a wealth of different file formats and resources, such as sounds and images. In other words, resource representations are diversifying.

Consider the documents generated by various word processing applications. Although most of word processing applications are similar in functionality, many use their own proprietary file format for storing documents. Interoperability between these applications suffers since these file formats differ so much. Some tools can convert these file formats into a single, common format like PostScript [Adobe 1990], but much of the original document's semantic information is lost in translation. In general, diverse file formats cannot be accurately transformed into a smaller subset of file formats. Therefore, since translations do not work well, Internet resource discovery systems that want to include documents with diverse file formats in their information base must understand a wide variety of file formats. In addition, the increasing number of different file formats shows no sign of slowing, since proprietary file formats are flourishing. As a result, the diversity of file formats will continue to plague Internet resource discovery systems.

2.1.3. Heterogeneity

Resources representations can vary from unstructured text as in electronic mail messages, to structured, tagged information as in bibliographic databases. Although tools can transform one representation into another, important semantic information about the resource might be lost in the translation.

Information system interfaces can vary from keyword-based searching as in WAIS, to browse-based systems such as X.500. Accessing information from each of these systems is fundamentally different, so homogeneous approaches fail.

Since homogeneous resource interpretations might lose semantic information during translation, and homogeneous access methods to information systems fail, Internet resource discovery systems must adapt a heterogeneous approach to heterogeneous resources, so that they can include as many resources as possible in their information base, and minimize the loss of semantic information from these resources.

2.1.4. Inconsistency

Internet resource discovery systems must quickly adapt to rapidly changing resources; otherwise, users will receive out of date information about resources. For example, many popular software packages are available through anonymous FTP sites. Initially a software package may only be available from a handful of sites, but in a matter of days, it may become available on dozens of sites throughout the Internet. When a new version of the software is released, all of the sites might not receive the updated software. Internet resource discovery systems, such as *archie*, need to update their information on resource locations quickly, and maybe even remove the locations of the outdated software packages, so that the user is referred to sites containing the updated software packages. *archie* updates its resource location information monthly, and therefore it adapts to the relatively static nature of anonymous FTP sites. For some Internet resource discovery applications, such as a Usenet news information location system, resources change much more quickly, thus requiring a more swift updating approach than *archie*.

2.1.5. Incompleteness

Netfind uses both user account information such as electronic mail hosts *and* user-defined information such as *finger* information [Zimmerman 1990] to extract information about users. In general, since most resources contain only a limited scope of information, Internet resource discovery systems may need to include information from several different resources to acquire enough relevant information.

This incompleteness also occurs with information available through information system interfaces. Internet resource discovery system may need to access multiple information systems to produce a more complete resource description. For example, an Internet resource discovery system might extract user information from both *whois*, which might include an electronic mail address, postal address, telephone number, and title, *and* Netfind, which might include research interests, home machine name, and previous login time, to form a more complete description of a user. Internet resource discovery systems need to adapt to incomplete resources to supply users with more complete resource information.

2.2. Approaches for Adapting to Information Diversity

The next generation of Internet resource discovery systems need scalable approaches to deal with diversifying information in the Internet. The following two subsections outline two promising approaches: *operation mapping*, for adapting to information system diversity, and *data mapping*, for adapting to resource representation diversity.

2.2.1. Operation Mapping

Many Internet resource discovery systems force the user to explicitly access information system resources. To access information available through autonomous information systems, Internet Gopher supports *menu-level gateways*. Users browse hierarchical menu items until they locate an interesting resource. When a user selects the menu item for a resource, if this resource is an information system, then Internet Gopher connects the user with the remote information system's user interface. At this interface, users extract information from the information system. Users are forced to learn a different interface for each remote information system that they want to access from Internet Gopher. In menu-level gateways, users explicitly access the information from the autonomous information system interface, but they need not remember the details to connect to the information system's user interface. As the

number of information systems increases, however, Internet Gopher users must continue to learn more interfaces.

More sophisticated approaches are needed to remove the user from both the process of connecting to the information system's user interface as well as the process of accessing information through an autonomous user interface. These approaches can provide automated information system access through a single interface. Future Internet resource discovery systems need to employ these approaches to support resources available through disparate, diverse information systems. By providing automated means for this information access, these approaches scale well as the Internet's information system diversity increases.

Operation mapping is such an approach. It provides automated, transparent access to information that is available through heterogeneous information system interfaces, by using *information-level gateways*. In essence, operation mapping maps the operations of many heterogeneous information system interfaces into a single interface without losing functionality. Operation mapping can adapt well to new information system interfaces, and save users from continually learning new information system interfaces. Knowbots is a step forward in this direction [Droms 1990]. Users query the Knowbots system for user information through a single user interface. Knowbots then extracts information from several, heterogeneous user directory services, and combines it into a structured response for the user. Other systems have taken similar automated approaches to heterogeneous resource access [Schwartz et al. 1987].

2.2.2. Data Mapping

By reducing all file resources to a common automated resource interpretation method (extracting a filename), *archie* does not need to worry about resource representation diversity, as all resources are treated the same. Rather *archie* forces the user to worry this diversity, since the user needs either to verify that the matching resources are in the appropriate representation, or to issue more sophisticated queries that match resources in the appropriate representation.

Like *archie*, WAIS interprets all file resources identically. WAIS extracts every word from a file and uses them as keywords in an index, regardless of the file's representation. WAIS does have limited support for diverse file resources such as electronic mail, but it is not automated. The user must explicitly instruct WAIS to interpret files differently. Since it treats all resources identically, however, semantic information is lost from the resource during its interpretation.

Although X.500 interprets diverse resources, how it interprets a resource is determined when it is added to the directory. The user (or program) that adds resources to the directory is responsible for explicitly identifying a resource's representation and its interpretation through rules [Lynch & Rose 1993].

In general, many Internet resource discovery systems skirt the problems of automated resource classification, by shifting the burden of automated resource classification from the system to the user. Future systems need more automated approaches to scale.

Data mapping is such an approach. It provides automated resource interpretation and classification of heterogeneous resource representations, while preserving semantic information about resources. Data mapping adapts to new resource representations well, since it employs automated resource classification.

Chapter 3

Operation Mapping – *Dynamic WAIS*

In this chapter, we discuss some approaches that build information system *gateways*, including menu-level and information-level gateways, to access information available through disparate information systems. In particular, constructing information-level gateways is an effective technique to cope with diversifying information systems. We've built *Dynamic WAIS* [Hardy & Schwartz 1993b] to experiment with such an operation mapping technique. Finally, we discuss the details of our prototype, and its effectiveness in seamlessly integrating information spaces through information-level gateways.

3.1. Information System Gateways

Information system *gateways* can provide users with access to information available through disparate information system interfaces. Some systems, however, do not provide any of these gateways. Consider the problem of looking for popular public-domain software that is available via anonymous FTP. First, the user might want to access information available through Usenet to discover the most popular software that the Internet community offers. In order to access this information, the user needs to know how to connect to the Usenet system. Usually users issue a command to execute a Usenet news reader like *rn*. Many sites use the Network News Transfer Protocol [Kantor & Lapsley 1986]. In this case, users must also supply the name of their NNTP server, although system administrators usually supply this information rather than the users. Then, the user needs to know how to use the news reader interface. A user might read the *comp.archives* news group to find out information about the anonymous FTP sites that have the latest, most popular, public-domain software releases. After finding this information, the user exits the news reader interface, and connects to the file transfer protocol (FTP) interface. The user also needs to supply FTP with the hostname of the anonymous FTP site that has the desired software. Then, the user uses the FTP interface to transfer the desired software.

In this example, the user needs to know how to connect to the information systems. To connect to Usenet, the user supplies a command name and a NNTP server hostname. To connect to the anonymous FTP site, the user supplies a command name and the hostname for the anonymous FTP site.

The user also needs to know how to use the information system interfaces. To access Usenet information, the user issues commands to read news articles from the *comp.archives* newsgroup. To access anonymous FTP information, the user issues commands to transfer files from the remote site to the local machine.

In this case, no information system gateway is provided, since users themselves act as a gateway between the systems. In practice, most sites have experienced users to help others with the details of connecting to information systems, and using their interfaces. Similarly, many Internet users refer to guides or documentation instead of local experienced users [Krol 1989, NSF 1989, Granrose 1990, LaQuey 1990, Kehoe 1992, Krol 1992, Martin 1993, LaQuey 1993].

In general, without an information systems gateway, users must know many details, including a host name, port number, and commands, to connect to an information system. Users must also learn the interface to each information system, which usually have non-trivial interfaces. As a result, this approach do not scale well as information systems diversify and grow. An enormous amount of user effort is required to keep pace with the Internet growth. Approaches that remove this burden are needed.

Information system *gateways* can provide users with access to information available through disparate information system interfaces. In the following subsections, we discuss two approaches to building gateways as shown in Table 1: *menu-level gateways* where users need to know only an information system's interface; and *information-level gateways* where the user does not need to know either how to connect to an information system, or to use its interface.

Table 1: Approaches to Information System Gateways

Gateway Type	Information System	
	Connection	Interface Interaction
<i>None</i>	user-provided	user-provided
<i>Menu-Level</i>	automated	user-provided
<i>Information-Level</i>	automated	automated

3.1.1. Menu-Level Gateways

Instead of requiring users to supply the details on how to connect to an information system, Internet Gopher only requires users to select a menu item. Internet Gopher handles the connection to the remote information system. As Figure 1[†] shows, when Internet Gopher users select the “Netfind server at the University of Colorado, Boulder” menu item, the Internet Gopher system initiates a remote login session to the remote Netfind information system. The user does not need to know how to initiate the remote login session (Internet Gopher uses the *telnet* command). Nor does the user need to know which Internet host has the Netfind information system interface (Internet Gopher uses *bruno.cs.colorado.edu*). In this example, Internet Gopher forms a *menu-level gateway* to the Netfind information system, since the details of connecting to the remote information system are handled by the system.

Users, however, still need to learn information system interfaces. As Figure 1 shows, users must understand the Netfind interface to access the information available through Netfind. Even though menu-level gateways burden users, they still provide an advantage. Internet Gopher’s immense popularity is a testament to this fact.

3.1.2. Information-Level Gateways

As information systems diversify, users have difficulty remembering the details of using varying information system interfaces. An approach that removes this burden from users is needed. *Information-level gateways* take this approach. Information-level gateways allow users to access information from a variety of information systems without having to know the details of connecting to each information system, *and* without having to learn new information system interfaces. These gateways have automated means for connecting to information systems, as well as accessing information available through the information system interface.

Knowbots implements an information-level gateway between user directory services, such as *whois* and X.500 [Droms 1990]. Users query the Knowbots system for user information through a single user interface. Knowbots then extracts information from several user directory services, and then combines the information into a structured response for the user. Knowbots users do not need to know how to connect to the different user directory services. Nor do they need to know how to use the different user directory services. The Knowbots system’s information-level gateways automates both of these processes.

3.2. The Dynamic WAIS System

*Dynamic WAIS*² uses an operation mapping approach which implements an information-level gateway between WAIS and remote search systems. In this section, we discuss the types of remote search systems that Dynamic WAIS supports. We also discuss how Dynamic WAIS implements its information-level gateways, and the scalability of this approach.

[†] Reprinted with permission from [Bowman et al. 1993].

² WAIS extracts information from a static collection of textual documents. Dynamic WAIS, however, extracts information from remote search systems whose information spaces are typically dynamic, and hence how it received its name.

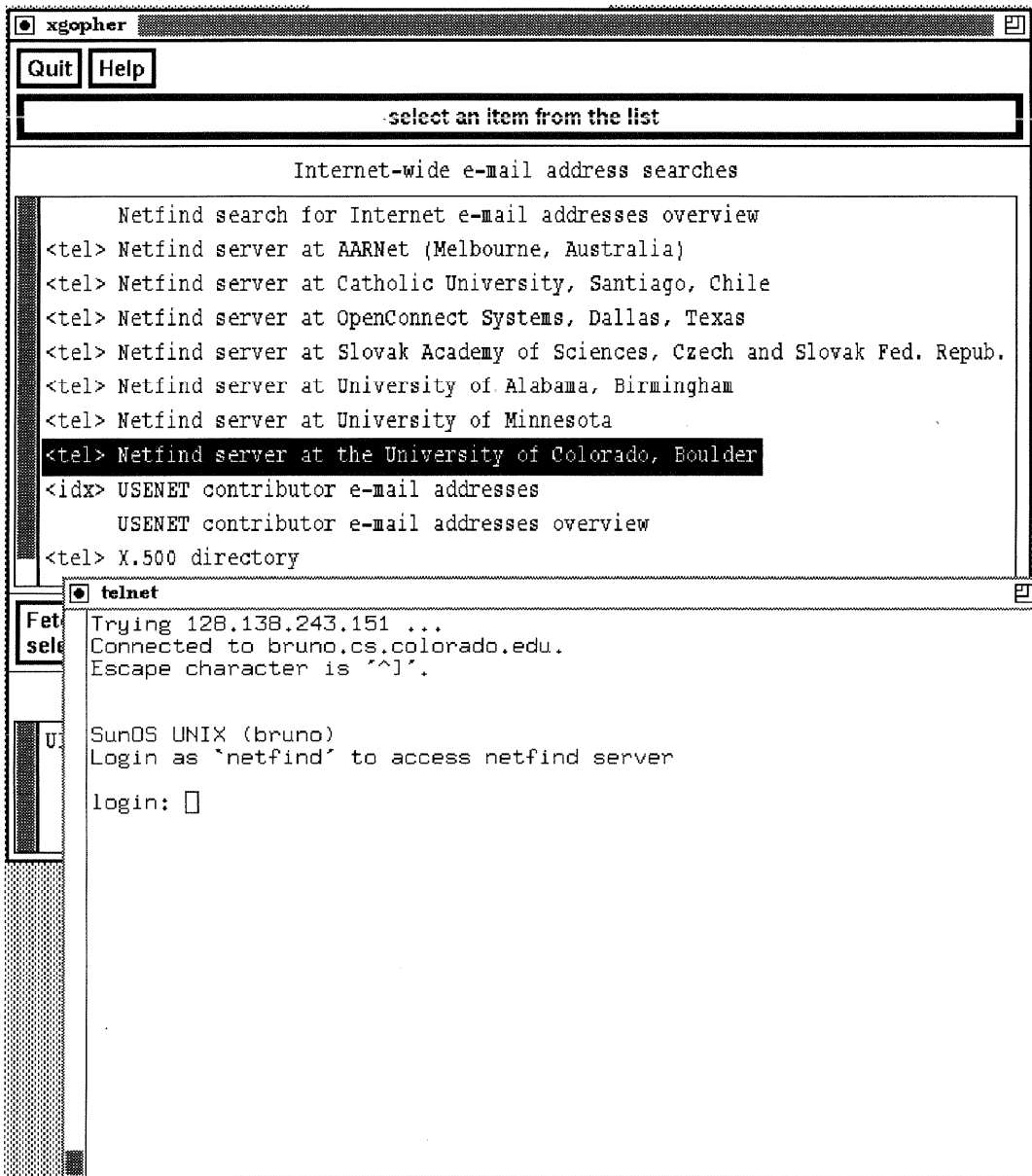


Figure 1: Menu-Level Gateway: Internet Gopher-to-Netfind

3.2.1. Supported Remote Search Systems

In Netfind, users first narrow the search scope to a handful of promising Internet domains. They then direct Netfind to search those domains for user information that matches user-supplied keywords. In *archie*, users first select one of the many *archie* servers, then they search the *archie* server for Internet file resources that match user-supplied keywords. In *whois*, users first select one of the many *whois* servers, then they search the *whois* server for user and domain information that match user-supplied keywords. These remote search systems, Netfind, *archie*, and *whois*, have a common attribute; they each require two search phases for a successful search. Dynamic WAIS takes

advantage of this common attribute to map this two-phase search into WAIS's search and retrieval paradigm. Mapping the operations of these remote search systems into a single paradigm prevents user from having to learn new interfaces to access information available from these remote search systems.

3.2.2. Dynamic WAIS Design

Dynamic WAIS modifies WAIS to support automated access to information available from remote search systems. Users perceive this information as if it were coming from a static textual database, rather than from a dynamic remote search system. The Dynamic WAIS design recognizes that the operations of certain remote search systems, like Netfind, *archie*, and *whois*, can be mapped into the WAIS search and retrieval paradigm. In this respect, Dynamic WAIS contributes an important *conceptual* mapping between WAIS's operations and the operations of remote search systems. Furthermore, an effective operation mapping approach requires this type of conceptual mapping. An appropriate conceptual mapping is fundamental to the seamless intergration between information systems, through information-level gateways.

In WAIS, users supply keywords for their search. Then, the WAIS server receives the keywords, looks them up in an index, and returns brief descriptions about the matching documents, known as *headlines*. Users may then choose among these headlines to retrieve an entire document.

In Dynamic WAIS, users supply keywords for their search. The Dynamic WAIS server receives the keywords, stores them, and returns brief descriptions about the choices that the remote search system requires during its first search phase. Dynamic WAIS, however, represents these choices as standard WAIS headlines. When the user chooses to retrieve one of the "documents," the Dynamic WAIS server searches using the keywords previously stored and the keywords from the chosen headline. Since the remote search system returns a single result for the search, the Dynamic WAIS server then is able to return the result in lieu of a static document.

3.3. Prototype

In this section, we discuss the Dynamic WAIS prototype, which implements information-level gateways between WAIS and Netfind, and WAIS and *archie*.

3.3.1. WAIS-to-Netfind Information-Level Gateway

Figure 2[†] shows the information-level gateway between WAIS and Netfind. The user first supplies the keywords "schwartz cs colorado" and then directs Dynamic WAIS to search for matching documents. Instead, Dynamic WAIS returns the choices of promising Internet domains as headlines. As shown in Figure 2, the user chooses the "cs.colorado.edu" headline. The retrieved document (shown in the lower rightmost window in Figure 2) is actually the result of the Netfind search for "schwartz" in the Internet domain "cs.colorado.edu," rather than a static textual document.

Unfortunately, Netfind queries have different semantics than WAIS queries. In WAIS, queries are simply a sequence of unordered keywords. In Netfind, however, queries are a sequence of ordered keywords in which a keyword to match a user comes first, then keywords to match an Internet domain come second. In our Dynamic WAIS prototype, we assume that the user is aware of the semantics of the remote search system queries, and that the user supplies an appropriate sequence of keywords.

A potential solution to adapting to differing query semantics is to associate a query template with each WAIS database. Whenever a user selects to search a WAIS database, the system retrieves a query "template" from the database server. Users then must use the template to structure their queries. In fact, Z39.50 [ANSI 1991], the search-and-retrieval protocol upon which WAIS is built, provides facilities to support this approach. WAIS,

[†] Reprinted with permission from [Bowman et al. 1993].

however, does not implement these facilities, and as a result, the differing query semantics are the responsibility of the user rather than the system.

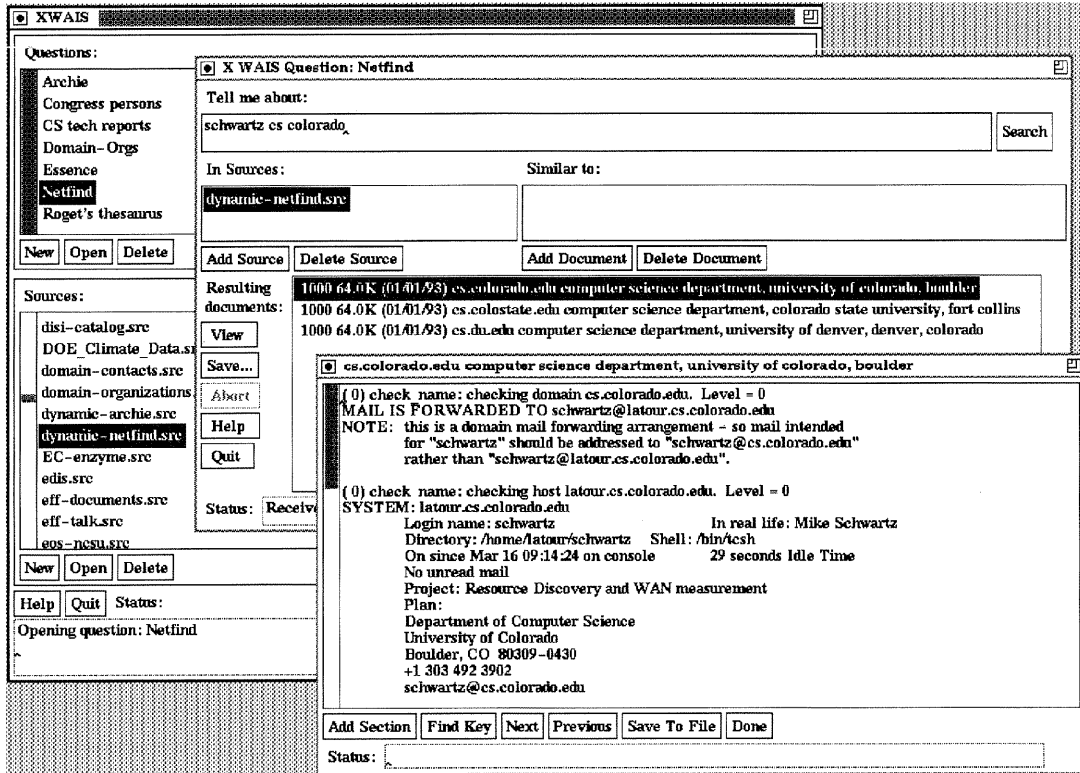


Figure 2: Information-Level Gateway: WAIS-to-Netfind

3.3.2. WAIS-to-archie Information-Level Gateway

Figure 3 shows the information-level gateway between WAIS and *archie*. The user first supplies the keyword “traceroute” and then directs Dynamic WAIS to search for matching documents. Instead, Dynamic WAIS returns the choices of the many *archie* servers throughout the Internet as headlines. As shown in Figure 3, the user chooses the “archie.ans.net” headline. The retrieved document (shown in the lower rightmost window in Figure 3) is actually the result of the *archie* search for “traceroute” as returned by the *archie* server “archie.ans.net,” rather than a static textual document.

As with the WAIS-to-Netfind information-level gateway, the query semantics for *archie* is different than those for WAIS. In *archie* queries, only a single keyword or regular expression is meaningful. In our Dynamic WAIS prototype, we assume that the user is aware of these semantics, and only uses a single keyword for WAIS-to-archie gateway queries.

3.3.3. Prototype Availability

The Dynamic WAIS prototype, including its documentation and WAIS modifications, is publically available by anonymous FTP from *ftp.cs.colorado.edu* in */pub/cs/distrib/dynamicwais*. The prototype is written in the C programming language [Kernighan & Ritchie 1988], and uses WAIS version 8-b5 [WAIS 1993].

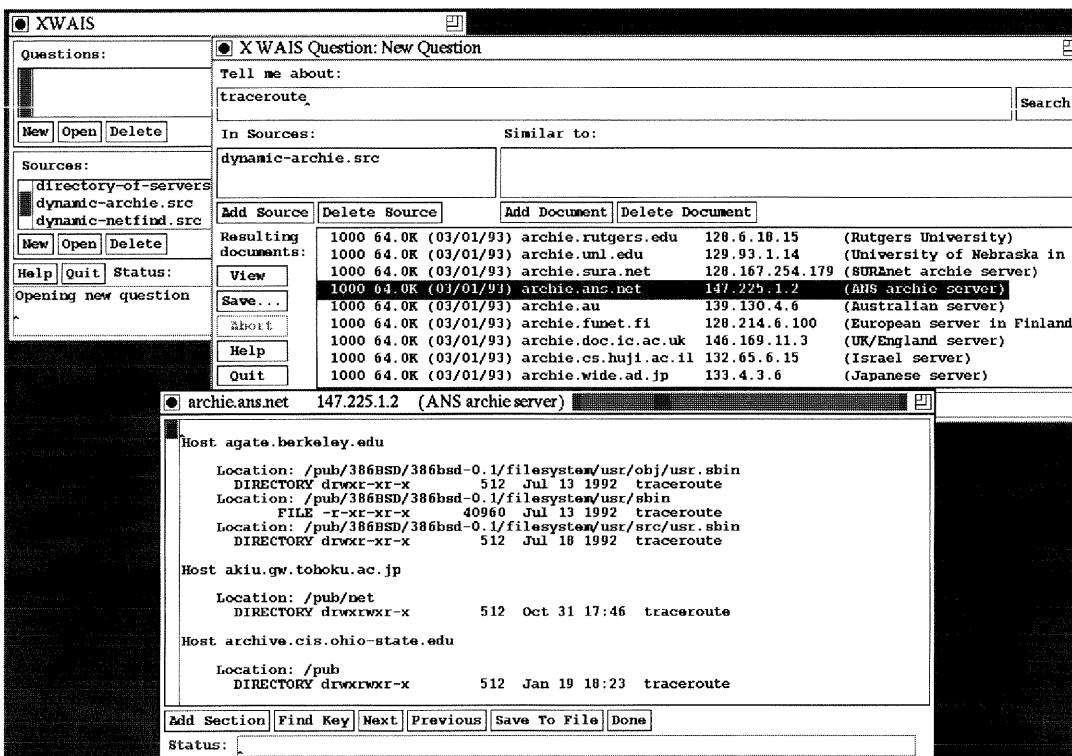


Figure 3: Information-Level Gateway: WAIS-to-Archie

3.4. Conclusions

Dynamic WAIS demonstrates that information-level gateways provide a powerful means for adapting to diversifying information system interfaces. This approach removes the user from the process of connecting to an information system, and from the process of extracting information through an information system interface. Furthermore, Dynamic WAIS seamlessly integrates WAIS and remote search systems. Query semantics, however, differ between information systems. Although Z39.50 supports facilities to support differing query semantics, WAIS does not implement them. As a result, our prototype assumes that users are aware of the different semantics, and that they conform to them. In this respect, Dynamic WAIS demonstrates the usefulness of supporting differing query semantics.

Chapter 4

Data Mapping – *Essence*

As disks have become larger, cheaper, and more plentiful, resource discovery has become a problem in general purpose file systems, such as the Sun Network File System (NFS). Yet, the current set of Internet discovery tools do not apply well to such an environment, for three reasons.

First, information in general file systems is typically very irregularly organized. Most Internet data is explicitly intended for sharing, and hence people often put some effort into organizing the information into a coherent whole (e.g., placing an entire file system into a meaningful hierarchical directory in an *anonymous* FTP site). In contrast, most general file system data are organized according to the individual whims of many people. Therefore, resource discovery systems that depend heavily on users to organize and browse through data (such as Prospero, WorldWideWeb [Berners-Lee et al. 1992], or Internet Gopher) do not work well for general purpose file system data. Instead, automated search procedures are needed. This typically means generating an index of the available information [Salton & McGill 1983].

Second, Internet resource discovery tools typically focus on information known to be of reasonably broad interest. For example, anonymous FTP archives typically contain popular documents and software packages, which exhibit heavy sharing [Schwartz et al. 1992a]. In contrast, general-purpose file systems typically contain mostly user-specific data that exhibit relatively little sharing [Muntz & Honeyman 1992, Ousterhout et al. 1985]. Current Internet resource discovery tools have difficulties with such low sharing-value data. For example, WAIS's full-text indexing mechanism may locate many uninteresting files if applied to an entire general file system, since keywords will be generated from files that are of interest to few users.

Third, general file systems contain a range of different resource representations, from unstructured text to structured data. Systems that use a generic indexing procedure (such as *archie* or WAIS) produce larger or less useful indexes under these circumstances. For example, WAIS is most effective when used on ASCII text files. Using WAIS to index executables and other files found in general file systems is not very effective. The indexes tend to do a poor job of locating information, and tend to be quite large.

In this chapter, we present a system for supporting resource discovery in general purpose file systems. The system addresses the above problems by generating indexes based on an understanding of the semantics of the files it indexes. This technique supports compact yet representative summaries for general collections of data. In addition to supporting file indexes, summaries can be browsed to help decide whether to retrieve a file across a slow network. This search and browse combination forms an effective resource discovery paradigm, as discussed in Section 1.1.2. We call our system *Essence* because of its ability to summarize large amounts of data with relatively small indexes [Hardy & Schwartz 1993a].

In particular, *Essence* focuses on an data mapping approach called *semantic indexing*. This approach automates heterogeneous resource interpretation and classification. In *Essence*, semantic indexing has two major benefits. First, it automates the identification and interpretation of diverse, heterogeneous file resource representations, or *file types*. In this sense, we show that *Essence* demonstrates an effective, scalable data mapping approach that uses automated resource interpretation techniques. Second, through effective resource interpretation techniques that preserve semantic information, *Essence* can exploit file semantics to produce compact, yet representative indexes. In this sense, we show that *Essence* demonstrates an effective, scalable data mapping approach that uses automated resource classification techniques which preserve important semantic information about file resources.

We begin with a discussion of indexing techniques. We discuss how *Essence* accomplishes semantic indexing. We also discuss the details of *Essence*'s data mapping approach, and how *Essence* uses this approach as a basis

Chapter 4 is an edited and expanded version of [Hardy & Schwartz 1993a].

for resource discovery. Finally, we discuss the details of our prototype, and present some measurements that compare Essence with WAIS and the MIT Semantic File System (SFS) [Gifford et al. 1991].

4.1. Full Text vs. Filename vs. Semantic Indexing

WAIS supports fine-grained information access by building full-text indexes, in which every keyword from a textual document appears in the index. As indicated above, this approach is primarily useful for purely textual, widely popular data. Moreover, WAIS has large space requirements, so WAIS distributes the indexes among the hosts that provide data.

A less space intensive indexing approach is used by *archie*, in which anonymous FTP files are summarized by name only (i.e., *archie* indexes contain no information derived from file content). This approach produces indexes that are roughly one thousandth the size of the data that they represent. In turn, this compact representation allows a great deal of index information to be collected onto a single machine, supporting far-reaching searches (currently reaching over 1000 archive sites). Yet, because *archie* indexes contain only filenames, they support only name-based searches. Searches based on more conceptual resource descriptions are not possible, except when the filenames happen to reflect some of these conceptual descriptions.

The range of structure and the low overall sharing value in general purpose file systems, coupled with the need for conceptual descriptions and the need for compact indexes, all suggest the use of a different means of indexing data. That means is *semantic* indexing.

Semantic indexing involves analyzing the structure of file data in different ways, depending on file type. For example, UNIX manual page files are broken into structured sections from which it is possible to extract information about a program's name and description, a usage synopsis, related programs or files, and author information. By generating information for different types of files in different manners, semantic indexing can generate representative keywords without including every word from a file. In addition to saving space, this technique can avoid including keywords that might degrade the quality of an index. For example, it makes little sense to include C language constructs like "struct" when indexing C source code, since these keywords do not distinguish the conceptual content of different C programs.

Semantic indexing involves two stages. The *classification* stage identifies promising files to index within a file system,³ as well as type information for each identified file. The *summarizing* stage applies an appropriate indexing procedure (called a *summarizer*, to emphasize the space reduction characteristic) to each identified file, based on the type information uncovered in the classification stage. The combination of these stages produces the foundation for an effective data mapping approach that automates diverse resource interpretation and classification.

Since summarizers understand file types, they can extract keyword information for both textual and binary files. For example, many binary executables have related textual documents describing their usage, from which keyword information can be extracted.

Since keyword information is extracted based on knowledge of where high-quality information might be located, semantic indexing extracts fewer keywords than full-text indexing, and thus generates smaller indexes. Yet, it retains the fine-grained, associative access capability of full-text indexes.

4.2. The Essence System

Essence provides an integrated system for classifying files, defining summarizer mechanisms, applying appropriate summarizers to each file, and traversing a portion of a file system to produce an index of its contents.

³ For example, this procedure might embody site-specific knowledge that certain parts of the file tree contain uninteresting administrative information, and hence needn't be indexed. Our current prototype does not select file system subsets - it simply indexes whatever file trees are specified.

Essence determines file types by exploiting file naming conventions (such as common filename extensions like “.c”), and locating identifying data or common structures within files (such as UNIX “magic numbers”). Once Essence determines a file’s type, it executes the appropriate summarizer to extract keywords from the file. Among other types, Essence understands *nested* file structures, such as compressed, uuencoded “tar” files. It recursively extracts files *hidden* within a nested file, and indexes them.

As a design goal, we wanted to allow summarizers to be constructed quickly and easily, so that Essence could be made to understand many different file types, and so individual sites could customize their summarizers. To accomplish this goal, we allow summarizers to be as simple as a one line script (perhaps containing a “grep” command).

Essence indexes can allow users to locate needed data. Moreover, Essence produces *summaries* of file data, which allow quick perusal of potentially large files.

Essence has many practical resource discovery applications:

- Systems administrators and users can use Essence to locate and learn about resources contained within their file systems without understanding the details of their local environment. This is particularly helpful in environments where mount points are “hidden” by the *amd* auto-mount system.
- Public archive administrators can use Essence to index archive contents, providing compact yet representative descriptions of files, including compressed archives. These indexes allow users to search for information more effectively, and examine summaries about files before retrieving them.
- People who wish to index data and search it using WAIS can use Essence to index more file types than WAIS itself currently supports, and to produce more space efficient indexes.

Once Essence generates an index for a portion of a file system, it exports its indexes via WAIS’s search and retrieval interface. This allows our indexes to be used within the context of a well established, easy to use information system.

4.3. Design Overview

Figure 4 shows how Essence is organized to accomplish semantic indexing. It operates as follows:

- Users supply Essence with the filenames from a select portion of a file system that they wish to index.
- The *Feeder* module iteratively passes each of these filenames to the *Classification* module, which determines the file’s type.
- The *Summarize* module chooses an appropriate summarizer based on the file’s type. It runs this summarizer on the file to extract keywords for the *Summary Files*.

The three modules, *Core Filename*, *Nested File Processor*, and *Nested File Feeder*, allow Essence to support nested files.

- Essence saves the initially supplied filename as the *Core Filename*.
- If the *Classification* module determines that the file has a nested file type (such as a compressed file), it passes the file to the *Nested File Processor*.
- The *Nested File Processor* extracts the hidden files from the nested file structure and passes the extracted files to the *Nested File Feeder*.
- The *Nested File Feeder* module performs the identical function as the *Feeder* but bypasses the *Core Filename* module.

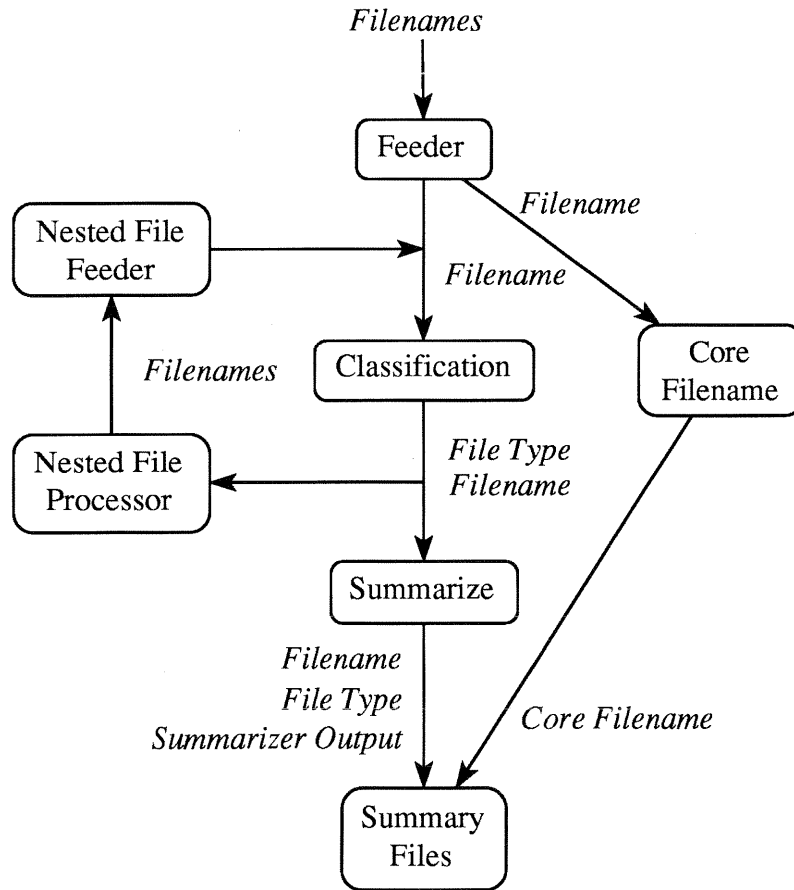


Figure 4: Organization of the Essence System

4.4. Scalable Techniques

Semantic indexing provides the basis for a sophisticated, scalable data mapping approach that employs techniques for both automated file resource interpretation and classification. In the following subsections, we examine the three main components of our approach: determining file types, nested file structures, and summarizers.

4.4.1. Determining File Types

Essence determines file types using a combination of exploiting file naming conventions and heuristically locating identifying data and common structures within files.

Unlike other systems such as WAIS, the user is not responsible for informing the system how to interpret file types, rather Essence automates the interpretation of diverse file types. Unlike other systems such as *archie* that use a generic data mapping approach that loses semantic information, Essence preserves semantic information through rigorous support of diverse file types. In fact, Essence later exploits the preserved semantic information to generate compact, yet representative indexes.

4.4.1.1. Exploiting File Naming Conventions

Observing even simple conventions in file naming can determine file types with fairly high certainty. The most basic file naming convention is filename extensions. For example, filenames with a “.c” extension are typically C source code files; filenames with a “.ps” extension are typically PostScript image files; and filenames with a “.txt” extension are typically ASCII text files. File naming conventions also include using specific words within a filename. For example, information about an entire source distribution or application is often found in files whose name contains the string “README”. Files named “Makefile” are typically associated with the UNIX *make* command [USENIX 1986].

In Essence, file naming conventions are represented as regular expressions. For example, **.ps* or **[Mm]akefile** represent the PostScript and Makefile file types, respectively. Expressing file naming conventions as regular expressions allows sites to easily integrate their local semantics into Essence.

4.4.1.2. Locating Identifying Data & Common Structures

In addition to using naming conventions, Essence examines file contents to try to determine file types. In particular, many files have an identifying *magic* number associated with them. For example, NeXT binary executables start with the hexadecimal number *0xfeedface*, and Sun Pixrect images start with the hexadecimal number *0x59a66a95*. Furthermore, common structures within a file may determine its file type. For example, PostScript images start with the string “%!”; UNIX shell programs start with the string “#!”; C source code files typically have comments denoted with the “/* */” delimiters; electronic mail files have distinctive header tags, such as *From:*, *Received by:*, and *Sender:*; and USENET news articles also have distinctive header tags, such as *Newsgroups:*, *Distribution:*, and *Path:*.

As with exploiting file naming conventions, locating identifying data and common structures within a file is a rule-based technique expressed with regular expressions. Sites can easily integrate their local semantics into the file type identification process by modifying these rules.

4.4.2. Nested File Structure

Nested files contain *hidden* files. Examples include compressed files, tar files, uuencoded files, ZIP files, and shell archive files. Furthermore, files can be arbitrarily nested within these file types. For example, compressed tar files or uuencoded compressed files are common. Understanding nested file structures is useful in file system environments (such as anonymous FTP file systems) in which the vast majority of files have nested structure.

When Essence determines that a file has nested structure, it extracts the hidden files, determines the resulting files’ types, and summarizes them. This process continues recursively, until no more nesting is found. Extracting hidden files from a nested file is accomplished by running a corresponding extraction program, such as the UNIX *uncompress* command for compressed files, the UNIX *tar* command with the “x” flag for tar files, or the UNIX *uudecode* command for uuencoded files. Furthermore, defining the interpretation of new nested file structures is a simple operation in Essence.

Through automated interpretation of nested file structures and file types, Essence scales well as the diversity of resource representations increases in the Internet.

4.4.3. Summarizers

Figure 5 shows how a summarizer works. The summarizer interprets the file data and exploits its semantics to extract a few keywords that are representative of the file’s contents. Summarizers increase the scalability of Essence’s data mapping approach as it confines resource interpretation into a small module that is easily written, integrated, and maintained.

Summarizers are simple stand-alone UNIX programs that are easy to write and integrate into the system. This design provides a powerful paradigm for exploiting file semantics. Each summarizer is associated with a specific file type, and understands the file’s format well enough to extract summary information from the file. For example, the summarizer for a UNIX troff-based manual page understands the troff syntax and the conventions used to

describe UNIX programs. It uses this understanding to extract summary information, such as the title of a program, related programs and files, the author(s) of the program, and a brief description of the program. Similar techniques can be used on many other moderately structured file types, such as source code. However, some file types do not easily lend themselves to automated interpretation. For example, ASCII text files typically contain unstructured data that is difficult to exploit effectively. Similarly, the UNIX *ps2txt* program can extract ASCII text from PostScript images, but the resulting information is unstructured text.

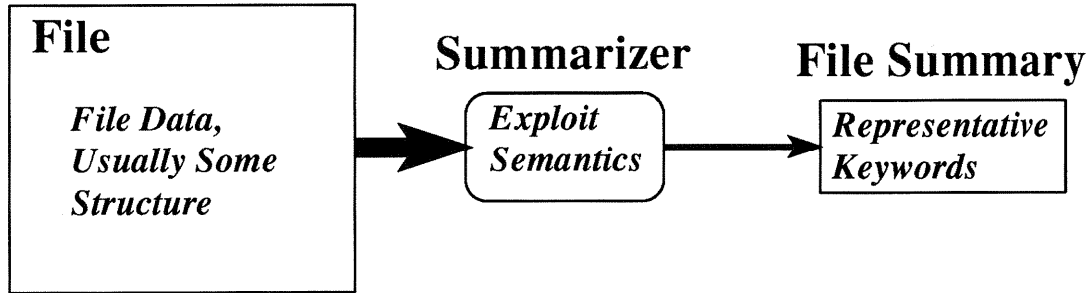


Figure 5: How a Summarizer Works

4.5. Prototype

In this section, we describe how our prototype determines file types, and exploits file semantics. We also discuss how we integrated Essence with WAIS.

4.5.1. Determining File Types

As described earlier, file types are determined by understanding naming conventions and locating identifying data and common structures within a file. In the prototype, naming conventions are expressed with case-insensitive regular expressions. The following example shows some entries from the configuration file that holds the expressions. In this file, the first field is the file type, and the second field is a case-insensitive regular expression for the corresponding file naming convention.

```
Compressed    .*\.Z
ManPage       .*\[12345678]
PostScript    .*\. (ps|eps)
README        .*readme.*
SCCS          s\..*
```

The prototype also uses the UNIX *file* command to determine file types, based on identifying data and common structures within a file [USENIX 1986]. *file* uses the */etc/magic* configuration file to specify recognizable file types. The following list shows some sample entries from */etc/magic*, where the first field is the data's offset into the file, the second field is the type of this data, the third field is the identifying data or common structure itself, and the last field is the corresponding file type.

```
0 string /*          C program text
0 string \037\235     Compressed data
0 long  0xfeedface   NeXT binary program
0 string #!/bin/perl  Perl program
0 string %!          PostScript image
```

Creating a suitable *magic* file is not trivial, because the identifying data or common structures must be distinctive. For example, the “/*” delimiter for C programming language comments is not sufficiently distinctive, and will likely appear in a variety of types of files. A lack of distinctive identifying data or common structures is common for binary formats, which usually depend on a single magic number. Although distinctive *magic* entries are difficult to formulate, careful selection of a *magic* file allows *file* to accurately identify file types. In Essence, building the *magic* file was accomplished through experimentation with various entries.

4.5.2. Summarizers

In the prototype, summarizers are simple UNIX programs that extract keyword information through understanding the syntax and semantics of a specific file type. Currently, the prototype supports summarizers for twenty-one file types and four nested file types. Table 2 describes these file types, their frequencies of occurrence by number of files, average file size, and which systems support them in two file system environments: an NFS file system that contains commonly shared data and programs in our local environment, and a fairly popular anonymous FTP file system (ftp.cs.colorado.edu). The most frequent file types in the NFS file system were *Text*, *CHeader*, and *MainPage*. In the anonymous FTP file system the most frequent file types were *CHeader*, *C*, and *Text*.

Essence supports more of the file types found in common NFS and anonymous FTP file systems than either WAIS or SFS, as shown in Table 2. Although WAIS and SFS support most of the frequently occurring file types (such as *Text*, *C*, and *CHeader*), Essence is the only system that supports the file types that contribute most to overall data size (such as *Binary*, *Tar*, and *Archive*). Occurrence frequencies will be used in our measurements, later in this chapter. Note that Table 2 does not list file types supported by WAIS or SFS that are not supported by Essence, since those types are all for fairly specialized formats that mostly do not occur in our measured file systems (and hence for which we had no measurements). Examples include MedLine and New York Times formats. There are 12 such formats understood by WAIS, and 4 understood by SFS. Also, as indicated in the table, Essence and SFS index “Unknown” file types. They do so by including the standard UNIX attributes in the index, such as owner, directory, group, and filename.

Table 3 briefly describes the techniques used by the Essence summarizers for the supported file types, except for nested types whose techniques were previously discussed. Many other potential summarizers are possible. For example, writing summarizers for other types of source code (such as Lisp or Pascal) would be an easy extension of the prototype’s source code summarizers. Writing summarizers for audio or image formats, however, would be difficult.⁴

The following sections describe some of the techniques used in various summarizers, representative of Essence’s supported file types.

4.5.2.1. Directory

Obtaining a listing of the files in a directory is an obvious method for a *directory summarizer*. However, Essence strives to obtain a higher-level understanding of a directory’s contents. Therefore, the prototype attempts to extract copyright information from files, in addition to the directory listing. Copyright information typically contains project, application, or author names. Keyword information from README files is also included in the directory summarizer, since these files usually contain high-quality information about the directory’s contents.

⁴ One possibility would be to sample a bitmap file down to an icon. While this does not easily support indexing, it could be used to support quick browsing before retrieving an entire image across a slow network.

4.5.2.2. Binary

An obvious method for a *binary summarizer* is to extract ASCII strings from the binary file, using the UNIX *strings* command. However, Essence filters these extracted ASCII strings using heuristics that only keep strings that convey the binary's purpose, such as usage, version, or copyright information. Essence also uses cross references to obtain high-quality summary information from binary executables. For example, the binary summarizer looks for associated manual pages for the given binary executable, and generates keywords using the *manual page summarizer* on them.

Table 2: Supported File Types in Common File System Environments

File Type	File Type Description	File Types Supported By			Frequency by Number of Files		Average File Size	
		Essence	WAIS	SFS	NFS	AFTP	NFS	AFTP
<i>Archive</i>	Library archives	×		×	0.36	0.12	626.31	47.52
<i>Binary</i>	Binary Executable	×		×	5.06	0.02	145.74	112.00
<i>C</i>	C source code	×	×	×	1.27	19.33	3.87	28.36
<i>CHeader</i>	C header files	×	×	×	14.73	22.42	4.29	2.40
<i>Command</i>	UNIX shell scripts	×	×		1.78	3.06	2.75	1.55
<i>Compressed</i>	Compressed file	×			0.69	11.81	114.98	73.29
<i>Directory</i>	Directory	×		×	4.87	5.05	0.81	0.50
<i>Dvi</i>	Device-indep. TeX	×	×		0.03	0.87	33.32	59.18
<i>Mail</i>	Electronic mail	×	×	×	0.02	0.17	1.79	35.30
<i>Makefile</i>	UNIX makefiles	×	×	×	0.26	3.87	0.85	3.04
<i>ManPage</i>	UNIX manual pages	×	×	×	13.78	0.70	6.76	295.92
<i>News</i>	USENET news articles	×	×	×	0.01	0.04	21.96	1.75
<i>Object</i>	Relocatable object file	×		×	0.00	1.12	0.00	28.11
<i>Patch</i>	File difference listing	×	×		0.02	0.00	1.88	0.00
<i>Perl</i>	Perl script	×	×		0.00	0.02	0.00	3.62
<i>PostScript</i>	PostScript images	×	×		1.42	3.31	64.64	194.45
<i>RCS</i>	RCS version ctrl files	×	×		0.00	1.41	0.00	8.41
<i>README</i>	High-quality info.	×	×	×	0.38	1.32	1.95	2.88
<i>SCCS</i>	SCCS version ctrl files	×	×		0.00	0.00	0.00	0.00
<i>ShellArchive</i>	Bourne shell archive	×			0.00	0.10	0.00	486.75
<i>Tar</i>	Tar archive	×			0.00	0.81	0.00	1734.21
<i>Tex</i>	TeX source docs	×	×	×	0.67	0.23	17.79	34.17
<i>Text</i>	Unstructured text	×	×	×	21.64	19.73	7.87	31.11
<i>Troff</i>	troff source docs	×	×		0.03	0.25	9.21	9.08
<i>Unknown</i>	Unknown file type	×		×	32.96	4.26	44.02	16.10

Table 3: Essence Summarizer Techniques

File Type	Summarizer Description
<i>Archive</i>	Extract symbol table
<i>Binary</i>	Extract meaningful strings, and manual page summary
<i>C</i>	Extract procedure names, #include'd filenames, and comments
<i>CHeader</i>	Extract procedure names, included filenames, and comments
<i>Command</i>	Extract comments
<i>Directory</i>	Extract directory listings, copyright information, and README files
<i>Dvi</i>	Convert to ASCII text
<i>Mail</i>	Extract select header fields
<i>Makefile</i>	Extract comments
<i>ManPage</i>	Extract author, title, etc., based on “-man” macros
<i>News</i>	Extract select header fields
<i>Object</i>	Extract symbol table
<i>Patch</i>	Extract filenames
<i>Perl</i>	Extract procedure names and comments
<i>PostScript</i>	Convert to ASCII text
<i>RCS</i>	Extract RCS-supplied summary
<i>README</i>	Use entire file
<i>SCCS</i>	Extract SCCS-supplied summary
<i>Tex</i>	Convert to ASCII text
<i>Text</i>	Extract first 100 lines
<i>Troff</i>	Extract author, title, etc., based on “-man”, “-ms”, “-me” macro packages, or extract section headers and topic sentences.

4.5.2.3. Formatted Text

Although formatted text (such as TeX, troff, or WordPerfect) has structured syntax, effectively summarizing these files is difficult unless semantic information is also available [Knuth 1984, Lamport 1986, USENIX 1986]. For example, plain troff files or troff files using the “-me” macros are difficult to exploit semantically, since their syntax is associated with formatting commands (such as font size or line spacing), rather than more conceptual commands (such commands to indicate an author’s name or paper title). troff files using the “-ms” or “-man” macros are much easier to summarize, since they contain conceptual commands (such as delimiting an abstract, author, and title).

Essence supports a sophisticated summarizer for troff and the “-me”, “-ms”, and “-man” troff macros. The *TeX summarizer* only extracts ASCII text from TeX files using *detex*, but exploiting TeX semantics would be a trivial extension of the methods used in our *troff summarizer*.

4.5.2.4. Simple Text

Simple text is difficult to summarize because it is unstructured. Essence assumes that the highest quality information in most unstructured text files is near the beginning of the file, as is common with paper abstracts or tables of contents. Therefore, the *text file summarizer* extracts keywords from the first one hundred lines of the text file. Natural language processing techniques might provide a better basis for a *text file summarizer*, although these techniques are usually domain-specific [Jacobs & Rau 1990].

README files, however, typically contain crucial, concise information about a distribution or application. Using a full-text index of README files provides high-quality keywords without occupying too much space. Therefore, the *README summarizer* uses the entire file to generate keywords.

The *Dvi*, *PostScript*, and *Tex summarizers* extract keywords from all of the ASCII text extracted from these files. Essence assumes that these file types contain generally useful information, and hence generates full text indexes for them.

4.5.2.5. Source and Object Code

Both source and object code are highly structured, and contain easily exploited semantic information. The *C summarizer* extracts procedure names, header filenames, and comments from a C source code file. Similarly, the *object summarizer* extracts the symbol table from an object file.

4.5.3. WAIS Interface

Essence exports its indexes through WAIS's search and retrieval interface, allowing users to use tools such as *waissearch* and the X Windows-based graphical user interface *xwais*. In order to generate WAIS-compatible indexes, Essence uses WAIS's indexing software to index the Essence summary files. This mechanism generates full-text WAIS indexes from the Essence summary files.

We modified WAIS's indexing mechanism to understand the format of the Essence summary files, so that it generates meaningful WAIS *headlines*. A headline is a short description of a single file, usually a filename. With Essence, headlines represent a file's core filename, its actual filename, and its file type.

To support additional file types, WAIS must be recompiled with new procedures that understand these file types. With Essence, one need only write a new summarizer, add its name to a configuration file, and add new heuristics for identifying the file type; no recompilation is necessary. In this sense, Essence modularizes the typed-file indexing extensions that WAIS can use, because it removes the keyword extraction process from WAIS and places it instead in Essence. Essence is better suited to incorporating new file types, and can be quickly adapted to become a *comprehensive* indexing system.

Figure 6 shows an example search of an index generated by Essence of the *ftp.cs.colorado.edu* anonymous FTP file system. It shows an ordered list of the ten files that best match the keyword *Netfind*. The headlines have up to three fields representing the matching file: the core filename, the filename (if different from the core filename), and the file type.

Consider the effectiveness of the example search in Figure 6. The best match is a PostScript paper that discusses a number of techniques for distributed information systems, with particular emphasis on techniques demonstrated by Netfind; the second match is the same file, but found in the compressed tar distribution *ALL.PS.tar.Z*. The third match is the C source code for the interactive user interface to Netfind. The fourth match is the README file found in the Netfind distribution directory; the fifth match is the same file, but found in the compressed tar distribution *netfind.3.10.tar.Z*. The sixth match is the UNIX manual page for Netfind. The remaining matches are PostScript papers in which Netfind is discussed.

In WAIS, a user retrieves files by selecting a matching headline. With Essence, if the headline represents a file hidden within a nested file (such as the first headline in Figure 6), the summary file is retrieved, instead of retrieving the hidden file itself. If the headline represents a plain file (such as the fourth headline in Figure 6), the file is retrieved. This functionality requires allocating storage for both the required summary files and the index. However, it allows users to browse through remote file systems by retrieving and viewing small summary files without having to retrieve complete files. This is useful when trying to decide whether to transfer large files across a slow network.

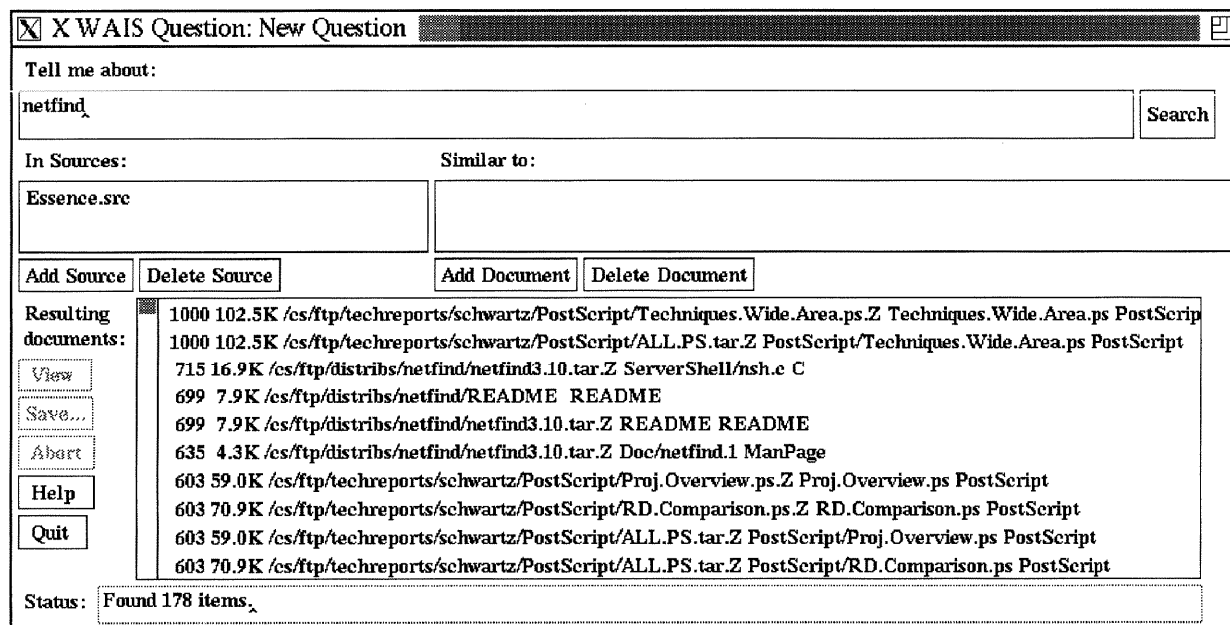


Figure 6: WAIS Search Using Essence-based Index

4.5.4. Performance

In this section, we present measurements of indexing speed and space efficiency, for Essence, WAIS, and SFS. We also discuss the usefulness and overhead of indexing nested files.

Before presenting measurements of the various systems, we note that it is difficult to interpret time and space efficiency measurements of the systems being compared, for two reasons.

First, indexing speed and compression are highly dependent on indexing techniques. For example, an indexer that skips most of the data (such as our Text summarizer) will achieve much higher indexing speeds and compression factors than one that uses all of these data (such as the Text indexers used by SFS and WAIS). In this case, the salient issue becomes recall/precision effectiveness of the generated indices (which is difficult to quantify). For example, a small, fast index would not be a reasonable tradeoff if one could not use this index to locate desired data.

Second, aggregate measurements (as shown in Table 5) are affected by the distribution of different file types in the sample file systems. Ideally, we would have measured each indexing system against the same file system data. We did this for WAIS and Essence, but the SFS code was not available at the time we made these measurements. Instead, we attempted to interpret the measurements given in [Gifford et al. 1991]. Notwithstanding these difficulties in interpreting the measurements, we feel it is worthwhile to present quantitative comparisons of these systems.

Table 4 presents the space and time measurements for Essence and WAIS, based on file type. (We do not show measurements for nested file types here. Those measurements are discussed in Table 7). As indicated in Table 2 and with a “-” in Table 4, WAIS and SFS cannot index all of the file types that Essence can. Table 4 shows that because there is a high amount of overhead associated with interpreting the semantics of a file type, Essence indexes slower than WAIS for some file types. Essence indexing is faster than WAIS for file types that have a low amount of semantic interpretation overhead.

Table 5 presents weighted averages of the space and time measurements in Table 4, based on the file type frequencies and average file sizes (as measured in Table 2). The weighted averages were computed using the formula:

$$\sum_{i=0}^n (f_i a_i) v_i, \text{ where } f_i \text{ is the frequency associated with file type } i, a_i \text{ is the average file size associated with file type } i,$$

v_i is the indexing rate or the compression factor value from Table 4 associated with file type i , and n is the number of file types supported by the system. $f_i a_i$ is used to normalize the measurements, to reflect only the system's supported file types. In particular, only non-nested file types are included in the aggregate measurements for WAIS and SFS (since those systems do not support nested files), while all types (including nested files) are included in the Essence measurements. We discuss the "unraveling" costs of dealing with nested file structures in Table 7.

Table 4: Time and Space Measurements for WAIS and Essence

File Type	Indexing Rate (KB/min)		Compression Factor vs. Index		Semantic Exploitation Overhead
	Essence	WAIS	Essence	WAIS	
<i>Archive</i>	3289.18	–	10.89	–	<i>low</i>
<i>Binary</i>	563.40	–	21.15	–	<i>high</i>
<i>C</i>	357.84	593.15	2.46	1.45	<i>high</i>
<i>CHeader</i>	164.87	342.93	1.33	0.65	<i>high</i>
<i>Command</i>	168.20	277.30	1.23	0.43	<i>high</i>
<i>Dvi</i>	278.71	2160.00	0.79	1.76	<i>high</i>
<i>Mail</i>	3718.12	1071.89	9.44	0.74	<i>low</i>
<i>Makefile</i>	421.05	648.65	2.33	0.86	<i>high</i>
<i>ManPage</i>	165.67	661.59	3.34	1.18	<i>high</i>
<i>News</i>	1913.29	329.24	20.82	0.63	<i>low</i>
<i>Object</i>	2588.75	–	15.93	–	<i>low</i>
<i>Patch</i>	7218.00	993.30	80.20	2.00	<i>low</i>
<i>Perl</i>	282.50	713.68	2.05	0.88	<i>high</i>
<i>PostScript</i>	1151.19	765.60	4.56	1.67	<i>low</i>
<i>RCS</i>	1293.16	614.25	5.91	1.27	<i>low</i>
<i>README</i>	390.00	400.83	0.68	0.65	<i>high</i>
<i>SCCS</i>	315.79	1500.00	3.12	3.52	<i>high</i>
<i>Tex</i>	598.93	385.52	2.09	0.92	<i>low</i>
<i>Text</i>	4699.59	1346.67	8.13	1.37	<i>low</i>
<i>Troff</i>	703.01	2036.92	7.04	1.97	<i>high</i>

Table 5: Weighted Time and Space Averages for WAIS and Essence

File System	Indexing Rate (KB/min)			Compression Factor vs. Index		
	Essence	WAIS	SFS	Essence	WAIS	SFS
<i>NFS</i>	1489.58	891.40	712.00	14.40	1.27	6.82
<i>Anonymous FTP</i>	1826.17	897.01	712.00	4.93	1.44	6.82
<i>Average</i>	1657.88	894.21	712.00	9.67	1.35	6.82

The Essence and WAIS measurements were performed on a Sun Microsystems 4/280 server running SunOS 4.1.1, with a local SMD disk. The SFS measurements were performed on a Microvax-3 running UNIX version 4.3BSD [Gifford et al. 1991]. This machine is approximately one-third as fast as the Sun 4/280.

Table 5 shows that Essence can index data faster than WAIS. Taking into account the slower machine on which SFS was measured, SFS appears to index data somewhat faster than Essence does.

Essence obtains about a 10:1 index compression factor on the file types that it supports, compared to WAIS (1:1), SFS (7:1), and *archie* (765:1) [Emtage & Deutsch 1992]. These measurements are not perfect, because detailed SFS measurements were not available.

Table 6 shows the percentage of data in the measured file systems that Essence, WAIS, and SFS were successfully able to interpret and index. The NFS file system contained many custom file formats that the indexing systems were unable to interpret. However, the anonymous FTP file system contained many more common file formats. Even though Essence only supports a relatively small number of common file types (21), it can index 75% of the data found in an average file system – far greater than WAIS (33%) or SFS (18%).

Table 6: Percentage of Interpretable Data

	Essence	WAIS	SFS
Anonymous FTP	98.51	48.47	27.56
NFS	50.70	17.88	8.11
Average	74.61	33.18	17.84

We found that seventy-eight percent of the files in our anonymous FTP had nested structure. These measurements indicate that supporting nested file structures is essential for such file systems. In contrast, only one percent of the files in the NFS file system had nested structure. In the future nested file structures may become less common, as they mostly represent inadequacies of current file systems and remote access protocols. For example, tar files are popular in FTP file systems because they make it easier to retrieve an entire directory tree, and FTP does not provide a recursive retrieval mechanism.

Table 7 shows how much overhead the prototype incurs when indexing nested files in the measured anonymous FTP file system. In this table, the *Original Data* row concerns the data which reside in the anonymous FTP file system. The *Processed Data* row concerns the data that Essence processes while indexing the *Original Data*. These data include all of the original files and each file within a nested file structure. For example, given the file *foo.tar.Z* from our previous example, *foo.tar.Z*, *foo.tar*, *foo.c*, *foo.h*, *Makefile*, and *README* are all included in the *Processed Data*. The *Summarized Data* row concerns the data on which summarizers are run. For example, *foo.c*, *foo.h*, *Makefile*, and *README* are all included in the *Summarized Data*. The *Summary Output* row concerns the resulting summary files. The resulting index of the summary files consumed 12.94 megabytes.

Note that this compression ratio (60.22/12.94) understates the actual compression, because the indexed data actually consumed 132.36 MB. In particular, indexing systems (like WAIS) that do not support nested structure would have to leave the data uncompressed. Hence, we actually achieve a twofold space reduction. WAIS would need to keep the uncompressed data around, and then would generate an index whose size was comparable to the uncompressed data. Essence generates a smaller index, and can function with compressed data. Putting the numbers together, WAIS would require approximately 265 MB of space for the uncompressed data plus index, while Essence requires only 73 MB total.

Table 7: Nested File Structure Overhead

	Total Number of Files	Total Size (in MB)
Original Data	1213	60.22
Processed Data	6409	262.03
Summarized Data	5334	132.36
Summary Output	5334	15.87

4.5.5. Analysis of Keyword Quality

Qualitative analysis of information retrieval systems is difficult. *Recall/precision* measurements are difficult to obtain, since they rely on hand-chosen reference sets [Salton & McGill 1983], and hence do not scale well to measuring large information collections. More effective measurements might be obtained by evaluating the effectiveness of a system from experience with an active user community. We have made the Essence prototype is publically available to allow users to make their own subjective judgements.

4.5.6. Prototype Availability

The Essence prototype, including its source code and WAIS modifications, is publically available by anonymous FTP from *ftp.cs.colorado.edu* in */pub/cs/distrib/essence*. The prototype is written in the C and Perl programming languages [Kernighan & Ritchie 1988, Wall & Schwartz 1991], and uses WAIS version 8-b5 [WAIS 1993].

4.5.7. Anonymous FTP Indexing

Currently, an Essence-generated index for file resources available from the anonymous FTP site at *ftp.cs.colorado.edu* is available through WAIS using the *afpt-cs-colorado-edu.src* WAIS source. Hundreds of users distributed throughout the Internet have used this WAIS interface to our anonymous FTP site. We would like, however, to make more anonymous FTP sites available through WAIS, using Essence-based indexes. Using Essence to index public archives allows remote users to search information based on conceptual descriptions and to view summaries before retrieving files. This would help decrease network traffic caused by retrieving unwanted files.

4.6. Conclusions

Automated file resource identification and simple, modular summarizers for automated file resource interpretation form the basis of an effective, scalable data mapping approach. The Essence prototype demonstrates that these scalable techniques are practical and efficient for supporting resource discovery in general purpose file systems. Furthermore, Essence's integration with the WAIS interface, and its use as an alternative interface to anonymous FTP sites demonstrate that Essence can effectively support Internet resource discovery through compact, yet representative indexes.

Chapter 5

Related Work

5.1. Identifying and Locating File Resources

Semantic indexing depends on successfully determining file types. Furthermore, Essence uses semantic indexing to locate file resources. Many systems can either determine file types or locate file resources, but Essence integrates both aspects into a single system.

- The Modules system is a sophisticated administrative approach to locating file resources associated with specific applications [Furlani 1991]. Applications are associated with a particular *module*, which can be easily incorporated or removed from a user's environment. Both the location and identification of the applications and their file resources are explicitly supplied by an administrator, and are hidden from the user.
- The NeXT file system browser determines common file types by exploiting filename extensions [NeXT 1991]. It then displays an icon representative of the file's type. Users can *launch* a specific application by supplying only a filename, as the application that is launched is determined by the file's type. Locating file resources is accomplished by browsing a UNIX file system hierarchy.
- The UNIX *file* command attempts to determine various file types based on file contents, but provides no mechanism for locating files [USENIX 1986].
- The UNIX *find* command locates files using an exhaustive search of a portion of a file system. It allows predicates to be specified concerning which files to locate. Among other things, these predicates can specify location based on the file types understood by the UNIX file system (such as ordinary file, directory, or symbolic link) [USENIX 1986, Leffler et al. 1989]. Higher-level types (such as image, script, or C source code) cannot be specified.
- Many programs use file naming conventions to infer file types. C compilers, for example, assume that a filename ending in ".c" is a C source file, while a file ending in ".o" is a relocatable object file. Similarly, *make* has various implicit rules based on file naming conventions.

5.2. Exploiting File Semantics

Semantic indexing also depends on the ability to extract good keyword information from files based on their file types. A number of UNIX commands can extract information with varying degrees of quality from files based on their file types [USENIX 1986].

- *ctags* extracts procedure, macro, and variable names from C source and header files. Some versions of *ctags* understand other programming languages, such as Lisp, Pascal, and C++.
- *strings* extracts embedded ASCII text strings from binary files.
- *deroff*, *detex*, and *ps2txt* extract ASCII text from troff, TeX, and PostScript files, respectively.
- *what* extracts embedded Source Code Control System (SCCS) information from files.

Essence provides a single cohesive system that integrates determining file types, locating file resources, and exploiting file semantics to extract good keyword information from files.

5.3. Semantic File Indexing

The MIT Semantic File System (SFS) uses semantic file indexing to provide a more effective storage abstraction than traditional hierarchical file systems [Gifford et al. 1991]. SFS exploits filename extensions to determine file types, and then runs *transducers* on files to extract keywords for building an index. SFS provides a *virtual directory* interface to search the resulting index and to access files. Virtual directory names are interpreted as queries against the index, and the contents of virtual directories are the results from these queries. Therefore, users perceive a search-based interface to explore file systems, rather than the more traditional hierarchical file system interface.

Although Essence and SFS use similar semantic indexing techniques, they differ in orientation, summarizer breadth, and space efficiency.

Orientation

SFS emphasizes semantic indexing as a storage abstraction. In contrast, Essence emphasizes semantic indexing as a basis for resource discovery. Concretely, while both systems support flexible associative access to file data, they export the data differently. Essence exports the data through a search and retrieval interface, while SFS exports the data through a file system interface. The advantage of the SFS approach is that it reuses an existing and familiar storage abstraction. The disadvantage is that doing so leads to undefined semantics. For example, if a user tries to copy data into a virtual directory (created as a result of an SFS query), the semantics are undefined.

Summarizer Breadth

Essence summarizers are autonomous UNIX programs, which are easy to implement, integrate, and maintain. The Essence prototype implements summarizers for many more file types than SFS does. Essence can index a wide variety of textual and binary data common in network file systems.

Space Efficiency

Our prototype provides better index compression than the SFS prototype.

5.4. Hypertext

Hypertext systems provide a less labor-intensive mechanism for users to browse information [Conklin 1987]. These systems use *links* to associate information within the system. Information is partitioned into *nodes*, usually a single paragraph or document. Related nodes are linked together through keyword-based links. Users then can select one of these keywords which triggers a link to access the related nodes automatically.


Some systems transparently link heterogeneous information resources through a hypertext interface [Noll & Scacchi 1991]. These systems employ gateways to access information from autonomous repositories. These gateways translate commands from the system into commands for the local repository. Similarly, Dynamic WAIS uses gateways to access information from autonomous remote search systems. Dynamic WAIS's gateways, however, provide more than simple command translations; they also provide a conceptual mapping between the WAIS search and retrieval paradigm and the paradigm for the remote search system.

For example, *NCSA Mosaic* is a hypertext-based, front-end user interface to many Internet information resources [Andreessen 1993]. It supports gateways to documents and other data available through Internet Gopher, WAIS, World Wide Web, FTP, Usenet, Archie, and many other sources. Figure 7 shows the NCSA Mosaic gateway interface from which users can query *archie*. The interface explicitly informs the user of the query semantics for the remote information system, *archie*. Since WAIS does not implement the Z39.50 facilities to handle differing query semantics, Dynamic WAIS similarly forces the user to deal with query semantics.

The Dynamic WAIS information-level gateway between WAIS and *archie*, however, provides a more powerful gateway than NCSA Mosaic. In Dynamic WAIS, users can select to query among many *archie* servers. In NCSA Mosaic, the system determines which *archie* server to query. This distinction is important, especially in the case of *archie* whose availability is unreliable. Due to its popularity, *archie* servers field a large number of queries each day. As a result, some *archie* servers take several minutes to reply. Dynamic WAIS users can quickly query another server that is less loaded if the query is taking too long or if the server is down. NCSA Mosaic users have no way of specifying another *archie* server to query.

Figure 8 shows how NCSA Mosaic returns *archie* results. NCSA Mosaic cross-references the hosts named within an *archie* result, so that users can immediately retrieve more information. Dynamic WAIS does not add any functionality to remote search system results; and therefore, browsing in Dynamic WAIS is more labor-intensive than NCSA Mosaic which provides hypertext links for users.

File Navigate Options Annotate Documents Manuals Help

Document Title: 

Document URL:

Archie Gateway

Archie is a search tool for locating documents and software packages at anonymous FTP sites on the Internet.

Enter a single-term search keyword.

Figure 7: NCSA Mosaic: *archie* Query

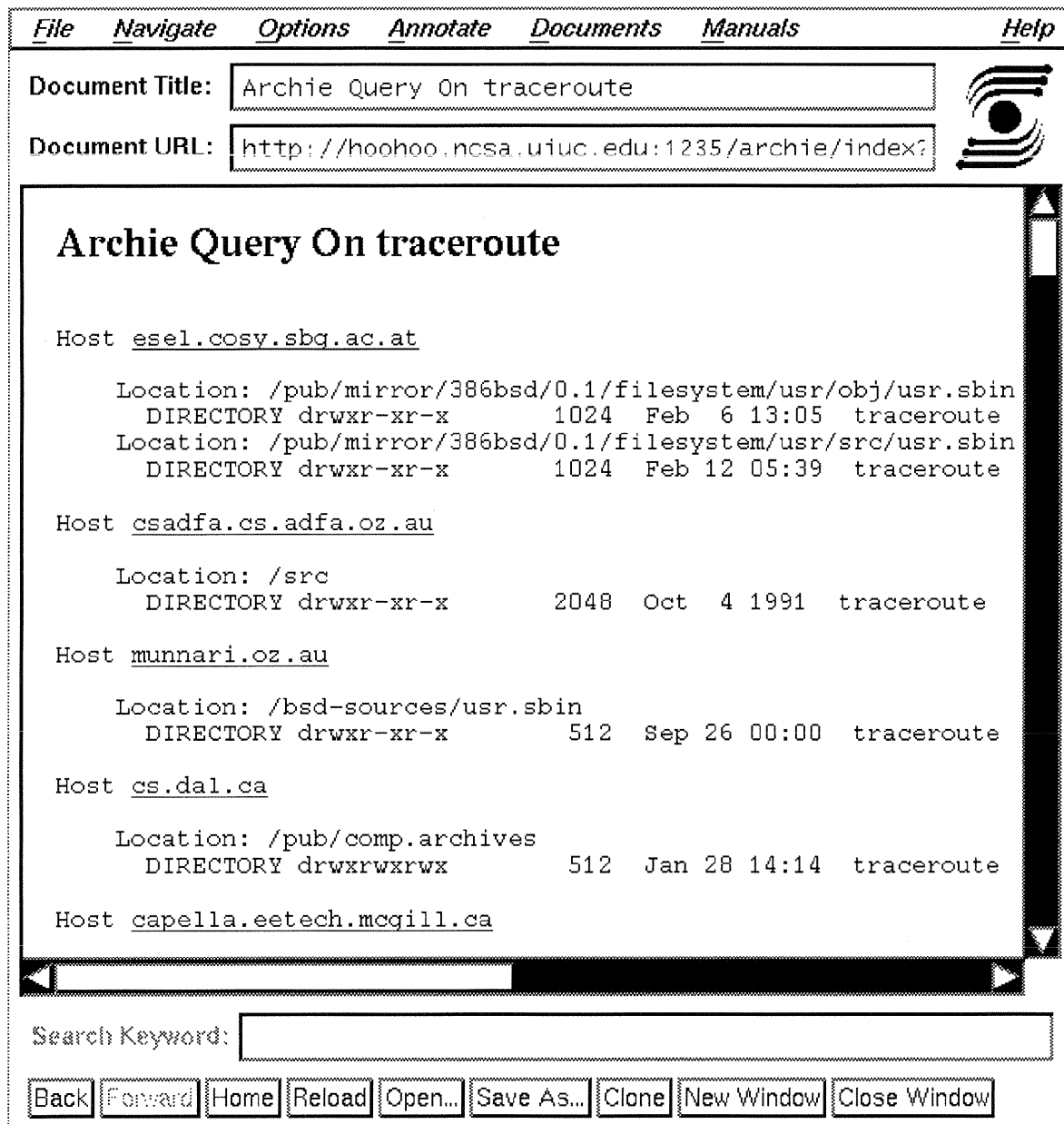


Figure 8: NCSA Mosaic: *archie* Response

5.5. Semantic Data Modeling

Semantic data models incorporate semantic information about data into a database [Hull & King 1987]. These models emphasize the content of data rather than its physical structure as in other models like the relational data model. Using semantic data models, database management systems can exploit data semantics to provide richer query languages that allow users to express their queries more conceptually. Essence exploits file semantics to provide users with a more expressive, content-based interface, but Essence only assumes the file semantics are present and it does not provide any mechanism to add such semantics to its information base.

5.6. Heterogeneous, Distributed Databases

Heterogeneous, distributed databases integrate several heterogeneous databases [Ram 1991, Ozsu & Valduriez 1991, Litwin et al. 1990, Sheth & Larson 1990, Thomas et al. 1990]. To support heterogeneous, distributed databases, systems must integrate the different databases.

One integration approach is to require that all of the supported databases conform to a *unified*, or a *global schema*. The global schema must support the data models for each of the integrated databases. This integration, however, is transparent to the user who can use the global schema to formulate queries. Through a global schema approach, heterogeneity problems are resolved through standardizing on a unified schema, but only at the cost of decreasing the autonomy of the supported databases. This approach differs from our work since it takes a homogeneous approach to heterogeneity – unifying schemas into a global schema which is a superset of all the supported schemas. As previously discussed, Essence does not take a homogeneous approach since semantic information may be lost while translating one file type into another. Since Essence depends on semantic information to product compact, yet representative indexes, it takes a heterogeneous approach, exploiting file semantics through the use of summarizers, to preserve semantic information better.

Another integration approach is to support heterogeneous, distributed, autonomous databases. This *multi-database*, or *federated*, approach supports sharing without a global schema. It provides mechanisms, or languages, by which a small portion of the schemas for a set of the supported databases can be integrated. The databases continue to operate independently, yet through these mechanisms, the system can correlate different schemas to allow sharing. Typically, multidatabases are read-only databases, since supporting updates is difficult without a global schema. In this respect, Dynamic WAIS is similar to the multidatabase approach. Dynamic WAIS supports read-only access to information available through autonomous, information systems. However, since Dynamic WAIS does not need to conform to database semantics, it does not need to support the same services that heterogeneous, distributed database systems do like transaction processing and query optimization. Therefore, Dynamic WAIS can employ simpler techniques which yield better scalability. In particular, distributed transaction processing has difficulty scaling with growing wide-area networks [Ozsu & Valduriez 1991]. Dynamic WAIS does not have these scalability problems since it does not adhere to database semantics, and therefore, does not implement any sophisticated transaction processing.

Chapter 6 Future Directions

6.1. Scalable Content-Based Searching

Combining Essence's semantic indexing techniques with more efficient index generation techniques, such as the two-level hybrid indexing techniques in the *Personal Information Retrieval System* (PIRS) [Manber & Wu 1993], might prove to significantly increase Essence's scalability. Figure 9[†] shows how PIRS combined with Essence might further the scalability of Internet resource discovery approaches by supporting very compact, yet representative indexes. With PIRS's space efficiency, Essence might support flat content-based searches through 2,000 one-gigabyte disks, approximately all of the data available via anonymous FTP, with a top-level index that fits on a single one-gigabyte disk.

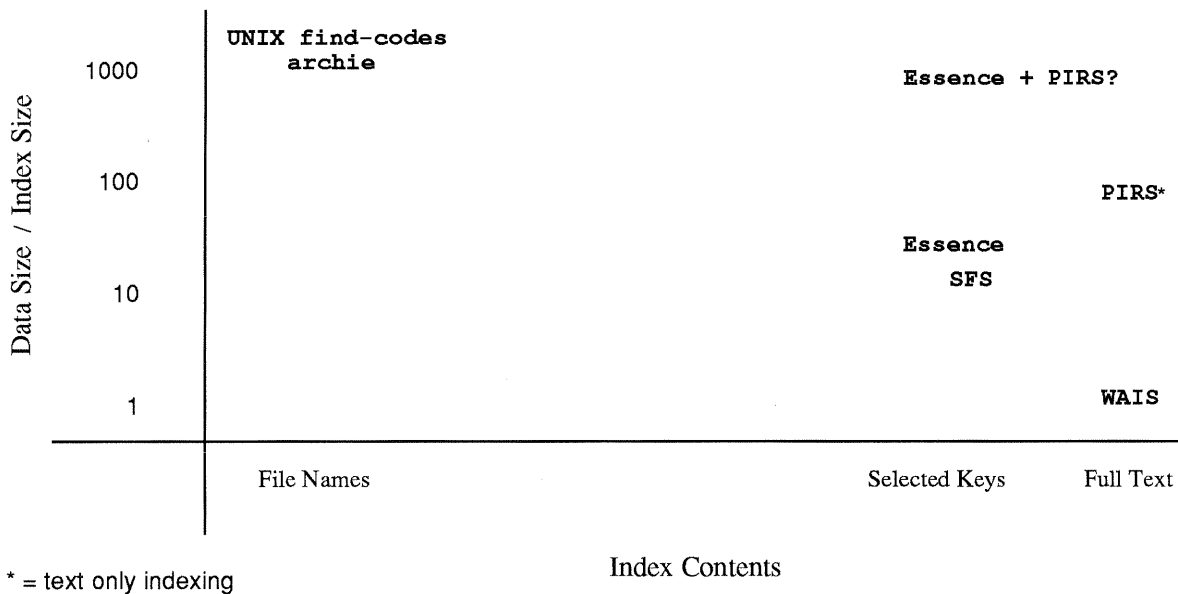


Figure 9: Indexing Space Efficiency vs. Representativeness

Even though a combination of Essence and PIRS might generate an extremely small index, gathering the enormous amount of information to be indexed over a wide-area network may be infeasible. Research into distributing the index computation might result in techniques that minimize the amount of information that is gathered over network links. Once the index is computed, replicating the index to improve access by many users is also a research problem.

[†] Reprinted with permission from [Bowman et al. 1993].

6.2. General Techniques for Information-Level Gateways

Dynamic WAIS is our only implementation of information-level gateways. As a result, we have too little experience with these gateways to provide much insight in developing *general* techniques for building them. Further research might lead to the development of some general techniques for building them.

6.3. Supporting Differing Query Semantics in WAIS

Although Z39.50 supports facilities to support differing query semantics, WAIS does not implement them. Dynamic WAIS demonstrates the usefulness of supporting differing query semantics, so implementing them in WAIS would be useful.

Chapter 7

Summary

In this chapter we present a brief summary of this thesis.

Internet resource discovery systems are inadequately prepared for the Internet's growth in information diversity, user base, and information volume. This thesis focuses on developing Internet resource discovery techniques to adapt to diversifying information which includes both diverse resource representations, such as multimedia file formats, and diverse information systems, such as various remote search systems like Netfind and *archie*. Developing these techniques provides a basis for the next generation of Internet resource discovery systems, since it allows the inclusion a more diverse information base. We have developed two Internet resource discovery systems, *Dynamic WAIS* and *Essence*, to experiment with some of these techniques. Both our Dynamic WAIS and Essence prototypes are publically available, and they have been used by people at over four hundred sites in two dozen countries.

Operation mapping uses information-level gateways to provide automated, transparent access to information that is available through heterogeneous information system interfaces. Dynamic WAIS employs this approach to provide information-level gateways between WAIS and Netfind, and WAIS and *archie*. The Dynamic WAIS design recognizes that the operations of certain remote search systems, like Netfind, *archie*, and *whois*, can be mapped into the WAIS search and retrieval paradigm. In this respect, Dynamic WAIS contributes an important conceptual mapping between WAIS's operations and the operations of remote search systems.

Data mapping provides automated resource interpretation and classification of heterogeneous resource representations, while preserving semantic information about resources. Essence takes this approach by using semantic indexing which has two major benefits. First, it automates the identification and interpretation of diverse, heterogeneous file resource representations, or *file types*. Second, through effective resource interpretation techniques that preserve semantic information, Essence can exploit file semantics to generate compact, yet representative indexes. Our Essence prototype generates indexes that are compatible with WAIS, so that they can be exported through the WAIS interface. Currently, we export an Essence-generated index for file resources available from the anonymous FTP site at *ftp.cs.colorado.edu* is available through the WAIS interface. Essence's integration with the WAIS interface, and its use as an alternative interface to anonymous FTP sites demonstrate that Essence can effectively support Internet resource discovery through compact, yet representative indexes.

Chapter 8

References

[Adobe 1990]

Adobe Systems Inc., *PostScript Language Reference Manual*, Addison-Wesley, Reading, MA, 1990.

[Andreessen 1993]

M. Andreessen, *NCSA Mosaic for the X Window System*, Software Development Group, National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, April 1993, available via anonymous FTP from ftp.ncsa.uiuc.edu:/Web/xmosaic.

[ANSI 1991]

ANSI, "ANSI Z39.50," Version 2, Third Draft, American National Standards Institute, May 1991.

[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*, 1(2), Meckler Publications, Spring 1992.

[Bowman et al. 1993]

C. M. Bowman, P. B. Danzig, and M. F. Schwartz, "Research Problems for Scalable Internet Resource Discovery," Technical Report, Department of Computer Science, University of Colorado, Boulder, CO, March 1993.

[CCITT/ISO 1988]

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

[Conklin 1987]

J. Conklin, "Hypertext: An Introduction and Survey," *Computer*, 20(9), pp. 17-41, IEEE Computer Society, September 1987.

[Droms 1990]

R. E. Droms, "Access to Heterogeneous Directory Services," *Proceedings of the 9th Joint Conference of IEEE Computer and Communications Societies (InfoCom)*, June 1990.

[Emtage & Deutsch 1992]

A. Emtage and P. Deutsch, "archie - An Electronic Directory Service for the Internet," *Proceedings of the 1992 Winter USENIX Technical Conference*, pp. 93-110, January 1992.

[Foster 1992]

S. Foster, "About the Veronica Service," Electronic bulletin board posting on the comp.infosystems.gopher newsgroup, November 1992.

[Furlani 1991]

J. L. Furlani, "Modules: Providing a Flexible User Environment," *Proceedings of the USENIX Large Installation Systems Administration V Conference*, pp. 141-152, October 1991.

[Gifford et al. 1991]

D. K. Gifford, P. Jouvelot, M. A. Sheldon, and J. W. O'Toole, Jr., "Semantic File Systems," *Proceedings of the 13th ACM Symposium Operating Systems Principles*, pp. 16-25, October 1991.

[Granrose 1990]

J. Granrose, "Anonymous FTP Site Listing," available via anonymous FTP from pilot.njin.net:/pub/ftp-list, Cowell College, University of California, Santa Cruz, CA, 1990.

[Hardy & Schwartz 1993a]

D. R. Hardy and M. F. Schwartz, "Essence: A Resource Discovery System Based on Semantic File Indexing," *Proceedings of the 1993 Winter USENIX Technical Conference*, pp. 361-373, January 1993.

[Hardy & Schwartz 1993b]

D. R. Hardy and M. F. Schwartz, "Extending the WAIS Paradigm to Support Dynamic Information Spaces," Technical Report, Department of Computer Science, University of Colorado, Boulder, CO, 1993, In preparation.

[Harrenstien et al. 1985]

K. Harrenstien, M. Stahl, and E. Feinler, "NICNAME/WHOIS," Request for Comments 954, SRI International, October 1985.

[Hull & King 1987]

R. Hull and R. King, "Semantic Database Modeling: Survey, Applications, and Research Issues," *ACM Computing Surveys*, **19**(3), pp. 201-260, September 1987.

[Jacobs & Rau 1990]

P. S. Jacobs and L. F. Rau, "SCISOR: Extracting Information from On-line News," *Communications of the ACM*, **33**(11), pp. 88-97, November 1990.

[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., November 1991.

[Kantor & Lapsley 1986]

B. Kantor and P. Lapsley, "Network News Transfer Protocol," Request for Comments 977, SRI International, February 1986.

[Kehoe 1992]

B. P. Kehoe, *Zen and the Art of the Internet: A Beginner's Guide to the Internet*, Widener University, Chester, PA, 1992.

[Kernighan & Ritchie 1988]

B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[Klingenstein 1992]

K. Klingenstein, "A Coming of Age: Design Issues in the Low-End Internet," in *Building Information Infrastructure*, ed. B. Kahin, McGraw-Hill, New York, NY, 1992.

[Knuth 1984]

D. E. Knuth, *The TeXbook*, Addison-Wesley, Reading, MA, 1984.

[Krol 1989]

E. Krol, "The Hitchhiker's Guide to the Internet," Request for Comments 1118, SRI International, September 1989.

[Krol 1992]

E. Krol, *The Whole Internet User's Guide and Catalog*, O'Reilly and Associates, Inc., Sebastopol, CA, 1992.

[Lamport 1986]

L. Lamport, *LaTeX: A Document Preparation System*, Addison-Wesley, Reading, MA, 1986.

[LaQuey 1990]

T. LaQuey, *The User's Directory of Computer Networks*, Digital Press, Burlington, MA, 1990.

[LaQuey 1993]

T. LaQuey, *The Internet Companion: A Beginner's Guide to Global Networking*, Addison-Wesley, Reading, MA, 1993.

[Leffler et al. 1989]

S. J. Leffler, M. K. McKusick, M. J. Karels, and J. S. Quarterman, *The Design and Implementation of the 4.3BSD UNIX Operating System*, Addison-Wesley, Reading, MA, 1989.

[Litwin et al. 1990]

W. Litwin, L. Mark, and N. Roussopoulos, "Interoperability of Multiple Autonomous Databases," *ACM Computing Surveys*, **22**(3), pp. 267-293, September 1990.

[Lottor 1992]

M. Lottor, "Internet Growth (1981-1991)," Request for Comments 1296, SRI International, January 1992.

[Lynch & Rose 1993]

D. C. Lynch and M. T. Rose, *Internet System Handbook*, Addison-Wesley, Reading, MA, 1993.

[Manber & Wu 1993]

U. Manber and S. Wu, "A Two-Level Approach to Information Retrieval," Technical Report, Department of Computer Science, University of Arizona, Tucson, AZ, January 1993.

[Martin 1993]

J. Martin, "There's Gold in them thar Networks! – or – Searching for Treasure in all the Wrong Places," Request for Comments 1402, January 1993.

[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.

[Muntz & Honeyman 1992]

D. Muntz and P. Honeyman, "Multi-Level Caching in Distributed File Systems - or - Your Cache Ain't Nuthin' But Trash," *Proceedings of the 1992 Winter USENIX Technical Conference*, pp. 305-313, January 1992.

[Neuman 1992]

B. C. Neuman, "The Prospero File System: A Global File System Based on the Virtual System Model," *Computing Systems*, 5(4), pp. 407-432, University of California Press, Fall 1992.

[NeXT 1991]

NeXT Computer, Inc., *NeXT User's Reference*, NeXT Computer, Inc., Redwood City, CA, 1991.

[Noll & Scacchi 1991]

J. Noll and W. Scacchi, "Integrating Diverse Information Repositories: A Distributed Hypertext Approach," *Computer*, 24(12), pp. 38-45, IEEE Computer Society, December 1991.

[NSF 1989]

NSF Network Service Center, *Internet Resources Guide*, Bolt, Beranek, and Newman Inc., January 1989.

[Ousterhout et al. 1985]

J. Ousterhout, H. Da Costa, D. Harrison, J. Kunze, M. Kupfer, and J. Thompson, "A Trace-Driven Analysis of the UNIX 4.2 BSD File System," *Proceedings of the 10th ACM Symposium Operating System Principles*, pp. 15-24, December 1985.

[Ozsu & Valduriez 1991]

M. T. Ozsu and P. Valduriez, "Distributed Database Systems: Where Are We Now?," *Computer*, 24(8), pp. 68-78, IEEE Computer Society, August 1991.

[Quarterman 1990]

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

[Ram 1991]

S. Ram, "Heterogenous Distributed Database Systems," *Computer*, 24(12), pp. 7-10, IEEE Computer Society, December 1991.

[Rose 1991]

M. T. Rose, *The Little Black Book: Mail-Bonding with OSI Directory Services*, Prentice Hall, Englewood Cliffs, NJ, 1991.

[Salton & McGill 1983]

G. Salton and M. J. McGill, *An Introduction to Modern Information Retrieval*, McGraw-Hill, New York, NY, 1983.

[Schatz 1990]

B. R. Schatz, "Interactive Retrieval in Information Spaces Distributed across a Wide-Area Network," Technical Report 90-35, Department of Computer Science, University of Arizona, Tucson, AZ, December 1990.

[Schwartz et al. 1987]

M. F. Schwartz, J. Zahorjan, and D. Notkin, "A Name Service for Evolving, Heterogeneous Systems," *Proceedings of the 11th ACM Symposium Operating Systems Principles*, pp. 52-62, November 1987.

[Schwartz et al. 1992a]

M. F. Schwartz, D. J. Ewing, and R. S. Hall, "A Measurement Study of Internet File Transfer Traffic," Technical Report CU-CS-571-92, Department of Computer Science, University of Colorado, Boulder, CO, January 1992.

[Schwartz et al. 1992b]

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, University of California Press, Fall 1992.

[Schwartz & Tsirigotis 1991]

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), pp. 23-50, March 1991.

[Sheth & Larson 1990]

A. P. Sheth and J. A. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Computing Surveys*, 22(3), pp. 183-236, September 1990.

[Thomas et al. 1990]

G. Thomas et al., "Heterogeneous Distributed Database Systems for Production Use," *ACM Computing Surveys*, 22(3), pp. 237-266, September 1990.

[USENIX 1986]

USENIX Association, *UNIX Supplementary Documents, 4.3 Berkeley Software Distribution*, USENIX Association, Berkeley, CA, November 1986.

[WAIS 1993]

Thinking Machines Corp., "WAIS Source Distribution, version 8-b5," available via anonymous FTP from [think.com/wais](ftp://think.com/wais), Thinking Machines Corp., Cambridge, MA, 1993.

[Wall & Schwartz 1991]

L. Wall and R. L. Schwartz, *Programming Perl*, O'Reilly and Associates, Inc., Sebastopol, CA, 1991.

[Zimmerman 1990]

D. Zimmerman, "The Finger User Information Protocol," Request for Comments 1288, SRI International, November 1990.